UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ELETRÔNICA E MICROELETRÔNICA

DISSERTAÇÃO DE DOUTORADO

DESENVOLVIMENTO DE NOVAS TÉCNICAS E UM APLICATIVO PARA SÍNTESE, MINIMIZAÇÃO E SIMULAÇÃO DE FUNÇÕES DIGITAIS MULTI-VALORES

MARCO AURÉLIO SELUQUE FREGONEZI

ORIENTADOR: ALBERTO MARTINS JORGE

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica.

Banca Examinadora:

Prof. Dr. Alberto Martins Jorge (Presidente)
Prof. Dr. Maria Nídia Ramos Daoud Yacoub
Prof. Dr. Nivaldo Vicençotto Serran
Prof. Dr. José Antonio Siqueira Dias
Prof. Dr. Elnatan Chagas Ferreira
Prof. Dr. Oseas Valente de Avilez Filho

Data da defesa: 10/07/2006
Última revisão: 22/08/2006

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

	Fregonezi, Marco Aurélio Seluque
F842d	Desenvolvimento de novas técnicas e um aplicativo para síntese, minimização e simulação de funções digitais multi-valores / Marco Aurélio Seluque Fregonezi. --Campinas, SP: [s.n.], 2006.
	Orientador: Alberto Martins Jorge Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.
	1. Lógica a múltiplos valores. 2. Sistema ternário. 3. Simulação (Computador). 4. Lógica algébrica. 5. Álgebra booleana. I. Jorge, Alberto Martins. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Titulo em Inglês: Software and new techniques for multi-valued digital functions synthesis, minimization and simulation
Palavras-chave em Inglês:
Área de concentração: Eletrônica e Microeletrônica
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: Maria Nídia Ramos Daoud Yacoub, Nivaldo Vicençotto Serran, José Antonio Siqueira Dias, Elnatan Chagas FerreiraOseas Valente de Avilez Filho
Data da defesa: 10/07/2006

Resumo

Este trabalho consiste na análise de regras de síntese e minimização de funções digitais multi-valores e na criação de um software para a realização automatizada da síntese empregando as regras criadas e, também, na criação de um software para simulação de portas lógicas e circuitos digitais multi-valores. Todas essas ações são coerentes entre si. A partir de uma tabela verdade, pode-se gerar uma expressão algébrica por meio do software que realiza a síntese; a partir dessa expressão, pode-se construir um circuito no simulador lógico criado para a observação do resultado; pode-se, também, usar um simulador analógico (*SPICE™*, por exemplo) para verificar a viabilidade da implementação física do circuito. Observa-se que o resultado obtido no simulador analógico é igual ao obtido no simulador lógico criado, que também é coerente com a tabela verdade original.

Abstract

This work consists in the investigation of rules for synthesis and minimization of digital multi-valued functions; in the creation of a software for automated performing of synthesis applying the proposed rules and also of a software for multi-valued logic gates and circuits simulation. All those features are jointly coherent. Starting with a truth-table, it is possible to generate an algebraic expression using the implemented software; starting from that expression, a circuit can be constructed using the created logic simulator to analyze the result. An analogical simulator (like *SPICE™*, for example) can be used to verify the viability of the circuit physical implementation. It is observed that the result obtained with the analogical simulator is equivalent to the result from the logic simulator; with is also coherent to the given truth table.

Agradecimentos

A Deus por ter dado a capacitação, a vocação e o interesse necessários para a realização deste trabalho e por ter colocado pessoas que tanta importância tiveram na realização desta obra, colocação que serviu para ensinar que nenhuma obra de valor pode ser realizada sozinho. Tais pessoas são citadas a seguir.

- Prof. Dr. Alberto Martins Jorge, pela orientação e pela criação da álgebra estendida de Post e da técnica de síntese utilizada, âmago deste trabalho.
- Os meus pais, pelo incentivo e suporte dado.
- Os colegas de pesquisa Oswaldo Hugo Bertone e Carlos Júnior Mingotto, pelas discussões pertinentes à lógica multi-valores e pelos momentos de descontração.
- Minha noiva Lina, pela revisão gramatical.
- Todos os integrantes do DEMIC (professores, funcionários e alunos), que proporcionaram um ambiente de trabalho favorável e humano.

À minha noiva

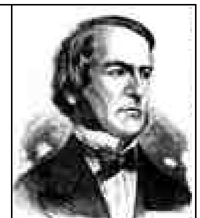
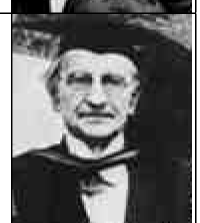
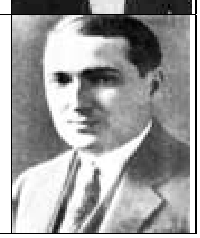
Lina

Índice

Introdução

Dados históricos:

No século XIX, o matemático inglês George Boole formalizou a álgebra de Boole. Em 1917, o lógico ucraniano Jan Lukasiewicz desenvolveu o cálculo proposicional de três valores e trabalhou na lógica multi-valores (MVL). A obra do matemático polonês Emil Leon Post, em 1921[1], foi a primeira álgebra publicada com completeza funcional para qualquer base[2]. A álgebra de Boole é um sob-conjunto da álgebra de Post. O teorema de De Morgan, aplicado à lógica de Post, permitiu aumentar os horizontes neste campo científico.

George Boole 2 Nov 1815. em Lincoln, Lincolnshire, Inglaterra 8 Dez 1864. em Ballintemple, County Cork, Irlanda http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Boole.html (21/07/2006)	
Augustus De Morgan 27 Jun. 1806 em Madurai, Tamil Nadu, Índia 18 Mar. 1871 em Londres, Inglaterra http://www-history.mcs.st-andrews.ac.uk/Mathematicians/De_Morgan.html (21/07/2006)	
Jan Lukasiewicz 21 Dez. 1878 em Lvov, Ucrânia 13 Fev. 1956 em Dublin, Irlanda http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Lukasiewicz.html (21/07/2006)	
Emil Leon Post 11 Fev. 1897 em Augustów, now Polônia 21 Abr. 1954 em Nova Iorque, EUA http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Post.html (21/07/2006)	

Motivação:

Quanto maior for a base adotada, menor será o número de dígitos necessários para expressar um certo número[1,3,4,5], porém a complexidade de sistemas, circuitos e manipulações algébricas aumenta[3].

A lógica MVL pode ter campo de aplicação em todas as áreas do conhecimento humano, mas, no presente momento, a maior motivação para o desenvolvimento de pesquisas nessa área está na engenharia elétrica, mais especificamente, na microeletrônica. Na medida em que processadores mais sofisticados vão sendo desenvolvidos, maior se torna o percentual da área interna do chip ocupada pelas trilhas das interconexões do circuito integrado[1,3,5,6,7,8,9] e, também, maior se torna o número de pinos (pads) de conexões externas[1]. Uma redução nesses dois números pode ajudar no projeto de processadores mais rápidos. Também ajuda na confecção de chips de memória[6], o que já vem sendo realizado[4] (processadores e memórias correspondem ao setor mais importante da microeletrônica). Um grande número de trilhas no chip aumenta a área do dispositivo e dificulta o projeto das máscaras do circuito integrado. Um grande número de interconexões entre chips causa problemas de alta freqüência, pois trilhas muito longas comportam-se como indutores e trilhas muito próximas comportam-se como capacitores. Aumentando-se a freqüência de operação, aumenta-se a reatância indutiva ($X_L=2\cdot\pi\cdot f\cdot L$), isto é, aumenta-se a resistência AC das trilhas; também se diminui a reatância capacitiva ($X_C=1/(2\cdot\pi\cdot f\cdot C)$), isto é, diminui-se a isolação AC entre as trilhas.

Este é um trabalho sobre matemática, computação e engenharia elétrica.

Em relação à matemática, o trabalho consiste no aperfeiçoamento das metodologias aplicadas na síntese de funções lógicas MVL. São apresentadas novas técnicas de síntese e de minimização; são explicadas, detalhadamente, as origens de cada propriedade de simplificação. Também é oferecida uma introdução aprofundada do tipo de lógica utilizado e de suas características elementares. Esta é a parte mais importante. Nela, são apresentados todos os conceitos pertinentes sobre a lógica adotada e sobre as técnicas de síntese e minimização. Ela é importante porque dá a oportunidade para que outros pesquisadores continuem este trabalho, realizando novas descobertas e aprimoramentos, ou, até mesmo, apresentando alternativas mais eficientes.

As explicações sobre o processo de síntese e, principalmente, de minimização, mostram que tal processo compreende uma quantidade grande de análises e tomadas de decisão, que tornariam o uso manual de tal processo inviável nas aplicações práticas[3]. Juntamente com o desenvolvimento das metodologias, foi criado um software capaz de executar essas tarefas. Isso significa que o usuário não precisa, necessariamente, compreender as técnicas de síntese e minimização apresentadas neste trabalho para fazer uso da lógica MVL. Esta é a parte mais interessante para o usuário final. O software está disponível para uso e distribuição e pode ajudar muito os cientistas envolvidos na lógica MVL, tal como já vem ajudando os colegas de departamento.

Apesar dos softwares não serem concebidos, especificamente, para a engenharia elétrica (nenhum parâmetro elétrico é incluído), é neste campo de atuação que ele tem o seu emprego mais comum. A idéia inicial é a de auxiliar os pesquisadores da área de lógica MVL, mais especificamente, os projetistas de circuitos digitais. Na seção 9.5, é apresentada a simulação de um circuito didático, em caráter de motivação. Tal simulação é interessantes porque mostra como o software pode prever, corretamente, o comportamento das portas lógicas e dos circuitos; como o uso correto das portas pode permitir a implementação de expressões lógicas mais complexas e como os princípios da lógica empregada podem ser observados por meio da construção de determinados circuitos. Uma vez que circuitos simulados em outros aplicativos não representam o ponto central deste trabalho, os problemas referentes às suas construções na forma de circuito integrado não são abordados. O circuito é apresentado na forma de blocos lógicos, sem abordagem detalhada a respeito do circuito interno dos blocos, pois não é esse o foco deste trabalho.

No que diz respeito ao software, há duas partes, ambas bastante distintas, tendo, em comum, apenas a lógica MVL.

LMVS: Lógica MVL – Síntese: Esse programa, tal como o nome diz, realiza, apenas, a síntese e a minimização de funções. Ele é o resultado da implementação computacional da teoria abordada na primeira parte, a matemática. Os conceitos teóricos são mais importantes do que sua implementação, pois, a partir deles, outro programador pode criar seu próprio software; a implementação faz parte deste trabalho porque, devido ao tamanho da complexidade do processo de síntese e minimização, o trabalho manual torna-se inviável, sendo necessário o uso de uma ferramenta automatizada. O software LMVS não apresenta nenhuma relação com portas lógicas nem com circuitos. Este software é um aperfeiçoamento da primeira versão, apresentada na tese de mestrado do mesmo autor, e está registrado na Agência de Inovação (INOVA - Unicamp) com o número 58.692.

LMVP: Lógica MVL – Portas/Circuitos lógicos: Este programa, tal como o nome diz, realiza a simulação das portas lógicas utilizando a lógica MVL, e dos circuitos que utilizam tais portas. Esse programa é dividido em duas partes: a primeira lida, apenas, com a definição e simulação da porta lógica; a segunda lida com a construção e simulação do circuito construído com a porta lógica escolhida na primeira parte. Não há nenhuma relação com expressões algébricas, síntese nem minimização. Ele é o resultado da implementação computacional dos dispositivos construídos a partir das expressões algébricas fornecidas pelo software LMVS. LMVP está registrado na Agência de Inovação (INOVA - Unicamp) com o número 58.680.

É apresentado, neste trabalho, um conhecimento não limitado a uma única aplicação. A matemática, a álgebra e a lógica podem ser empregadas em qualquer ramo do conhecimento humano (engenharia, física, biologia, química, psicologia, sociologia, medicina, etc).

Com o passar do tempo e o aperfeiçoamento da ciência, poderão surgir novas aplicações para este trabalho. Pode ser que, muitos anos mais tarde, surja um nicho científico no qual este trabalho venha a ser empregado. Quando os principais matemáticos trabalharam na lógica booleana e, posteriormente, na MVL, jamais se imaginava a construção de circuitos integrados lógicos MVL. Um trabalho desenvolvido para uma aplicação específica pode perder seu valor quando tal aplicação deixa de ser empregada, mas, quando não há este vínculo, o trabalho continua tendo seu valor, como uma ferramenta disponível para futuras necessidades. Pode ser, também, que, dispondo de um certo conhecimento, a humanidade somente encontre aplicação para este conhecimento anos após a morte de seu descobridor.

Até mesmo o programa de simulação de circuitos lógicos possui essa desvinculação, pois um circuito pode apresentar qualquer fenômeno real. Uma porta lógica pode ficar no lugar de, desde um circuito elétrico, como de um dispositivo mecânico, pneumático, hidráulico, ótico, como, também, das possibilidades de tomadas de decisões realizadas por uma pessoa. Um cruzamento de duas avenidas com quatro semáforos de três estágios pode ser representado por um circuito ternário. O programa oferece a possibilidade da definição do tempo de atraso das portas, mas, até mesmo essa característica é desvinculada das aplicações; dispositivos microeletrônicos, elétricos, mecânicos, hidráulicos, pneumáticos, dentre outros, possuem tempo de atraso. Economia, sociologia, biologia e outras ciências também apresentam fenômenos com tempo de atraso.

Como parte deste trabalho, foi requerida uma patente junto à Agência de Inovação (INOVA - Unicamp) sob título “Circuito Controlador De Semáforo Baseado Na Lógica Multi-Valores MVL” de autoria de Alberto Martins Jorge e Marco Aurélio Seluque Fregonezi em 28/07/2005 sob número PI0504717-0.

Os capítulos 1,2 e 3 apresentam conceitos que já foram introduzidos no trabalho de mestrado, embora com menos profundidade. Na versão divulgada no mestrado, focalizou-se o resultado, enquanto neste trabalho focaliza-se, também, a formalização dos procedimentos. No capítulo 3, novos conceitos são introduzidos. Os demais capítulos consistem de assuntos inéditos.

No final desta obra, está o glossário, onde estão as explicações sobre todas as terminologias, siglas e símbolos empregados ao longo do texto.

Este trabalho pode ser dividido em três blocos:

- Capítulos 1 a 3: Aprofundamento da teoria apresentada no trabalho de mestrado (apresentação de aperfeiçoamentos na teoria da lógica MVL e inovações).
- Capítulos 4 e 5 e apêndices 1, 2 e 3: Novas técnicas de minimização, de síntese e de manipulação algébrica.
- Capítulos 6, 7 e 8 e apêndice 4: Os aplicativos.

Capítulo 1
Introdução à Lógica MVL de Post

1.1 Teoria dos conjuntos

A teoria dos conjuntos se baseia no diagrama da figura 1.1.

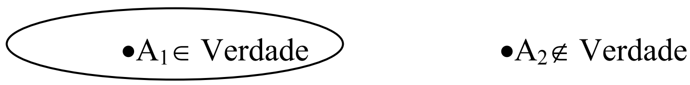


Figura 1.1 – O conjunto A.

O elemento A_1 pertence ao conjunto *Verdade*, CV, mas A_2 não. Pode-se definir, então, a região interna ao conjunto como a área verdadeira, o conjunto *Verdade*, e a região externa como a área falsa, que não pertence a CV. Define-se como *Verdade* a satisfação de uma sentença ou proposição. Um elemento pertencente ao CV assume o valor "V", *Verdadeiro*, e se não pertencer ao CV, assume o valor "F", *Falso*. Define-se o negador, complemento ou inversor, um operador, simbolizado por uma barra, que tira os elementos contidos no conjunto e coloca os elementos não contidos. O inverso do valor "V" é o "F" e vice-versa. A variável "A" complementada pode ser representada por "-A", " A' ", "/A", etc.

- $(A_1 \in CV) \rightarrow$ A=V (1a) Se A=V, /A=F(2a)
- $(A_1 \notin CV) \rightarrow$ A=F (1b) Se A=F, /A=V(2b)

Para dois conjuntos (ou duas sentenças ou proposições), tem-se:

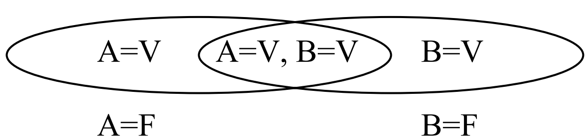


Figura 1.2 – Os conjunto A e B.

- A intersecção dos conjuntos é dada por $A \cap B$=V (3a)
- A união dos conjuntos é dada por $A \cup B$=V (3b)

- $A \cap B$=V $\rightarrow$ $A \cap B$ = (A=V e B=V) (4a)
- $A \cup B$=V $\rightarrow$ $A \cup B$ = (A=V ou B=V) (4b)

Se, ao valor falso, for atribuído o número "0" e, ao valor verdadeiro, o número "1", pode-se fazer a seguinte dedução:

- A região A∩B é dada por (A=1 e B=1). (5a)
- A região A∪B é dada por (A=1 ou B=1). (5b)

A união e a intersecção são conectivos. Conectivo[3,7] é a operação que conecta duas ou mais coordenadas (variáveis de entrada) gerando a variável de saída. Há, ainda, mais dois conectivos[2,8,10]:

- Implicação ou condicional (se), "→", também chamado de "*está contido*", "⊂".
- Igualdade, equivalência ou bicondicional (se e somente se) "↔".

- A intersecção "∩" também é chamada de conjunção "∧" ou mínimo ou AND "*".
- A união "∪" também é chamada de disjunção "∨" ou máximo ou OR "+".

(A=1 e B=1) ↔ A∧B=1.(6a)	(A=0 ou B=0) ↔ A∧B=0.(7a)	(A=1 e B=1) → A∨B=1.(8a)
(A=0 e B=0) ↔ A∨B=0.(6b)	(A=1 ou B=1) ↔ A∨B=1.(7b)	(A=0 e B=0) → A∧B=0.(8b)

Tabela 1.1 – Propriedades.

	∧	∨
Elemento dominante	0	1
Elemento neutro ou indiferente	1	0

Tabela 1.2– Definição dos elementos dominante e indiferente.

O condicional (também chamado de unicondicional) é o único conectivo que não se aplica à propriedade comutativa. Essa teoria é resumida nas tabelas 1.3 e 1.4, na forma de tabela-verdade e mapa de Karnaugh, respectivamente, e na figura 1.3, na forma de conjuntos. Nos mapas, as colunas representam a variável "A" e as linhas "B". Os conectivos "∧", "∨" comparam as duas variáveis e podem chamados de comparadores de dominância[6]. Além desses, há um outro tipo de operador, os quantificadores[10]:

- Universal: B=(α)A Para qualquer "A", B(A)=1. (9a)
- Existencial: B=(∃α)A Existe pelo menos um "A" para o qual B(A)=1. (9b)

- Operadores: "–", "∧", "∨", "→", "↔", "(α)", "(∃α)".
- Conectivos: "∧", "∨", "→" e "↔".
- Comparadores de dominância: "∧", "∨".

A	B	A∧B	A∨B	A→B	A↔B
0	0	0	0	1	1
1	0	0	1	0	0
0	1	0	1	1	0
1	1	1	1	1	1

Tabela 1.3 – A definição dos conectivos na forma de tabela-verdade.

∧	**0**	**1**
0	0	0
1	0	1

∨	**0**	**1**
0	0	1
1	1	1

→	**0**	**1**
0	1	0
1	1	1

↔	**0**	**1**
0	1	0
1	0	1

Tabela 1.4 – A definição dos conectivos na forma de mapa de Karnaugh.

Por essas tabelas, pode-se definir a intersecção "∩" como sendo a multiplicação lógica "*" e a união como sendo a adição lógica "+". Também com base na tabela, pode-se verificar que, na multiplicação, o valor menor domina sobre o maior e que, na adição, o valor maior domina sobre o menor; essas funções podem ser chamadas de MÍNIMO e MÁXIMO, respectivamente.

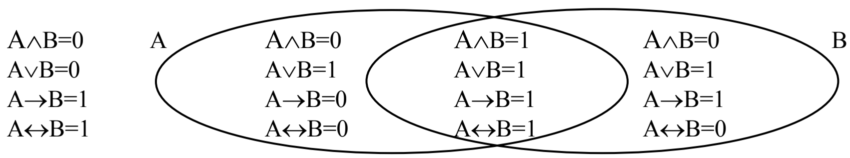


Figura 1.3 – A definição dos conectivos na forma de conjuntos.

A	B	C	∧	∨	→	↔
0	0	0	0	0	1	1
1	0	0	0	1	0	0
0	1	0	0	1	0	0
1	1	0	0	1	0	0
0	0	1	0	1	1	0
1	0	1	0	1	0	0
0	1	1	0	1	1	0
1	1	1	1	1	1	1

Tabela 1.5 – A definição dos conectivos para três entradas

Os elementos desse sistema lógico assumem duas situações: Estar contido ou não em certo conjunto, ser verdadeiro ou falso.

Assumindo um valor intermediário, tem-se um novo valor, "½", que indica que a situação não é nem verdadeira nem falsa, é um estado intermediário. Cada variável pode assumir três valores (0, ½ e 1). Isso permite formar nove pares ordenados ou coordenadas ou combinações. Os quatro conectivos mantém suas propriedades, podendo ser aplicados para qualquer base. O unicondicional e o bicondicional possuem pouca aplicação para bases maioress do que 2.

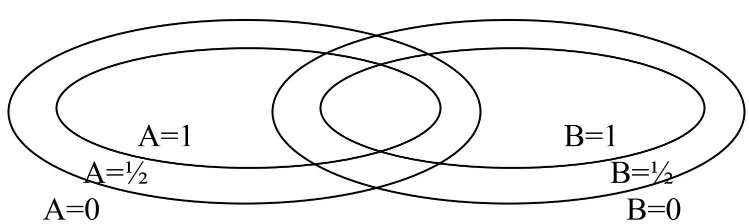


Figura 1.4 – Os conjuntos A e B.

A	B	A∧B	A∨B	A→B	A↔B
0	0	0	0	1	1
½	0	0	½	½	½
1	0	0	1	0	0
0	½	0	½	1	½
½	½	½	½	½	½
1	½	½	1	½	½
0	1	0	1	1	0
½	1	½	1	1	½
1	1	1	1	1	1

As tabelas para o condicional e o bicondicional diferem daquelas fornecidas em [2] e foram obtidas por meio das fórmulas[10]:

$$A \rightarrow B = /A \vee B \qquad (10a)$$

$$A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A) \qquad (10b)$$

Tabela 1.6 – A definição dos conectivos na forma de tabela-verdade, para três valores.

∧	**0**	**½**	**1**
0	0	0	0
½	0	½	½
1	0	½	1

∨	**0**	**½**	**1**
0	0	½	1
½	½	½	1
1	1	1	1

→	**0**	**½**	**1**
0	1	½	0
½	1	½	½
1	1	1	1

↔	**0**	**½**	**1**
0	1	½	0
½	½	½	½
1	0	½	1

Tabela 1.7 – A definição dos conectivos na forma de mapa de Karnaugh, para três valores.

As fórmulas binárias para definição dos conectivos podem ser aplicadas. Os elementos neutros e dominantes também são os mesmos. Essas regras valem para qualquer quantidade de valores.

A inversão do valor "½" fornece esse mesmo valor em lógica ternária, pois ele está no centro da escala de valores; o inverso do meio termo é o meio termo. Esse fato está relacionado com a famosa pergunta "o copo está meio cheio ou meio vazio?". Com base nesse procedimento, pode-se definir os operadores para qualquer quantidade de valores.

0	Falso	Não	Nada	Nunca	Circuito aberto	Claridade	Seco
½	Meio-falso	Talvêz	Metade	Algumas vezes	Comportamento ôhmico	Penumbra	Úmido
1	Verdadeiro	Sim	Tudo	Sempre	Circuito fechado	Escuridão	Molhado

Tabela 1.8 – Exemplos de interpretação para os valores.

A	**0**	**½**	**1**
/A	1	½	0

A	**0**	**$^1/_3$**	**$^2/_3$**	**1**
/A	1	$^2/_3$	$^1/_3$	0

Tabela 1.9 – O inversor para três e quatro valores.

∧	**0**	**$^1/_3$**	**$^2/_3$**	**1**
0	0	0	0	0
$^1/_3$	0	$^1/_3$	$^1/_3$	$^1/_3$
$^2/_3$	0	$^1/_3$	$^2/_3$	$^2/_3$
1	0	$^1/_3$	$^2/_3$	1

∨	**0**	**$^1/_3$**	**$^2/_3$**	**1**
0	0	$^1/_3$	$^2/_3$	1
$^1/_3$	$^1/_3$	$^1/_3$	$^2/_3$	1
$^2/_3$	$^2/_3$	$^2/_3$	$^2/_3$	1
1	1	1	1	1

→	**0**	**$^1/_3$**	**$^2/_3$**	**1**
0	1	$^2/_3$	$^1/_3$	0
$^1/_3$	1	$^2/_3$	$^1/_3$	$^1/_3$
$^2/_3$	1	$^2/_3$	$^2/_3$	$^2/_3$
1	1	1	1	1

↔	**0**	**$^1/_3$**	**$^2/_3$**	**1**
0	1	$^2/_3$	$^1/_3$	0
$^1/_3$	$^2/_3$	$^2/_3$	$^1/_3$	$^1/_3$
$^2/_3$	$^1/_3$	$^1/_3$	$^2/_3$	$^2/_3$
1	0	$^1/_3$	$^2/_3$	1

Tabela 1.10 – A definição dos conectivos para quatro valores.

Um sistema de regras de inferência é funcionalmente completo se e somente se for possível derivar, a partir de um conjunto de sentenças, todas as conseqüências daquele conjunto[10]. Em outras palavras, um sistema completo é aquele para o qual, a partir dos operadores definidos, pode-se obter todas as funções definíveis por mapa, tabela ou outra forma de definição que venha a ser criada. Isso equivale a afirmar que o sistema lógico completo permite que qualquer mapa seja definido por meio de uma expressão algébrica[1]. Sendo assim, a expressão algébrica é uma outra forma de definição de função. O sistema lógico apresentado, também chamado de Lógica de Lukasiewicz[1], só é completo para lógica binária. Para lógica ternária ou superior, ele não é completo, ou seja, não se podem converter todos os mapas para expressão. Isso se deve ao fato de que a quantidade de funções definíveis por meio desses operadores é limitada. Por exemplo, a função B(A) da tabela 1.11 não pode ser obtida.

A conjunção pode se transformar em disjunção e vice-versa por meio do teorema de De Morgan (fórmula (11)). O sistema lógico apresentado, na forma binária, é completo usando-se, apenas o inversor e um dos conectivos (qualquer um dos quatro, porém é mais fácil empregar um dos comparadores de dominância). Essa lógica pode ser formalizada partindo-se do inversor junto com o mínimo ou do inversor junto com o máximo.

A	**0**	**$^1/_3$**	**$^2/_3$**	**1**
/A	$^1/_3$	$^2/_3$	1	0

Tabela 1.11 – Exemplo de função que não pode ser gerada.

$$\overline{A \wedge B} = \overline{A} \vee \overline{B} \quad (11a)$$
$$\overline{A \vee B} = \overline{A} \wedge \overline{B} \quad (11b)$$

Para lógicas de três ou mais valores tal sistema lógico é incompleto porque usa o inversor, e este operador não permite que se façam todas as transformações de uma entrada. Por exemplo, para três valores, pode-se transformar “0” em “1” e vice versa, mas não se pode transformar “0” em “½” e vice-versa, nem “½” em “1” e vice-versa. A inversão permite que um valor seja substituído, apenas, pelo seu inverso, ou seja, quanto maior for a base, menor é o percentual de funções definíveis algebricamente por meio deste sistema lógico.

1.2 Lógica fechada, cíclica ou circular

A lógica cíclica fechada (ou circular conectada) possui uma distribuição de valores seqüencial repetitiva na qual uma ascensão progressiva leva à repetição de valores, como será mostrado mais adiante, na explicação sobre os deslocadores. Na lógica fechada, usam-se números inteiros. Neste sistema lógico, não há uma atribuição fixa dos valores "verdadeiro" e "falso". Por exemplo, para a lógica negativa não cíclica de três valores, o valor "0" é o verdadeiro, o valor "1" é o falso, e o valor "½" é o intermediário. Na lógica cíclica:

- Se "0" for o valor verdadeiro, "1" é o falso.
- Se "1" for o valor verdadeiro, "2" é o falso.
- Se "2" for o valor verdadeiro, "0" é o falso.

O operador de uma entrada não é um inversor, mas um deslocador[3].

- BASE[3,4,8,11,12,13,14,15,16,17] (ou predecessor): deslocador para baixo
- TOPO[3,4,8,11,12,13,14,15,16,17] (ou sucessor): deslocador para cima

- BASE: Se A>0 então B=A–1 senão B=n–1 (12a)
- TOPO: Se A<n–1 então B=A+1 senão B=0 (12b)

Ao contrário do inversor, o deslocador pode ser de dois tipos. O operador BASE é o deslocador para baixo porque decresce o valor de entrada em uma unidade. O operador TOPO é o deslocador para cima porque eleva o valor de entrada em uma unidade. Se for considerado um conjunto em ordem crescente, com todos os valores admissíveis na lógica adotada, como, por exemplo, {0,1,2} para lógica ternária, então o operador BASE é chamado de deslocador para a esquerda[11,15] e o operador TOPO é chamado de deslocador para a direita[11,15]. O operador BASE também pode ser considerado uma subtração aritmética unitária, e o TOPO uma adição aritmética unitária. O operador BASE é representado por uma barra em baixo da expressão e o operador TOPO por uma barra em cima da expressão. O conjunto de valores para uma lógica de n valores é {0,1,...,n–1}. Por ser uma lógica fechada, o deslocamento TOPO do valor máximo (n–1) é o valor mínimo (0) e o deslocamento BASE do valor mínimo é o valor máximo. O operador TOPO também pode ser chamado de deslocador crescente[8] e o BASE de deslocador decrescente[8].

Os deslocadores também podem ser definidos por meio da operação MOD[1,5,7] (módulo, proveniente da teoria dos números), equação (13), um recurso muito utilizado em linguagens de programação que oferece como resposta o resto da divisão inteira. Por exemplo, 2 mod 3 = 2, 4 mod 3 = 1.

- O deslocador TOPO é um negador[3,4,6,11,12,13,14,15], ou seja, se a entrada receber o valor “verdadeiro”, a saída dá o valor “falso”.
- O deslocador BASE é um positivador, ou seja, ou seja, se a entrada receber o valor “falso”, a saída dá o valor “verdadeiro”.

Os valores provenientes do deslocamento BASE do valor verdadeiro e do deslocamento TOPO do valor falso não são nem verdadeiros, nem falsos. Por exemplo, se, em lógica ternária fechada, o valor verdadeiro for atribuído a “0”, o falso é “1”; o deslocamento TOPO de “1” é “2”, que não recebe atribuição de verdadeiro nem de falso. Se o valor verdadeiro for “1”, o falso é “2”; o deslocamento TOPO de “2” é “0”, que não recebe atribuição de verdadeiro nem de falso. Se, em lógica quaternária fechada, o valor verdadeiro for atribuído a “0”, o falso é “1”; o deslocamento TOPO de “1” é “2”, que não recebe atribuição de verdadeiro nem de falso.

Por meio de qualquer um desses dois deslocadores, todas as transformações (de um valor para outro) são possíveis, trata-se de um sistema lógico completo [11,18].

- BASE: B= (A–1) mod n (13a)
- TOPO: B= (A+1) mod n (13b)

- BASE: $\underline{F} = V$ - Operador YES (14a)
- TOPO: $\overline{V} = F$ - Operador NOT (14b)

Funções que geram, apenas, dois valores consecutivos podem ser chamadas de binárias MVL, pois, fornecem o valor verdadeiro e o seu respectivo deslocamento TOPO, o valor falso, assumindo um comportamento parecido com o das funções booleanas. Tais funções podem interagir com funções booleanas por meio da adequação das grandezas físicas atribuídas a cada valor lógico. As funções binárias MVL são muito importantes no processo de síntese MVL, como será apresentado no capítulo 3.

1.3 Álgebra de Post

O sistema Lógico de Post, em sua formulação original, utiliza o conectivo máximo "$\vee$" e o deslocador negador [3,4], genéricos para qualquer base. Esse conectivo é mostrado na tabela 1.12. As colunas representam a variável "A" e as linhas a variável "B".

Dois valores

$\vee$	**0**	**1**
0	0	1
1	1	1

Três valores

$\vee$	**0**	**1**	**2**
0	0	1	2
1	1	1	2
2	2	2	2

Quatro valores

$\vee$	**0**	**1**	**2**	**2**
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

Cinco valores

$\vee$	**0**	**1**	**2**	**3**	**4**
0	0	1	2	3	4
1	1	1	2	3	4
2	2	2	2	3	4
3	3	3	3	3	4
4	4	4	4	4	4

Tabela 1.12 – Tabelas-verdade para o operador $\vee$ de dois, três, quatro e cinco valores.

Pode-se agrupar várias negações, como mostrado na tabela 1.13.

Dois valores

A	$\bar{A}$
0	1
1	0

Três valores

A	$\bar{A}$	$\bar{\bar{A}}$
0	1	2
1	2	0
2	0	1

Quatro valores

A	$\bar{A}$	$\bar{\bar{A}}$	$\bar{\bar{\bar{A}}}$
0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	2

Cinco valores

A	$\bar{A}$	$\bar{\bar{A}}$	$\bar{\bar{\bar{A}}}$	$\bar{\bar{\bar{\bar{A}}}}$
0	1	2	3	4
1	2	3	4	0
2	3	4	0	1
3	4	0	1	2
4	0	1	2	3

Tabela 1.13 – Negador.

Devido ao fato da negação do maior valor da base adotada resultar no menor valor e da positivação do menor valor resultar no maior valor este operador recebe o nome de negação cíclica de Post [3,4,6,11,12,13,14,15].

1.4 Formalismo aplicado à álgebra de Post

A álgebra apresentada neste trabalho corresponde à mesma álgebra de Post original, mas com uma notação estendida. Essa extensão na notação visa facilitar a escrita das expressões[1]. Com base nesta notação, cria-se as regras de manipulação algébrica para a realização da síntese e minimização. O mesmo procedimento é aplicado à lógica booleana e na de Lukasiewicz quando se escreve uma expressão na forma disjuntiva ou conjuntiva, pois ambas as formas usam dois tipos de conectivos (AND e OR), sendo que a expressão pode ser escrita usando-se, apenas, um conectivo. A síntese baseada no uso da notação estendida também é chamada de "método dos operadores"[11,15].

O teorema de De Morgan pode ser usado para definir conectivos auxiliares[3,12]. Para tal, deve-se escolher um deslocador e um conectivo do tipo comparador de dominância. Com um deslocador e um conectivo pode-se criar um sistema lógico completo. Neste trabalho, escolheu-se o deslocador TOPO e o conectivo Mínimo "∧" [3,11,12,15]. Uma vez que os conectivos derivados dos operadores escolhidos por meio do teorema de De Morgan são diferentes do Mínimo e do Máximo, é conveniente escolher outros nomes, de forma a padronizá-los. Optou-se por usar letras gregas [3,4,11,12,13,14,15] (α,β,γ,δ). A primeira letra (ALFA- "α"[3,4,11,12,13,14,15]) é atribuída ao primeiro conectivo, aquele a partir do qual os demais são determinados (Mínimo, neste caso). O nome "negador cíclico" é substituído por TOPO[3,4,11,13]. O número de conectivos gerados dessa forma coincide com o número de valores (por exemplo, tem-se dois conectivos em lógica binária e três em ternária), tal como é mostrado na tabela 1.14. No caso particular da lógica booleana, o conectivo BETA corresponde ao comparador Máximo.

$\overline{x}\,\beta\,\overline{y} = \overline{x\,\alpha\,y}$ (15a) $\quad \overline{x}\,\gamma\,\overline{y} = \overline{x\,\beta\,y}$ (15b) $\quad \overline{x}\,\delta\,\overline{y} = \overline{x\,\gamma\,y}$ (15c)

2valores	3 valores	4 valores	5 valores	Símbolo	Dominante
ALFA	ALFA	ALFA	ALFA	α	0
BETA	BETA	BETA	BETA	β	1
	GAMA	GAMA	GAMA	γ	2
		DELTA	DELTA	δ	3
			TETA	τ	4

Tabela 1.14 – Os conectivos.

Qualquer conectivo pode ser usado para formar um sistema lógico, desde que satisfaça a necessidade da completeza funcional. Por exemplo, poder-se-ia criar o sistema a partir da função ALFA-TOPO, isto é, a função ALFA com o deslocamento TOPO.

Partindo-se do conectivo Mínimo, não se pode obter o conectivo Máximo, nem vice versa, por meio da aplicação do teorema de De Morgan. Isso deve ser feito por meio de diversas operações algébricas (síntese), como pode ser visto na literatura.

Para cada conectivo, existe uma escala de dominância. Quando duas variáveis são ligadas por meio de um conectivo, a saída assume o valor que tiver a maior dominância. A entrada cujo valor tiver maior dominância prevalece sobre as outras. Para um determinado conectivo, o valor de maior dominância é chamado de <u>dominante</u> [3,8,18] (**D**) ou principal[4] e o de menor dominância (elemento neutro), <u>indiferente</u> [8,11,18] (**I**). Valores intermediários recebem nomes ordinais pertinentes à sua posição na escala de dominâncias.

- O valor dominante é dado pelo deslocamento TOPO do valor indiferente.
- O valor indiferente é dado pelo deslocamento BASE do valor dominante.

O valor dominante é definido para cada conectivo independente da base adotada, mas o valor indiferente depende da base. Por exemplo, para o conectivo ALFA, o valor dominante é sempre “0”, mas o valor indiferente é “1” para base 2, “2” para base “3” e asim por diante. Usando-se o teorema de De Morgan, define-se todos os demais conectivos[3,12,13]:

É definido, também, outro deslocador, BASE[3,4,8,11,12,13,14,15,17], de ação antagônica à do deslocador TOPO, simbolizado por uma barra inferior. Ele serve para simplificar a grafia, evitando grande acúmulo de barras acima dos caracteres. Algumas igualdades são mostradas na tabela 1.15.

Dois valores	Três valores	Quatro valores
$\overline{\overline{A}} = A$ [16a]	$\overline{\overline{\overline{A}}} = A$ [16d]	$\overline{\overline{\overline{\overline{A}}}} = A$ [16h]
$\overline{A} = \underline{A}$ [16b]	$\overline{\overline{A}} = \underline{A}$ [16e]	$\overline{\overline{\overline{A}}} = \underline{A}$ [16i]
$A = \underline{\underline{A}}$ [16c]	$\overline{A} = \underline{\underline{A}}$ [16f]	$\overline{\overline{A}} = \underline{\underline{A}}$ [16j]
	$A = \underline{\underline{\underline{A}}}$ [16g]	$\overline{A} = \underline{\underline{\underline{A}}}$ [16k]
		$A = \underline{\underline{\underline{\underline{A}}}}$ [16l]

Tabela 1.15 – Igualdades.

2 valores

$$\overline{A\,\alpha\,B} = \overline{A}\,\beta\,\overline{B} \quad [17a]$$
$$\overline{A\,\beta\,B} = \overline{A}\,\alpha\,\overline{B} \quad [17b]$$

3 valores

$$\overline{A\,\alpha\,B} = \overline{A}\,\beta\,\overline{B} \quad [17c] \qquad \underline{A\,\alpha\,B} = \underline{A}\,\gamma\,\underline{B} \quad [17f]$$
$$\overline{A\,\beta\,B} = \overline{A}\,\gamma\,\overline{B} \quad [17d] \qquad \underline{A\,\gamma\,B} = \underline{A}\,\beta\,\underline{B} \quad [17g]$$
$$\overline{A\,\gamma\,B} = \overline{A}\,\alpha\,\overline{B} \quad [17e] \qquad \underline{A\,\beta\,B} = \underline{A}\,\alpha\,\underline{B} \quad [17h]$$

4 valores

$\overline{A\,\alpha\,B} = \overline{A}\,\beta\,\overline{B}$ [17i]	$\overline{\overline{A\,\alpha\,B}} = \overline{\overline{A}}\,\gamma\,\overline{\overline{B}}$ [17m]	$\underline{A\,\alpha\,B} = \underline{A}\,\delta\,\underline{B}$ [17q]
$\overline{A\,\beta\,B} = \overline{A}\,\gamma\,\overline{B}$ [17j]	$\overline{\overline{A\,\beta\,B}} = \overline{\overline{A}}\,\delta\,\overline{\overline{B}}$ [17n]	$\underline{A\,\delta\,B} = \underline{A}\,\gamma\,\underline{B}$ [17r]
$\overline{A\,\gamma\,B} = \overline{A}\,\delta\,\overline{B}$ [17k]	$\overline{\overline{A\,\gamma\,B}} = \overline{\overline{A}}\,\alpha\,\overline{\overline{B}}$ [17o]	$\underline{A\,\gamma\,B} = \underline{A}\,\beta\,\underline{B}$ [17s]
$\overline{A\,\delta\,B} = \overline{A}\,\alpha\,\overline{B}$ [17l]	$\overline{\overline{A\,\delta\,B}} = \overline{\overline{A}}\,\beta\,\overline{\overline{B}}$ [17p]	$\underline{A\,\beta\,B} = \underline{A}\,\alpha\,\underline{B}$ [17t]

Os conectivos podem, também, ser obtidos um a partir de outro diretamente:

$$A\beta B = \overline{\underline{A}\,\alpha\,\underline{B}} \quad [18a]$$

$$A\gamma B = \overline{\overline{\underline{\underline{A}}\,\alpha\,\underline{\underline{B}}}} \quad [18b]$$

$$A\delta B = \overline{\overline{\overline{\underline{\underline{\underline{A}}}\,\alpha\,\underline{\underline{\underline{B}}}}}} \quad [18c]$$

$$A\gamma B = \overline{\underline{A}\,\beta\,\underline{B}} \quad [18d]$$

$$A\delta B = \overline{\overline{\underline{\underline{A}}\,\beta\,\underline{\underline{B}}}} \quad [18e]$$

$$A\delta B = \overline{\underline{A}\,\gamma\,\underline{B}} \quad [18f]$$

As fórmulas (19c) e (19d) também podem ser deduzidas a partir das fórmulas (19a) e (19b); a fórmula (19f) pode ser deduzida a partir da fórmula (19e), como mostrado na tabela 1.16. Observa-se, então, que todos os conectivos podem ser definidos para qualquer base, aplicando-se o operador MOD aos valores fornecidos. Por exemplo, o valor “2” para base 2 é dado por *2 MOD 2 = 0*.

Equivalências para dois valores:

[16a]	$\rightarrow 2 = 0$	[19a]
[16a]	$\rightarrow 3 = 1$	[19b]
[16a],[18b]	$\rightarrow A\gamma B = A\alpha B$	[19c]
[16a],[18b],[18c]	$\rightarrow A\delta B = A\beta B$	[19d]

Equivalências para três valores:

[16d]	$\rightarrow 3 = 0$	[19e]
[16d],[18c]	$\rightarrow A\delta B = A\alpha B$	[19f]

δ	**3**	**0**	**1**	**2**
3	3	3	3	3
0	3	0	0	0
1	3	0	1	1
2	3	0	1	2

3=0 →

δ	**0**	**0**	**1**	**2**
0	0	0	0	0
0	0	0	0	0
1	0	0	1	1
2	0	0	1	2

→

α	**0**	**1**	**2**
0	0	0	0
1	0	1	1
2	0	1	2

Tabela 1.16 – Dedução de (19f) a partir de (19e).

2 valores:

A lógica de dois valores coincide com a lógica booleana, e sua exposição, embora não apresente nenhum conceito novo, serve como introdução para a lógica de três e quatro valores. Ela também serve como uma forma de comparação entre a lógica booleana e a MVL, mostrando que a segunda é uma expansão da primeira.

- São definidos dois valores: 0 e 1.

- São definidos dois conectivos: ALFA e BETA.

α	α	β
Dominante (**D**)	0	1
Indiferente (**I**)	1	0

cn	D	S
D	D	D
S	D	S

α	0	1
0	0	0
1	0	1

β	0	1
0	0	1
1	1	1

Tabela 1.17 – Conectivos (cn) de dois valores.

3 valores:

- São definidos três valores: 0,1 e 2.
- São definidos três conectivos: ALFA, BETA e GAMA.

	α	β	γ
Dominante (**D**)	0	1	2
Secundário (**S**)	1	2	0
Indiferente (**I**)	2	0	1

cn	D	S	I
D	D	D	D
S	D	S	S
I	D	S	I

Tabela 1.18 – Hierarquia de dominâncias para os conectivos (cn) de três valores.

	α	β	γ
D	0	1	2
S	1	2	0
I	2	0	1

α	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

β	0	1	2
0	0	1	2
1	1	1	1
2	2	1	2

γ	0	1	2
0	0	0	2
1	0	1	2
2	2	2	2

Tabela 1.19 – Conectivos de três valores [4].

4 valores:

- São definidos quatro valores: 0,1,2 e 3.
- São definidos quatro conectivos: ALFA, BETA, GAMA e DELTA.

	α	β	γ	δ
Dominante (**D**)	0	1	2	3
Secundário (**S**)	1	2	3	0
Terciário (**T**)	2	3	0	1
Indiferente (**I**)	3	0	1	2

cn	D	S	T	I
D	D	D	D	D
S	D	S	S	S
T	D	S	T	T
I	D	S	T	I

Tabela 1.20 – Hierarquia de dominâncias para os conectivos de quatro valores.

α	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

β	0	1	2	3
0	0	1	2	3
1	1	1	1	1
2	2	1	2	2
3	3	1	2	3

γ	0	1	2	3
0	0	0	2	3
1	0	1	2	3
2	2	2	2	2
3	3	3	2	3

δ	0	1	2	3
0	0	0	0	3
1	0	1	1	3
2	0	1	2	3
3	3	3	3	3

Tabela 1.21 – Conectivos de quatro valores [11].

n valores:

- São definidos “n” valores: 0,1,2, ... , n-1.
- São definidos “n” conectivos: ALFA, BETA, ... , OP_{n-1}.

	α	β	...	OP_{n-1}

α	0	1	...	n-1

Dominante	0	1	...	n-1
Secundário	1	2	...	0
...	...	...	...	...
Indiferente	n-1	0	...	n-2

0	0	0	...	0
1	0	1	...	1
...	...	...	...	...
n-1	0	1	...	n-1

Tabela 1.22 –
Hierarquia de dominâncias para os conectivos de "n" valores e o conectivo ALFA de "n" valores.

- BASE: B = (A – 1) mod n (20a)
- TOPO: B = (A + 1) mod n (20b)

α	**0 mod n**	**1 mod n**	**2 mod n**	**...**
0 mod n	0 mod n	0 mod n	0 mod n	...
1 mod n	0 mod n	1 mod n	1 mod n	...
2 mod n	0 mod n	1 mod n	2 mod n	...
...	...	...	...	...

Tabela 1.23 – Conectivos ALFA de "n" valores.

Uma função de 3 valores e "m" entradas pode ser representada pelo vetor (0,1,2) onde cada elemento do trio ordenado indica um valor possível para a saída da função. Para 4 valores, a representação é (01,2,3). Uma função representada por (0,1,1), por exemplo, indica que todas as ocorrências do valor "2" foram substituídas pelo valor "1". Uma função representada por (1,2,0), por exemplo, indica que a função original sofreu um deslocamento TOPO. Essa notação é útil para representar transformações de valores na tabela-verdade, mas não pode ser usada para representar transformações de posição.

Uma notação mais genérica é (D,S,I) para lógica ternária e (D,S,T,I) para quaternária, onde as letras significam "dominante", "secundário", "terciário" e "indiferente". Ela é útil quando a análise é feita sobre a escala de dominâncias de um determinado conectivo e não sobre os valores particulares; dessa forma, é possível expressar transformações genéricas. Por exemplo, a função (I,S,D) indica que a função original sofreu uma permuta entre os valores D e S, sejam eles quais forem.

1.5 Criação do sistema algébrico a partir do máximo

Se, para se definir o operador ALFA, ao invés do conectivo mínimo, tivesse sido utilizado o conectivo máximo, ter-se-ia outro sistema algébrico. Uma vez que o circuito que realiza a operação "mínimo" é o dual daquele que realiza a operação "máximo", pode-se dizer que o circuito que representa uma função utilizando conectivos derivados do máximo é o dual do circuito que representa a mesma função, porém utilizando conectivos derivados do mínimo. Pode ser que um projetista prefira implementar seu circuito empregando os conectivos derivados do máximo[19,20]. O sistema Lógico de Post, em sua formulação original, utiliza o conectivo máximo. Sendo assim, é importante apresentar os conectivos derivados do máximo. O tratamento dado a tais conectivos é análogo ao apresentado neste trabalho.

α’	**0**	**1**	**2**
0	0	1	2
1	1	1	2
2	2	2	2

β’	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	2

γ’	**0**	**1**	**2**
0	0	1	0
1	1	1	1
2	0	1	2

Tabela 1.24 – Sistema lógico derivado do máximo, para três valores.

α’	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

β’	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	2	3
3	0	3	3	3

γ’	**0**	**1**	**2**	**3**
0	0	1	0	0
1	1	1	1	1
2	0	1	2	3
3	0	1	3	3

δ’	**0**	**1**	**2**	**3**
0	0	1	2	0
1	1	1	2	1
2	2	2	2	2
3	0	1	2	3

Tabela 1.25 – Sistema lógico derivado do máximo, para quatro valores.

1.6 Síntese

Toda função lógica (booleana ou MVL) pode ser definida de duas maneiras:

1. Tabela verdade ou Mapa de Karnaugh[6,13,21]
2. Expressão algébrica

A tabela corresponde a uma lista (vertical ou horizontal) com todas as combinações de variáveis de entrada e seu respectivo valor na variável de saída. O mapa consiste em uma matriz cuja dimensão corresponde ao número de variáveis de entrada, cada posição da matriz é preenchida pelo valor da variável de saída. A definição por tabela verdade ou Mapa de Karnaugh permite uma compreensão rápida das características da função, dando uma noção intuitiva de seu comportamento. A definição por expressão algébrica permite a verificação das leis de formação empregadas na formação da função; também permite escrever a função em uma linha, como sendo uma fórmula. Esta forma de exibição é útil quando se pretende implementar a função em um dispositivo físico (circuito integrado, por exemplo), pois, a partir das operações fundamentais, pode-se obter todas as possibilidades de funções. A desvantagem deste modo de exibição consiste na dificuldade de percepção do comportamento da função e na minimização não intuitiva (não visual).

Ao processo de conversão da visualização por tabela verdade para a visualização por expressão algébrica dá-se o nome de síntese[11,15] pois a definição por expressão é mais sintética do que a por tabela. A síntese pode ou não utilizar técnicas de minimização. Minimização consiste em diversos artifícios que tornam o tamanho da expressão algébrica menor, sem garantir, contudo, que a expressão esteja na forma mínima absoluta. Somente métodos de exaustão e inspecionais podem fornecer essa garantia.

1.7 Propriedades da álgebra MVL

Algumas propriedades da álgebra MVL devem ser mostradas a fim de embasar as operações feitas nos demais capítulos deste trabalho.

1 Operação com o valor indiferente (elemento neutro): Uma variável (X) operada com o valor indiferente (I) do conectivo (op) fornece a própria variável. X op I = X[3]

Geral	Dois valores	Três valores	Quatro valores
$A \text{ op } I = A \quad (21a)$	$A\,\alpha\,1 = A \quad (21b)$	$A\,\alpha\,2 = A \quad (21f)$	$A\,\alpha\,3 = A \quad (21i)$
	$A\,\beta\,0 = A \quad (21c)$	$A\,\beta\,0 = A \quad (21g)$	$A\,\beta\,0 = A \quad (21j)$
	$A \cdot 1 = A \quad (21d)$	$A\,\gamma\,1 = A \quad (21h)$	$A\,\gamma\,1 = A \quad (21k)$
	$A + 0 = A \quad (21e)$		$A\,\delta\,2 = A \quad (21l)$

2 Operação com o valor dominante (elemento nulo): Uma variável (X) operada com o valor dominante (D) do conectivo (op) fornece valor o dominante. X op D = D[3]

Geral	Dois valores	Três valores	Quatro valores
$A \text{ op } D = D \quad (22a)$	$A\,\alpha\,0 = 0 \quad (22b)$	$A\,\alpha\,0 = 0 \quad (22f)$	$A\,\alpha\,0 = 0 \quad (22i)$
	$A\,\beta\,1 = 1 \quad (22c)$	$A\,\beta\,1 = 1 \quad (22g)$	$A\,\beta\,1 = 1 \quad (22j)$
	$A \cdot 0 = 0 \quad (22d)$	$A\,\gamma\,2 = 2 \quad (22h)$	$A\,\gamma\,2 = 2 \quad (22k)$
	$A + 1 = 1 \quad (22e)$		$A\,\delta\,3 = 3 \quad (22l)$

3 Obtenção do valor dominante: Uma variável operada com todas as possibilidades de deslocamento fornece o valor dominante do conectivo empregado[3].

Dois valores	Três valores	Quatro valores
$A\,\alpha\,\overline{A}=0 \quad (23a)$	$A\,\alpha\,\overline{A}\,\alpha\,\underline{A}=0 \quad (23e)$	$A\,\alpha\,\overline{A}\,\alpha\,\overline{\overline{A}}\,\alpha\,\underline{A}=0 \quad (23h)$
$A\,\beta\,\overline{A}=1 \quad (23b)$	$A\,\beta\,\overline{A}\,\beta\,\underline{A}=1 \quad (23f)$	$A\,\beta\,\overline{A}\,\beta\,\overline{\overline{A}}\,\beta\,\underline{A}=1 \quad (23i)$
$A\cdot\overline{A}=0 \quad (23c)$	$A\,\gamma\,\overline{A}\,\gamma\,\underline{A}=2 \quad (23g)$	$A\,\gamma\,\overline{A}\,\gamma\,\overline{\overline{A}}\,\gamma\,\underline{A}=2 \quad (23j)$
$A+\overline{A}=1 \quad (23d)$		$A\,\delta\,\overline{A}\,\delta\,\overline{\overline{A}}\,\delta\,\underline{A}=3 \quad (23k)$

4 Comutação: A op B = B op A (24)

O único conectivo para o qual a propriedade comutativa não é válida é o unicondicional, o que torna difícil a formalização de um conjunto de conectivos a partir dele.

5 Associação: A op (B op C) = (A op B) op C (25)

4 Distributividade crescente: Dado um conectivo (op1) e o conectivo (op2) tal que $\overline{A\,\mathrm{op}_1\,B}=\overline{A}\,\mathrm{op}_2\,\overline{B}$, tem-se a distributividade crescente; op1 é distributivo sobre op2[3,12].

Geral

$$A\,\mathrm{op1}\,(B\,\mathrm{op2}\,C)=(A\,\mathrm{op1}\,B)\,\mathrm{op2}\,(A\,\mathrm{op1}\,C) \quad (26a)$$

Dois valores

$$A\,\alpha\,(B\,\beta\,C)=(A\,\alpha\,B)\,\beta\,(A\,\alpha\,C) \quad (26b)$$
$$A\,\beta\,(B\,\alpha\,C)=(A\,\beta\,B)\,\alpha\,(A\,\beta\,C) \quad (26c)$$
$$A\cdot(B+C)=(A\cdot B)+(A\cdot C) \quad (26d)$$
$$A\cdot(B+C)=(A+B)\cdot(A+C) \quad (26e)$$

Três valores

$$A\,\alpha\,(B\,\beta\,C)=(A\,\alpha\,B)\,\beta\,(A\,\alpha\,C) \quad (26f)$$
$$A\,\beta\,(B\,\gamma\,C)=(A\,\beta\,B)\,\gamma\,(A\,\beta\,C) \quad (26g)$$
$$A\,\gamma\,(B\,\alpha\,C)=(A\,\gamma\,B)\,\alpha\,(A\,\gamma\,C) \quad (26h)$$

Quatro valores

$$A\,\alpha\,(B\,\beta\,C)=(A\,\alpha\,B)\,\beta\,(A\,\alpha\,C) \quad (26i)$$
$$A\,\beta\,(B\,\gamma\,C)=(A\,\beta\,B)\,\gamma\,(A\,\beta\,C) \quad (26j)$$
$$A\,\gamma\,(B\,\delta\,C)=(A\,\gamma\,B)\,\delta\,(A\,\gamma\,C) \quad (26k)$$
$$A\,\delta\,(B\,\alpha\,C)=(A\,\delta\,B)\,\alpha\,(A\,\delta\,C) \quad (26l)$$

5 Não distributividade decrescente: Dadas as mesmas definições anteriores, para três ou mais valores, tem-se: $A\,\mathrm{op2}\,(B\,\mathrm{op1}\,C)\neq(A\,\mathrm{op2}\,B)\,\mathrm{op1}\,(A\,\mathrm{op2}\,C)$ (27a)[3]

Esta propriedade não pode ser observada para base 2.

Três valores

$$A\,\beta\,(B\,\alpha\,C)\neq(A\,\beta\,B)\,\alpha\,(A\,\beta\,C) \quad (27b)$$
$$A\,\gamma\,(B\,\beta\,C)\neq(A\,\gamma\,B)\,\beta\,(A\,\gamma\,C) \quad (27c)$$
$$A\,\alpha\,(B\,\gamma\,C)\neq(A\,\alpha\,B)\,\gamma\,(A\,\alpha\,C) \quad (27d)$$

Quatro valores

$$A\,\beta\,(B\,\alpha\,C)\neq(A\,\beta\,B)\,\alpha\,(A\,\beta\,C) \quad (27e)$$
$$A\,\gamma\,(B\,\beta\,C)\neq(A\,\gamma\,B)\,\beta\,(A\,\gamma\,C) \quad (27f)$$
$$A\,\delta\,(B\,\gamma\,C)\neq(A\,\delta\,B)\,\gamma\,(A\,\delta\,C) \quad (27g)$$
$$A\,\alpha\,(B\,\delta\,C)\neq(A\,\alpha\,B)\,\delta\,(A\,\alpha\,C) \quad (27h)$$

6 Conversão para binário (CPB): Esta propriedade é derivada da distributividade crescente e da obtenção do dominante. Embora esta propriedade não tenha serventia para dois valores, as fórmulas (28a-d) foram incluídas a título de observação. Função binária, no campo da lógica MVL (função binária MVL) é aquela que fornece, apenas, dois valores consecutivos.

Dois valores

$$(A\alpha B)\beta(A\alpha\overline{B}) = A\alpha(B\beta\overline{B}) = A\alpha 1 \quad (28a)$$
$$(A\beta B)\alpha(A\beta\overline{B}) = A\beta(B\alpha\overline{B}) = A\beta 0 \quad (28b)$$
$$(A\cdot B)+(A\cdot\overline{B}) = A\cdot(B+\overline{B}) = A\cdot 1 \quad (28c)$$
$$(A+B)\cdot(A+\overline{B}) = A+(B\cdot\overline{B}) = A+0 \quad (28d)$$

Três valores

$$(A\alpha B)\beta(A\alpha\overline{B})\beta(A\alpha\underline{B}) = A\alpha(B\beta\overline{B}\beta\underline{B}) = A\alpha 1 \quad (28e)$$
$$(A\beta B)\gamma(A\beta\overline{B})\gamma(A\beta\underline{B}) = A\beta(B\gamma\overline{B}\gamma\underline{B}) = A\beta 2 \quad (28f)$$
$$(A\gamma B)\beta(A\gamma\overline{B})\beta(A\gamma\underline{B}) = A\gamma(B\alpha\overline{B}\alpha\underline{B}) = A\gamma 0 \quad (28g)$$

Quatro valores

$$(A\alpha B)\beta(A\alpha\overline{B})\beta(A\alpha\overline{\overline{B}})\beta(A\alpha\underline{B}) = A\alpha(B\beta\overline{B}\beta\overline{\overline{B}}\beta\underline{B}) = A\alpha 1 \quad (28h)$$
$$(A\beta B)\gamma(A\beta\overline{B})\gamma(A\beta\overline{\overline{B}})\gamma(A\beta\underline{B}) = A\beta(B\gamma\overline{B}\gamma\overline{\overline{B}}\gamma\underline{B}) = A\beta 2 \quad (28i)$$
$$(A\gamma B)\delta(A\gamma\overline{B})\delta(A\gamma\overline{\overline{B}})\delta(A\gamma\underline{B}) = A\gamma(B\delta\overline{B}\delta\overline{\overline{B}}\delta\underline{B}) = A\gamma 3 \quad (28j)$$
$$(A\delta B)\alpha(A\delta\overline{B})\alpha(A\delta\overline{\overline{B}})\alpha(A\delta\underline{B}) = A\delta(B\alpha\overline{B}\alpha\overline{\overline{B}}\alpha\underline{B}) = \delta\gamma 0 \quad (28k)$$

7 Colunas completas: Esta propriedade, utilizada juntamente com a conversão para binário, é importante para justificar a simplificação das colunas completas no processo de síntese. Pode ser aplicado, também, às linhas.

Dois valores

$$(A\alpha B)\beta(A\alpha\overline{B}) = A\alpha 1 = A \quad (29a)$$
$$(A\beta B)\alpha(A\beta\overline{B}) = A\beta 0 = A \quad (29b)$$
$$(A\cdot B)+(A\cdot\overline{B}) = A\cdot 1 = A \quad (29c)$$
$$(A+B)\cdot(A+\overline{B}) = A+0 = A \quad (29d)$$

Três valores

$$(A\beta B)\alpha(A\beta\overline{B})\alpha(A\beta\underline{B}) = (A\alpha B)\beta(A\alpha\overline{B})\beta(A\alpha\underline{B}) = A\alpha 1 \quad (29e)$$
$$(A\gamma B)\beta(A\gamma\overline{B})\beta(A\gamma\underline{B}) = (A\beta B)\gamma(A\beta\overline{B})\gamma(A\beta\underline{B}) = A\beta 2 \quad (29f)$$
$$(A\alpha B)\gamma(A\alpha\overline{B})\gamma(A\alpha\underline{B}) = (A\gamma B)\alpha(A\gamma\overline{B})\alpha(A\gamma\underline{B}) = A\gamma 0 \quad (29g)$$

Quatro valores

$$(A\beta B)\alpha(A\beta\overline{B})\alpha(A\beta\overline{\overline{B}})\alpha(A\beta\underline{B}) = (A\alpha B)\beta(A\alpha\overline{B})\beta(A\alpha\overline{\overline{B}})\beta(A\alpha\underline{B}) = A\alpha 1 \quad (29h)$$
$$(A\gamma B)\beta(A\gamma\overline{B})\beta(A\gamma\overline{\overline{B}})\beta(A\gamma\underline{B}) = (A\beta B)\gamma(A\beta\overline{B})\gamma(A\beta\overline{\overline{B}})\gamma(A\beta\underline{B}) = A\beta 2 \quad (29i)$$
$$(A\delta B)\gamma(A\delta\overline{B})\gamma(A\delta\overline{\overline{B}})\gamma(A\delta\underline{B}) = (A\gamma B)\delta(A\gamma\overline{B})\delta(A\gamma\overline{\overline{B}})\delta(A\gamma\underline{B}) = A\gamma 3 \quad (29j)$$
$$(A\alpha B)\delta(A\alpha\overline{B})\delta(A\alpha\overline{\overline{B}})\delta(A\alpha\underline{B}) = (A\delta B)\alpha(A\delta\overline{B})\alpha(A\delta\overline{\overline{B}})\alpha(A\delta\underline{B}) = A\delta 0 \quad (29k)$$

A progressão a seguir, para três valores, é um exemplo que mostra que a distributividade decrescente não é verdadeira. Essa progressão também mostra que a distributividade decrescente consiste em uma CPB. Em se tratando de funções binárias, a distributividade decrescente é válida, pode ser aplicada livremente.

Ponto de partida : A

$1 - A = A\beta 0$	Operação com o indiferente
$2 - A\beta 0 = A\beta(B\alpha\overline{B}\alpha\underline{B})$	Obtenção do dominante
$3 - A\beta(B\alpha\overline{B}\alpha\underline{B}) \neq (A\beta B)\alpha(A\beta\overline{B})\alpha(A\beta\underline{B})$	Não distributividade decrescente
$4 - (A\beta B)\alpha(A\beta\overline{B})\alpha(A\beta\underline{B}) = (A\alpha B)\beta(A\alpha\overline{B})\beta(A\alpha\underline{B})$	Colunas completas
$5 - (A\alpha B)\beta(A\alpha\overline{B})\beta(A\alpha\underline{B}) = A\alpha(B\beta\overline{B}\beta\underline{B})$	Conversão para binário
$6 - A\alpha(B\beta\overline{B}\beta\underline{B}) = A\alpha 1$	Obtenção do dominante

Conclusão : $A \neq A\alpha 1$

Há, ainda, outras propriedades, porém sem aplicação neste trabalho. Muitas outras propriedades poderão ser descobertas, ajudando no processo de síntese MVL.

1.8 Generalização das funções MVL

Neste trabalho, aborda-se funções do tipo **`B = B(A)`** ou **`C = C(A,B)`**.

Binário:	**A,B,C ∈ {0,1}**
Ternário:	**A,B,C ∈ {0,1,2}**
Quaternário:	**A,B,C ∈ {0,1,2,3}**

Essa definição assume que todas as variáveis estejam dentro do mesmo escopo de valores. No entanto, pode-se assumir que as variáveis tenham abrangências diferentes (embora estejam todas dentro da mesma base). Por exemplo, uma função quaternária de duas entradas (**A,B**) pode ser definida para **A** ∈ **{0,1,2}** e **B** ∈ **{0,1,2,3}**, a função não é definida para **A=3**. Outro exemplo é o de uma função quaternária de uma entrada que gere, apenas, os valores **{0,2,3}**; neste caso, **A** ∈ **{0,1,2,3}** e **B** ∈ **{0,2,3}**.

Um exemplo típico de abrangências diferentes é quando se deve tomar uma decisão booleanan (sim ou não) com base em dois fatos, sendo que um deles pode assumir três possibilidades e o outro quatro. Mais especificamente, este exemplo pode ser "ir à praia" (variável de saída de dois valores), "ser de manhã, de tarde ou de noite" (primeira variável de entrada, ternária), "ser primavera, verão, outono ou inverno" (segunda variável de entrada, quaternária). Para cada uma das doze combinações de entrada, uma decisão será tomada.

Variável C (Saída / Decisão)
0. Não ir à praia
1. Ir à praia

Variável A (Entrada / Condição)
0. De manhã
1. De tarde
2. De noite

Variável B (Entrada / Condição)
0. Primavera
1. Verão
2. Outono
3. Inverno

A tabela 1.26 o exemplo acima.

C	A=0	A=1	A=2
B=0	0	1	1
B=1	1	1	1
B=2	1	1	0
B=3	0	0	0

A ∈ **{0,1,2}**
B ∈ **{0,1,2,3}**
C ∈ **{0,1}**

Tabela 1.26 – Exemplo 1.

C	A=0	A=1	A=2	A=3
B=0	0	0	2	2
B=1	0	1	1	1

A ∈ **{0,1,2,3}**
B ∈ **{0,1}**
C ∈ **{0,1,2}**

Tabela 1.27 – Exemplo 2.

C	A=0	A=1
B=0	0	1
B=1	2	3

A ∈ **{0,1}**
B ∈ **{0,1}**
C ∈ **{0,1,2,3}**

Tabela 1.28 – Exemplo 3.

C	A=0	A=1	A=2	A=3
B=0	0	0	0	0
B=1	0	1	1	1
B=2	0	1	0	0
B=3	0	1	0	1

A ∈ **{0,1,2,3}**
B ∈ **{0,1,2,3}**
C ∈ **{0,1}**

Tabela 1.29 – Exemplo 4.

Um tipo interessante de função MVL é aquela que fornece apenas dois valores consecutivos, tal como os exemplos 1 e 4 acima. Tais funções recebem o nome particular de “binárias-MVL” e são abordadas no capítulo 2. Sua utilidade é apresentada no capítulo 3.

Capítulo 2
Funções Binárias-MVL

2.1 Tipos de função MVL

As funções MVL, quanto à quantidade de valores que elas cobrem, podem ser classificadas nos tipos abaixo. É apresentado um exemplo para cada tipo (tabelas 2.1 e 2.2).

$$\text{lógica binária}\begin{cases}\text{1 - um valor}\\ \text{2 - dois valores}\end{cases}\qquad \text{lógica ternária}\begin{cases}\text{1 - um valor}\\ \text{2 - dois valores}\\ \text{3 - três valores}\end{cases}$$

$$\text{lógica quaternária}\begin{cases}\text{1 - um valor}\\ \text{dois valores}\begin{cases}\text{2 - dois valores consecutivos}\\ \text{3 - dois valores não consecutivos}\end{cases}\\ \text{4 - três valores}\\ \text{5 - quatro valores}\end{cases}$$

Exemplo 1

0	0	0
0	0	0
0	0	0

Exemplo 2

0	0	0
0	1	1
0	1	1

Exemplo 3

0	0	0
0	1	1
0	1	2

Tabela 2.1 – Exemplos dos três tipos de funções de três valores.

Exemplo 1

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Exemplo 2

0	0	0	0
0	1	1	1
0	1	1	1
0	1	1	1

Exemplo 3

0	0	0	0
0	0	0	0
0	0	2	2
0	0	2	2

Exemplo 4

0	0	0	0
0	1	1	1
0	1	2	2
0	1	2	2

Exemplo 5

0	0	0	0
0	1	1	1
0	1	2	2
0	1	2	3

Tabela 2.2 – Exemplos dos cinco tipos de funções de quatro valores.

Funções que fornecem um valor não requerem nenhuma técnica de síntese, são constantes. Sendo assim, as funções MVL mais simples são aquelas que fornecem apenas dois valores consecutivos. Essas funções são ditas <u>binárias-MVL</u>.

É importante não confundir uma <u>função binária-MVL</u> com uma <u>função booleana</u>.

Uma função binária-MVL fornece dois valores vizinhos, sendo que o segundo valor é o deslocamento TOPO do primeiro, mas a recíproca não é verdadeira, pois o primeiro valor não é o deslocamento TOPO do segundo (exceto para funções de dois valores); o primeiro valor é o deslocamento BASE do segundo (para dois valores, o deslocamento BASE é o deslocamento TOPO). Neste trabalho, sempre que se falar em “função binária-MVL”, está se tratando de uma função cujas variáveis estejam no domínio MVL, mas que a variável de saída somente assuma dois valores vizinhos. Funções pertencentes à lógica de Boole são referenciadas como “booleanas”.

Na lógica booleana, pode-se analisar o valor “0” – lógica negativa, na qual “0” é o valor dominante – e o valor “1” – lógica positiva, na qual “1” é o valor dominante. Em funções binárias-MVL, somente um valor pode ser considerado verdadeiro, não há duas opções. O valor “falso” é determinado pelo deslocamento TOPO do valor verdadeiro.

Na lógica de Boole (funções booleanas) e na álgebra de Post (funções binárias-MVL), a expressão é constituída por termos[3]. A palavra “termo” refere-se a uma pequena expressão (célula) que representa uma combinação das variáveis de entrada de uma função para a qual a saída corresponde ao valor verdadeiro (dominante). Para facilitar a compreensão futura, a função completa recebe, como nome, uma letra minúscula (“b” para uma entrada e “c” para duas entradas) seguida por um número que indica qual o valor verdadeiro[11,15].

2.2 Endereçamento de colunas para uma entrada

Para facilitar a implementação computacional, a grafia usada nas expressões MVL é convertida em códigos ASCII. As barras “TOPO” e “BASE” são substituídas pelo símbolo “/” antes e após, respectivamente. As letras “α”, “β”, “γ” e “δ” são substituídas por “a”, “b”, “c” e “d”, respectivamente. Essas letras minúsculas, seguidas de um espaço, representam os conectivos, e, quando seguidas por um número, representam funções binárias-MVL, tal como explicado no parágrafo anterior. Em funções de uma entrada (`B(A)`) há apenas uma linha e tantas colunas quanto for o tamanho da base adotada (tabela 2.3). Representa-se a combinação de entrada por meio de expressões que gerem o valor analisado (tabela 2.4). No entanto, analisando-se cada termo separadamente, percebe-se que binária-MVL. Para corrigir esse erro, adiciona-se, à expressão, uma conversão para binário-MVL (CPB) [11,13,15], que converte os valores terciário e indiferente no valor secundário, tornando a função binária-MVL. (tabela 2.6). No caso de funções booleanas, esse cuidado não é necessário por motivos óbvios.

A – entrada	0	1
B(A) – saída	B(A=0)	B(A=1)

A – entrada	0	1	2
B(A) – saída	B(A=0)	B(A=1)	B(A=2)

A – entrada	0	1	2	3
B(A) – saída	B(A=0)	B(A=1)	B(A=2)	B(A=3)

Tabela 2.3 – Funções de uma entrada (dois, três e quatro valores).

A	0	1
B(A) = 0	A	/A
B(A) = 1	/A	A

A	0	1	2
B(A) = 0	A	A/	/A
B(A) = 1	/A	A	A/
B(A) = 2	A/	/A	A

A	0	1	2	3
B(A) = 0	A	A/	//A	/A
B(A) = 1	/A	A	A/	//A
B(A) = 2	//A	/A	A	A/
B(A) = 3	A/	//A	/A	A

Tabela 2.4 – Representação parcial dos termos (dois, três e quatro valores).

Tipo	Dominante Verdadeiro	Secundário Falso
b0	0	1
b1	1	0

Tipo	Dominante Verdadeiro	Secundário Falso
b0	0	1
b1	1	2
b2	2	0

Tipo	Dominante Verdadeiro	Secundário Falso
b0	0	1
b1	1	2
b2	2	3
b3	3	0

Tabela 2.5 – Especificação dos valores verdadeiro e falso (dois, três e quatro valores).

Tipo	A = 0	A = 1
b0	A a 1	/A a 1
b1	/A b 0	A b 1

Tipo	A = 0	A = 1	A = 2
b0	A a 1	A/ a 1	/A a 1
b1	/A b 2	A b 2	A/ b 2
b2	A/ c 0	/A c 0	A c 0

Tipo	A = 0	A = 1	A = 2	A = 3
b0	A a 1	A/ a 1	//A a 1	/A a 1
b1	/A b 2	A b 2	A/ b 2	//A b 2
b2	//A c 3	/A c 3	A c 3	A/ c 3
b3	A/ d 0	//A d 0	/A d 0	A d 0

Tabela 2.6 – Representação completa dos termos (dois, três e quatro valores).

Uma vez determinada qual expressão corresponde a cada termo, é preciso ligá-los, formando a expressão binária-MVL completa. Escolhe-se um operador, chamado de operador entre termos. Essa escolha deve ser feita segundo o critério da existência.

Existência: Uma função digital existe se ela for verdadeira para, pelo menos, uma combinação de entrada. A expressão binária-MVL completa, formada pela junção das expressões relativas a cada termo (ou célula), deve ser verdadeira para todos os casos em que uma das funções "binárias-MVL" parciais for verdadeira.

Para que isso aconteça, o conectivo utilizado entre os termos deve ser aquele para o qual o valor analisado seja o dominante, pois, dessa maneira, toda vez que um dos termos for verdadeiro, a função completa é verdadeira.

Tipo	Dominante Verdadeiro	Secundário Falso	Operador entre Termos
b0	0	1	a
b1	1	2	b
b2	2	0	c

Tipo	Dominante Verdadeiro	Secundário Falso	Operador entre Termos
b0	0	1	a
b1	1	2	b
b2	2	3	c
b3	3	0	d

Tabela 2.7 – Operador entre termos (três e quatro valores).

Para uma determinada combinação de entrada verdadeira, somente um termo é verdadeiro; todos os demais são falsos, mas como o termo verdadeiro fornece o valor dominante, então a expressão inteira se torna verdadeira. A expressão somente se torna falsa se nenhum dos termos for verdadeiro, o que corresponde a uma combinação falsa.

Ao se juntar dois ou mais termos, a CPB não precisa ser repetida. Basta usar uma única CPB para tornar a expressão toda binária-MVL, não importando quantos sejam os termos usados. Nem sempre a CPB é necessária. No capítulo 3, o critério para saber se a CPB é necessária ou não é abordado, mas, por hora, assume-se que ela seja sempre necessária. A união de todos os termos possíveis, gerando o valor dominante, é mostrada na tabela 2.8. As funções binárias-MVL de uma entrada são mostradas na tabela 2.9. Por convenção, a CPB é colocada no final da expressão, mas pode ficar em qualquer lugar.

`b0 = A a /A a 1`
`b1 = /A b A b 0`

`b0 = A a A/ a /A a 1`
`b1 = /A b A b A/ b 2`
`b2 = A/ c /A c A c 0`

`b0 = A a A/ a //A a /A a 1`
`b1 = /A b A b A/ b //A b 2`
`b2 = //A c /A c A c A/ c 3`
`b3 = A/ d //A d /A d A d 0`

Tabela 2.8 – União dos termos (dois, três e quatro valores).

Sempre que possível, neste trabalho, os exemplos são mostrados para b0, a fim de facilitar comparações entre os exemplos dados. Além disso, essa função binária-MVL utiliza os valores “0” e “1”, mais parecida com as funções booleanas, mais familiares. As expressões para b0 são mostradas na tabela 2.10. Deve-se ter em mente que todos os conceitos apresentados para b0 valem, também, para b1, b2, etc.

Genérico	b0	b1
V	0	1
F	1	0
0 **VV**	**00**	**11**
1 **VF**	**01**	**10**
2 **FV**	**10**	**01**

Genérico	b0	b1	b2
V	0	1	2
F	1	2	0
0 **VVV**	**000**	**111**	**222**
1 **VVF**	**001**	**112**	**220**
2 **VFV**	**010**	**121**	**202**
3 **VFF**	**011**	**122**	**200**
4 **FVV**	**100**	**211**	**022**
5 **FVF**	**101**	**212**	**020**
6 **FFV**	**110**	**221**	**002**

Genérico	b0	b1	b2	b3
V	0	1	2	3
F	1	2	3	0
0 **VVVV**	**0000**	**1111**	**2222**	**3333**
1 **VVVF**	**0001**	**1112**	**2223**	**3330**
2 **VVFV**	**0010**	**1121**	**2232**	**3303**
3 **VVFF**	**0011**	**1122**	**2233**	**3300**
4 **VFVV**	**0100**	**1211**	**2322**	**3033**
5 **VFVF**	**0101**	**1212**	**2323**	**3030**
6 **VFFV**	**0110**	**1221**	**2332**	**3003**
7 **VFFF**	**0111**	**1222**	**2333**	**3000**
8 **FVVV**	**1000**	**2111**	**3222**	**0333**
9 **FVVF**	**1001**	**2112**	**3223**	**0330**
10 **FVFV**	**1010**	**2121**	**3232**	**0303**
11 **FVFF**	**1011**	**2122**	**3233**	**0300**
12 **FFVV**	**1100**	**2211**	**3322**	**0033**
13 **FFVF**	**1101**	**2212**	**3323**	**0030**
14 **FFFV**	**1110**	**2221**	**3332**	**0003**

Tabela 2.9 – Funções binárias-MVL de uma entrada (dois, três e quatro valores).

Há 3·3·3 = 27 funções de três valores e uma entrada e 4·4·4 = 64 de quatro valores. A tabela 2.9 apresenta 21 funções para três valores e 60 para quatro valores, as binárias MVL. As funções que faltam são aquelas que não se enquadram na definição de função binária MVL. As funções FF, FFF e FFFF (dois, três e quatro valores, respectivamente) não foram apresentadas porque, se elas nunca são verdadeiras, elas não existem.

Vetor	Expressão
00	**c0 = A a /A** a 1
01	**c0 = A** a 1
10	**c0 = /A** a 1

Vetor	Expressão
000	**c0 = A a A/ a /A** a 1
001	**c0 = A a A/** a 1
010	**c0 = A a /A** a 1
011	**c0 = A** a 1
100	**c0 = A/ a /A** a 1
101	**c0 = A/** a 1
110	**c0 = /A** a 1

Vetor	Expressão
0000	**c0 = A a A/ a //A a /A** a 1
0001	**c0 = A a A/ a //A** a 1
0010	**c0 = A a A/ a /A** a 1
0011	**c0 = A a A/** a 1
0100	**c0 = A a //A a /A** a 1
0101	**c0 = A a //A** a 1
0110	**c0 = A a /A** a 1
0111	**c0 = A** a 1
1000	**c0 = A/ a //A a /A** a 1
1001	**c0 = A/ a //A** a 1
1010	**c0 = A/ a /A** a 1
1011	**c0 = A/** a 1
1100	**c0 = //A a /A** a 1
1101	**c0 = //A** a 1
1110	**c0 = /A** a 1

Tabela 2.10 – Expressões para c0 (dois, três e quatro valores).

2.3 Endereçamento de linhas para duas entradas

Em funções MVL de duas entradas (**C(A,B)**) há tantas linhas e tantas colunas quanto for o tamanho da base adotada. Uma vez que, no caso de funções de uma entrada, as colunas já foram atribuídas à variável "A", para funções de duas entradas, as linhas são atribuídas à variável “B”. A técnica para endereçamento de linhas é igual à de colunas. No caso de funções que só dependam da variável “B”, ou seja, (`C(B)`),“A = constante”, pode-se fazer a mesma dedução feita para as colunas. As funções da tabela 2.12 são aquelas que retornam um valor verdadeiro para qualquer entrada, e, nesse caso, c0 = 0, c1 = 1, etc.

B	C(B)
0	C(B=0)
1	C(B=1)

B	C(B)
0	C(B=0)
1	C(B=1)
2	C(B=2)

B	C(B)
0	C(B=0)
1	C(B=1)
2	C(B=2)
3	C(B=3)

Tabela 2.11 – Funções de uma entrada (dois, três e quatro valores).

c0 = B a /B a 1
c1 = /B b B b 0

c0 = B a B/ a /B a 1
c1 = /B b B b B/ b 2
c2 = B/ c /B c B c 0

c0 = B a B/ a //B a /B a 1
c1 = /B b B b B/ b //B b 2
c2 = //B c /B c B c B/ c 3
c3 = B/ d //B d /B d B d 0

Tabela 2.12 – União dos termos (dois, três e quatro valores).

2.4 Representação de termos para duas entradas

Até agora, foi abordado somente o caso da existência de, apenas, uma variável relevante. Já foi analisado e determinado qual é o operador a ser utilizado entre os termos, mas ainda é preciso determinar qual o operador a ser utilizado dentro dos termos, operador que une as coordenadas, a parte referente às colunas (variável “A”) e a parte referente às linhas (variável “B”), chamado de operador entre coordenadas. Escolhendo-se corretamente o conectivo, obtém-se uma expressão que endereça biunivocamente a posição da célula na tabela (ou matriz).

Na lógica booleana, a disponibilidade dos operadores OR e AND juntos permite que se expresse a função na forma disjuntiva e conjuntiva.

Forma disjuntiva	Forma conjuntiva
analisa-se os zeros	analisa-se os uns
V=0, F=1	V=1, F=0
produtos de somas	soma de produtos
produtório de maxtermos	somatório de mintermos
Π max	Σ min

Tabela 2.13 – Lógica negativa e positiva.

$$0 = (A+B)*(\overline{A}+B)*(A+\overline{B})*(\overline{A}+\overline{B}) \quad (1a)$$
$$1 = (\overline{A}*\overline{B})+(A*\overline{B})+(\overline{A}*B)+(A*B) \quad (1b)$$

As tabelas 2.14 a 16 mostram como a união deve ser feita, para c0. A escolha do operador entre coordenadas deve seguir o critério da unicidade, explicado a seguir.

c0(A)	A = 0	A = 1
B = 0	`A a 1`	`A/ a 1`
B = 1	`A a 1`	`A/ a 1`

c0(B)	A = 0	A = 1
B = 0	`B a 1`	`B a 1`
B = 1	`B/ a 1`	`B/ a 1`

Tabela 2.14 – Termos isolados para dois valores (colunas e linhas).

c0(A)	A = 0	A = 1	A = 2
B = 0	`A a 1`	`A/ a 1`	`/A a 1`
B = 1	`A a 1`	`A/ a 1`	`/A a 1`
B = 2	`A a 1`	`A/ a 1`	`/A a 1`

c0(B)	A = 0	A = 1	A = 2
B = 0	`B a 1`	`B a 1`	`B a 1`
B = 1	`B/ a 1`	`B/ a 1`	`B/ a 1`
B = 2	`/B a 1`	`/B a 1`	`/B a 1`

Tabela 2.15 – Termos isolados para três valores (colunas e linhas).

c0(A)	A = 0	A = 1	A = 2	A = 3
B = 0	`A a 1`	`A/ a 1`	`//A a 1`	`/A a 1`
B = 1	`A a 1`	`A/ a 1`	`//A a 1`	`/A a 1`
B = 2	`A a 1`	`A/ a 1`	`//A a 1`	`/A a 1`
B = 3	`A a 1`	`A/ a 1`	`//A a 1`	`/A a 1`

c0(B)	A = 0	A = 1	A = 2	A = 3
B = 0	`B a 1`	`B a 1`	`B a 1`	`B a 1`
B = 1	`B/ a 1`	`B/ a 1`	`B/ a 1`	`B/ a 1`
B = 2	`//B a 1`	`//B a 1`	`//B a 1`	`//B a 1`
B = 3	`/B a 1`	`/B a 1`	`/B a 1`	`/B a 1`

Tabela 2.16 – Termos isolados para quatro valores (colunas e linhas).

Unicidade: Cada termo (ocorrência do valor dominante ou verdadeiro) da tabela binária-MVL corresponde a uma interseção de uma coluna (variável "A") e uma linha (variável "B") e deve corresponder a uma expressão que seja verdadeira (valor dominante). Para que isso aconteça, o conectivo utilizado entre as coordenadas deve ser aquele para o qual o valor analisado seja o indiferente (elemento neutro)[3], pois, dessa maneira, o termo somente é verdadeiro para uma única combinação de valores de entrada, aquela que corresponde aos dois lados da expressão iguais ao valor indiferente. Para todas as demais combinações, o termo é falso. Para cada termo deve haver uma única combinação de entrada para a qual ele seja verdadeiro, e cada combinação de entrada deve ser representada por um único termo (relação biunívoca). Os operadores são mostrados nas tabelas 2.17-19.

Para a coordenada (A=0, B=0) de c0, tem-se a coluna A=0 e a linha B=0, representadas, respectivamente, por `A a 1` e `B a 1`. Interseccionando-se a coluna com a linha, obtém-se `(A a 1) b (B a 1)`. Por meio da propriedade da distributividade crescente, a CPB pode ser agrupada, formando: `(A b B) a 1`; este é o agrupamento entre coordenadas. Unindo, então, as coordenadas, obtém-se as tabelas 2.20 e 2.21. Toda expressão binária-MVL de duas entradas utiliza dois conectivos, que são:

- Operação entre coordenadas, determinada pelo conceito da unicidade.
- Operação entre termos, determinada pelo conceito da existência.

Expressões de uma entrada utilizam apenas o conectivo da operação entre termos.

Tipo	Dominante Verdadeiro	Secundário Falso	Operador entre coordenadas	Operador entre termos
c0	0	1	b	a
c1	1	0	a	b

Tabela 2.17 – Dois valores.

Tipo	Dominante Verdadeiro	Secundário Falso	Operador entre coordenadas	Operador entre termos
c0	0	1	b	a
c1	1	2	c	b
c2	2	0	a	c

Tabela 2.18 – Três valores.

Tipo	Dominante Verdadeiro	Secundário Falso	Operador entre coordenadas	Operador entre termos
c0	0	1	b	a
c1	1	2	c	b
c2	2	3	d	c
c3	3	0	a	d

Tabela 2.19 – Quatro valores.

c0(A,B)	A = 0	A = 1	A = 2
B = 0	`(A b B) a 1`	`(A/ b B) a 1`	`(/A b B) a 1`
B = 1	`(A b B/) a 1`	`(A/ b B/) a 1`	`(/A b B/) a 1`
B = 2	`(A b /B) a 1`	`(A/ b /B) a 1`	`(/A b /B) a 1`

c1(A,B)	A = 0	A = 1	A = 2
B = 0	`(/A c /B) b 2`	`(A c /B) b 2`	`(A/ c /B) b 2`
B = 1	`(/A c B) b 2`	`(A c B) b 2`	`(A/ c B) b 2`
B = 2	`(/A c B/) b 2`	`(A c B/) b 2`	`(A/ c B/) b 2`

c2(A,B)	A = 0	A = 1	A = 2
B = 0	`(A/ a B/) c 0`	`(/A a B/) c 0`	`(A a B/) c 0`
B = 1	`(A/ a /B) c 0`	`(/A a /B) c 0`	`(A a /B) c 0`
B = 2	`(A/ a B) c 0`	`(/A a B) c 0`	`(A a B) c 0`

Tabela 2.20 – Determinação dos termos para três valores.

Para a coordenada (0,0) e (1,0) de c0, tem-se: **`((A b B) a 1) a ((A/ b B) a 1)`**. A CPB pode ser agrupada, gerando **`(A b B) a (A/ b B) a 1`**. Este é o agrupamento entre termos. Há, então, dois níveis de agrupamento da CPB, entre coordenadas e entre termos.

0 = (A b B) a (A/ b B) a (/A b B) a (A b B/) a (A/ b B/) a (/A b B/) a (A b /B) a (A/ b /B) a (/A b /B) a 1	1 = (/A c /B) b (A c /B) b (A/ c /B) b (/A c B) b (A c B) b (A/ c B) b (/A c B/) b (A c B/) b (A/ c B/) b 2	2 = (A/ a B/) c (/A a B/) c (A a B/) c (A/ a /B) c (/A a /B) c (A a /B) c (A/ a B) c (/A a B) c (A a B) c 0

Tabela 2.21 – Todos os termos, formando 0, 1 e 2.

c0(A,B)	A = 0	A = 1	A = 2	A = 3
B = 0	(A b B) a 1	(A/ b B) a 1	(//A b B) a 1	(/A b B) a 1
B = 1	(A b B/) a 1	(A/ b B/) a 1	(//A b B/) a 1	(/A b B/) a 1
B = 2	(A b //B) a 1	(A/ b //B) a 1	(//A b //B) a 1	(/A b //B) a 1
B = 3	(A b /B) a 1	(A/ b /B) a 1	(//A b /B) a 1	(/A b /B) a 1

c1(A,B)	A = 0	A = 1	A = 2	A = 3
B = 0	(/A c /B) b 2	(A c /B) b 2	(A/ c /B) b 2	(//A c /B) b 2
B = 1	(/A c B) b 2	(A c B) b 2	(A/ c B) b 2	(//A c B) b 2
B = 2	(/A c B/) b 2	(A c B/) b 2	(A/ c B/) b 2	(//A c B/) b 2
B = 3	(/A c //B) b 2	(A c //B) b 2	(A/ c //B) b 2	(//A c //B) b 2

c2(A,B)	A = 0	A = 1	A = 2	A = 3
B = 0	(//A d //B) c 3	(/A d //B) c 3	(A d //B) c 3	(A/ d //B) c 3
B = 1	(//A d /B) c 3	(/A d /B) c 3	(A d /B) c 3	(A/ d /B) c 3
B = 2	(//A d B) c 3	(/A d B) c 3	(A d B) c 3	(A/ d B) c 3
B = 3	(//A d B/) c 3	(/A d B/) c 3	(A d B/) c 3	(A/ d B/) c 3

c3(A,B)	A = 0	A = 1	A = 2	A = 3
B = 0	(A/ a B/) d 0	(//A a B/) d 0	(/A a B/) d 0	(A a B/) d 0
B = 1	(A/ a //B) d 0	(//A a //B) d 0	(/A a //B) d 0	(A a //B) d 0
B = 2	(A/ a /B) d 0	(//A a /B) d 0	(/A a /B) d 0	(A a /B) d 0
B = 3	(A/ a B) d 0	(//A a B) d 0	(/A a B) d 0	(A a B) d 0

Tabela 2.22 – Determinação dos termos para quatro valores.

A célula com valor verdadeiro e sua respectiva expressão algébrica apresentam uma relação biunívoca, ou seja, a única posição da tabela que torna a expressão verdadeira é aquela endereçada pela expressão e, quando uma célula assume valor verdadeiro, a única expressão que se torna verdadeira é aquela que endereça tal célula (unicidade). Em outras palavras, a célula é verdadeira se e somente se a expressão que a endereça for verdadeira.

2.5 Representação de colunas e linhas completas

No capítulo 1, são apresentadas as fórmulas das colunas completas. Essas fórmulas são reproduzidas a seguir, relativas à variável "A". A mesma verificação pode ser feita mantendo-se a variável "B" constante. Partindo-se, por exemplo, da equação (3), relativa à coluna A=0 na forma c0, chega-se à forma compacta para linhas e colunas completas (equação (5)). A demonstração das fórmulas (3,4) não foi incluída porque elas ainda não foram obtidas; tem-se, por enquanto, apenas a verificação inspecional.

Três valores

`(A b B) a (A b B/) a (A b /B) = (A a B) b (A a B/) b (A a /B)` (3a)
`(A c B/) b (A c /B) b (A c B) = (A b B/) c (A b /B) c (A b B)` (3b)
`(A a /B) c (A a B) c (A a B/) = (A c /B) a (A c B) a (A c B/)` (3c)

Quatro valores

`(A b B) a (A b B/) a (A b //B) a (A b /B) = (A a B) b (A a B/) b (A a //B) b (A a /B)` (4a)
`(A c B/) b (A c //B) b (A c /B) b (A c B) = (A b B/) c (A b //B) c (A b /B) c (A b B)` (4b)
`(A d //B) c (A d /B) c (A d B) c (A d B/) = (A c //B) d (A c /B) d (A c B) d (A c B/)` (4c)
`(A a /B) d (A a B) d (A a B/) d (A a //B) = (A d /B) a (A d B) a (A d B/) a (A d //B)` (4d)

`c0 = (A b B) a (A b B/) a (A b B/)`
`c0 = (A a B) b (A a B/) b (A a B/)` Colunas completas (29a cap1)
`c0 = A a (B b B/ b /B)` Distributividade crescente (26d cap1)
`c0 = A a 1` (5) Obtenção do cominante (29a cap1)

A variável “B” não aparece na expressão final; isso mostra que uma linha ou coluna completa pode ser tratada como uma função de uma variável. Ficou, também, mostrado que, na expressão na forma expandida, `c0 = (A a B) b (A a B/) b (A a B/)`, está implícita a CPB, que ficou explícita na expressão na forma compacta, `c0 = A a 1`, ou seja, a primeira forma não requer CPB. Sempre que uma expressão binária-MVL dispensa CPB, é porque, dentro dela, está embutida a CPB. A seguir, são apresentados alguns exemplos. Por convenção, as linhas ou colunas completas são escritas após os termos isolados e a CPB no final.

Exemplo 1: `c0 = A a B a 1`
Exemplo 2: `c0 = (A/ b B) a (/A b B) a (A b B/) a (A b /B) a (/A b /B) a 1`
Exemplo 3: `c0 = (A/ b B) a (A b B/) a /A a /B a 1`
Exemplo 4: `c0 = (A/ b B/) a (/A b //B) a A a B a /B a 1`

0	0	0
0	1	1
0	1	1

Tabela 2.23 – Exemplo 1.

1	0	0
0	1	1
0	1	0

Tabela 2.24 – Exemplo 2.

1	0	0
0	1	0
0	0	0

Tabela 2.25 – Exemplo 3.

0	0	0	0
0	0	1	1
0	1	1	0
0	0	0	0

Tabela 2.26 – Exemplo 4.

A simplificação percentual por linhas e colunas completas é determinada pela proporção de termos que foram englobados por uma linha e/ou coluna completa.

Exemplo	Simplificação	Simplificação Percentual (%)
1	5/5	100
2	0/5	0
3	3/7	42,86
4	10/12	83,33

Tabela 2.27 – Exemplo 5.

2.6 Conveniência da conversão para binário-MVL (CPB)

Em se tratando de funções binárias-MVL, nem sempre a CPB é necessária[3]. É conveniente saber se a CPB pode ser removida a fim de diminuir o tamanho da expressão. Se a expressão requer a CPB, sua remoção acarreta em erro, faz uma alteração na função, que deixa de ser binária-MVL. Se a expressão não requer a CPB, a CPB está implícita na expressão, tal como foi exemplificado anteriormente, e a colocação da CPB explícita implica em uma redundância. Foram examinados e estabelecidos os casos que determinam a necessidade da CPB. Para funções de uma entrada, a análise é mais simples.

Três valores

- Condição 1 – Quando houver apenas um termo, a CPB é necessária.
- Condição 2 – Quando houver pelo menos dois termos, a CPB não é necessária.

Quatro valores

- Condição 1 – Quando houver apenas um termo a CPB é necessária.
- Condição 2 – Quando houver apenas dois termos vizinhos a CPB é necessária.
- Condição 3 – Quando houver pelo menos dois termos intercalados a CPB não é necessária.

b0 = A a 1

0	1	1

b0 = A a A/

0	0	1

Tabela 2.28 – Exemplos das condições 1 e 2, três valores.

b0 = A a 1

0	1	1	1

b0 = A a A/ a 1

0	0	1	1

b0 = A a //A

0	1	0	1

Tabela 2.29 – Exemplos das condições 1,2 e 3, quatro valores.

A tabela 2.10 passa a ser a tabela 2.32. Na lógica ternária, 57% das funções de uma entrada dispensam a CPB Na quaternária, 47%. Isso sugere que, quanto maior é a base, menor é o percentual de funções de uma entrada que dispensam a CPB.

Termos	Expressão
0	**1**
1	**A a 1**
2	**A a A/**
3	**A a A/ a /A**

Tabela 2.30 – Expressões com termos a partir da posição A=0, três valores.

Termos	Expressão
0	**1**
1	**A a 1**
2	**A a A/ a 1**
3	**A a A/ a //A**
4	**A a A/ a //A a /A**

Tabela 2.31 – Expressões com termos a partir da posição A=0, quatro valores.

Vetor	Expressão
000	c0 = A a A/ a /A
001	c0 = A a A/
010	c0 = A a /A
011	c0 = A a 1
100	c0 = A/ a /A
101	c0 = A/ a 1
110	c0 = /A a 1

Vetor	Expressão
0000	c0 = A a A/ a //A a /A
0001	c0 = A a A/ a //A
0010	c0 = A a A/ a /A
0011	c0 = A a A/ a 1
0100	c0 = A a //A a /A
0101	c0 = A a //A
0110	c0 = A a /A a 1
0111	c0 = A a 1
1000	c0 = A/ a //A a /A
1001	c0 = A/ a //A a 1
1010	c0 = A/ a /A
1011	c0 = A/ a 1
1100	c0 = //A a /A a 1
1101	c0 = //A a 1
1110	c0 = /A a 1

Tabela 2.32 – Expressões para c0 (três e quatro valores).

2.7 CPB para funções de duas entradas

Para duas entradas, as condições de uso da CPB são mais detalhadas.

2.7.1 – 3 valores:

1. Quando houver apenas três termos ou menos alinhados a CPB é necessária.
2. Quando houver pelo menos três termos alinhados e eles não forem agrupados em uma coluna ou linha completa, a CPB não é necessária (ver item 2.5).
3. Quando houver pelo menos dois termos em diagonal "\", a CPB não é necessária.
4. Quando houver apenas dois termos em diagonal "/", a CPB é necessária.
5. Quando houver pelo menos três termos que não estejam todos na mesma coluna ou linha a CPB não é necessária.

c0 = (A b B) a 1

0	1	1
1	1	1
1	1	1

c0 = (A b B) a (A/ b B) a 1

0	0	1
1	1	1
1	1	1

c0 = B a 1

0	0	0
1	1	1
1	1	1

Tabela 2.33 – Exemplos da condição 1.

c0 = (A b B) a (A/ b B) a (/A b B)

0	0	0
1	1	1
1	1	1

Tabela 2.34 – Exemplo da condição 2.

c0 = (A b B) a (A/ b B/)

0	1	1
1	0	1
1	1	1

Tabela 2.35 – Exemplo da condição 3.

c0 = (A/ b B) a (A b B/) a 1

1	0	1
0	1	1
1	1	1

Tabela 2.36 – Exemplo da condição 4.

c0 = (A b B) a (A/ b B) a (A b B/)
c0 = (A/ b B) a (/A b B) a (A b B/)
c0 = (A/ b B) a (A b B/) a (/A b /B)

0	0	1
0	1	1
1	1	1

1	0	0
0	1	1
1	1	1

1	0	1
0	1	1
1	1	0

Tabela 2.37 – Exemplos da condição 5.

Uma exceção à condição 5 é mostrada abaixo.

c0 = = (A/ b B) a (A b B/) a (A/ b B/) a 1

1	0	1
0	0	1
1	1	1

Tabela 2.38 – Contra-exemplo da condição 5

(A/ b B)

2	0	1
1	1	1
2	2	1

a

(A b B/)

2	1	2
0	1	2
1	1	1

a

(A/ b B/)

2	2	1
2	0	1
1	1	1

=

2	0	1
0	0	1
1	1	1

Tabela 2.39 – Demonstração para o exemplo da condição 5

2.7.2 – 4 valores:

1. Quando houver apenas quatro termos ou menos alinhados a CPB é necessária.

2. Quando houver pelo menos quatro termos alinhados e eles não tiverem sido agrupados em uma coluna ou linha completa, a CPB não é necessária.

3. Quando houver pelo menos três termos em diagonal “\”, a CPB não é necessária.

4. Quando houver apenas um quadrado 3x3 faltando o termo superior esquerdo, a CPB é necessária.

c0 = (A b B) a 1
c0 = (A b B) a (A/ b B) a 1
c0 = (A b B) a (A/ b B) a (//A b B) a 1
c0 = B a 1

0	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

0	0	1	1
1	1	1	1
1	1	1	1
1	1	1	1

0	0	0	1
1	1	1	1
1	1	1	1
1	1	1	1

0	0	0	0
1	1	1	1
1	1	1	1
1	1	1	1

Tabela 2.40 – Exemplos da condição 1.

c0 = (A b B) a (A/ b B) a (//A b B) a (/A b B)

0	0	0	0
1	1	1	1
1	1	1	1
1	1	1	1

Tabela 2.41 – Exemplos da condição 2.

c0 = (A b B) a (A/ b B/) a (//A b //B)

0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

Tabela 2.42 – Exemplo da condição 3.

c0 = (A/ b B) a (//A b B) a (A b B/) a (A/ b B/) a (//A b B/) a (A b //B) a (A/ b //B) a (//A b //B) a 1

1	0	0	1
0	0	0	1
0	0	0	1
1	1	1	1

Tabela 2.43 – Exemplo da condição 4.

Inúmeras possibilidades aparecem quando se tenta estabelecer um padrão, mostrando que, em se tratando de uma ferramenta automatizada, é mais fácil fazer essa verificação por inspeção, do que analisar, caso a caso, se a CPB é necessária. Além disso, o agrupamento em linhas e colunas completas pode tornar necessária a CPB em situações em que não seria necessário, tornando a análise mais complicada ainda.

A primeira versão do software LMVS, apresentada no trabalho de mestrado, aplica as análises acima para identificar a necessidade da CPB, porém inúmeras exceções ocorrem, e, nesses casos, a CPB é colocada indevidamente. Na nova versão (capítulo 6), é empregado um método inspecional (apêndice 3), onde se determina a tabela da expressão sem a CPB; a CPB é colocada somente se for verificada a presença de um valor espúrio na tabela.

O motivo pelo qual o conteúdo deste capítulo é apresentado é que, a partir de funções binárias-MVL, pode-se obter funções MVL de três ou mais valores (e não, apenas, dois). Este é o assunto do capítulo 3.

Capítulo 3
Leis de formação para funções MVL

3.1 Introdução

Neste capitulo é apresentada uma técnica de síntese de funções MVL de três ou mais valores. Tais funções podem ser decompostas em subfunções binárias-MVL (SFB) dentro do escopo da lógica MVL, como explicado no capítulo 2 (e que não devem ser confundidas com as funções booleanas). Da mesma forma, toda função MVL pode ser obtida por meio de SFBs[11,13,15].

A criação de uma SFB a partir de uma função MVL requer a escolha de um valor como sendo o dominante, e o valor seguinte (o TOPO do outro valor escolhido) como o secundário. Quanto aos demais valores, há duas opções:

- Substituição dos demais valores pelo valor secundário (o TOPO do dominante).
- Substituição dos demais valores pelo valor dominante (o BASE do secundário).

Para representar a ocorrência de valores em funções MVL de "m" entradas, é utilizado um vetor, com todos os valores (tal como foi explicado no capítulo 1). Para três valores, tem-se `(0,1,2)`; para quatro valores, tem-se `(0,1,2,3)`. Essa grafia representa um trio ou quarteto ordenado, onde cada elemento representa todas as ocorrências do valor determinado pela sua posição. O valor contido em cada posição representa o valor resultante, na função, após uma determinada operação. Para a representação de funções MVL, utilizam-se letras maiúsculas seguidas por um número que identifica a forma pela qual a função foi gerada (B1, C2, etc); para a representação de SFBs, utilizam-se letras minúsculas seguidas por um número que representa o valor dominante (b0, c1, etc). Este padrão permite que o tipo de função seja determinado pelo seu nome, evitando problemas de interpretação.

3.2 Substituição pelo valor secundário

As CPBs por meio da substituição pelo valor secundário são mostradas nas equações (1). Tomando, como exemplo, a (1h), e, supondo tratar-se de uma função de uma entrada [B=B(A)] pode-se fazer a explicação por meio das equações (2). No caso de 3 valores, é fácil perceber que a função geral MVL **(0,1,2)** não pode ser obtida somente por meio da combinação das SFBs (equações (3)). Sendo assim, a substituição pelo valor secundário não é vantajosa. O mesmo acontece na lógica quaternária.

3 valores:		4 valores	
(D,S,I) → c0 = (D,S,S)	(1a)	**(D,S,T,I) → c0 = (D,S,S,S)**	(1d)
(I,D,S) → c1 = (S,D,S)	(1b)	**(I,D,S,T) → c1 = (S,D,S,S)**	(1e)
(S,I,D) → c2 = (S,S,D)	(1c)	**(T,I,D,S) → c2 = (S,S,D,S)**	(1f)
		(S,T,I,D) → c3 = (S,S,S,D)	(1g)
(0,1,2) → c0 = (0,1,1)	(1h)	**(0,1,2,3) → c0 = (0,1,1,1)**	(1k)
(0,1,2) → c1 = (2,1,2)	(1i)	**(0,1,2,3) → c1 = (2,1,2,2)**	(1l)
(0,1,2) → c2 = (0,0,2)	(1j)	**(0,1,2,3) → c2 = (3,3,2,3)**	(1m)
		(0,1,2,3) → c3 = (0,0,0,3)	(1n)

- Se B = 0 então b = 0 (2a)
- Se B = 1 então b = 1 (2b)
- Se B = 2 então b = 1 (2c)

Se B = **D** então b = **D** (2d).
Se B = **S** então b = **S** (2e)
Se B = **I** então b = **S** (2f).

- B: Função MVL
- b: Função binária-MVL derivada da função MVL B original.

c0 a c1 = (0,1,1) (3a)	**c1 b c2 = (2,1,2)** (3d)	**c2 c c0 = (0,0,2)** (3g)
c0 b c1 = (2,1,1) (3b)	**c1 c c2 = (2,0,2)** (3e)	**c2 a c0 = (0,0,1)** (3h)
c0 c c1 = (2,1,2) (3c)	**c1 a c2 = (0,0,2)** (3f)	**c2 b c0 = (0,1,1)** (3i)

3.3 Substituição pelo valor dominante

Uma vez verificado que a substituição dos valores terciário e indiferente pelo valor secundário não é interessante, resta, então, analisar a substituição pelo valor dominante.

3.3.1 – 3 valores:

As fórmulas para três valores são dadas a seguir, (4). Pode-se verificar, por inspeção, quais são as combinações de SFBs que geram a função geral MVL **(0,1,2)**.

(D,S,I) → cO = (D,S,D) (4a)	**(0,1,2) → c0 = (0,1,0)** (4d)
(I,D,S) → c1 = (D,D,S) (4b)	**(0,1,2) → c1 = (1,1,2)** (4e)
(S,I,D) → c2 = (S,D,D) (4c)	**(0,1,2) → c2 = (0,2,2)** (4f)

```
c0 a c1 = (0,1,0) (5a)     c1 a c2 = (0,1,2) (5d)     c2 a c0 = (0,1,0) (5g)
c0 b c1 = (1,1,2) (5b)     c1 b c2 = (1,1,2) (5e)     c2 b c0 = (0,1,2) (5h)
c0 c c1 = (0,1,2) (5c)     c1 c c2 = (0,2,2) (5f)     c2 c c0 = (0,2,2) (5i)

                           c1 a c2 = (0,1,2) (5d)
                           c2 b c0 = (0,1,2) (5h)
                           c0 c c1 = (0,1,2) (5c)
```

As fórmulas que fornecem a função MVL são as (5c),(5d) e (5h). As SFBs possuem o valor indiferente transformado no dominante, ou seja, formam duas ocorrências do valor dominante. Se o operador utilizado tem, como dominante, o valor dominante de uma das duas SFBs, a função resultante tem duas ocorrências desse valor e uma do valor secundário, não é MVL, tal como observado das seis outras fórmulas. Para gerar a função MVL correta, o operador entre as duas SFBs não deve ser ter, como dominante, o valor dominante de nenhuma das duas SFBs, sobrando, apenas, uma alternativa. Por exemplo, no caso de se usar **c0** e **c1**, o valor dominante para essas SFBs são "0" e "1", respectivamente. O operador escolhido não pode ter, como dominante, nenhum desses dois valores; sobra, então, apenas o valor "2". O operador que tem esse valor como dominante é o "c" (fórmula (5c)).

As leis de formação para as funções MVL de três valores são [3,12]:

```
C1 = c0 c c1 (6a)          (0,1,0) c (1,1,2) = (0,1,2) (6d)
C2 = c1 a c2 (6b)          (1,1,2) a (0,2,2) = (0,1,2) (6e)
C3 = c2 b c0 (6c)          (0,2,2) b (0,1,0) = (0,1,2) (6f)

          C1: (S,I,S) op (I,I,D) = (S,I,D) (6g)
          C2: (S,S,I) op (D,I,I) = (D,S,I) (6h)
          C3: (I,S,S) op (I,D,I) = (I,D,S) (6i)
```

Esta dedução está baseada em três regras:

- **D op S = D**
- **S op I = S**
- **I op I = I**

3.3.2 – 4 valores:

As fórmulas para quatro valores são dadas a seguir. Pode-se verificar, por inspeção, quais são as combinações de SFBs que geram a função geral MVL **(0,1,2,3)**.

```
(D,S,T,I) → c0 = (D,S,D,D) (7a)     (0,1,2,3) → c0 = (0,1,0,0) (7e)
(I,D,S,T) → c1 = (D,D,S,D) (7b)     (0,1,2,3) → c1 = (1,1,2,1) (7f)
(T,I,D,S) → c2 = (D,D,D,S) (7c)     (0,1,2,3) → c2 = (2,2,2,3) (7g)
(S,T,I,D) → c3 = (S,D,D,D) (7d)     (0,1,2,3) → c3 = (0,3,3,3) (7h)
```

`c0 a c1 = (0,1,0,0)` (8a)	`c1 a c2 = (1,1,2,1)` (8e)	`c2 a c3 = (0,2,2,3)` (8i)
`c0 b c1 = (1,1,2,1)` (8b)	`c1 b c2 = (1,1,2,1)` (8f)	`c2 b c3 = (2,2,2,3)` (8j)
`c0 c c1 = (0,1,2,0)` (8c)	`c1 c c2 = (2,2,2,3)` (8g)	`c2 c c3 = (2,2,2,3)` (8k)
`c0 d c1 = (0,1,0,0)` (8d)	`c1 d c2 = (1,1,2,3)` (8h)	`c2 d c3 = (0,3,3,3)` (8l)
`c3 a c0 = (0,1,0,0)` (8m)	`c0 a c2 = (0,1,0,0)` (8q)	`c1 a c3 = (0,1,2,1)` (8u)
`c3 b c0 = (0,1,3,3)` (8n)	`c0 b c2 = (2,1,2,3)` (8r)	`c1 b c3 = (1,1,2,1)` (8v)
`c3 c c0 = (0,3,3,3)` (8o)	`c0 c c2 = (2,2,2,3)` (8s)	`c1 c c3 = (0,3,2,3)` (8w)
`c3 d c0 = (0,3,3,3)` (8p)	`c0 d c2 = (0,1,0,3)` (8t)	`c1 d c3 = (0,3,3,3)` (8x)

A quantidade de possibilidades sobe bastante de três para quatro valores, dando uma idéia da complexidade das lógicas de bases maiores. Nenhuma das 24 combinações gera a função geral MVL (`c = (0,1,2,3)`); mas oito delas geram funções que fornecem três valores, por isso elas devem ser escolhidas. As opções que geram dois valores são descartadas.

`c0 c c1 = (0,1,2,0)` (8c)	`c0 d c2 = (0,1,0,3)` (8t)
`c1 d c2 = (1,1,2,3)` (8h)	`c1 a c3 = (0,1,2,1)` (8u)
`c2 a c3 = (0,2,2,3)` (8i)	`c2 b c0 = (2,1,2,3)` (8r)
`c3 b c0 = (0,1,3,3)` (8n)	`c3 c c1 = (0,3,2,3)` (8w)

`(0,1,0,0) c (1,1,2,1) = (0,1,2,0)` (9a)	`(0,1,0,0) d (2,2,2,3) = (0,1,0,3)` (9e)
`(1,1,2,1) d (2,2,2,3) = (1,1,2,3)` (9b)	`(1,1,2,1) a (0,3,3,3) = (0,1,2,1)` (9f)
`(2,2,2,3) a (0,3,3,3) = (0,2,2,3)` (9c)	`(2,2,2,3) b (0,1,0,0) = (2,1,2,3)` (9g)
`(0,3,3,3) b (0,1,0,0) = (0,1,3,3)` (9d)	`(0,3,3,3) c (1,1,2,1) = (0,3,2,3)` (9h)
`(T,I,T,T) op (I,I,D,I) = (T,I,D,T)` (9i)	`(S,T,S,S) op (T,T,T,D) = (S,T,S,D)` (9m)
`(T,T,I,T) op (I,I,I,D) = (T,T,I,D)` (9j)	`(S,S,T,S) op (D,T,T,T) = (D,S,T,S)` (9n)
`(T,T,T,I) op (D,I,I,I) = (D,T,T,I)` (9k)	`(S,S,S,T) op (T,D,T,T) = (S,D,S,T)` (9o)
`(I,T,T,T) op (I,D,I,I) = (I,D,T,T)` (9l)	`(T,S,S,S) op (T,T,D,T) = (T,S,D,S)` (9p)

Na 1ª coluna, estão as combinações similares às da lógica ternária (fórmulas (6)), chamadas de “diretas” nas quais a 2ª SFB possui, como dominante, o valor secundário da 1ª SFB. Na 2ª coluna, estão as combinações diferentes, chamadas de “interpoladas”, nas quais a 2ª SFB possui, como valor dominante, o valor terciário da 1ª SFB. As fórmulas (8) representam funções quaternárias que retornam, apenas, três valores; recebem o nome de “sub-função ternário-quaternária”, SFT. Estas funções são obtidas com base em três regras:

Normal:

- `D op T = D`
- `T op I = T`
- `I op I = I`

Interpolado:

- `D op S = D`
- `S op T = S`
- `T op T = T`

Comparando-se com as três regras usadas na lógica ternária, vê-se que, na lógica quaternária no modo normal, usa-se o valor terciário ao invés do secundário. No modo interpolado, utiliza-se o valor terciário ao invés do indiferente. No modo normal, a SFT apresenta falta do valor secundário, enquanto que, no interpolado, apresenta falta do valor indiferente.

Para simplificar a grafia das SFTs, utiliza-se uma letra minúscula e dois números, os valores dominantes das duas SFBs, na ordem. Continua valendo o princípio de que o operador utilizado deve ter, como dominante, um valor diferente dos valores dominantes das suas SFBs. Percebe-se que, tanto na 1ª coluna (forma direta), como na 2ª (forma interpolada), o valor dominante do operador é o valor secundário da 2ª SFB.

Direto	Interpolado
c01 = (0,1,2,0) (10a);	**c02 = (0,1,0,3)** (10e)
c12 = (1,1,2,3) (10b);	**c13 = (0,1,2,1)** (10f)
c23 = (0,2,2,3) (10c);	**c20 = (2,1,2,3)** (10g)
c30 = (0,1,3,3) (10d);	**c31 = (0,3,2,3)** (10h)

As SFBs c0,c1,c2 e c3 podem sofrer qualquer operações, desde que não se empregue um conectivo cujo valor dominante coincida com o dominante da SFB. Por exemplo, **c0** pode sofrer as operações “b”, “c” e “d”, mas não suporta a operação “a”, pois, para **c0**, o valor dominante é “0”, que é o valor dominante da operação “a”. Como **c0** possui três valores transformados em “0”, se for operado com “a”, o resultado continua tendo esses três valores em “0”, a SFB não se transforma em uma SFT. Por exemplo, no caso de **c0**, as duas formas são: **c0 c c1 = (0,1,2,0)** (10a); **c0 d c2 = (0,1,0,3)** (10e);

As SFTs são obtidas por meio de três princípios:

1. Na forma direta, o valor dominante da 2ª SFB deve ser o valor secundário da 1ª SFB.
2. Na forma interpolada, o valor dominante da 2ª SFB deve ser o valor terciário da 1ª SFB.
3. O operador utilizado deve ter, como dominante, o valor secundário da 2ª SFB.

Para formar a função MVL, deve-se adicionar, às funções acima, mais uma SFB e um conectivo. As combinações possíveis são mostradas a seguir. Para que todo texto caiba na tabela, os parênteses e as vírgulas foram retirados. As combinações que fornecem todos os valores estão sublinhadas. As SFTs diretas geram duas formas de função MVL, as interpoladas geram apenas uma.

		(0,1,0,0)			(1,1,2,1)			(2,2,2,3)			(0,3,3,3)		
		b c0	c c0	d c0	a c1	c c1	d c1	a c2	b c2	d c2	a c3	b c3	c c3
Diretas	c0 c c1 (0,1,2,0)	0120	0120	0100	0120	0120	0120	0120	2123	0123	0120	0123	0333
	c1 d c2 (1,1,2,3)	1123	0123	0103	1121	1123	1123	1123	1123	1123	0123	1123	1323
	c2 a c3 (0,2,2,3)	0123	0223	0103	0121	0223	0123	0223	2223	0223	0223	0223	0223
	c3 b c0 (0,1,3,3)	0133	0133	0103	0121	0123	0133	0123	2123	0133	0133	0133	0333
Interpoladas	c0 d c2 (0,1,0,3)	0103	0103	0103	0101	0123	0103	0103	2123	0103	0103	0103	0333
	c1 a c3 (0,1,2,1)	0121	0120	0100	0121	0121	0121	0121	2121	0123	0121	0121	0323
	c2 b c0 (2,1,2,3)	2123	2123	0103	1121	2123	1123	2123	2123	2123	0123	2123	2323
	c3 c c1 (0,3,2,3)	0123	0323	0303	0121	0323	0323	0223	2223	0323	0323	0323	0323

Tabela 3.1 – Todas as possibilidades de operações (quatro valores).

A análise também pode ser feita observando-se, apenas, as SFTs.

c01 b c30 = (0,1,2,0) b (0,1,3,3) = (0,1,2,3)
c01 b (c3 b c0) = (0,1,2,3)
(c01 b c3) b c0 = (0,1,2,3)
((0,1,2,0) b (0,3,3,3)) b c0 = (0,1,2,3)
((0,1,2,3)) b c0 = (0,1,2,3)
Conclusão: (c0 c c1) b c3 = (0,1,2,3)

As leis de formação para as funções MVL de quatro valores são dadas pelas fórmulas (11) e (12). A verificação é feita pelas fórmulas (13).

C = (c0 c c1) d c2 (11a)
C = (c1 d c2) a c3 (11b)
C = (c2 a c3) b c0 (11c)
C = (c3 b c0) c c1 (11d)

C = (c0 c c1) b c3 (11e)
C = (c1 d c2) c c0 (11f)
C = (c2 a c3) d c1 (11g)
C = (c3 b c0) a c2 (11h)

C = (c0 d c2) c c1 (11i)
C = (c1 a c3) d c2 (11j)
C = (c2 b c0) a c3 (11k)
C = (c3 c c1) b c0 (11l)

C = c01 d c2 (12a)
C = c12 a c3 (12b)
C = c23 b c0 (12c)
C = c30 c c1 (12d)

C = c01 b c3 (12e)
C = c12 c c0 (12f)
C = c23 d c1 (12g)
C = c30 a c2 (12h)

C = c13 d c2 (12i)
C = c20 a c3 (12j)
C = c31 b c0 (12k)
C = c02 c c1 (12l)

[(0,1,0,0) c (1,1,2,1)] d (2,2,2,3) = (0,1,2,3) (13a)
[(1,1,2,1) d (2,2,2,3)] a (0,3,3,3) = (0,1,2,3) (13b)
[(2,2,2,3) a (0,3,3,3)] b (0,1,0,0) = (0,1,2,3) (13c)
[(0,3,3,3) b (0,1,0,0)] c (1,1,2,1) = (0,1,2,3) (13d)

[(0,1,0,0) c (1,1,2,1)] b (0,3,3,3) = (0,1,2,3) (13e)
[(1,1,2,1) d (2,2,2,3)] c (0,1,0,0) = (0,1,2,3) (13f)
[(2,2,2,3) a (0,3,3,3)] d (1,1,2,1) = (0,1,2,3) (13g)
[(0,3,3,3) b (0,1,0,0)] a (2,2,2,3) = (0,1,2,3) (13h)

[(0,1,0,0) d (2,2,2,3)] c (1,1,2,1) = (0,1,2,3) (13i)
[(1,1,2,1) a (0,3,3,3)] d (2,2,2,3) = (0,1,2,3) (13j)
[(2,2,2,3) b (0,1,0,0)] a (0,3,3,3) = (0,1,2,3) (13k)
[(0,3,3,3) c (1,1,2,1)] b (0,1,0,0) = (0,1,2,3) (13ll

As quatro primeiras formas são chamadas de diretas (fórmulas (11 a (13) de (a) a (d)) porque as SFBs têm, como valor dominante, uma seqüência TOPO, e os operadores também. Nos oito primeiros tipos ((a) a (h)), utiliza-se as SFTs similares às da lógica ternária, nas quais a 2ª SFB possui, como valor dominante, o valor secundário da 1ª SFB. As quatro últimas utilizam as SFTs nas quais a 2ª SFB possui, como valor dominante, o valor terciário da 1ª SFB. O nome dado para cada possibilidade é mostrado na tabela 3.3.

Forma 1 Utilizando c0,c1 e c2:	Forma 2 Utilizando c1,c2 e c3:	Forma 3 Utilizando c2,c3 e c0:	Forma 4 Utilizando c3,c0 e c1:
`C = (c0 c c1) d c2`	`C = (c1 d c2) a c3`	`C = (c2 a c3) b c0`	`C = (c3 b c0) c c1`
`C = (c1 d c2) c c0`	`C = (c2 a c3) d c1`	`C = (c3 b c0) a c2`	`C = (c0 c c1) b c3`
`C = (c0 d c2) c c1`	`C = (c1 a c3) d c2`	`C = (c2 b c0) a c3`	`C = (c3 c c1) b c0`

Tabela 3.2 – As quatro formas

`C1 = C1(c0,c1)` (14a) `C2 = C2(c1,c2)` (14b) `C3 = C3(c2,c0)` (14c)	`C1 = C1(c0,c1,c2)` (14d) `C2 = C2(c1,c2,c3)` (14e) `C3 = C3(c2,c3,c0)` (14f) `C4 = C4(c3,c0,c1)` (14g)

Tabela 3.3 – Formas (três e quatro valores).

Tomando por base a forma 1 quaternária, verifica-se as seguintes igualdades:

- `(c0 c c1) d c2 = (c1 d c2) c c0 ;` `(A c B) d C = (B d C) c A` (15a)
- `(c0 c c1) d c2 = (c0 d c2) c c1 ;` `(A c B) d C = (A d C) c B` (15b)

Isso significa que as três fórmulas para cada fórmula são, na verdade, a mesma fórmula, reescritas de acordo com a propriedade (15). Toda função MVL é função de um grupo de SFBs. A quantidade dessas SFBs é dada pela base menos um (uma para funções booleanas, duas para funções ternárias e três para quaternárias).

Para cada uma das quatro formas de escolha das quatro SFBs a serem utilizadas, para quatro valores, há três formas de se construir a função MVL (linhas da tabela 3.2); todas as três possuem fórmulas com o mesmo tamanho. Porém, as quatro SFBs podem ter tamanhos diferentes, e, em decorrência disso, as quatro possibilidades de escolha das três SFBs (colunas da tabela 3.2) podem fornecer expressões com tamanhos diferentes. Mais adiante, será mostrado que, para quatro valores, das 12 formas de construção, quatro não são interessantes, ficando, assim, apenas oito formas de construção.

3.4 Exemplos

Os exemplos a seguir ilustram os conceitos apresentados. Tais exemplos foram obtidos por meio do software LMVS (capítulo 6).

Exemplo 1:

A tabela 3.4 apresenta a função a ser sintetizada, que pode ser separada em três SFBs conforme mostrado na tabela 3.5. Para cada uma dessas SFBs, pode-se escrever uma expressão (c0, c1 e c2). A expressão MVL é obtida pela junção de duas SFBs, perfazendo três possibilidades. Deve-se escolher a expressão que ofereça maior vantagem segundo algum critério de comparação. Normalmente, esse critério é o tamanho da expressão.

0	0	2
0	1	1
2	1	2

Tabela 3.4 – Exemplo 1.

c0	c1	c2
000	112	002
011	111	022
010	212	222

Tabela 3.5 – As tabelas das SFBs.

```
c0 = (/A b /B) a A a B
c1 = (/A c /B) b A b B
c2 = (/A a /B) c A c B

C1 = c0 c c1
C2 = c1 a c2
C3 = c2 b c0
```

```
C1 = [ (/A b /B) a A a B ] c [ (/A c /B) b A b B ]
C2 = [ (/A c /B) b A b B ] a [ (/A a /B) c A c B ]
C3 = [ (/A a /B) c A c B ] b [ (/A b /B) a A a B ]
```

Neste exemplo, todas as SFBs e expressões completas apresentam o mesmo tamanho, sendo necessário um outro critério e escollha. A expressão escolhida pode, então, ser implementada na aplicação desejada.

Exemplo 2:

0	0	0
0	1	1
0	1	2

Tabela 3.6 – Exemplo 2.

c0	c1	c2
000	111	000
011	111	022
010	112	022

Tabela 3.7 – As tabelas das SFBs.

```
c0 = (/A b /B) a A a B
c1 = /A b A b /B b B
c2 = (/A a /B) c (A a /B) c (/A a B) c (A a B)
```

```
C1 = [ (/A b /B) a A a B ] c [ /A b A b /B b B ]
C2 = [ /A b A b /B b B ] a [ (/A a /B) c (A a /B) c (/A a B) c (A a B) ]
C3 = [ (/A a /B) c (A a /B) c (/A a B) c (A a B) ] b [ (/A b /B) a A a B ]
```

Neste exemplo, a menor expressão é a C_1.

Exemplo 3:

1	2	1	2

1	3	0	1
2	1	3	2
1	0	2	0

Tabela 3.8 – Exemplo 3.

c0	c1	c2	c3
1010	1212	2222	3333
1001	1111	2322	3303
0100	2112	2232	3333
1000	1121	2222	3030

Tabela 3.9 – As tabelas das SFBs.

```
c0 = (A/ b B) a (/A b B) a (A/ b B/) a (//A b B/) a (A b //B) a (//A b //B) a (/A b //B) a (A/ b
/B) a (//A b /B) a (/A b /B)
c1 = (/A c /B) b (A/ c /B) b (A c B/) b (A/ c B/) b (/A c //B) b (A c //B) b (//A c //B) b B
c2 = (A d /B) c (/A d B) c //A c A/ c //B c B/
c3 = (//A a //B) d (A a //B) d (/A a B) d A/ d B/ d /B
```

```
C1 = { [ (A/ b B) a (/A b B) a (A/ b B/) a (//A
b B/) a (A b //B) a (//A b //B) a (/A b //B) a
(A/ b /B) a (//A b /B) a (/A b /B) ] c [ (/A c
/B) b (A/ c /B) b (A c B/) b (A/ c B/) b (/A c
//B) b (A c //B) b (//A c //B) b B ] } d [ (A d
/B) c (/A d B) c //A c A/ c //B c B/ ]

C2 = { [ (/A c /B) b (A/ c /B) b (A c B/) b (A/
c B/) b (/A c //B) b (A c //B) b (//A c //B) b
B ] d [ (A d /B) c (/A d B) c //A c A/ c //B c
B/ ] } a [ (//A a //B) d (A a //B) d (/A a B) d
A/ d B/ d /B ]

C3 = { [ (A d /B) c (/A d B) c //A c A/ c //B c B/
] a [ (//A a //B) d (A a//B) d (/A a B) d A/ d B/
d /B ] } b [ (A/ b B) a (/A b B) a (A/ b B/) a
(//A b B/) a (A b //B) a (//A b //B) a (/A b //B)
a (A/ b /B) a (//A b /B) a (/A b /B) ]

C4 = { [ (//A a //B) d (A a //B) d (/A a B) d A/ d
B/ d /B ] b [ (A/ b B) a (/A b B) a (A/ b B/) a
(//A b B/) a (A b //B) a (//A b //B) a (/A b //B)
a (A/ b /B) a (//A b /B) a (/A b /B) ] } c [ (/A c
/B) b (A/ c /B) b (A c B/) b (A/ c B/) b (/A c
//B) b (A c //B) b (//A c //B) b B ]
```

A menor expressão é a C_2. A síntese de funções quaternárias é muito mais trabalhosa do que a de funções ternárias. Percebe-se, então, a necessidade de se obter uma ferramenta automatizada para a realização desta tarefa.

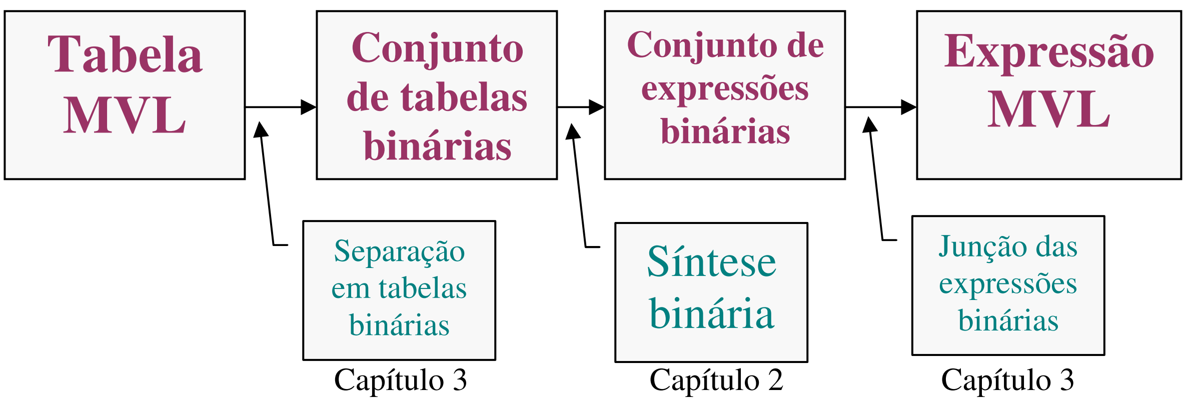


Figura 3.1 – Diagrama de blocos do processo de síntese MVL.

3.5 Valores irrelevantes fixos

As ocorrências do valor indiferente (lógica.ternária) ou dos valores terciário e indiferente (lógica quaternária) que podem ser substituídos por qualquer valor e não, obrigatoriamente, pelo valor dominante, durante a etapa da construção das SFBs são chamados de valores irrelevantes fixos. Essas irrelevâncias provém das leis de formação das funções MVL a partir de SFBs e da aplicação da propriedade da obtenção do valor dominante (capítulo 1). Muito embora a explicação apresentada aborde, apenas, os casos ternários e quaternários, os conceitos valem para qualquer quantidade de valores acima de dois.

3.5.1 – 3 valores:

As fórmulas (6) são as leis de formação para as funções MVL apresentadas anteriormente. Considerando que, para os operadores “a”, “b” e “c”, os valores dominantes são, respectivamente, “0”, “1” e “2”, a ocorrência desses valores em uma das SFBs faz com que o valor, na outra SFB, na mesma posição, seja irrelevante (propriedade da operação com o dominante). O valor dominante do operador só aparece na 2ª SFB, há ocorrência de valores irrelevantes somente na 1ª SFB. As leis de formação para as funções MVL de três valores devem levar em conta o uso dos valores irrelevantes. Atribui-se o sufixo “*” ao nome da SFB para mencionar o fato de que a ocorrência de valores irrelevantes foi considerada. Atribui-se o símbolo “X” – *don’t care* – ao valor irrelevante.

```
C1 = c0 c c1 (6a)          (0,1,0) c (1,1,2) = (0,1,2) (6d)
C2 = c1 a c2 (6b)          (1,1,2) a (0,2,2) = (0,1,2) (6e)
C3 = c2 b c0 (6c)          (0,2,2) b (0,1,0) = (0,1,2) (6f)

         C1: (S,I,S) op (I,I,D) = (S,I,D) (6g)
         C2: (S,S,I) op (D,I,I) = (D,S,I) (6h)
         C3: (I,S,S) op (I,D,I) = (I,D,S) (6i)

C1 = c0* c c1 (16a)        (0,1,X) c (1,1,2) = (0,1,2) (16d)
C2 = c1* a c2 (16b)        (X,1,2) a (0,2,2) = (0,1,2) (16e)
C3 = c2* b c0 (16c)        (0,X,2) b (0,1,0) = (0,1,2) (16f)

         C1: (S,I,X) op (I,I,D) = (S,I,D) (16g)
         C2: (X,S,I) op (D,I,I) = (D,S,I) (16h)
         C3: (I,X,S) op (I,D,I) = (I,D,S) (16i)

c0* = (0,1,X) (17a)        c0 = (0,1,0) (17d)
c1* = (X,1,2) (17b)        c1 = (1,1,2) (17e)
c2* = (0,X,2) (17c)        c2 = (0,2,2) (17f)

c0* = (D,S,X) (17g)        c0 = (D,S,D) (17j)
c1* = (X,D,S) (17h)        c1 = (D,D,S) (17k)
c2* = (S,X,D) (17i)        c2 = (S,D,D) (17l)
```

	Grupo	
	Primeiro	**Segundo**
Dominante	Dominante	Dominante
Secundário	Secundário	Secundário
Indiferente	Irrelevante	Dominante

	Grupo	
	1º	**2º**
Dominante	V	V
Secundário	F	F
Indiferente	X	V

Tabela 3.10 – Regras de substituição para três valores.

O nome das funções MVL completas (C_1, C_2 e C_3) não muda, não recebe o símbolo "*" porque não é alterada com o uso dos valores irrelevantes na 1ª SFB; aliás, nem é para mudar, pois, se mudar, torna-se outra função e o de emprego de valores irrelevantes implica em erro. São dois grupos de SFBs. Cada um possui uma regra na substituição dos valores à partir da função MVL `(0,1,2)` original.

3.5.2 – 4 valores:

Para quatro valores, vale o mesmo raciocínio. Como há, em cada fórmula, dois operadores que ligam as três SFBs, é necessário fazer a análise em duas etapas. Analisando o 1º operador, ou seja, a SFT, tem-se:

```
c01* = (0,1,X,0) c (1,1,2,1) (18a)      c02* = (0,1,0,X) d (2,2,2,3) (18e)
c12* = (1,1,2,X) d (2,2,2,3) (18b)      c13* = (X,1,2,1) a (0,3,3,3) (18f)
c23* = (X,2,2,3) a (0,3,3,3) (18c)      c20* = (2,X,2,3) b (0,1,0,0) (18g)
c30* = (0,X,3,3) b (0,1,0,0) (18d)      c31* = (0,3,X,3) c (1,1,2,1) (18h)

(T,I,X,T) op (I,I,D,I) = (T,I,D,T) (18i)      (S,T,S,X) op (T,T,T,D) = (S,T,S,D) (18i)
(T,T,I,X) op (I,I,I,D) = (T,T,I,D) (18i)      (X,S,T,S) op (D,T,T,T) = (D,S,T,S) (18i)
(X,T,T,I) op (D,I,I,I) = (D,T,T,I) (18i)      (S,X,S,T) op (T,D,T,T) = (S,D,S,T) (18i)
(I,X,T,T) op (I,D,I,I) = (I,D,T,T) (18i)      (T,S,X,S) op (T,T,D,T) = (T,S,D,S) (18i)

C = (c0* c c1) d c2 (19a)   C = (c0* c c1) b c3 (19e)   C = (c0* d c2) c c1 (19i)
C = (c1* d c2) a c3 (19b)   C = (c1* d c2) c c0 (19f)   C = (c1* a c3) d c2 (19j)
C = (c2* a c3) b c0 (19c)   C = (c2* a c3) d c1 (19g)   C = (c2* b c0) a c3 (19k)
C = (c3* b c0) c c1 (19d)   C = (c3* b c0) a c2 (19h)   C = (c3* c c1) b c0 (19l)

[(0,1,X,0) c (1,1,2,1)] d (2,2,2,3) = (0,1,2,3) (20a)
[(1,1,2,X) d (2,2,2,3)] a (0,3,3,3) = (0,1,2,3) (20b)
[(X,2,2,3) a (0,3,3,3)] b (0,1,0,0) = (0,1,2,3) (20c)
[(0,X,3,3) b (0,1,0,0)] c (1,1,2,1) = (0,1,2,3) (20d)

[(0,1,X,0) c (1,1,2,1)] b (0,3,3,3) = (0,1,2,3) (20e)
[(1,1,2,X) d (2,2,2,3)] c (0,1,0,0) = (0,1,2,3) (20f)
[(X,2,2,3) a (0,3,3,3)] d (1,1,2,1) = (0,1,2,3) (20g)
[(0,X,3,3) b (0,1,0,0)] a (2,2,2,3) = (0,1,2,3) (20h)

[(0,1,0,X) d (2,2,2,3)] c (1,1,2,1) = (0,1,2,3) (20i)
[(X,1,2,1) a (0,3,3,3)] d (2,2,2,3) = (0,1,2,3) (20j)
[(2,X,2,3) b (0,1,0,0)] a (0,3,3,3) = (0,1,2,3) (20k)
[(0,3,X,3) c (1,1,2,1)] b (0,1,0,0) = (0,1,2,3) (20l)
```

Mantendo as ocorrências dos valores irrelevantes gerados na etapa anterior e analisando o 2º operador, obtém-se as equações (20). Quatro das doze formas, (20e) a (20h) não criam valores irrelevantes por meio do 2º operador, pois o valor dominante deste operador não é encontrado na 3ª SFB; isso acarreta a perda de duas ocorrências de valores irrelevantes (uma na 1ª SFB, outra na 2ª SFB) prejudicando o processo de minimização. Não há vantagem na utilização dessas quatro formas, elas são descartadas.

Forma 1 - Utilizando c0,c1 e c2:
`C1 = (c0** c c1*) d c2`
`C5 = (c0** d c2*) c c1`
	Forma 2 - Utilizando c1,c2 e c3:
	`C2 = (c1** d c2*) a c3`
	`C6 = (c1** a c3*) d c2`

Forma 3 - Utilizando c2,c3 e c0:
`C3 = (c2** a c3*) b c0`
`C7 = (c2** b c0*) a c3`
	Forma 4 - Utilizando c3,c0 e c1:
	`C4 = (c3** b c0*) c c1`
	`C8 = (c3** c c1*) b c0`

Tabela 3.11 – As oito formas

```
C1 = (c0** c c1*) d c2 (21a)        C5 = (c0** d c2*) c c1 (21e)
C2 = (c1** d c2*) a c3 (21b)        C6 = (c1** a c3*) d c2 (21f)
C3 = (c2** a c3*) b c0 (21c)        C7 = (c2** b c0*) a c3 (21g)
C4 = (c3** b c0*) c c1 (21d)        C8 = (c3** c c1*) b c0 (21g)
```

```
[(0,1,X,X) c (1,1,2,X)] d (2,2,2,3) = (0,1,2,3) (22a)
[(X,1,2,X) d (X,2,2,3)] a (0,3,3,3) = (0,1,2,3) (22b)
[(X,X,2,3) a (0,X,3,3)] b (0,1,0,0) = (0,1,2,3) (22c)
[(0,X,X,3) b (0,1,X,0)] c (1,1,2,1) = (0,1,2,3) (22d)

[(0,1,X,X) d (2,2,X,3)] c (1,1,2,1) = (0,1,2,3) (22i)
[(X,1,2,X) a (0,3,3,X)] d (2,2,2,3) = (0,1,2,3) (22j)
[(X,X,2,3) b (X,1,0,0)] a (0,3,3,3) = (0,1,2,3) (22k)
[(0,X,X,3) c (1,X,2,1)] b (0,1,0,0) = (0,1,2,3) (22l)
```

```
c01** = (0,1,X,X) c (1,1,2,X) (23a)   ;   C1 = c01** d c2 (24a)
c12** = (X,1,2,X) d (X,2,2,3) (23b)   ;   C2 = c12** a c3 (24b)
c23** = (X,X,2,3) a (0,X,3,3) (23c)   ;   C3 = c23** b c0 (24c)
c30** = (0,X,X,3) b (0,1,X,0) (23d)   ;   C4 = c30** c c1 (24d)

c02** = (0,1,X,X) d (2,2,X,3) (23e)   ;   C5 = c02** c c1 (24e)
c13** = (X,1,2,X) a (0,3,3,X) (23f)   ;   C6 = c13** d c2 (24f)
c20** = (X,X,2,3) b (X,1,0,0) (23g)   ;   C7 = c20** a c3 (24g)
c31** = (0,X,X,3) c (1,X,2,1) (23h)   ;   C8 = c31** b c0 (24h)
```

Embora a primeira SFB (ci**) seja a mesma para as duas formas (normal e interpolada), na forma normal o valor secundário torna-se irrelevante devido ao primeiro operador e o valor indiferente devido ao segundo operador. Na forma interpolada o valor secundário é irrelevante devido ao segundo operador, e o valor indiferente devido ao primeiro.

Embora as fórmulas (21) estejam corretas, há um problema na nomenclatura, pois duas funções diferentes não podem receber o mesmo nome, e é isso que acontece, pois, a 2ª SFB para as quatro primeiras formas não corresponde à equivalente para as quatro últimas formas. A substituição diferente dos valores irrelevantes leva à formação de funções diferentes. Por esse motivo, atribui-se, às segundas SFBs das quatro últimas formas, os nomes de c4*, c5*, c6* e c7* no lugar de c0*, c1*, c2* e c3*, respectivamente. Nestes casos, o número que segue a letra representa o valor dominante mais quatro. As leis de formação para as funções MVL de quatro valores são dadas pelas fórmulas (25) e (26). As SFBs são dadas por (27). São quatro grupos de SFBs, cada um possui uma regra na substituição dos valores à partir da função MVL `(0,1,2,3)` original (tabela 3.12).

	Grupo			
	Primeiro	**Segundo**	**Terceiro**	**Quarto**
Dominante	Dominante	Dominante	Irrelevante	Dominante
Secundário	Secundário	Secundário	Secundário	Secundário
Terciário	Irrelevante	Irrelevante	Dominante	Dominante
Indiferente	Irrelevante	Dominante	Dominante	Dominante

	Grupo			
	1º	**2º**	**3º**	**4º**
D	V	V	X	V
S	F	F	F	F
T	X	X	V	V
I	X	V	V	V

Tabela 3.12 – Regras de substituição para quatro valores.

```
C1 = (c0** c c1*) d c2 (25a)   ;   C5 = (c0** d c6*) c c1 (26a)
C2 = (c1** d c2*) a c3 (25b)   ;   C6 = (c1** a c7*) d c2 (26b)
C3 = (c2** a c3*) b c0 (25c)   ;   C7 = (c2** b c4*) a c3 (26c)
C4 = (c3** b c0*) c c1 (25d)   ;   C8 = (c3** c c5*) b c0 (26d)
```

Primeiro grupo

```
c0** = (0,1,X,X) (27a)
c1** = (X,1,2,X) (27b)
c2** = (X,X,2,3) (27c)
c3** = (0,X,X,3) (27d)
```

Segundo grupo

```
c0*  = (0,1,X,0) (27e)
c1*  = (1,1,2,X) (27f)
c2*  = (X,2,2,3) (27g)
c3*  = (0,X,3,3) (27h)
```

Terceiro grupo

```
c4*  = (X,1,0,0) (27i)
c5*  = (1,X,2,1) (27j)
c6*  = (2,2,X,3) (27k)
c7*  = (0,3,3,X) (27l)
```

Quarto grupo

```
c0   = (0,1,0,0) (27m)
c1   = (1,1,2,1) (27n)
c2   = (2,2,2,3) (27o)
c3   = (0,3,3,3) (27p)
```

3.6 Técnica de substituição dos valores irrelevantes

Nesta seção, formaliza-se a maneira como os valores irrelevantes são substituídos.

3.6.1 – 1ª técnica: A princípio, toda ocorrência do valor irrelevante deve ser substituída pelo valor secundário, pois este valor não produz termos na expressão. A substituição pelo valor dominante acrescenta termos à expressão, aumentando seu tamanho.

0	0	0
0	1	1
0	1	X

Tabela 3.13 – Exemplo da 1ª técnica.

- Substituição pelo dominante:
  `C1 = (/A b /B) a A a B`
- Substituição pelo secundário:
  `C1 = A a B a 1`

A substituição do valor irrelevante pelo valor dominante provocou o surgimento do termo `(/A b /B)`. Eventualmente, a substituição pelo valor secundário pode requerer o uso da explícita conversão para binário-MVL (CPB), como neste caso.

3.6.2 – 2ª técnica: Se a linha ou a coluna que contiver a ocorrência do valor irrelevante não contiver nenhuma ocorrência do valor secundário, então a substituição deve ser feita pelo valor dominante, pois isso forma uma linha e/ou coluna completa, diminuindo o tamanho da expressão. Se tanto a linha como a coluna contiverem, pelo menos, uma ocorrência do valor secundário, então a substituição deve ser feita pelo valor secundário, pois não há como formar linha ou coluna completa, e fica valendo a 1ª técnica.

0	0	0
0	1	0
0	1	X

Tabela 3.14 – Exemplo da 2ª técnica.

- Substituição pelo dominante:
  `C1 = A a /A a B`
- Substituição pelo secundário:
  `C1 = (/A b B/) a A a B`

3.6.3 – 3ª técnica: Se a ocorrência de um valor irrelevante satisfizer a formação de uma linha completa e for substituído pelo valor dominante, ela não deve ser considerada como irrelevante para uma formação de coluna completa. Se a ocorrência de um valor irrelevante satisfizer a formação de uma coluna completa e for substituído pelo valor dominante, ela não deve ser considerada como irrelevante para uma formação de linha completa.

0	0	0
0	1	0
0	X_A	X_B

Tabela 3.15 – Exemplo da 3ª técnica.

- X_A e X_B pelo dominante: `C1 = A a /A a B a /B`
- X_B pelo dominante: `C1 = A a /A a B`
- X_A e X_B pelo secundário: `C1 = (/A b B/) a A a B`

3.6.4 – 4ª técnica: É preciso estabelecer uma hierarquia entre as ocorrências dos valores irrelevantes. Quanto maior for a quantidade de termos eliminados por meio da formação de linha ou coluna completa criada pela substituição do valor irrelevante pelo valor verdadeiro, maior é o poder minimizador da substituição, tal substituição devem ser prioritárias e realizadas antes, pois, uma vez realizadas, tais valores irrelevantes transformados em dominantes não são considerados como dominantes na análise da próxima hierarquia. No exemplo anterior, a 3ª técnica é eficaz somente se as colunas forem analisadas antes das linhas. Se for feito o contrário, a 3ª linha é inteiramente preenchida pelo valor verdadeiro, e a técnica falha. Pela 4ª técnica, esse problema é solucionado, pois, a seguinte tabela é gerada:

	Linha	Coluna
X_A	0 termo	Possui secundário - anulado
X_B	1 termo	2 termos

Tabela 3.16 – Tabela comparativa de ganhos com substituições.

X_B está em uma hierarquia superior, pois pode eliminar 2 termos em sua coluna. Após sua substituição, X_A não tem como eliminar nenhum termo, X_A é substituído pelo valor secundário. A seguir, tem-se um exemplo para quatro valores. Os termos da 1ª coluna não devem ser considerados, pois já fazem parte de uma linha completa.

$\cancel{0}$	1	0	0
$\cancel{0}$	1	X_A	0
$\cancel{0}$	0	X_B	0
$\cancel{0}$	X_C	0	X_D

Tabela 3.17 – Exemplo da 4ª técnica para 4 valores.

	Linha	Coluna	Total
X_A	possui secundário – anulado	2	2
X_B	2	2	4
X_C	1	possui secundário – anulado	1
X_D	1	3	4

Tabela 3.18 – Tabela comparativa de ganhos com substituições.

X_B e X_D possuem hierarquia superior, devem ser substituídos primeiro.

$\cancel{0}$	1	0	$\cancel{0}$
$\cancel{0}$	1	X_A	$\cancel{0}$
$\cancel{0}$	$\cancel{0}$	$\cancel{0}$	$\cancel{0}$
$\cancel{0}$	X_C	0	$\cancel{0}$

Tabela 3.19 – Primeira substituição.

	Linha	Coluna	Total
X_A	possui secundário – anulado	2	2
X_C	1	possui secundário – anulado	1

Tabela 3.20 – Tabela comparativa de ganhos com substituições.

$\cancel{0}$	1	$\cancel{0}$	$\cancel{0}$
$\cancel{0}$	1	$\cancel{0}$	$\cancel{0}$
$\cancel{0}$	$\cancel{0}$	$\cancel{0}$	$\cancel{0}$
$\cancel{0}$	X_C	$\cancel{0}$	$\cancel{0}$

Tabela 3.21 – Segunda substituição

$\cancel{0}$	1	$\cancel{0}$	$\cancel{0}$
$\cancel{0}$	1	$\cancel{0}$	$\cancel{0}$
$\cancel{0}$	$\cancel{0}$	$\cancel{0}$	$\cancel{0}$
$\cancel{0}$	1	$\cancel{0}$	$\cancel{0}$

Tabela 3.22 – Terceira substituição

```
C1 = A a //A a /A a //B
```

X_A possui hierarquia superior, deve ser substituído primeiro (tabela 3.21). Não sobrou nenhum termo para X_C anular, ou seja, X_C deve ser substituído pelo valor secundário (tabela 3.22). Por meio destes exemplos pode-se ter uma noção do quão complicada pode ser a análise das substituições e como pode ser difícil formalizar um procedimento para essa etapa da síntese de funções MVL.

Estas quatro técnicas não são suficientes para cobrir todos os casos da forma mais eficiente; há situações específicas nas quais a aplicação destas quatro técnicas não fornece a melhor substituição de irrelevâncias, porém elas são eficientes para a grande maioria dos casos. Devido ao grande número de combinações, uma verificação inspecional tornaria o aplicativo demasiadamente lento e, por isso, foi descartada. No software LMVS, este algoritmo é a etapa mais sofisticada e sofreu diversos aperfeiçoamentos a partir da primeira versão apresentada no trabalho de mestrado do mesmo autor.

3.7 Exemplos de síntese

Os exemplos a seguir mostram o resultado da aplicação dos conceitos apresentados. Diversas situações diferentes podem surgir dependendo do caso, de modo que não se pode abordar todos eles em poucos exemplos, porém, como esses exemplos, pretende-se dar uma noção de como os procedimentos devem ser empregados. Tais exemplos foram obtidos por meio do software LMVS (capítulo 6).

Exemplo 1:

0	0	2
0	1	1
2	1	2

Tabela 3.23 – Exemplo 1.

c0*	c1*	c2*	c0	c1	c2
000	212	002	000	112	002
011	111	002	011	111	022
011	212	222	010	212	222

Tabela 3.24 – As tabelas das SFBs.

```
  c0* = A a B a 1           ;      c0 = (/A b /B) a A a B
  c1* = A b B b 2           ;      c1 = (/A c /B) b A b B
  c2* = A c B c 0           ;      c2 = (/A a /B) c A c B
C1 = c0* c c1  ;     C1 = [ A a B a 1 ] c [ (/A c /B) b A b B ]
C2 = c1* a c2  ;     C2 = [ A b B b 2 ] a [ (/A a /B) c A c B ]
C3 = c2* b c0  ;     C3 = [ A c B c 0 ] b [ (/A b /B) a A a B ]
```

As SFBs com valores irrelevantes c0*, c1* e c2* são diferentes das sem valores irrelevantes c0, c1 e c2 (tabela 3.24); essa diferença faz com que as aquelas sofram uma minimização por linhas e colunas completas maior do que a destas.

Exemplo 2:

Esta é a função C = A a B ternária, porém, as expressões obtidas são maiores do que esta. Isto se deve ao fato de que existem, ainda, outras técnicas de minimização, apresentadas no capítulo 4, necessárias para a obtenção da expressão mínima, neste exemplo.

C
000
011
012

c0*	c1*	c2*	c0	c1	c2
000	212	000	000	111	000
011	111	000	011	111	022
011	212	002	010	112	022

Tabela 3.25 – Exemplo 2.

```
c0* = A a B a 1     ;     c0  = (/A b /B) a A a B
c1* = A b B b 2     ;     c1  = /A b A b /B b B
c2* = (A a B) c 0   ;     c2  = (/A a /B) c (A a /B) c (/A a B) c (A a B)
C1 = [ A a B a 1 ] c [ /A b A b /B b B ]
C2 = [ A b B b 2 ] a [ (/A a /B) c (A a /B) c (/A a B) c (A a B) ]
C3 = [ (A a B) c 0 ] b [ (/A b /B) a A a B ]
```

Exemplo 3:

1	2	1	2
1	3	0	1
2	1	3	2
1	0	2	0

Tabela 3.26 – Exemplo 3.

c0**	c1**	c2**	c3**	c0*	c1*	c2*	c3*
1111	1212	2222	0000	1111	1212	2222	3333
1101	1111	2332	0300	1001	1111	2332	0300
1111	2122	2332	3333	1101	2122	2232	3333
1010	1222	2222	0000	1010	1121	2222	0030

c0	c1	c2	c3	c4*	c5*	c6*	c7*
1010	1212	2222	3333	1010	2222	2222	3333
1001	1111	2322	3303	1011	1111	2322	3003
0100	2112	2232	3333	0100	2212	2232	3333
1000	1121	2222	3030	1101	2121	2222	3030

Tabela 3.27 – As tabelas das SFBs.

```
c0** = (//A b B/) a (A/ b /B) a (/A b /B) a 1
c1** = (/A c /B) b (A/ c /B) b (A c B/) b (/A c //B) b B
c2** = //A c A/ c //B c B/ c 3
c3** = (//A a //B) d /B d 0

c0*  = (A/ b B/) a (//A b B/) a (//A b //B) a (A/ b /B) a (/A b /B)
c1*  = (/A c /B) b (A/ c /B) b (A c B/) b (/A c //B) b (A c //B) b (//A c //B) b B
c2*  = (/A d B) c //A c A/ c //B c B/
c3*  = (//A a //B) d (/A a B) d B/ d /B

c0   = (A/ b B) a (/A b B) a (A/ b B/) a (//A b B/) a (A b //B) a (//A b //B) a (/A b //B) a
(A/ b /B) a (//A b /B) a (/A b /B)
c1   = (/A c /B) b (A/ c /B) b (A c B/) b (A/ c B/) b (/A c //B) b (A c //B) b (//A c //B) b B
c2   = (A d /B) c (/A d B) c //A c A/ c //B c B/
c3   = (//A a //B) d (A a //B) d (/A a B) d A/ d B/ d /B

c4*  = (A/ b B) a (/A b B) a (A/ b B/) a (A b //B) a (//A b //B) a (/A b //B) a (//A b /B)
c5*  = (A/ c B/) b (A c //B) b (//A c //B) b B
c6*  = (A d /B) c (/A d B) c //A c A/ c //B c B/
c7*  = (A a //B) d (/A a B) d A/ d B/ d /B

C1 = { [ (//A b B/) a (A/ b /B) a (/A b /B) a 1 ] c [ (/A c /B) b (A/ c /B) b (A c B/) b (/A
c //B) b (A c //B) b (//A c //B) b B ] } d [ (A d /B) c (/A d B) c //A c A/ c //B c B/ ]

C2 = { [ (/A c /B) b (A/ c /B) b (A c B/) b (/A c //B) b B ] d [ (/A d B) c //A c A/ c //B c
B/ ] } a [ (//A a //B) d (A a //B) d (/A a B) d A/ d B/ d /B ]

C3 = { [ //A c A/ c //B c B/ c 3 ] a [ (//A a //B) d (/A a B) d B/ d /B ] } b [ (A/ b B) a
(/A b B) a (A/ b B/) a (//A b B/) a (A b //B) a (//A b //B) a (/A b //B) a (A/ b /B) a (//A
b /B) a (/A b /B) ]

C4 = { [ (//A a //B) d /B d 0 ] b [ (A/ b B/) a (//A b B/) a (//A b //B) a (A/ b /B) a (/A b
/B) ] } c [ (/A c /B) b (A/ c /B) b (A c B/) b (A/ c B/) b (/A c //B) b (A c //B) b (//A c
//B) b B ]

C5 = { [ (//A b B/) a (A/ b /B) a (/A b /B) a 1 ] d [ (A d /B) c (/A d B) c //A c A/ c //B c
B/ ] } c [ (/A c /B) b (A/ c /B) b (A c B/) b (A/ c B/) b (/A c //B) b (A c //B) b (//A c
//B) b B ]

C6 = { [ (/A c /B) b (A/ c /B) b (A c B/) b (/A c //B) b B ] a [ (A a //B) d (/A a B) d A/ d
B/ d /B ] } d [ (A d /B) c (/A d B) c //A c A/ c //B c B/ ]

C7 = { [ //A c A/ c //B c B/ c 3 ] b [ (A/ b B) a (/A b B) a (A/ b B/) a (A b //B) a (//A b
//B) a (/A b //B) a (//A b /B) ] } a [ (//A a //B) d (A a //B) d (/A a B) d A/ d B/ d /B ]

C8 = { [ (//A a //B) d /B d 0 ] c [ (A/ c B/) b (A c //B) b (//A c //B) b B ] } b [ (A/ b B)
a (/A b B) a (A/ b B/) a (//A b B/) a (A b //B) a (//A b //B) a (/A b //B) a (A/ b /B) a
(//A b /B) a (/A b /B) ]
```

Este exemplo reflete o quão trabalhosa pode ser uma síntese quaternária.

Exemplo 4 (AaB):

C
0000
0111
0122
0123

c0**	c1**	c2**	c3**	c0	c1	c2	c3
0000	2122	3323	0000	0000	1111	2222	0000
0111	1111	3323	0000	0111	1111	2222	0333
0111	2122	2222	0000	0100	1122	2222	0333
0111	2122	3323	0003	0100	1121	2223	0333

c0*	c1*	c2*	c3*	c4*	c5*	c6*	c7*
0000	1111	3223	0000	1111	1111	2222	0000
0111	1111	2222	0000	1111	1222	2222	0333
0111	1122	2222	0033	1100	1222	2223	0333
0110	1122	3223	0033	1100	1221	2223	0330

Tabela 3.28 – Exemplo 4.

```
c0** = A a B a 1              c0*  = (/A b /B) a A a B a 1
c1** = A b B b 2              c1*  = /A b A b /B b B b 2
c2** = A c B c 3              c2*  = /A c A c /B c B c 3
c3** = (A a B) d 0            c3*  = (/A a /B) d (A a /B) d (/A a B) d (A a B) d 0

c0   = (//A b //B) a (/A b //B) a (//A b /B) a (/A b /B) a A a B
c1   = (//A c //B) b /A b A b /B b B
c2   = //A c /A c A c //B c /B c B
c3   = (//A a //B) d (/A a //B) d (A a //B) d (//A a /B) d (/A a /B) d (A a /B) d (//A a B)
d (/A a B) d (A a B)

c4*  = (//A b //B) a (/A b //B) a (//A b /B) a (/A b /B) a 1
c5*  = (//A c //B) b /A b /B b 2
c6*  = //A c /A c A c //B c /B
c7*  = (//A a //B) d (/A a //B) d (A a //B) d (//A a /B) d (/A a /B) d (A a /B) d (//A a B)
d (/A a B)
```

Para não deixar o texto extenso, é mostrada, apenas, a forma C1.

```
C1 = { [ A a B a 1 ] c [ /A b A b /B b B b 2 ] } d [ //A c /A c A c //B c /B c B ]
```

É preciso construir todas as expressões (C1, C2, C3, C4, C5, C6, C7 e C8) e escolher a menor delas. Trata-se de um trabalho extenso, sujeito a erros se feito manualmente. Uma ferramenta automatizada é recomendada para a realização dessa tarefa.

Com o conhecimento apresentado até este ponto, é possível sintetizar e minimizar qualquer função MVL. Porém, há casos nos quais algumas técnicas adicionais de minimização podem ser aplicadas; tais técnicas são introduzidas no capítulo 4.

Capítulo 4
Minimizações adicionais

4.1 Eliminação da conversão para binário – 2º critério

Os capítulos 4 e 5 consistem de novas descobertas que foram agregadas ao processo de síntese. A primeira versão do software LMVS aplica, somente, os conceitos apresentados nos capítulos 1, 2 e 3. A nova versão, além de conter aprimoramentos daqueles métodos, inclui, também, os conceitos apresentados nestes dois capítulos.

Para fins de formalização e evitar dificuldades de interpretação do texto, o termo "eliminação da conversão para binário MVL pelo 1º critério" é usado para indicar o processo de verificar se a SFB necessita da CPB para ser binária-MVL (em outras palavras, verificar se a remoção da CPB faz a SF deixar de ser binária-MVL), e, caso não seja necessário, omiti-la. Esse processo é abordado no capítulo 3. Se a verificação não for feita, uma omissão indevida da CPB poderá transformar a SFB numa função ternária ou quaternária.

O termo "eliminação da conversão para binário MVL pelo 2º critério" é usado para indicar o processo de remoção da CPB, fazendo a SF tornar-se ternária ou quaternária, sem, contudo, alterar a função MVL completa. Ao decompor a função MVL em SFBs, essas SFBs geralmente requerem o uso da CPB. Porém, em certos casos, a remoção da CPB (2º critério) não altera a expressão MVL, ou seja, é dispensável. Isso consiste numa ferramenta de minimização. A remoção da CPB pelo 2º critério consiste da comprovar se a CPB (2º critério), que já passou pelo 1º critério, pode ser omitida sem alterar a expressão MVL.

4.2 Eliminação da CPB, 2º critério – Três valores

Quanto à necessidade da CPB, tem-se quatro casos (tabela 4.1). No 1ª caso, a 1ª SFB não requer CPB nem a 2ª. No 2ª caso, a 1ª SFB não requer CPB mas a 2ª sim, e assim por diante. Na tabela, um valor falso indica que a SFB não requer CPB enquanto que um valor verdadeiro indica que ela requer. A quarta coluna indica a quantidade de CPB necessárias na função MVL completa. Na tabela 4.2, tem-se um exemplo de ocorrência de cada caso, para C_1. A mesma análise pode ser feita para os modos C_2 e C_3. A situação pode variar conforme o modo, como mostrado a tabela 4.3. Ainda não foi descoberto nenhum indício, na função MVL, que permita prever em qual dos quatro casos ela se enquadra.

	SFB		
Caso	**1ª**	**2ª**	**CPB**
1	Falso	Falso	0
2	Falso	Verdadeiro	1
3	Verdadeiro	Falso	1
4	Verdadeiro	Verdadeiro	2

Tabela 4.1 – Necessidade da CPB em lógica ternária (caso quaternário: tabela 4.23).

	1	**FF**
0	0	1
1	1	1
0	1	2

	2	**FV**
0	0	1
2	1	2
2	0	2

	3	**VF**
0	0	0
0	1	1
0	1	2

	4	**VV**
2	0	2
2	1	2
1	1	1

Tabela 4.2 – Ocorrências de cada tipo, para três valores (caso quaternário: tabela 4.24).

0	0	1
1	1	2
0	0	2

C_1: FF
C_2: FV
C_3: VF

Tabela 4.3 – Exemplo de diferenças entre as formas.

Com o conhecimento atual, a única maneira de fazer essa verificação é separando a função MVL em SFBs e analisá-las individualmente. Partindo-se de uma função MVL ternária (2 SFBs, capítulo 3) cuja 1ª SFB requeira a CPB, a remoção da CPB causará a formação de um valor indiferente (terciário) para a SF, e este valor é o dominante para o operador utilizado entre as duas SFs. Se as ocorrências do valor indiferente na 1ª SFB ocorrerem em posições irrelevantes (X), tal fato não causará erro na função MVL. Esta propriedade baseia-se no fato de que um valor irrelevante (X) pode ser substituído por qualquer valor dentro da base adotada e não, apenas, pelo verdadeiro ou pelo falso. A função MVL não se altera com essas substituições porque tais valores são operados com valores (na 2ª SFB) que são dominantes, para o operador entre as SFBs. Aliás, essa é a origem das irrelevâncias na 1ª SFB, explicada detalhadamente no capítulo 3. Se um valor irrelevante for substituído por um valor diferente do verdadeiro e do falso, a sub-função obtida não será binária-MVL.

Uma maneira de indicar os casos é mencionar se cada uma das três SFBs (c0, c1 e c2) empregadas nas três formas (C_1, C_2 e C_3) utiliza (V = verdadeiro) ou não (F = falso) CPB antes e depois da remoção da CPB (2º critério) (tabela 4.4); o ganho é a diferença entre a quantidade de CPB encontradas na função MVL completa antes e depois da análise de remoção da CPB (2º critério). Esta tabela permite observar se o 2º critério surtiu algum efeito na minimização. Os casos 3, 8 e 9 (tabela 4.4) não existem porque uma CPB (2º critério) na 2ª SFB não pode ser eliminada, pois não há valores irrelevantes. O ganho pode variar de acordo com a forma adotada (C_1, C_2 e C_3), como mostrado nos exemplos da tabela 4.5.

Caso	Antes	Depois	Ganho
1	FF	FF	0
2	FV	FV	0
3	FV	FF	1
4	VF	VF	0
5	VF	FF	1
6	VV	VV	0
7	VV	FV	1
8	VV	VF	1
9	VV	FF	2

Tabela 4.4 – Possibilidades de simplificação.

	1	**000**
1	0	2
2	2	1
1	2	1

	2	**100**
1	0	2
1	1	2
2	2	0

	3	**110**
2	2	1
2	2	2
1	2	0

	4	**111**
2	0	1
2	0	1
1	1	1

Tabela 4.5 – Exemplo de ocorrências de cada combinação de ganhos entre C_1, C_2 e C_3, respectivamene.

Deslocando-se os valores para a esquerda ou para a direita, obtém-se, também, outras variações. O processo de remoção da CPB (2º critério) em lógica ternária é melhor compreendido partindo-se de um exemplo.

	Antes		Depois		
Exemplo	**Caso**	**CPB**	**Caso**	**CPB**	**Ganho**
1	FF	0	FF	0	0
2	FV	1	FV	1	0
3	VF	1	FF	0	1
4	VF	1	VF	1	0
5	VV	2	FV	1	1
6	VV	2	FF	0	2

Tabela 4.6 – Os exemplos.

São mostrados exemplos para os casos da tabela 4.6. A coluna "antes" representa a situação antes da análise da remoção da CPB (2º critério); a coluna "depois" mostra o resultado. A coluna "ganho" mostra quantas CPB foram removidas pelo 2º critério.

Exemplo 1: FF→FF

Neste exemplo, nenhuma das duas SFBs necessita da CPB, a análise da remoção da CPB (2º critério) não pode ser realizada.

0	1	1
1	1	1
2	0	0

Tabela 4.7 – Exemplo 1

```
C1 = [ (A b B) a /B ] c [ A b A/ b /B b B ]
```

0	1	1		0	1	1		1	1	1
1	1	1	=	1	1	1	c	1	1	1
2	0	0		0	0	0		2	1	1

Tabela 4.8 – Método convencional.

Exemplo 2: FV→FV

Quando a CPB estiver na 2ª SFB (que não possui valores irrelevantes), a CPB não pode ser removida, pois o valor indiferente (terciário) ocorrerá em posição relevante, gerando erro na função MVL completa.

2	0	2
2	1	2
1	0	0

Tabela 4.9 – Exemplo 2.

```
C1 = [ (A/ b /B) a /A a B ] c [ A b B/ b 2 ]
```

2	0	2		0	0	0		2	X=1	2
2	1	2	=	1	1	0	c	2	1	2
1	0	0		1	0	0		1	X=1	X=1

Tabela 4.10 – Método convencional.

Com base na proposição feita no exemplo 2, pode-se intuir que algumas transformações, por meio da remoção da CPB (2º critério) sejam impossíveis.

Antes		Depois		
Caso	**CPB**	**Caso**	**CPB**	**Ganho**
FV	1	FF	0	1
VV	2	VF	1	1
VV	2	FF	0	2

Tabela 4.11– Casos impossíveis.

Exemplo 3: VF→FF

A remoção da CPB da 1ª SFB cria um valor indiferente; mas esta ocorrência está em uma posição irrelevante, não alterando a função MVL C. Para o exemplo, o valor indiferente gerado é “2”, a função MVL C_1 não muda porque `Xc2=2`.

1	2	2
0	2	2
2	2	0

Tabela 4.12 – Exemplo 3.

`C1 = [ /A a B/ a 1 ] c [ (/A c /B) b (/A c B) b (A/ c B/) ]`

1	2	2		1	X=1	X=0		1	2	2
0	2	2	=	0	X=0	X=0	c	1	2	2
2	2	0		X=1	X=1	0		2	2	1

Tabela 4.13 – Método convencional.

`C1 = [ /A a B/ ] c [ (/A c /B) b (/A c B) b (A/ c B/) ]`

1	2	2		1	X=2	X=0		1	2	2
0	2	2	=	0	X=0	X=0	c	1	2	2
2	2	0		X=1	X=1	0		2	2	1

Tabela 4.14 – Remoção da CPB da 1ª SFB.

Exemplo 4: VF→VF

Caso antagônico ao do exemplo 3. Devido à remoção da CPB da 1ª SFB, surge uma ocorrência do valor indiferente em posição relevante, gerando erro. A CPB da 1ª SFB não pode ser removida. Para o exemplo, o valor indiferente gerado é “2”.

`C1 = [ (/A b B) a (/A b /B) a 1 ] c [ (/A c /B) b (/A c B) b (A c B/) b A/ ]`

1	2	0		1	X=1	0		1	2	1
1	2	1	=	1	X=1	1	c	1	2	1
2	1	0		X=1	1	0		2	1	1

Tabela 4.15 – Método convencional.

`C1 = [ (/A b B) a (/A b /B) ] c [ (/A c /B) b (/A c B) b (A c B/) b A/ ]`

1	2	0		1	X=1	0		1	2	1
1	2	1	≠	1	X=1	1	c	1	2	1
2	1	0		X=1	2	0		2	1	1

Tabela 4.16 – Remoção da CPB da 1ª SFB.

Exemplo 5: VV→FV

As ocorrências do valor indiferente são todas em posições irrelevantes, não alterando a função MVL C porque **`Xc2=2`**. Para o exemplo, valor indiferente gerado é “2”.

`C1 = [ /B a 1 ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`

2	1	1		X=1	1	1		2	1	1
2	2	2	=	X=1	X=1	X=1	c	2	2	2
2	2	0		X=0	X=0	0		2	2	1

Tabela 4.17 – Método convencional.

`C1 = [ /B ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`

2	1	1		X=1	1	1		2	1	1
2	2	2	=	X=2	X=2	X=2	c	2	2	2
2	2	0		X=0	X=0	0		2	2	1

Tabela 4.18 – Remoção da CPB da 1ª SFB.

Exemplo 6: VV→VV

Caso antagônico ao do exemplo 5. Há uma ocorrências do valor indiferente em posição relevantes, gerando erro. Nenhuma CPB pode ser removida. Para o exemplo, valor indiferente gerado é “2”.

`C1 = [ A a 1 ] c [ (/A c /B) b (A/ c /B) b 2 ]`

0	2	1		0	X=1	1		1	2	1
2	2	2	=	X=0	X=1	X=1	c	2	2	2
2	2	2		X=0	X=1	X=1		2	2	2

Tabela 4.19 – Método convencional.

`C1 = [ A ] c [ (/A c /B) b (A/ c /B) b 2 ]`

0	2	1		0	X=1	2		1	2	1
2	2	2	≠	X=0	X=1	X=2	c	2	2	2
2	2	2		X=0	X=1	X=2		2	2	2

Tabela 4.20 – Remoção da CPB da 1ª SFB.

Exemplo 7: Substituição indevida (erro)

Neste exemplo, é mostrado como um erro pode surgir se as regras de remoção da CPB pelo 2º critério forem desobedecidas. A remoção da CPB da 1ª SFB gera um resultado errado, mostrado na tabela 4.22. A causa do erro está no fato de que a remoção da CPB da 1ª SFB gera uma ocorrência do valor indiferente numa combinação de entrada para a qual o valor da 1ª SF é relevante, e, se não é irrelevante, é porque o respectivo valor na 2ª SF não é o dominante para o operador entre as expressões, mas, infelizmente, o valor dominante aparece na 1ª SF, gerando um valor errado na função MVL.

`C1 = [ (A/ b B) a (A b B/) a 1 ] c`
`[ (/A c /B) b (A c /B) b (/A c B) b (A/ c B) b (A c B/) ]`

1	0	2		1	0	X=1		1	1	2
0	2	1	=	0	X=1	1	c	1	2	1
2	1	2		X=1	1	X=1		2	1	1

Tabela 4.21 – Método convencional.

`C1 = [ (A/ b B) a (A b B/) ] c`
`[ (/A c /B) b (A c /B) b (/A c B) b (A/ c B) b (A c B/) ]`

2	0	2		2	0	X=1		1	1	2
0	2	1	=	0	X=1	1	c	1	2	1
2	1	2		X=1	1	X=1		2	1	1

Tabela 4.22 – Erro.

Uma vez que a análise da remoção da CPB pelo 2º critério envolve a verificação das irrelevâncias e que tal verificação envolve a determinação das ocorrências do valor dominante na 2ª SFB para o conectivo empregado entre as SFBs, o software LMVS realiza a análise por inspeção, ou seja, remove-se a 1ª CPB e constrói-se a tabela; se não houver erro, mantém-se a remoção. Os dois critérios de remoção da CPB são aplicados por inspeção.

4.3 Eliminação da CPB, 2º critério – Quatro valores

Quanto à necessidade da CPB, há oito possibilidades (tabela 4.23). Na tabela 4.24 tem-se um exemplo de ocorrência de cada tipo, para C_1; não foram encontrados, por inspeção, exemplos para os casos 4 e 8. A mesma análise pode ser feita para C_2, C_3, C_4, C_5, C_6, C_7 e C_8. SFBs que possuem dois valores irrelevantes (c0**, c1**, etc) têm maior chance de exigir CPB (1º critério) do que aquelas com apenas um valor irrelevante (c0*, c1*, etc), que, por sua vez, têm maior chance de exigir CPB (1º critério) do que aquelas sem valores irrelevantes (c0, c1, etc). Isso se deve ao fato de que o uso de valores irrelevantes propicia a formação de linhas e colunas completas, como foi visto no capítulo 2, e essa formação de linhas e colunas completas aumenta a probabilidade da expressão, sem CPB, fornecer três valores ou mais. São mostrados exemplos para doze casos (tabela 4.25), para C_1 (com exceção do último exemplo, que é para C5).

	SFB			
Caso	**1ª**	**2ª**	**3ª**	**CPB**
1	Falso	Falso	Falso	0
2	Falso	Falso	Verdadeiro	1
3	Falso	Verdadeiro	Falso	1
4	Falso	Verdadeiro	Verdadeiro	2
5	Verdadeiro	Falso	Falso	1
6	Verdadeiro	Falso	Verdadeiro	2
7	Verdadeiro	Verdadeiro	Falso	2
8	Verdadeiro	Verdadeiro	Verdadeiro	3

Tabela 4.23 – Necessidade da CPB em lógica quaternária (caso ternário: tabela 4.1).

1

0	0	1	0
2	1	1	2
1	3	1	0
2	2	0	3

2

0	1	2	0
3	0	3	3
0	0	3	3
2	0	0	0

3

0	0	0	1
2	3	1	1
2	2	3	1
2	0	3	3

5

0	0	0	0
0	1	1	1
0	1	2	3
0	1	3	3

6

3	3	0	3
2	2	1	2
1	2	1	1
3	3	1	3

7

0	0	0	0
0	1	1	1
0	1	2	2
0	1	2	3

Tabela 4.24 – Ocorrências de cada tipo, para quatro valores (caso ternário: tabela 4.2).

Exemplo	Caso	Exemplo	Caso	Exemplo	Caso	Exemplo	Caso
1	FFF→FFF	4	VFF→VFF	7	VVF→FFF	10	VVF→VVF
2	FFV→FFV	5	FVF→FFF	8	VVF→VFF	11	VFV→VFV
3	VFF→FFF	6	FVF→FVF	9	VVF→FVF	12	VVF→VVF

Tabela 4.25 – Os exemplos.

Exemplo 1: FF→FF

Nenhuma das três SFBs necessita da CPB, a análise da remoção da CPB (2º critério) não pode ser realizada (análogo ao exemplo 1 ternário, tabela 4.7).

```
C1 = { [ (A b //B) a B/ a /B ] c [ (/A c /B) b A b A/ b B/ b //B ] } d
[ (/A d //B) c (A/ d //B) c (A d B/) c //A c /B c B ]
```

1	1	3	2		1	1	1	1		1	1	1	2		2	2	3	2
2	0	0	2	= [	0	0	0	0	c	2	1	1	2	] d	2	2	2	2
0	1	1	1		0	1	1	1		1	1	1	1		2	2	2	2
0	3	0	3		0	0	0	0		1	1	1	1		2	3	2	3

Tabela 4.26 – Exemplo 1.

Exemplo 2: FV→FV

Sempre que a CPB estiver na 3ª SFB (que não possui irrelevantes), a CPB não pode ser removida, pois o valor indiferente (terciário) ocorrerá em posição relevante, gerando erro (análogo ao exemplo 2 ternário, tabela 4.9).

```
C1 = { [ A a /A a B/ a //B a /B ] c [ (/A c /B) b (A/ c //B) b A b //A b
B/ ] } d [ (//A d B) c /A c //B c B/ c 3 ]
```

0	1	2	0
3	0	3	3
0	0	3	3
2	0	0	0

Tabela 4.27 – Exemplo 2.

Os exemplos 3,4,5 e 6 formam um grupo distinto. Os dois primeiros atuam sobre a 1ª SFB, os dois últimos atuam sobre a 2ª SFB. Os ímpares mostram situações onde a remoção pode ser realizada, os pares mostram casos onde a remoção pode ser feita.

Exemplo 3: VFF→FFF

A remoção da CPB da 1ª SFB cria um valor indiferente; mas esta ocorrência está em uma posição irrelevante, não alterando a função MVL C. Para o exemplo, o valor indiferente gerado é "2", a função MVL C_1 não muda porque `Xd3=3` (análogo ao exemplo 3 ternário, tabela 4.11).

```
{ [ (//A b B/) a (A/ b /B) a (/A b /B) a 1 ] c [ (/A c /B) b (A/ c /B) b (A c B/) b (/A c
//B) b (A c //B) b (//A c //B) b B ] } d [ (A d /B) c (/A d B) c //A c A/ c //B c B/ ]
```

1	2	1	2		1	X=1	1	X=1		1	2	1	2		2	2	2	2
1	3	0	1	= [	1	X=1	0	1	c	1	X=1	1	1	] d	2	3	2	2
2	1	3	2		X=1	1	X=1	X=1		2	1	X=2	2		2	2	3	2
1	0	2	0		1	0	X=1	0		1	1	2	1		2	2	2	2

Tabela 4.28 – Método convencional.

```
C1 = { [ (//A b B/) a (A/ b /B) a (/A b /B) ] c [ (/A c /B) b (A/ c /B) b (A c B/) b (/A c
//B) b (A c //B) b (//A c //B) b B ] } d [ (A d /B) c (/A d B) c //A c A/ c //B c B/ ]
```

1	2	1	2		1	X=1	1	X=1		1	2	1	2		2	2	2	2
1	3	0	1	= [	1	X=2	0	1	c	1	X=1	1	1	] d	2	3	2	2
2	1	3	2		X=1	1	X=1	X=1		2	1	X=2	2		2	2	3	2
1	0	2	0		1	0	X=1	0		1	1	2	1		2	2	2	2

Tabela 4.29 – Remoção da CPB da 1ª SFB.

Exemplo 4: VFF→VFF

Caso antagônico ao do exemplo 3. Devido à remoção da CPB da 1ª SFB, surge uma ocorrência do valor indiferente em posição relevante, gerando erro. A CPB da 1ª SFB não pode ser removida (análogo ao exemplo 4 ternário, tabela 4.14).

```
C1 = { [ (A b B/) a (A/ b /B) a 1 ] c [ (A c /B) b (A/ c /B) b (A/ c B) b
(A c //B) b /A ] } d [ A c //B c /B c B/ ]
```

1	1	1	2		1	1	1	X=1		1	1	1	2		2	2	2	2
0	2	1	2	= [	0	X=1	1	X=1	c	1	2	1	2	] d	2	2	2	2
3	3	2	3		X=1	X=1	X=1	X=1		X=1	X=2	2	X=2		3	3	2	3
1	0	2	2		1	0	X=1	X=1		1	1	2	2		2	2	2	2

Tabela 4.30 – Método convencional.

```
C1 = { [ (A b B/) a (A/ b /B) ] c [ (A c /B) b (A/ c /B) b (A/ c B) b (A
c //B) b /A ] } d [ A c //B c /B c B/ ]
```

1	1	1	2		1	1	1	X=1		1	1	1	2		2	2	2	2
0	2	1	2	= [	0	X=1	1	X=2	c	1	2	1	2	] d	2	2	2	2
3	3	2	3		X=1	X=1	X=1	X=1		X=1	X=2	2	X=2		3	3	2	3
2	0	2	2		2	0	X=1	X=2		1	1	2	2		2	2	2	2

Tabela 4.31 – Remoção da CPB da 1ª SFB.

Exemplo 5: FVF→FFF

Este é um caso parecido com o do exemplo 3. A remoção da CPB da 2ª SFB cria um valor indiferente; mas esta ocorrência está em uma posição irrelevante, não alterando a função MVL C. Para o exemplo, o valor indiferente gerado é "3", a função MVL C_1 não muda porque `Xd3=3`.

```
C1 = { [ A/ a B/ a //B a /B ] c [ A b A/ b B b B/ b 2 ] } d [ (/A d //B)
c (A/ d //B) c (//A d B/) c A c /B c B ]
```

3	0	1	2		X=1	0	1	X=1		X=2	1	1	2		3	2	2	2
0	0	0	0	= [	0	0	0	0	c	1	1	1	1	] d	2	2	2	2
0	0	0	0		0	0	0	0		1	1	1	1		2	2	2	2
2	3	0	3		X=0	X=0	0	X=0		2	X=1	1	X=2		2	3	2	3

Tabela 4.32 – Método convencional.

```
C1 = { [ A/ a B/ a //B a /B ] c [ A b A/ b B b B/ ] } d [ (/A d //B) c
(A/ d //B) c (//A d B/) c A c /B c B ]
```

3	0	1	2		X=1	0	1	X=1		X=3	1	1	2		3	2	2	2
0	0	0	0	= [	0	0	0	0	c	1	1	1	1	] d	2	2	2	2
0	0	0	0		0	0	0	0		1	1	1	1		2	2	2	2
2	3	0	3		X=0	X=0	0	X=0		2	X=1	1	X=2		2	3	2	3

Tabela 4.33 – Remoção da CPB da 2ª SFB.

Exemplo 6: FVF→FVF

Caso antagônico ao do exemplo 5. Devido à remoção da CPB da 2ª SFB, surge uma ocorrência do valor indiferente em posição relevante, gerando erro. A CPB da 2ª SFB não pode ser removida.

```
C1 = { [ A a A/ a //A ] c [ A b B b B/ b 2 ] } d [ (A/ d B) c //B c /B ]
```

2	0	2	2		X=0	0	X=0	X=1		2	1	2	2		2	2	2	2
0	0	0	1	= [	0	0	0	1	c	1	1	1	1	] d	2	2	2	2
3	3	3	1		X=0	X=0	X=0	1		X=1	X=1	X=1	1		3	3	3	2
3	3	3	3		X=0	X=0	X=0	X=1		X=2	X=1	X=2	X=2		3	3	3	3

Tabela 4.34 – Método convencional.

```
C1 = { [ A a A/ a //A ] c [ A b B b B ] } d [ (A/ d B) c //B c /B ]
```

3	0	2	3		X=0	0	X=0	X=1		3	1	2	3		2	2	2	2
0	0	0	1	= [	0	0	0	1	c	1	1	1	1	] d	2	2	2	2
3	3	3	1		X=0	X=0	X=0	1		X=1	X=1	X=1	1		3	3	3	2
3	3	3	3		X=0	X=0	X=0	X=1		X=2	X=1	X=2	X=2		3	3	3	3

Tabela 4.35 – Remoção da CPB da 2ª SFB.

Os exemplos 7,8,9 e 10 formam outro grupo distinto, no qual as duas primeiras SFBs são analisadas. Os exemplos correspondem às quatro combinações de casos. Para o restante dos exemplos, a explicação é feita após a exibição das tabelas, com citações específicas.

Exemplo 7: VVF→FFF

```
C1 = { [ //A a /A a B/ a //B a 1 ] c [ A b A/ b /B b //B b 2 ] } d [ (//A
d //B) c (A/ d //B) c (/A d B) c A c /B c B/ ]
```

1	3	0	0		1	X=1	0	0		1	X=1	1	1		2	3	2	2
2	0	0	2	= [	X=0	0	0	X=0	c	2	1	1	2	] d	2	2	2	2
3	0	0	3		X=0	0	0	X=0		X=2	1	1	X=2		3	2	2	3
1	1	0	0		1	1	0	0		1	1	1	1		2	2	2	2

Tabela 4.36 – Método convencional.

```
c0** = //A a /A a B/ a //B
c1* = A b B b B/
```

1	X=2	0	0		1	X=1	1	1
X=0	0	0	X=0		2	1	1	2
X=0	0	0	X=0		X=3	1	1	X=2
1	1	0	0		1	1	1	1

Tabela 4.37 – Remoção das CPB.

Ambas remoções de CPB criam uma ocorrência de valor terciário (2 para c0** e 3 para c1*) em posições irrelevantes, não alterando a função MVL C_1, pois **`Xd3=3`**.

Exemplo 8: VVF→VFF

```
C1 = { [ //B a 1 ] c [ (A/ c B/) b A b /B b 2 ] } d [ (/A d //B) c (/A d
/B) c (A/ d /B) c (A/ d B) c (/A d B/) c A ]
```

3	1	1	3		X=1	1	1	X=1		X=1	1	1	X=1		3	2	2	3
3	1	2	2	= [	X=1	1	X=1	X=1	c	X=2	1	2	2	] d	3	2	2	2
3	3	0	2		X=0	X=0	0	X=0		X=2	1	1	2		3	3	2	2
3	1	2	3		X=1	1	X=1	X=1		X=2	1	2	X=2		3	2	2	3

Tabela 4.38 – Método convencional.

```
c0** = //B  ;      c1* = (A/ c B/) b A b /B
```

X=2	2	2	X=2		X=1	1	1	X=1
X=3	3	X=3	X=3		X=2	1	2	2
X=0	X=0	0	X=0		X=3	1	1	2
X=1	1	X=1	X=1		X=2	1	2	X=2

Tabela 4.39 – Remoção das CPB.

Devido à remoção da CPB da 1ª SFB, surgem valores terciários (2) e indiferentes (3) em posições relevantes, alterando a função MVL C_1, gerando erro. Devido à remoção da CPB da 2ª SFB, surgiu um valor terciário (3) em uma posição irrelevante, não alterando a função MVL C_1, pois **`Xd3=3`**. Somente a CPB da 2ª SF pode ser removida.

Exemplo 9: VVF→FVF

C1 = { [//A a /A a B/ a 1] c [(/A c /B) b B b B/ b 2] } d [(//A d //B) c (//A d /B) c (/A d B) c (A d B) c A/ c B/]

1	3	3	2		1	X=1	X=0	X=0		1	X=2	X=2	2		2	3	3	2
0	3	3	0	= [	0	X=0	X=0	0	c	1	X=1	X=1	1	] d	2	3	3	2
3	1	0	0		X=1	1	0	0		X=1	1	1	1		3	2	2	2
2	2	2	2		X=1	X=1	X=0	X=0		2	2	2	2		2	2	2	2

Tabela 4.40 – Método convencional.

c0 = //A a /A a B/ ;** **c1* = (/A c /B) b B b B/**

1	X=2	X=0	X=0		1	X=2	X=3	3
0	X=0	X=0	0		1	X=1	X=1	1
X=1	1	0	0		X=1	1	1	1
X=1	X=2	X=0	X=0		2	2	2	2

Tabela 4.41 – Remoção das CPB.

Devido à remoção da CPB da 1ª SFB, surgiu um valor terciário (2); mas esta ocorrência está em posição irrelevante, não alterando a função MVL C_1, pois **Xc2=2**. Devido à remoção da CPB (2º critério) da 2ª SFB, surgiu um valor terciário (3) em uma posição relevante, gerando erro. Somente a CPB da 1ª SF pode ser removida.

Exemplo 10: VVF→VVF

C1 = { [(//A b B/) a (A/ b //B) a 1] c [(A c B) b (A/ c B) b (//A c B) b (A c B/) b (A/ c B/) b 2] } d [/A c A c A/ c //B c /B c B/]

2	2	2	2		X=1	X=1	X=1	X=1		2	2	2	2		2	2	2	2
2	1	0	1	= [	X=1	1	0	1	c	2	1	1	1	] d	2	2	2	2
3	0	1	2		X=1	0	1	X=1		X=2	1	1	2		3	2	2	2
2	2	2	2		X=1	X=1	X=1	X=1		2	2	2	2		2	2	2	2

Tabela 4.42 – Método convencional.

c0 = (//A b B/) a (A/ b //B)**
c1* = (A c B) b (A/ c B) b (//A c B) b (A c B/) b (A/ c B/)

X=2	X=2	X=1	X=1		2	3	2	2
X=2	3	0	1		2	1	1	1
X=1	0	1	X=1		X=2	1	1	2
X=1	X=1	X=1	X=1		2	2	2	2

Tabela 4.43 – Remoção das CPB.

Devido à remoção da CPB da 1ª SFB, surgem valores terciários (2) e indiferentes (3), um deles em posição relevante, gerando erro. Devido à remoção da CPB (2º critério) da 2ª SFB, surgiu um valor terciário (3) em uma posição relevante gerando erro.

Exemplo 11: VFV→VFV

`C1 = { [ B a 1 ] c [ (/A c B/) b (//A c B/) b A/ ] } d [ A c /B c B c 3 ]`

3	3	0	3		X=0	X=0	0	X=0		X=2	X=2	1	X=2		3	3	2	3
2	2	1	2	= [	X=1	X=1	1	X=1	c	2	2	1	2	] d	2	2	2	2
1	2	1	1		1	X=1	1	1		1	2	1	1		2	2	2	2
3	3	1	3		X=1	X=1	1	X=1		X=2	X=2	1	X=2		3	3	2	3

Tabela 4.44 – Método convencional.

`C1 = { [ B ] c [ (/A c B/) b (//A c B/) b A/ ] } d [ A c /B c B c 3 ]`

3	3	0	3		X=0	X=0	0	X=0		X=2	X=2	1	X=2		3	3	2	3
2	2	1	2	= [	X=1	X=1	1	X=1	c	2	2	1	2	] d	2	2	2	2
2	2	2	2		2	X=2	2	2		1	2	1	1		2	2	2	2
3	3	3	3		X=3	X=3	3	X=3		X=2	X=2	1	X=2		3	3	2	3

Tabela 4.45 – Remoção da CPB da 1ª SFB.

A CPB da 3ª SFB não pode ser removida pois não há valor irrelevante. Devido à remoção da CPB da 1ª SF, surgem ocorrências do valor terciário (2) e do valor indiferente (3) em posições relevantes, gerando erro.

Exemplo 12: VVV→VVV

`C5 = { [ //A a 1 ] d [ A c //B c 3 ] } c [ (A c /B) b /A b //B b 2 ]`

1	1	2	2		1	1	X=0	X=1		2	2	X=2	X=2		1	1	2	2
3	2	2	2	= [	X=1	X=1	X=0	X=1	d	3	X=3	X=2	X=3	] c	1	2	2	2
3	2	2	2		X=1	X=1	X=0	X=1		3	X=3	X=2	X=3		1	2	2	2
3	3	0	3		X=1	X=1	0	X=1		3	3	2	3		1	1	1	1

Tabela 4.46 – Método convencional.

`c0** = //A`
`c6* = A c //B`

2	3	X=0	X=1		2	2	X=2	X=2
X=2	X=3	X=0	X=1		3	X=3	X=2	X=3
X=2	X=3	X=0	X=1		0	X=0	X=2	X=3
X=2	X=3	0	X=1		0	1	2	3

Tabela 4.47 – Remoção das CPB.

Ambas remoções de CPB criam uma ocorrência de valor terciário (2 para c0** e 3 para c6*) e indiferente (3 para c0** e 1 para c6*) em posições relevantes, gerando erro. A CPB da 3ª SFB não pode ser removida pois não há valor irrelevante. Nenhuma das três CPB pode ser removida. (análogo ao exemplo 6 ternário, tabela 4.18).

4.4 Outro tipo de remoção da CPB pelo 2º critério

Há um fenômeno que pode permitir a remoção da CPB (2º critério) mesmo quando tal remoção implique na formação de valores terciários ou indiferentes em posições relevantes, mostrando como esta análise pode se tornar complexa, motivo pelo qual se preferiu um algoritmo inspecional no software LMVS. Os exemplos a seguir ilustram esse fato.

Exemplo 1:

`C1 = { [ (A b B/) a (A/ b B/) a (A/ b //B) a 1 ] c [ A b B b B/ b 2 ] } d [ (A d /B) c //A c /A ]`

2	1	3	3		X=1	1	X=1	X=1		2	1	X=2	X=2		2	2	3	3
0	0	1	3	= [	0	0	1	X=1	c	1	1	1	X=1	] d	2	2	2	3
1	0	3	3		1	0	X=1	X=1		1	1	X=1	X=1		2	2	3	3
2	1	3	3		X=1	1	X=1	X=1		2	1	X=2	X=2		2	2	3	3

Tabela 4.48 – Método convencional.

As duas primeiras SFBs utilizam CPB. Ao remover a 1ª SFB, observa-se que tal remoção provoca o surgimento de valores terciários em posições irrelevantes, ou seja, tal remoção pode ser feita. Ao remover a 2ª SFB, observa-se que tal remoção provoca o surgimento de um valores terciário em posição relevante, ou seja, tal remoção não pode ser feita.

`C1 = { [ (A b B/) a (A/ b B/) a (A/ b //B)] c [ A b B b B/ b 2 ] } d [ (A d /B) c //A c /A ]`

2	1	3	3		X=2	1	X=1	X=2		2	1	X=2	X=2		2	2	3	3
0	0	1	3	= [	0	0	1	X=2	c	1	1	1	X=1	] d	2	2	2	3
1	0	3	3		1	0	X=1	X=1		1	1	X=1	X=1		2	2	3	3
2	1	3	3		X=1	1	X=1	X=1		2	1	X=2	X=2		2	2	3	3

Tabela 4.49 – Remoção da 1ª CPB.

`C1 = { [ (A b B/) a (A/ b B/) a (A/ b //B) a 1 ] c [ A b B b B/ ] } d [ (A d /B) c //A c /A ]`

3	1	3	3		X=1	1	X=1	X=1		3	1	X=2	X=3		2	2	3	3
0	0	1	3	= [	0	0	1	X=1	c	1	1	1	X=1	] d	2	2	2	3
1	0	3	3		1	0	X=1	X=1		1	1	X=1	X=1		2	2	3	3
2	1	3	3		X=1	1	X=1	X=1		2	1	X=2	X=2		2	2	3	3

Tabela 4.50 – Remoção da 2ª CPB.

Se a CPB da 1ª SFB for removida, ocorre o surgimento, na 1ª SFB, do valor dominante (2) para o operador "c", ou seja, o valor terciário (3) da 2ª SF (cuja CPB foi removida), ao ser operado com o valor 2 da 1ª SFB, volta a ser 2, não alterando a função MVL C1.

```
c0** = ( A b B/ ) a ( A/ b B/ ) a ( A/ b //B )
c1* = A b B b B/
```

X=2	1	X=1	X=2
0	0	1	X=2
1	0	X=1	X=1
X=1	1	X=1	X=1

c

X=3	1	X=2	X=3
1	1	1	X=1
1	1	X=1	X=1
2	1	X=2	X=2

Tabela 4.51 – Remoção das CPB.

A remoção na 1ª SF corrige o erro vindo da remoção na 2ª. A 1ª SF pode gerar valores terciários que anulem o erro causado por uma remoção indevida na 2ª SF (indevida porque gera valores terciários em posições relevantes). Todas as ocorrências do valor “3” na 2ª SF são, neste exemplo, eliminadas por meio da operação “c” com ocorrências do valor “2” na 1ª SF. Essa é uma técnica de minimização derivada da remoção da CPB (2º critério).

Exemplo 2:

```
C1 = { [ /A a B a 1 ] c [ (A c /B) b (A/ c /B) b (A c B) b (//A c B) b 2 ] } d [
(//A d //B) c (A d //B) c (//A d /B) c (A/ d /B) c (A d B/) c (A/ d B/) c /A]
```

2	0	0	3
2	1	3	0
3	2	3	3
3	2	2	2

= [

X=0	0	0	X=0
X=1	1	X=1	0
X=1	X=1	X=1	X=0
X=1	X=1	X=1	X=0

c

2	1	1	X=2
2	1	X=2	1
X=2	2	X=2	X=2
X=2	2	2	2

] d

2	2	2	3
2	2	3	2
3	2	3	3
3	2	2	2

Tabela 4.52 – Método convencional.

```
c0** = /A a B
c1* = (A c /B) b (A/ c /B) b (A c B) b (//A c B)
```

X=0	0	0	X=0
X=1	1	X=1	0
X=1	X=2	X=2	X=0
X=1	X=2	X=3	X=0

c

2	1	1	X=2
2	1	X=2	1
X=2	X=2	X=2	X=2
X=2	X=3	2	2

Tabela 4.53 – Remoção das CPB.

Pelo mesmo motivo do exemplo 1, se as duas CPB forem removidas simultaneamente, a função MVL C_1 não é alterada.

Fica difícil, entretanto, prever quando esse fato ocorre. Pode ser, também, que existam outros fenômenos, ainda não descobertos, que permitam remoções especiais de CPB. Isso mostra como a realização manual da síntese pode ser um trabalho difícil e demorado, é preciso fazer inúmeras análises. Uma ferramenta automatizada que realize as remoções das CPB por inspeção é bastante desejável. No tópico 4.5, a seguir, é introduzido o conceito de irrelevâncias móveis, que não devem ser confundidas com o caso desses dois exemplos.

4.5 Valores irrelevantes móveis – Três valores

Os valores irrelevantes móveis são uma conseqüência da remoção da CPB (2º critério). Aquele processo consiste numa etapa de minimização da expressão, mas os valores irrelevantes móveis também geram uma eventual etapa na minimização. Pelo mesmo motivo pelo qual as ocorrências do valor dominante (para o operador utilizado entre as SFB) encontradas na 2ª SFB levam á formação de irrelevâncias na 1ª SFB, essas criações do valor dominante (para o operador utilizado entre as SFB) devido à remoção da CPB (2º critério) na 1ª SF também levam á formação de irrelevâncias na 2ª SF. Esse fenômeno é melhor explicado por meio de exemplos.

O processo pode ser resumido em três etapas:

1. Síntese pelo método convencional.
2. Remoção da CPB (2º critério) da 1ª SFB.
3. Substituição dos valores irrelevantes móveis da 2ª SFB.

Exemplo 1:

Neste exemplo, por causa da substituição de um valor irrelevante pelo valor indiferente na 1ª SF, surgiu um valor irrelevante na segunda SF, pois o valor 2, na combinação (a=2,b=2), é o dominante para o operador “c”, utilizado entre as duas SFs; sendo assim, o valor da posição (2,2) na segunda SF torna-se irrelevante

`C1 = [ A a B a 1 ] c [ /A b A b B ]`

0	0	2		0	0	X=0		1	1	2
0	1	1	=	0	1	1	c	1	1	1
0	1	2		0	1	X=1		1	1	2

Tabela 4.54 – 1ª etapa.

`C1 = [ A a B ] c [ /A b A b B ]`

0	0	2		0	0	X=0		1	1	2
0	1	1	=	0	1	1	c	1	1	1
0	1	2		0	1	X=2		1	1	2

Tabela 4.55 – 2ª etapa.

`C1 = [ A a B ] c [ /A b A b B ]`

0	0	2		0	0	X=0		1	1	2
0	1	1	=	0	1	1	c	1	1	1
0	1	2		0	1	X=2		1	1	X=2

Tabela 4.56 – 3ª etapa.

Neste exemplo, a formação de um valor irrelevante na 2ª SF não diminuiu a expressão final, pois qualquer outra substituição do valor irrelevante leva a uma minimização menos eficiente. Os dois exemplos seguintes são casos de diminuição.

Exemplo 2:

Neste exemplo e no próximo, a remoção da CPB da 1ª SF pelo 2º critério forma uma linha ou coluna de valores irrelevantes na 2ª SF. Uma das três células irrelevantes pode ser substituída pelo valor dominante da SF, a fim de formar uma linha ou coluna completa.

`C1 = [ /B a 1 ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`

2	1	1		X=1	1	1		2	1	1
2	2	2	=	X=1	X=1	X=1	c	2	2	2
2	2	0		X=0	X=0	0		2	2	1

Tabela 4.57 – 1ª etapa.

`C1 = [ /B ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`

2	1	1		X=1	1	1		2	1	1
2	2	2	=	X=2	X=2	X=2	c	2	2	2
2	2	0		X=0	X=0	0		2	2	1

Tabela 4.58 – 2ª etapa.

`C1 = [ /B ] c [ (A c /B) b A/ b 2 ]`

2	1	1		1	1	1		2	1	1
2	2	2	=	2	2	2	c	X=2	X=2	X=1
2	2	0		0	0	0		2	2	1

Tabela 4.59 – 3ª etapa.

1. `C1 = [ /B a 1 ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`
2. `C1 = [ /B ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`
3. `C1 = [ /B ] c [ (A c /B) b A/ b 2 ]`

Exemplo 3:

`C1 = [ A/ a 1 ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`

2	0	1		X=1	0	1		2	1	1
2	2	2	=	X=1	X=0	X=1	c	2	2	2
2	2	1		X=1	X=0	1		2	2	1

Tabela 4.60 – 1ª etapa.

`C1 = [ A/ ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`

2	0	1		X=2	0	1		2	1	1
2	2	2	=	X=2	X=0	X=1	c	2	2	2
2	2	1		X=2	X=0	1		2	2	1

Tabela 4.61 – 2ª etapa.

`C1 = [ A/ ] c [ (A/ c B/) b /B b 2 ]`

2	0	1		2	0	1		X=1	1	1
2	2	2	=	2	0	1	c	X=2	2	2
2	2	1		2	0	1		X=2	2	1

Tabela 4.62 – 3ª etapa.

1. `C1 = A/ a 1 ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`
2. `C1 = [ A/ ] c [ (A c /B) b (A/ c /B) b (A/ c B/) b 2 ]`
3. `C1 = [ A/ ] c [ (A/ c B/) b /B b 2 ]`

Sempre que a remoção da CPB (2º critério) da 1ª SF levar a uma expressão do tipo "A", "B" e seus deslocamentos TOPO e BASE, tal remoção causará o surgimento de três ocorrências do valor irrelevante na 2ª SF.

1. A 2ª SFB gera ocorrências de valores irrelevantes na 1ª SFB.
2. A remoção da CPB na 1ª SFB gera um valor terciário em posição irrelevante.
3. O valor terciário na 1ª SF gera um valor irrelevante na 2ª SF.

Nem sempre os valores irrelevantes móveis surgem sob a situação apresentada acima, quando se formam linha ou coluna completa. Os dois próximos exemplos ilustram casos diferentes, onde se tem células isoladas.

Exemplo 4: C = (A a B)

Neste exemplo, surge, apenas, uma célula irrelevante na 2ª SF.

`C1 = [ A a B a 1 ] c [ /A b A b /B b B ]` (método tradicional).

`C1 = [ A a B ] c [ /A b A b /B b B ]`

0	0	0		0	0	0		1	1	1
0	1	1	=	0	1	1	c	1	1	1
0	1	2		0	1	X=2		1	1	2

Tabela 4.63 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

`C1 = [ A a B ] c [ 1 ]` → `C1 = [ A a B ]`

0	0	0		0	0	0		1	1	1
0	1	1	=	0	1	1	c	1	1	1
0	1	2		0	1	2		1	1	X=1

Tabela 4.64 – Substituição dos valores irrelevantes móveis da 2ª SF (3ª e 4ª etapas).

Exemplo 5: C = (A b B)

Neste exemplo, surgem três células irrelevante na 2ª SF não alinhadas.

`C1 = [ (A b B) a 1 ] c [ (/A c /B) b A b B ]` (método tradicional).

`C1 = [ A b B ] c [ (/A c /B) b A b B ]`

0	1	2		0	1	X=2		1	1	2
1	1	1	=	1	1	1	c	1	1	1
2	1	2		X=2	1	X=2		2	1	2

Tabela 4.65 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

`C1 = [ A b B ] c [ 1 ]` → `C1 = [ A b B ]`

0	1	2		0	1	2		1	1	X=1
1	1	1	=	1	1	1	c	1	1	1
2	1	2		2	1	2		X=1	1	X=1

Tabela 4.66 – Substituição dos valores irrelevantes móveis da 2ª SF (3ª e 4ª etapas).

Etapa	Exemplo 4: (A a B)	Exemplo 5: (A b B)
1	`[ A a B a 1 ] c [ /A b A b /B b B ]`	`[ (A b B) a 1 ] c [ (/A c /B) b A b B ]`
2	`[ A a B ] c [ /A b A b /B b B ]`	`[ A b B ] c [ (/A c /B) b A b B ]`
3	`[ A a B ] c [ 1 ]`	`[ A b B ] c [ 1 ]`
4	`[ A a B ]`	`[ A b B ]`

Tabela 4.67 – Comparação entre os exemplos.

<u>Exemplo 6:</u> C = (A c B)

Neste exemplo, o valor irrelevante móvel na 2ª SF não muda (X=2), pois, se mudar para “1”, aumentará o tamanho da expressão. A técnica do uso dos valores irrelevantes móveis, no modo C_1, permite a obtenção das expressões `C1 = [ A a B ]` e `C1 = [ A b B ]`, mas não permite a obtenção de `C1 = [ A c B ]`.

`C1 = [ A a B a 1 ] c [ (/A c /B) b (A c /B) b (/A c B) b (A c B) ]` (método tradicional).

`C1 = [ A a B ] c [ (/A c /B) b (A c /B) b (/A c B) b (A c B) ]`

0	0	2		0	0	X=0		1	1	2
0	1	2	=	0	1	X=1	c	1	1	2
2	2	2		X=0	X=1	X=2		2	2	2

Tabela 4.68 – Remoção da CPB (2º critério) da 1ª SF.

<u>Exemplo 7:</u> B = (A)

Os valores irrelevantes móveis também podem surgir em funções de uma entrada.

0	1	2

Tabela 4.69 – Exemplo 7 (uma entrada, o resultado deve ser a própria função inicial).

`B1 = [ A a 1 ] c [ /A b A ]`	0	1	2	=	0	1	X=1	c	1	1	2
`B2 = [ A b 2 ] a [ /A c A ]`	0	1	2	=	X=2	1	2	a	0	2	2
`B3 = [ A c 0 ] b [ A a /A ]`	0	1	2	=	0	X=0	2	b	0	1	0

Tabela 4.70 – Método convencional (1ª etapa).

`B1 = [ A ] c [ /A b A ]`	0	1	2	=	0	1	X=2	c	1	1	2
`B2 = [ A ] a [ /A c A ]`	0	1	2	=	X=0	1	2	a	0	2	2
`B3 = [ A ] b [ A a /A ]`	0	1	2	=	0	X=1	2	b	0	1	0

Tabela 4.71 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

B1 = [A] c [1]	0	1	2	=	0	1	2	c	1	1	X=1
B2 = [A] a [2]	0	1	2	=	0	1	2	a	X=2	2	2
B3 = [A] b [0]	0	1	2	=	0	1	2	b	0	X=0	0

Tabela 4.72 – Substituição dos valores irrelevantes móveis da 2ª SF (3ª etapa).

Com base na propriedade da operação com o valor indiferente (fórmulas (21) do capítulo 1), as expressões tornam-se:

B1 = [A] ; B2 = [A] ; B3 = [A]

O processo é composto de quatro etapas:

1. Método convencional.
2. Remoção da CPB (2º critério) da 1ª SFB.
3. Substituição dos valores irrelevantes móveis da 2ª SF.
4. Verificação da possibilidade de eliminação da 2ª SF.

	1	2	3	4
B1	**[A a 1] c [/A b A]**	**[A] c [/A b A]**	**[A] c [1]**	**A**
B2	**[A b 2] a [/A c A]**	**[A] a [/A c A]**	**[A] a [2]**	**A**
B3	**[A c 0] b [A a /A]**	**[A] b [A a /A]**	**[A] b [0]**	**A**

Tabela 4.73 – As quatro etapas do processo.

4.6 Valores irrelevantes móveis – Quatro valores

Para lógica quaternária, podem surgir valores irrelevantes móveis na 2ª e 3ª SF. Primeiramente, é apresentado um exemplo didático para uma entrada.

Exemplo 1: B = A

0	1	2	3

Tabela 4.74 – Exemplo 1 (uma entrada, o resultado deve ser a própria função inicial).

B1 = { [A a 1] c [/A b A b 2] } d [//A c /A c A]
B2 = { [A b 2] d [/A c A c 3] } a [//A d /A d A]
B3 = { [A c 3] a [/A d A d 0] } b [A a //A a /A]
B4 = { [A d 0] b [A a /A a 1] } c [/A b A b //A]

0	1	2	3	= (	0	1	X=1	X=1	c	1	1	2	X=2	) d	2	2	2	3
0	1	2	3	= (	X=2	1	2	X=2	d	X=3	2	2	3	) a	0	3	3	3
0	1	2	3	= (	X=3	X=3	2	3	a	0	X=0	3	3	) b	0	1	0	0
0	1	2	3	= (	0	X=0	X=0	3	b	0	1	X=1	0	) c	1	1	2	1

Tabela 4.75 – Método convencional (1ª etapa).

0	1	2	3	=(	0	1	X=2	X=3	c	1	1	X=2	X=2	)d	2	2	2	3
0	1	2	3	=(	X=0	1	2	X=3	d	X=3	2	2	X=3	)a	0	3	3	3
0	1	2	3	=(	X=0	X=1	2	3	a	X=0	X=0	3	3	)b	0	1	0	0
0	1	2	3	(	0	X=1	X=2	3	b	0	X=1	X=1	0	)c	1	1	2	1

Tabela 4.76 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

0	1	2	3	= (	0	1	X=2	X=3	c	1	1	X=2	X=3	) d	2	2	2	X=3
0	1	2	3	= (	X=0	1	2	X=3	d	X=0	2	2	X=3	) a	X=0	3	3	3
0	1	2	3	= (	X=0	X=1	2	3	a	X=0	X=1	3	3	) b	0	X=1	0	0
0	1	2	3	= (	0	X=1	X=2	3	b	0	X=1	X=2	0	) c	1	1	X=2	1

Tabela 4.77 – Remoção da CPB (2º critério) da 2ª SFB (3ª etapa).

0	1	2	3	= (	0	1	X=2	X=3	c	1	1	X=1	X=3	) d	2	2	2	X=2
0	1	2	3	= (	X=0	1	2	X=3	d	X=0	2	2	X=2	) a	X=3	3	3	3
0	1	2	3	= (	X=0	X=1	2	3	a	X=3	X=1	3	3	) b	0	X=0	0	0
0	1	2	3	= (	0	X=1	X=2	3	b	0	X=0	X=2	0	) c	1	1	X=1	1

Tabela 4.78 – Substituição dos valores irrelevantes móveis (4ª etapa).

```
B1 = { [ A ] c [ 1 ] } d [ 2 ]          B2 = { [ A ] d [ 2 ] } a [ 3 ]
B3 = { [ A ] a [ 3 ] } b [ 0 ]          B4 = { [ A ] b [ 0 ] } c [ 1 ]
```

0	1	2	3	= (	0	1	X=2	X=3	c	1	1	X=1	X=1	) d	2	2	2	X=2
0	1	2	3	= (	X=0	1	2	X=3	d	X=2	2	2	X=2	) a	X=3	3	3	3
0	1	2	3	= (	X=0	X=1	2	3	a	X=3	X=3	3	3	) b	0	X=0	0	0
0	1	2	3	= (	0	X=1	X=2	3	b	0	X=0	X=0	0	) c	1	1	X=1	1

Tabela 4.79 – Substituição do valor irrelevante fixo da 2ª SF (5ª etapa).

Com base na propriedade da operação com o indiferente, as expressões tornam-se:

`B1 = [ A ];  B2 = [ A ]; B3 = [ A ]; B3 = [ A ]` (6ª etapa).

Exemplo 2: C = (A a B) – forma C_1

Os valores terciários (2) e indiferentes (3), na 1ª SF, estão em posições irrelevantes fixas, a 1ª CPB. pode ser removida (tabela 4.78). O valor terciário (3), na 2ª SF, está em posição irrelevante fixa, a 2ª CPB pode ser removida (tabela 4.79). O valor irrelevante móvel na 3ª SF surge do fato de que, nas duas primeiras SFs, os valores na posição (3,3) são substituídos pelo dominante do operador "d", "3" (tabela 4.80).

```
C1 = { [ A a B a 1 ] c [ /A b A b /B b B b 2 ] } d [ //A c /A c A c //B c /B c B ]
```

0	0	0	0		0	0	0	0		1	1	1	1		2	2	2	2
0	1	1	1	= (	0	1	1	1	c	1	1	1	1	) d	2	2	2	2
0	1	2	2		0	1	X=1	X=1		1	1	2	2		2	2	2	2
0	1	2	3		0	1	X=1	X=1		1	1	2	X=2		2	2	2	3

Tabela 4.80 – Método convencional (1ª etapa).

```
C1 = { [ A a B ] c [ /A b A b /B b B b 2 ] } d [ //A c /A c A c //B c /B c B ]
```

0	0	0	0		0	0	0	0		1	1	1	1		2	2	2	2
0	1	1	1	= (	0	1	1	1	c	1	1	1	1	) d	2	2	2	2
0	1	2	2		0	1	X=2	X=2		1	1	X=2	X=2		2	2	2	2
0	1	2	3		0	1	X=2	X=3		1	1	X=2	X=2		2	2	2	3

Tabela 4.81 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

`C1 = { [ A a B ] c [ /A b A b /B b B ] } d [ //A c /A c A c //B c /B c B ]`

0	0	0	0		0	0	0	0		1	1	1	1		2	2	2	2
0	1	1	1	= (	0	1	1	1	c	1	1	1	1	) d	2	2	2	2
0	1	2	2		0	1	X=2	X=2		1	1	X=2	X=2		2	2	2	2
0	1	2	3		0	1	X=2	X=3		1	1	X=2	X=3		2	2	2	X=3

Tabela 4.82 – Remoção da CPB (2º critério) da 2ª SFB (3ª etapa).

`C1 = { [ A a B ] c [ ... ] } d [ 2 ]`

0	0	0	0		0	0	0	0		1	1	1	1		2	2	2	2
0	1	1	1	= (	0	1	1	1	c	1	1	1	1	) d	2	2	2	2
0	1	2	2		0	1	2	2		1	1	X=1	X=1		2	2	2	2
0	1	2	3		0	1	2	3		1	1	X=1	X=3		2	2	2	X=2

Tabela 4.83 – Substituição dos valores irrelevantes móveis (4ª etapa).

Com a substituição X=2 na posição (3,3) da 3ª SF, o valor "3" da 2ª SF deixa de ser irrelevante (não pode ser substituído por "2", que é o dominante de "c"), mas pode ser substituído por "1" que é o indiferente de "c". Trata-se de uma <u>irrelevância parcial</u>. Essa possibilidade vem da percepção, é intuitiva. A princípio, se algum valor irrelevante fixo é usado na geração de um valor irrelevante móvel em outra SF e esse irrelevante móvel sofre alguma substituição, o irrelevante fixo não pode mais ser alterado, pois isso pode implicar na eliminação da irrelevância móvel e, por causa da substituição, haver erro. Neste exemplo, porém, esse problema não acontece e a mudança pode ser aplicada.

`C1 = { [ A a B ] c [ 1 ] } d [ 2 ]`

0	0	0	0		0	0	0	0		1	1	1	1		2	2	2	2
0	1	1	1	= (	0	1	1	1	c	1	1	1	1	) d	2	2	2	2
0	1	2	2		0	1	2	2		1	1	X=1	X=1		2	2	2	2
0	1	2	3		0	1	2	3		1	1	X=1	X=1		2	2	2	X=2

Tabela 4.84 – Substituição do valor irrelevante fixo da 2ª SF (5ª etapa).

O valor "1" é o indiferente do operador "c", o valor "2" é o indiferente do operador "d", a expressão torna-se: `C1 = [ A a B ]`.

Dois fenômenos independentes podem ser observados:

1. A remoção da CPB (2º critério) da 1ª SFB (`a1`) gera ocorrências do valor terciário "2", e essas ocorrências geram a formação de irrelevâncias móveis na 2ª SFB.
2. A remoção da CPB (2º critério) da 2ª SFB (`b2`) gera ocorrências do valor terciário "3", e essas ocorrências geram a formação de irrelevâncias móveis na 3ª SFB.

Os valores irrelevantes móveis só ocorrem na 2ª e na 3ª SF. Na 2ª SF, eles surgem com a remoção da CPB (2º critério) da 1ª SF; na 3ª SF, eles surgem com a remoção da CPB (2º critério) da 2ª SF.

O processo é composto de seis etapas.

1. Método convencional.
2. Remoção da CPB (2º critério) da 1ª SFB.
3. Remoção da CPB (2º critério) da 2ª SFB.
4. Substituição dos valores irrelevantes móveis.
5. Substituição do valor irrelevante fixo da 3ª SF.
6. Verificação da possibilidade de remoção da 2ª e 3ª SF.

Exemplo 3: C = A a B – forma C_4

Os valores terciários (1) e indiferentes (2), na 1ª SF, aparecem em posições irrelevantes fixas, a CPB pode ser removida; os valores terciários (2), na 2ª SF, aparecem em posições irrelevantes fixas, a CPB pode ser removida. Na tabela 4.85, tem-se irrelevâncias parciais na 2ª SF nas posições (2,2),(2,3) e (3,2). O valor “2”, apesar de não ser irrelevante (não pode ser substituído por “1”, que é o dominante para o operador “b”), pode ser substituído por “0” pois esse é o valor indiferente de “b”.

C4 = { [(A a B) d 0] b [(/A b /B) a A a B a 1] } c [(//A c //B) b /A b A b /B b B]

0	0	0	0		0	0	0	0		0	0	0	0		1	1	1	1
0	1	1	1	= (	0	X=0	X=0	X=0	b	0	1	1	1	) c	1	1	1	1
0	1	2	2		0	X=0	X=0	X=0		0	1	X=1	X=1		1	1	2	2
0	1	2	3		0	X=0	X=0	3		0	1	X=1	0		1	1	2	1

Tabela 4.85 – Método convencional (1ª etapa).

C4 = { [A a B] b [(/A b /B) a A a B a 1] } c [(//A c //B) b /A b A b /B b B]

0	0	0	0		0	0	0	0		0	0	0	0		1	1	1	1
0	1	1	1	= (	0	X=1	X=1	X=1	b	0	X=1	X=1	X=1	) c	1	1	1	1
0	1	2	2		0	X=1	X=2	X=2		0	X=1	X=1	X=1		1	1	2	2
0	1	2	3		0	X=1	X=2	3		0	X=1	X=1	0		1	1	2	1

Tabela 4.86 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

C4 = { [A a B] b [(/A b /B) a A a B] } c [(//A c //B) b /A b A b /B b B]

0	0	0	0		0	0	0	0		0	0	0	0		1	1	1	1
0	1	1	1	= (	0	X=1	X=1	X=1	b	0	X=1	X=1	X=1	) c	1	1	1	1
0	1	2	2		0	X=1	X=2	X=2		0	X=1	X=2	X=2		1	1	X=2	X=2
0	1	2	3		0	X=1	X=2	3		0	X=1	X=2	0		1	1	X=2	1

Tabela 4.87 – Remoção da CPB (2º critério) da 2ª SFB (3ª etapa).

`C4 = { [ A a B ] b [ ... ] } c [ 1 ]`

0	0	0	0		0	0	0	0		0	0	0	0		1	1	1	1
0	1	1	1	= (	0	X=1	X=1	X=1	b	0	X=0	X=0	X=0	) c	1	1	1	1
0	1	2	2		0	X=1	X=2	X=2		0	X=0	X=2	X=2		1	1	X=1	X=1
0	1	2	3		0	X=1	X=2	3		0	X=0	X=2	0		1	1	X=1	1

Tabela 4.88 – Substituição dos valores irrelevantes móveis (4ª etapa).

`C4 = { [ A a B ] b [ 0 ] } c [ 1 ]   →   C4 = [ A a B ]`

0	0	0	0		0	0	0	0		0	0	0	0		1	1	1	1
0	1	1	1	= (	0	X=1	X=1	X=1	b	0	X=0	X=0	X=0	) c	1	1	1	1
0	1	2	2		0	X=1	X=2	X=2		0	X=0	X=0	X=0		1	1	X=1	X=1
0	1	2	3		0	X=1	X=2	3		0	X=0	X=0	0		1	1	X=1	1

Tabela 4.89 – Substituição do valor irrelevante fixo da 2ª SF (5ª e 6ª etapas).

Exemplo 4: C = A a B – forma C_5

Mesmo sem haver remoção de CPB na 2ª SF, surgem irrelevâncias móveis na 3ª SF. Tal como nos dois exemplos anteriores, os valores irrelevantes na 3ª SF vêm do fato de que, nas duas 1ª s SFs, os valores naquelas posições são substituídos pelo dominante do operador "c", "2". Como não há CPB na 2ª SFB, não se aplica a 3ª etapa.

`C5 = { [ A a B a 1 ] d [ //A c /A c A c //B c /B ] } c [ (//A c //B) b /A b A b /B b B ]`

0	0	0	0		0	0	0	0		2	2	2	2		1	1	1	1
0	1	1	1	= (	0	1	1	1	d	2	2	2	2	) c	1	1	1	1
0	1	2	2		0	1	X=1	X=1		2	2	X=2	X=3		1	1	2	2
0	1	2	3		0	1	X=1	X=1		2	2	X=2	3		1	1	2	1

Tabela 4.90 – Método convencional (1ª etapa).

`C5 = { [ A a B ] d [ //A c /A c A c //B c /B ] } c [ (//A c //B) b /A b A b /B b B ]`

0	0	0	0		0	0	0	0		2	2	2	2		1	1	1	1
0	1	1	1	= (	0	1	1	1	d	2	2	2	2	) c	1	1	1	1
0	1	2	2		0	1	X=2	X=2		2	2	X=2	X=3		1	1	2	2
0	1	2	3		0	1	X=2	X=3		2	2	X=2	X=3		1	1	2	1

Tabela 4.91 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

`C5 = { [ A a B ] d [ 2 ] } c [ (//A c //B) b /A b A b /B b B ]`

0	0	0	0		0	0	0	0		2	2	2	2		1	1	1	1
0	1	1	1	= (	0	1	1	1	d	2	2	2	2	) c	1	1	1	1
0	1	2	2		0	1	X=2	X=2		2	2	X=2	X=2		1	1	2	2
0	1	2	3		0	1	X=2	X=3		2	2	X=2	X=2		1	1	2	1

Tabela 4.92 – Substituição dos valores irrelevantes fixos e móveis na 2ª SF.

C5 = { [A a B] d [2] } c [1]   →   C5 = [A a B]

0	0	0	0		0	0	0	0		2	2	2	2		1	1	1	1
0	1	1	1	= (	0	1	1	1	d	2	2	2	2	) c	1	1	1	1
0	1	2	2		0	1	X=2	X=2		2	2	X=2	X=2		1	1	X=1	X=1
0	1	2	3		0	1	X=2	3		2	2	X=2	X=2		1	1	X=1	1

Tabela 4.93 – Substituição dos valores irrelevantes móveis na 3ª SF.

Exemplo 5: C = A a B – forma C8

Ao contrário do que acontece na síntese de C_1 e C_4, os valores terciários na 2ª SF (3) aparecem nas irrelevâncias móveis, não nas fixas, ou seja, a 2ª remoção só pode ser realizada depois da 1ª. Além disso, tais valores terciários não geram irrelevâncias móveis na 3ª SF. Novamente, recorre-se à substituição intuitiva. Os valores irrelevantes na 3ª SF vêm do fato de que, nas duas primeiras SFs, os valores naquelas posições são substituídos pelo dominante do operador "b", "1".

C8 = { [(A a B) d 0] c [(//A c //B) b /A b /B b 2] } b
[(//A b //B) a (/A b //B) a (//A b /B) a (/A b /B) a A a B]

0	0	0	0		0	0	0	0		1	1	1	1		0	0	0	0
0	1	1	1	= (	0	X=0	X=0	X=0	c	1	X=2	X=2	X=2	) b	0	1	1	1
0	1	2	2		0	X=0	X=0	X=0		1	X=2	2	2		0	1	0	0
0	1	2	3		0	X=0	X=0	3		1	X=2	2	1		0	1	0	0

Tabela 4.94 – Método convencional (1ª etapa).

C8 = { (A a B) c [(//A c //B) b /A b /B b 2] } b
[(//A b //B) a (/A b //B) a (//A b /B) a (/A b /B) a A a B]

0	0	0	0		0	0	0	0		1	1	1	1		0	0	0	0
0	1	1	1	= (	0	X=1	X=1	X=1	c	1	X=2	X=2	X=2	) b	0	1	1	1
0	1	2	2		0	X=1	X=2	X=2		1	X=2	X=2	X=2		0	1	0	0
0	1	2	3		0	X=1	X=2	3		1	X=2	X=2	1		0	1	0	0

Tabela 4.95 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

C8 = { (A a B) c [(//A c //B) b /A b /B] } b
[(//A b //B) a (/A b //B) a (//A b /B) a (/A b /B) a A a B]

0	0	0	0		0	0	0	0		1	1	1	1		0	0	0	0
0	1	1	1	= (	0	X=1	X=1	X=1	c	1	X=2	X=2	X=2	) b	0	1	1	1
0	1	2	2		0	X=1	X=2	X=2		1	X=2	X=3	X=3		0	1	0	0
0	1	2	3		0	X=1	X=2	3		1	X=2	X=3	1		0	1	0	0

Tabela 4.96 – Remoção da CPB (2º critério) da 2ª SFB (3ª etapa).

A terceira etapa, neste exemplo, é dispensável, pode-se ir diretamente para a 4ª etapa.

C8 = { (A a B) c [1] } b
[(//A b //B) a (/A b //B) a (//A b /B) a (/A b /B) a A a B]

0	0	0	0		0	0	0	0		1	1	1	1		0	0	0	0
0	1	1	1	= (	0	X=1	X=1	X=1	c	1	X=1	X=1	X=1	) b	0	1	1	1
0	1	2	2		0	X=1	X=2	X=2		1	X=1	X=1	X=1		0	1	0	0
0	1	2	3		0	X=1	X=2	3		1	X=1	X=1	1		0	1	0	0

Tabela 4.97 – Substituição dos valores irrelevantes fixos e móveis na 2ª SF.

C8 = { [A a B] c [1] } b [0] → C8 = [A a B]

0	0	0	0		0	0	0	0		1	1	1	1		0	0	0	0
0	1	1	1	= (	0	X=1	X=1	X=1	c	1	X=1	X=1	X=1	) b	0	X=0	X=0	X=0
0	1	2	2		0	X=1	X=2	X=2		1	X=1	X=1	X=1		0	X=0	0	0
0	1	2	3		0	X=1	X=2	3		1	X=1	X=1	1		0	X=0	0	0

Tabela 4.98 – Substituição dos valores irrelevantes móveis na 3ª SF.

Exemplo 5: C = A a B – forma C2

Novamente, é preciso recorrer à intuição. Pode-se substituir todos os valores irrelevantes da 2ª SF por 2; isso a torna apenas “2”, e, sendo “2” o valor irrelevante do operador “d”, ele pode ser eliminado. Infelizmente, a menor expressão não pode ser obtida, mas, mesmo assim, esta técnica traz uma boa minimização para este exemplo.

C2 = { [A b B b 2] d [/A c A c /B c B c 3] } a [(//A a //B) d (/A a //B) d (A a //B) d (//A a /B) d (/A a /B) d (A a /B) d (//A a B) d (/A a B) d (A a B)]

0	0	0	0		X=2	X=1	X=2	X=2		X=3	X=2	X=2	X=3		0	0	0	0
0	1	1	1	= (	X=1	1	1	1	d	X=2	2	2	2	) a	0	3	3	3
0	1	2	2		X=2	1	2	2		X=2	2	2	2		0	3	3	3
0	1	2	3		X=2	1	2	X=2		X=3	2	2	3		0	3	3	3

Tabela 4.99 – Método convencional (1ª etapa).

C2 = { [A b B] d [/A c A c /B c B c 3] } a [(//A a //B) d (/A a //B) d (A a //B) d (//A a /B) d (/A a /B) d (A a /B) d (//A a B) d (/A a B) d (A a B)]

0	0	0	0		X=0	X=1	X=2	X=3		X=3	X=2	X=2	X=3		0	0	0	0
0	1	1	1	= (	X=1	1	1	1	d	X=2	2	2	2	) a	0	3	3	3
0	1	2	2		X=2	1	2	2		X=2	2	2	2		0	3	3	3
0	1	2	3		X=3	1	2	X=3		X=3	2	2	X=3		0	3	3	3

Tabela 4.100 – Remoção da CPB (2º critério) da 1ª SFB (2ª etapa).

C2 = { [A b B] d [/A c A c /B c B] } a [(//A a //B) d (/A a //B) d (A a //B) d (//A a /B) d (/A a /B) d (A a /B) d (//A a B) d (/A a B) d (A a B)]

0	0	0	0		X=0	X=1	X=2	X=3		X=0	X=2	X=2	X=3		X=0	0	0	0
0	1	1	1	= (	X=1	1	1	1	d	X=2	2	2	2	) a	0	3	3	3
0	1	2	2		X=2	1	2	2		X=2	2	2	2		0	3	3	3
0	1	2	3		X=3	1	2	X=3		X=3	2	2	X=3		0	3	3	3

Tabela 4.101 – Remoção da CPB (2º critério) da 2ª SFB (3ª etapa).

A terceira etapa, neste exemplo, é dispensável, pode-se ir diretamente para a 4ª etapa.

```
C2 = { [ A b B ] d [ 2 ] } a [ (//A a //B) d (/A a //B) d (A a //B) d
(//A a /B) d (/A a /B) d (A a /B) d (//A a B) d (/A a B) d (A a B) ]
```

0	0	0	0
0	1	1	1
0	1	2	2
0	1	2	3

= (

X=0	X=1	X=2	X=3
X=1	1	1	1
X=2	1	2	2
X=3	1	2	X=3

d

X=2	X=2	X=2	X=2
X=2	2	2	2
X=2	2	2	2
X=2	2	2	X=2

) a

X=0	0	0	0
0	3	3	3
0	3	3	3
0	3	3	3

Tabela 4.102 – Substituição dos valores irrelevantes da 2ª SF.

```
C2 = { A b B } a [ (//A a //B) d (/A a //B) d (A a //B) d (//A a /B) d
(/A a /B) d (A a /B) d (//A a B) d (/A a B) d (A a B) ]
```

0	0	0	0
0	1	1	1
0	1	2	2
0	1	2	3

= (

X=0	X=1	X=2	X=3
X=1	1	1	1
X=2	1	2	2
X=3	1	2	3

) a

0	0	0	0
0	3	3	3
0	3	3	3
0	3	3	3

Tabela 4.103 – Eliminação da 2ª SF.

No quinto exemplo, a remoção da 2ª SF faz com que o valor "3" na posição (3,3) na 1ª SF deixe de ser irrelevante, pois, como ele é operado com "a 3", qualquer outro valor provoca erro na função MVL completa.

4.7 Notas finais

Devido às análises intuitivas, a minimização por uso dos valores irrelevantes móveis não pode ser implementada em um software. Somente quando as leis que regem essas minimizações forem amplamente conhecidas (e não for mais necessário recorrer à intuição) isso poderá ser feito. É necessário formalizar as propriedades e os procedimentos empregados nesta etapa de minimização; surge aqui, então, uma sugestão para futuros trabalhos.

Tal como já foi dito no item 4.2, software LMVS realiza a análise por inspeção, ou seja, remove-se a 1ª CPB e constrói-se a tabela; se não houver erro, mantém-se a remoção. A substituição das irrelevâncias móveis não foi implementada.

Além do problema da não formalização, esta etapa de minimização não garante (tal como as demais) fornecer a minimização ótima, tal como foi mostrado no último exemplo. Isso demonstra a vastidão de possibilidades de minimização da lógica MLV.

Por meio dos exemplos, ficou claro que a eliminação da CPB (2º critério) pode gerar ocorrências de irrelevâncias móveis nas outras SFs e que, em certos casos, uma substituição diferente destes valores irrelevantes móveis pode gerar uma expressão menor. O segundo fato é conseqüência do primeiro. Entretanto, há casos em que esse condicional é invertido, quando a substituição de uma ocorrência do valor irrelevante móvel por um valor diferente do dominante e do secundário proporciona a eliminação da CPB, por tê-la tornado desnecessária devido ao fato da função ter deixado de ser binária-MVL, por causa da substituição realizada. Exemplos desse caso são abordados no item 4.4.

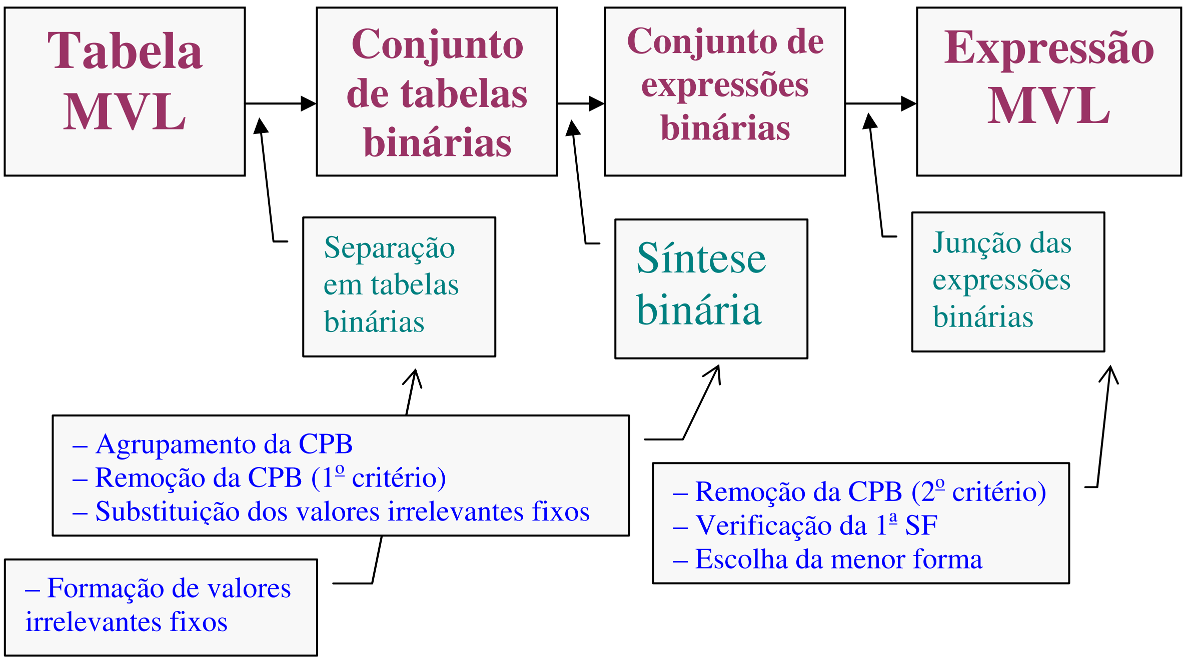


Figura 4.1 – Minimizações no diagrama de blocos do processo de síntese MVL.

Há outros processos de síntese, que, por utilizarem outros artifícios para a obtenção da expressão MVL completa, são apresentados no capítulo 5.

Capítulo 5
Processos alternativos

5.1 Inversão de dominâncias

Neste capítulo, são apresentados dois processos alternativos: a inversão de dominâncias e a forma paramétrica. O primeiro é aplicado sobre as SFBs (cf. capítulos 2 e 3); o segundo (seção 5.6) não utiliza divisão em SFB.

Por meio da decomposição da função MVL em SFBs, vista no capítulo 2, analisa-se o valor dominante, como sendo verdadeiro e o secundário como falso. Supondo o exemplo 1 (tabela 5.1), o valor "0" é o dominante e o valor "1" é o secundário. Infelizmente, nenhuma linha ou coluna completa pode ser obtida, nenhuma simplificação pode ser feita.

0	1	0	0
1	1	1	1
0	1	0	0
0	1	0	0

Tabela 5.1 – Exemplo 1.

```
C = (A b B) a (//A b B) a (/A b B) a
(A b //B) a (//A b //B) a (/A b //B)
a (A b /B) a (//A b /B) a (/A b /B)
```

Este é um exemplo de caso extremo, em que todos os termos são representados isoladamente na expressão, situação na qual não há nenhuma linha ou coluna completa, a minimização não ocorre. Supondo, agora, outra função binária-MVL (tabela 5.2), o valor "0" é o dominante (verdadeiro) e o valor "1" é o secundário (falso). A função é:

```
C = A/ a B/ a 1
```

1	0	1	1
0	0	0	0
1	0	1	1
1	0	1	1

Tabela 5.2 – Exemplo 2.

Este é um outro caso extremo, em que todos os termos são representados agrupadamente em uma linha ou coluna completa, a minimização por este critério é total.

Comparando-se os dois exemplos, percebe-se que a única mudança foi a permutação entre o valor dominante e o secundário. No 1º exemplo, a simplificação por linhas e colunas completas não aconteceu, mas, no 2º, sim, fazendo com que a 2ª expressão (7 caracteres, sem os espaços) seja muito menor do que a 1ª (72). A diferença entre esses dois exemplos é que, no 1º, a simplificação por linhas e colunas completas é de 0%, e, no 2º, de 100%. No 1º exemplo, é desejável que as dominâncias sejam invertidas, pois isso acarreta em grande minimização.

A idéia é separar a função binária-MVL em duas funções parciais que possuam melhor simplificação por linhas e colunas completas; uma delas fornece o valor dominante, a outra fornece o valor secundário. Entretanto, para a função parcial que fornece o valor dominante da função global, este valor tem que ser o seu valor secundário, e, para a função parcial que fornece o valor secundário da função global, este valor tem que ser o seu valor dominante, senão não é obtida a vantagem quanto à formação de linhas e colunas completas. Se V=0 e F=1, o vetor binário é **(0,1)**. Fazendo-se a decomposição em duas SFBs, seguindo a idéia sugerida, tem-se: **(0,1) = (?,1) operador (0,?)**

Para que o valor secundário "1" seja, na 1ª função parcial, o valor dominante, o seu valor secundário deverá ser "2", e, para que o valor dominante "0" seja, na 2ª função parcial, o valor secundário, o seu valor dominante deverá ser "2" (em lógica ternária, escolhida) ou "3" (em lógica quaternária). A SFB é fornecida por duas outras SFBs, mas com inversão de dominâncias. Observa-se que, para três valores, a 2ª SFB é o TOPO da primeira, e, para quatro valores, o DUPLO TOPO (fórmulas 3).

A determinação do operador pode ser feita por inspeção, e deve ser "a". Então, as fórmulas para inversão de dominâncias são definidas.

(0,1) = (2,1) operador (0,2) (1)

(2,1) a (0,2) = (0,1) (2a)
(2,1) b (0,2) = (2,1) (2b)
(2,1) c (0,2) = (2,2) (2c)

3 valores:
(D,S) = (I,S) a (D,I) (3a)
(0,1) = (2,1) a (0,2) (3b)
(1,2) = (0,2) b (1,0) (3c)
(2,0) = (1,0) c (2,1) (3d)

4 valores
(D,S) = (T,S) a (D,I) (3e)
(0,1) = (2,1) a (0,3) (3f)
(1,2) = (3,2) b (1,0) (3g)
(2,3) = (0,3) c (2,1) (3h)
(3,0) = (1,0) d (3,2) (3i)

A expressão obtida possui 19 caracteres, comparando-se com aquela fornecida pelo método tradicional (tabela 5.1), com 72 caracteres, obteve-se uma redução de 73%.

C = [A b B b 2] a [//A d //B d 0]

0	1	0	0		2	1	2	2		0	3	0	0
1	1	1	1	=	1	1	1	1	a	3	3	3	3
0	1	0	0		2	1	2	2		0	3	0	0
0	1	0	0		2	1	2	2		0	3	0	0

Tabela 5.3 – Inversão de dominâncias, para o exemplo 1.

Neste capítulo, ao contrário do resto deste trabalho, apresentou-se, primeiramente, o caso quaternário, e não o ternário. O motivo disso é o fato de que a inversão de dominâncias propicia uma melhora mais acentuada na minimização de funções quaternárias do que na das ternárias. Abaixo, tem-se um exemplo ternário.

C = [A b B b 2] a [/A c /B c 0]

0	1	0		2	1	2		0	2	0
1	1	1	=	1	1	1	a	2	2	2
0	1	0		2	1	2		0	2	0

Tabela 5.4 – Exemplo 3.

Método convencional: `C = (A b B) a (/A b B) a (A b /B) a (/A b /B)`.

Com o método da inversão de dominância, obteve-se uma expressão com 17 caracteres, e, com o método convencional, 27, uma redução de 37%, inferior aos 73% de redução obtidos no exemplo 1. Ainda assim, vale a pena analisar, para lógica ternária, a expressão obtida por inversão de dominâncias, pois, tal como neste caso, ela pode ser menor do que aquela obtida sem inversão. O exemplo 4 mostra que a inversão de dominâncias nem sempre leva a uma expressão menor, é preciso fazer a verificação.

1	0	1
0	0	0
1	0	1

Tabela 5.5 – Exemplo 4

Método convencional:
`C = A/ a B/ a 1`

Por inversão de dominância: `C = [(/A c /B) b (A/ c /B) b (/A c B/) b (A/ c B/)] a [(A/ a B/) c (A a B/) c (A/ a B) c (A a B)]`

5.2 Eliminação da CPB em lógica ternária

Suponhamos que uma função MVL ternária **(D,S,I)** tenha sofrido a CPB e se obtenha **(D,S,D)**, por exemplo **(0,1,0)**. Observa-se que, <u>na inversão de dominâncias, para três valores, a CPB das duas funções parciais sempre pode ser eliminada</u>. O motivo disso é que os valores espúrios encontrado nas funções parciais, obtidos pela remoção da necessária CPB, sempre geram o valor dominante para o conectivo empregado.

Inversão de dominâncias:
(D,S,D) = (I,S,I) op (D,I,D) (4a) **(0,1,0) = (2,1,2) a (0,2,0)** (4e)
Omitindo-se a primeira CPB:
(D,S,D) = (D,S,I) op (D,I,D) (4b) **(0,1,0) = (0,1,2) a (0,2,0)** (4f)
Omitindo-se a segunda CPB:
(S,S,D) = (I,S,I) op (S,I,D) (4c) **(1,1,0) = (2,1,2) a (1,2,0)** (4g)
Omitindo-se as duas CPB:
(D,S,D) = (D,S,I) op (S,I,D) (4d) **(0,1,0) = (0,1,2) a (1,2,0)** (4h)

C = [A b B] a [/A c /B]

0	1	0		0	1	2		1	2	0
1	1	1	=	1	1	1	a	2	2	2
0	1	0		2	1	2		0	2	0

Tabela 5.6 – Exemplo 3 com eliminação das CPB.

O tamanho da expressão passou de 17 para 13 e a redução, com a inversão de dominâncias, passou de 37% para 52%. Podem existir, também, casos em que a SFB não requeira CPB; não havendo minimização por eliminação de CPB, como no exemplo abaixo.

C = [(A/ c B/) b /A b /B] a [(A a B) c A/ c B/]

1	1	1		1	1	1		2	2	2
1	0	0	=	1	2	2	a	2	0	0
1	0	1		1	2	1		2	0	1

Tabela 5.7 – Exemplo 5.

Agora, um exemplo de função MVL. Esta é a função Máximo, dual da função "a", Mínimo.

0	1	2
1	1	2
2	2	2

Tabela 5.8 – Exemplo MVL.

```
    Normal
C1 = [ A b B ] c [ (/A c /B) b (A c /B) b (/A c B) b (A c B) ]
C2 = [ (A c /B) b (/A c B) b (A c B) ] a [ /A c A c /B c B ]
C3 = [ A c B ] b [ (A b B) a /A a /B ]

    Invertido
C1 = [ [A b B] a [/A c /B] ] c [ [A c B] b [/A a /B] ]
C2 = [ [A c B] b [/A a /B] ] a [ [(A b B)] c [(/A c /B)] ]
C3 = [ [(A b B)] c [(/A c /B)] ] b [ [(A c /B) b (/A c B) b (A c B)] a
[(/A a B/) c (A/ a /B) c (/A a /B)] ]
```

Das 3 formas (C_1,C_2,C_3), a menor forma invertida de todas é a C_1.

```
C1 = [ [A b B] a [/A c /B] ] c [ [A c B] b [/A a /B] ]
```

Pelo método convencional, a menor forma é a C_3, mas a C_1 é:

```
C1 = [ A b B ] c [ (/A c /B) b (A c /B) b (/A c B) b (A c B) ]
```

No entanto, não é necessário que as SFBs c0* e c1 estejam, ambas, na forma convencional ou na forma invertida. Pode-se mesclar as formas. Percebe-se que c0* é menor na forma normal, mas que c1 é menor na forma invertida.

```
Sub-Funções - Normal
c0* = (A b B) a 1
c1  = (/A c /B) b (A c /B) b (/A c B) b (A c B)

Sub-Funções - Invertido
c0* = [A b B] a [/A c /B]
c1  = [A c B] b [/A a /B]

Sub-Funções - Otimizado
c0* = (A b B) a 1
c1  = [A c B] b [/A a /B]
```

A forma <u>otimizada</u> é aquela que utiliza a menor possibilidade para cada SFB.

No caso de c0* normal, observa-se que a CPB existe, mas que, na expressão MVL C_1, no modo normal, foi omitida, conforme o procedimento visto no capítulo 4.

```
C1 = [ (A b B) a 1 ] c [ [A c B] b [/A a /B] ]
```

Nesse caso, a CPB de c0* também pode ser eliminada, por inspeção, pelo processo da remoção da necessária CPB.

```
C1 = [ A b B ] c [ [A c B] b [/A a /B] ]
```

As SFBs invertidas não correspondem às não invertidas, como mostrado abaixo, para c0* do exemplo anterior. O motivo está na diferente substituição dos valores irrelevantes.

Normal

`( A b B ) a 1`

0	1	1
1	1	1
1	1	1

Invertido

`[A b B] a [/A c /B]`

0	1	0
1	1	1
0	1	0

Sem Substituição

0	1	X
1	1	X
X	X	X

Tabela 5.9 – Exemplo de desigualdade.

Para o modo normal, os valores irrelevantes devem ser substituídos pelo valor dominante, de modo a criar linhas e colunas completas em "0", que é o dominante, utilizando os termos já existentes, e, quando não for possível, substituir pelo valor secundário.

No exemplo anterior, como nenhuma linha e coluna pode ser formada, todas as ocorrências do valor irrelevante são substituídas pelo valor secundário, "1". Para o caso invertido, os valores irrelevantes devem ser substituídos pelo valor secundário, de modo a criar linhas e colunas completas em “1”, utilizando as ocorrências de “1” já existentes, e, quando não for possível, substituir pelo valor dominante “0”. Uma linha e uma coluna dodem ser formadas. Para a 2ª SFB do exemplo, c1, não há diferença, pois não há valor irrelevante.

5.3 Eliminação da CPB em lógica quaternária

Pela inversão de dominâncias, tem-se:

`(D,S,D,D) = (T,S,T,T) op (D,I,D,D)` (5a) `(0,1,0,0) = (2,1,2,2) a (0,3,0,0)` (5e)

Omitindo-se a primeira CPB:

`(D,S,D,D) = (D,S,T,I) op (D,I,D,D)` (5b) `(0,1,0,0) = (0,1,2,3) a (0,3,0,0)` (5f)

Omitindo-se a segunda CPB:

`(T,S,D,S) = (T,S,T,T) op (T,I,D,S)` (5c) `(2,1,0,1) = (2,1,2,2) a (2,3,0,1)` (5g)

Omitindo-se as duas CPB:

`(D,S,D,S) = (D,S,T,I) op (T,I,D,S)` (5d) `(0,1,0,1) = (0,1,2,3) a (2,3,0,1)` (5h)

Na inversão de dominâncias, para quatro valores, a CPB da 1ª SFB sempre pode ser eliminada. Para o exemplo 1, a expressão torna-se:

`C = [A b B] a [//A d //B d 0]`

0	1	0	0		0	1	2	3		0	3	0	0
1	1	1	1	=	1	1	1	1	a	3	3	3	3
0	1	0	0		2	1	2	2		0	3	0	0
0	1	0	0		3	1	2	3		0	3	0	0

Tabela 5.10 – Exemplo 1 com eliminação da CPB.

0	1	2	3
1	1	2	3
2	2	2	3
3	3	3	3

Tabela 5.11 – Exemplo MVL.

É mostrada, somente, a forma C_1. O mesmo procedimento precisa ser realizado para as formas C_2, C_3, C_4, C_5, C_6, C_7 e C_8.

```
Função – Invertido:
C1 = { [ [A b B] a [//A d //B d 0] ] c [ [A c B] b [//A a //B a 1] ] } d
[ [A d B] c [//A b //B b 2] ]

Função – Normal:
C1 = { [ A b B ] c [ (/A c /B) b (A c /B) b (/A c B) b (A c B) ] } d [
(//A d //B) c (/A d //B) c (A d //B) c (//A d /B) c (/A d /B) c (A d /B)
c (//A d B) c (/A d B) c (A d B) ]

    Sub-Funções – Normal:
    c0** = (A b B) a 1
    c1*  = (/A c /B) b (A c /B) b (/A c B) b (A c B) b 2
    c2   = (//A d //B) c (/A d //B) c (A d //B) c (//A d /B) c (/A d /B)
c (A d /B) c (//A d B) c (/A d B) c (A d B)
```

```
Sub-Funções - Invertido:
c0** = [A b B] a [//A d //B d 0]
c1*  = [A c B] b [//A a //B a 1]
c2   = [A d B] c [//A b //B b 2]
```

No caso de c0** normal, observa-se que a CPB existe, mas que, na expressão MVL, foi omitida, conforme o procedimento visto no capítulo 4. Observa-se que c0** é menor pelo método convencional, mas que c1* e c2 são menores pelo método invertido. A forma otimizada é aquela que utiliza a menor possibilidade para cada SFB.

```
Sub-Funções - Otimizado
c0** = (A b B) a 1
c1*  = [A c B] b [//A a //B a 1]
c2   = [A d B] c [//A b //B b 2]
```

```
C1 = { [ (A b B) a 1 ] c [ [A c B] b [//A a //B a 1] ] } d [ [A d B] c [//A b //B b 2]]
```

Nesse caso, a CPB de c0* também pode ser eliminada.

```
C1 = { [ A b B ] c [ [A c B] b [//A a //B a 1] ] } d [ [A d B] c [//A b //B b 2]]
```

Tal como no caso ternário, as SFBs que possuem irrelevâncias diferem entre os modos normal e invertido. Para c0** do exemplo anterior, tem-se:

Normal

`( A b B ) a 1`

0	1	X=1	X=1
1	1	X=1	X=1
X=1	X=1	X=1	X=1
X=1	X=1	X=1	X=1

Invertido

`[A b B] a [//A d //B d 0]`

0	1	X=0	X=0
1	1	X=1	X=1
X=0	X=1	X=0	X=0
X=0	X=1	X=0	X=0

Tabela 5.12 – Substituições dos valores irrelevantes.

5.4 Valores irrelevantes gerados pela inversão

Três valores:

Relembrando o exemplo 1, sem remoção das CPB, tem-se a tabela 5.4. A remoção da CPB das SFBs pode ser realizada naquele exemplo porque os valores espúrios gerados por essas remoções são operados com "`a 0`", proveniente da outra SFB.

`C = [A b B b 2] a [/A c /B c 0]`

0	1	0		2	1	2		0	2	0
1	1	1	=	1	1	1	A	2	2	2
0	1	0		2	1	2		0	2	0

Tabela 5.13 – Exemplo 3.

Os valores irrelevantes são mostrados na tabela 5.14. A melhor substituição para os valores irrelevantes é a tabela 5.15.

0	1	0		X	1	X		0	2	0
1	1	1	=	1	1	1	A	2	2	2
0	1	0		X	1	X		0	2	0

Tabela 5.14 – Surgimento dos valores irrelevantes na inversão.

`C = 1 a [/A c /B c 0]`

0	1	0		1	1	1		0	2	0
1	1	1	=	1	1	1	A	2	2	2
0	1	0		1	1	1		0	2	0

Tabela 5.15 – Substituição dos valores irrelevantes.

Comparando-se as duas expressões, a segunda forma é mais interessante.

`C = [A b B b 2] a [/A c /B c 0]` Inversão de dominâncias
`C = [A b B] a [/A c /B]` Com a remoção das CPBs.
`C = 1 a [/A c /B c 0]` Com a substituição dos valores irrelevantes.

Analisando-se a função "máximo", tem-se:

0	1	2
1	1	2
2	2	2

Tabela 5.16 – A função "*máximo*" de três valores.

```
c0* = [A b B] a [/A c /B]
c1  = [A c B] b [/A a /B]
C1 = [ [A b B] a [/A c /B] ] c [ [A c B] b [/A a /B] ]
```

Fazendo-se a substituição dos valores irrelevantes, obtém-se:

```
c0* = 1 a [/A c /B c 0]
c1  = 2 b [/A a /B a 1]
C1 = [ 1 a [/A c /B c 0] ] c [ 2 b [/A a /B a 1] ]
```

Quatro valores:

Relembrando o exemplo 1, sem remoção das CPB, tem-se a tabela 5.17. A remoção da CPB da 1ª SFB pode ser realizado porque os valores espúrios gerados por essa remoção são operados com "`a 0`", proveniente da 2ª SFB. Os valores irrelevantes são mostrados na tabela 5.18. A melhor substituição para os valores irrelevantes é a tabela 5.19.

`C = [A b B b 2] a [//A d //B d 0]`

0	1	0	0		2	1	2	2		0	3	0	0
1	1	1	1	=	1	1	1	1	a	3	3	3	3
0	1	0	0		2	1	2	2		0	3	0	0
0	1	0	0		2	1	2	2		0	3	0	0

Tabela 5.17 – Inversão de dominâncias, para o exemplo 1.

0	1	0	0		X	1	X	X		0	3	0	0
1	1	1	1	=	1	1	1	1	a	3	3	3	3
0	1	0	0		X	1	X	X		0	3	0	0
0	1	0	0		X	1	X	X		0	3	0	0

Tabela 5.18 – Surgimento dos valores irrelevantes.

C = 1 a [//A d //B d 0]

0	1	0	0		1	1	1	1		0	3	0	0
1	1	1	1	=	1	1	1	1	a	3	3	3	3
0	1	0	0		1	1	1	1		0	3	0	0
0	1	0	0		1	1	1	1		0	3	0	0

Tabela 5.19 – Substituição dos valores irrelevantes.

Analisando-se a função “máximo”, tem-se:

0	1	2	3
1	1	2	3
2	2	2	3
3	3	3	3

Tabela 5.20 –
A função “máximo” de quatro valores.

c0 = [A b B] a [//A d //B d 0]**
c1* = [A c B] b [//A a //B a 1]
c2 = [A d B] c [//A b //B b 2]

C1 = { [[A b B] a [//A d //B d 0]] c [[A c B] b [//A a //B a 1]] } d [[A d B] c [//A b //B b 2]]

Fazendo-se a substituição dos valores irrelevantes, tem-se:

c0 = 1 a [//A d //B d 0]**
c1* = 2 b [//A a //B a 1]
c2 = 3 c [//A b //B b 2]

C1 = { [1 a [//A d //B d 0]] c [2 b [//A a //B a 1]] } d [3 c [//A b //B b 2]]

Em geral:

Esta análise pode ser feita, também, por meio dos vetores (análise para c0).

Três valores:

(D,S,D) = (I,S,I) op (D,I,D) (6a)
(D,S,D) = (X,S,X) op (D,I,D) (6b)
(D,S,D) = (S,S,S) op (D,I,D) (6c)
(D,S,D) = S op (D,I,D) (6d)
(0,1,0) = (2,1,2) a (0,2,0) (6e)
(0,1,0) = (X,1,X) a (0,2,0) (6f)
(0,1,0) = (1,1,1) a (0,2,0) (6g)
(0,1,0) = 1 a (0,2,0) (6h)

Quatro valores:

(D,S,D,D) = (T,S,T,T) op (D,I,D,D) (6i)
(D,S,D,D) = (X,S,X,X) op (D,I,D,D) (6j)
(D,S,D,D) = (S,S,S,S) op (D,I,D,D) (6k)
(D,S,D,D) = S op (D,I,D,D) (6l)
(0,1,0,0) = (2,1,2,2) a (0,3,0,0) (6m)
(0,1,0,0) = (X,1,X,X) a (0,3,0,0) (6n)
(0,1,0,0) = (1,1,1,1) a (0,3,0,0) (6o)
(0,1,0,0) = 1 a (0,3,0,0) (6p)

Para c0,c1,c2 e c3, tem-se:

Três valores:

c0 = (0,1,0) = 1 a (0,2,0) (6q)
c1 = (1,1,2) = 2 b (1,1,0) (6r)
c2 = (2,2,0) = 0 c (2,2,1) (6s)

Quatro valores:

c0 = (0,1,0,0) = 1 a (0,3,0,0) (6t)
c1 = (1,1,2,1) = 2 b (1,1,0,1) (6u)
c2 = (2,2,2,3) = 3 c (2,2,2,1) (6v)
c3 = (0,3,3,3) = 0 d (2,3,3,3) (6w)

5.5 Funções de uma entrada

Para uma entrada, a inversão de dominâncias não oferece vantagem, primeiramente porque fornece expressões iguais ou maiores do que o método convencional, e também porque, para funções de uma entrada, o melhor método é o indexado, explicado na próxima seção. Além disso, a simplificação pelo uso de valores irrelevantes gerados pela inversão de dominâncias pode aumentar o tamanho da expressão, para funções de uma entrada. Abaixo, estão alguns exemplos. Para cada exemplo, é mostrado:

1. O trio ou quarteto ordenado
2. A forma convencional
3. A forma invertida
4. A forma com valores irrelevantes gerados pela inversão de dominâncias

(0,1,2)
```
C1 = [ A ] c [ /A b A ]
C1 = [ [A] a [/A] ] c [ [A] b [/A] ]
C1 = [ 1 a [/A c 0] ] c [ 2 b [/A a 1] ]
```

(2,1,0)
```
C1 = [ /A a 1 ] c [ A b A/ ]
C1 = [ [A] a [/A] ] c [ [A/] b [A] ]
C1 = [ 1 a [/A c 0] ] c [ 2 b [A a 1] ]
```

(0,1,0)
```
C1 = A a /A
C1 = A a /A
C1 = 1 a [/A c 0]
```

(0,1,1)
```
C1 = A a 1
C1 = [A b A/] a [/A c A]
C1 = 1 a [/A c A]
```

(0,1,0,0)
```
C1 = A a //A a /A
C1 = A a [//A d 0]
C1 = 1 a [//A d 0]
```

(0,1,1,1)
```
C1 = A a 1
C1 = [A b A/ b //A] a [//A d /A d A]
C1 = 1 a [//A d /A d A]
```

(0,1,1,0)
```
C1 = A a /A a 1
C1 = [A b A/] a [//A d /A d 0]
C1 = 1 a [//A d /A d 0]
```

(0,1,0,1)
```
C1 = A a //A
C1 = [A b //A] a [//A d A]
C1 = 1 a [//A d A]
```

(0,1,2,3)
```
C1 = { [ A ] c [ /A b A ] } d [ //A c /A c A ]
C1 = { [ [A] a [//A d 0] ] c [ [A] b [//A a 1] ] } d [ [A] c [//A b 2] ]
C1 = { [ 1 a [//A d 0] ] c [ 2 b [//A a 1] ] } d [ 3 c [//A b 2] ]
```

(3,2,1,0)
```
C1 = { [ /A a 1 ] c [ A/ b //A b 2 ] } d [ /A c A c A/ ]
C1 = { [ [A/] a [/A d 0] ] c [ [/A] b [A/ a 1] ] } d [ [A/] c [/A b 2] ]
C1 = { [ 1 a [/A d 0] ] c [ 2 b [A/ a 1] ] } d [ 3 c [/A b 2] ]
```

(0,1,3,2)
```
C1 = { [ A ] c [ /A b A b 2 ] } d [ //A c /A c A/ ]
C1 = { [ [A] a [//A d 0] ] c [ [A/] b [/A a 1] ] } d [ [/A] c [A/ b 2] ]
C1 = { [ 1 a [//A d 0] ] c [ 2 b [/A a 1] ] } d [ 3 c [A/ b 2] ]
```

A forma convencional é, sempre, a menor.

5.6 Forma paramétrica – Três valores

Além da inversão de dominâncias, há outro processo alternativo: a forma paramétrica. Quando se desenha o gráfico de uma função tridimensional, é usual separar a função em várias funções bidimensionais, tendo, cada função, um determinado valor (parâmetro) para uma determinada variável (variável paramétrica), que assume valores discretos, ainda que retrate uma função contínua. Igualmente, uma função MVL de duas entradas pode ser separada em duas, três ou quatro funções de uma entrada (nos casos binário, ternário e quaternário, respectivamente). O mesmo pode ser feito com funções de três entradas ou mais. Sendo assim, é possível trabalhar, apenas, com funções de uma entrada.

Índice		Expressão
000	0	B1 = 0
001	1	B1 = A a A/
002	2	B3 = A c 0
010	3	B1 = A a /A
011	4	B1 = A a 1
012	5	B1 = A
020	6	B3 = /A c 0
021	7	B1 = [A a 1] c [/A b A/]
021	7	B2 = [A/ b 2] a [/A c A]
021	7	B3 = [/A c 0] b [A a A/]
022	8	B3 = /A c A
100	9	B1 = A/ a /A
101	10	B1 = A/ a 1
102	11	B1 = [A/ a 1] c [/A b A]
102	11	B2 = [/A b 2] a [A/ c A]
102	11	B3 = [A c 0] b [A/ a /A]
110	12	B1 = /A a 1
111	13	B2 = 1
112	14	B2 = /A b A
120	15	B2 = /A
121	16	B2 = /A b A/
122	17	B2 = /A b 2
200	18	B3 = A/ c 0
201	19	B3 = A/
202	20	B3 = A/ c A
210	21	B1 = [/A a 1] c [A b A/]
210	21	B2 = [A b 2] a [A/ c /A]
210	21	B3 = [A/ c 0] b [A a /A]
211	22	B2 = A b A/
212	23	B2 = A b 2
220	24	B3 = A/ c /A
221	25	B2 = A/ b 2
222	26	B3 = 2

Tabela 5.21 – Padrões de formação para três valores.

Uma vez que as funções de uma entrada são poucas (n^n, onde n é a quantidade de valores), é possível obtê-las previamente e indexa-las em uma tabela de busca (*look up table*).

Existem $3^3 = 27$ funções de três valores e uma entrada, B(A) [6] (tabela 5.21). O nome B1, B2 ou B3 indica qual forma fornece a expressão (capítulo 3) (tabela 5.22). Com exceção das funções (210), (102) e (021), todas as demais são obtidas de forma intuitiva, por inspeção. As três exceções são o espelhamento da função B=A e seus deslocamentos, cujas expressões foram obtidas pelo processo de síntese elementar, não proporcionam uma simplificação intuitiva; foram obtidas por meio do procedimento descrito nos capítulos 2 e 3. Percebe-se que as 27 expressões respeitam determinados padrões[3,6], mostrados na tabela 5.23, em ordem de complexidade. Se todas essas expressões puderem ser obtidas previamente e indexadas, elas podem ser buscadas diretamente (*look up table*), ao invés de se refazer todo o processo de síntese da função de uma entrada, isso reduz bastante o tempo de execução do programa. Outra vantagem é a formação da tabela a partir dos padrões mais simples, obtendo-se, com isso, expressões menores, para uma entrada.

O algoritmo de criação da tabela de expressões de uma entrada parte das expressões mais simples em direção às mais complexas, e, sempre que uma função é satisfeita por uma expressão, tal função não é observada na análise das expressões mais complexas, ou seja, a tabela apresenta as expressões menores para cada função. Esse processo garante fornecer a expressão totalmente minimizada.

Forma 1	Forma 2	Forma 3
000 B1 = 0	111 B2 = 1	002 B3 = A c 0
001 B1 = A a A/	112 B2 = /A b A	020 B3 = /A c 0
010 B1 = A a /A	120 B2 = /A	022 B3 = /A c A
011 B1 = A a 1	121 B2 = /A b A/	200 B3 = A/ c 0
012 B1 = A	122 B2 = /A b 2	201 B3 = A/
100 B1 = A/ a /	211 B2 = A b A/	202 B3 = A/ c A
101 B1 = A/ a 1	212 B2 = A b 2	220 B3 = A/ c /A
110 B1 = /A a 1	221 B2 = A/ b 2	222 B3 = 2
210 B1 = [/A a 1] c [A b A/]	210 B2 = [A b 2] a [A/ c /A]	210 B3 = [A/ c 0] b [A a /A]

Tabela 5.22 – Funções agrupadas por forma

Padrão 0	Padrão 1	Padrão 2	Padrão 3	Padrão 4
000 B1 = 0	012 B1 = A	011 B1 = A a 1 110 B1 = /A a 1 101 B1 = A/ a 1	010 B1 = A a /A 001 B1 = A a A/ 100 B1 = A/ a /A	021 B1,B2,B3
111 B2 = 1	120 B2 = /A	212 B2 = A b 2 122 B2 = /A b 2 221 B2 = A/ b 2	112 B2 = /A b A 211 B2 = A b A/ 121 B2 = /A b A/	102 B1,B2,B3
222 B3 = 2	201 B3 = A/	002 B3 = A c 0 020 B3 = /A c 0 200 B3 = A/ c 0	202 B3 = A/ c A 022 B3 = /A c A 220 B3 = A/ c /A	210 B1,B2,B3

Tabela 5.23 – Padrões de formação para três valores

A tabela 5.24 mostra a maneira genérica de se mostrar cada padrão.

- *VAR* – variável {*A,/A,A/*}
- *NUM* – número {*0,1,2*}
- *OP* – operador {*a,b,c*}

Padrão	Esquema	Combinações
0	*NUM*	$3^1 = 3$
1	*VAR*	$3^1 = 3$
2	*VAR OP NUM*	$3^3 = 27$
3	VAR_1 *OP* VAR_2	$3^3 = 27$
46	$(VAR_1\ OP_1\ NUM_1)\ OP_3\ (VAR_2\ OP_2\ VAR_3)$	$3^7 = 2187$

Tabela 5.24 – Uso dos padrões.

Os padrões são contados a partir de 0 (e não 1) pois o software LMVS usa essa numeração, que está vinculada a uma variável do algoritmo de geração da tabela de indexação.

Quando se analisa graficamente uma função analógica de três variáveis, define-se:

- X - Uma variável contínua independente (abscissa)
- Y - Uma variável contínua dependente (mantissa)
- Z - Uma variável discreta paramétrica (cota)

Em funções digitais, todas variáveis são discretas. A variável paramétrica serve para dividir o gráfico 3-D em gráficos 2-D, que podem ser representados no papel. Essa mesma idéia pode ser aplicada às funções MVL de duas variáveis. Ao invés de se analisar uma tabela 3x3, *C(A,B)*, analisa-se três tabelas 3x1 *C(A)* ou três tabelas 1x3 *C(B)*. Por exemplo:

		A		
		0	1	2
	0	0	0	2
B	1	0	1	1
	2	0	1	2

Tabela 5.25 – Exemplo para três valores.

0	0	2

0	1	1

0	1	2

Tabela 5.26 – Parametrização por B - Linhas.

0
0
0

0
1
1

2
1
2

Tabela 5.27 – Parametrização por A - Colunas.

Parametrização em função da variável B (linhas):

Separam-se cada valor de B, e, para cada um, atribui-se uma função *C(A)*.

Parâmetro	Trio Ordenado	Índice	Expressão
`B=0`	`C(A)={0,0,2}`	$C(A) = \mathbf{0}*3^2 + \mathbf{0}*3^1 + \mathbf{2}*3^0 = 2$	`C(A)= A c 0`
`B=1`	`C(A)={0,1,1}`	$C(A) = \mathbf{0}*3^2 + \mathbf{1}*3^1 + \mathbf{1}*3^0 = 4$	`C(A)= A a 1`
`B=2`	`C(A)={0,1,2}`	$C(A) = \mathbf{0}*3^2 + \mathbf{1}*3^1 + \mathbf{2}*3^0 = 5$	`C(A)= A`

Tabela 5.28 – Exemplo de parametrização da variável *B*.

Para cada função, calcula-se um índice a partir do qual se busca, na tabela de expressões, aquela atribuída a ele. A matriz é dividida em linhas. Para que a indexação forneça índices intuitivos, a linha **`C(B)=2`** representa o dígito menos significativo, e a linha **`C(B)=0`** representa o mais significativo, pois algarismos à esquerda possuem uma ordem de grandeza acima em comparação ao da direita, para a maioria das culturas.

Parametrização em função da variável A (colunas):

Separam-se cada opção de valor de A, e, para cada uma, atribui-se uma função *C(B)*. A tabela é dividida em colunas. **`C(A)=2`** representa o dígito menos significativo.

Parâmetro	Trio Ordenado	Índice	Expressão
A=0	**C(B)={0,0,0}**	$C(B) = \mathbf{0}*3^2 + \mathbf{0}*3^1 + \mathbf{0}*3^0 = 0$	**C(B)= 0**
A=1	**C(B)={0,1,1}**	$C(B) = \mathbf{0}*3^2 + \mathbf{1}*3^1 + \mathbf{1}*3^0 = 4$	**C(B)= B a 1**
A=2	**C(B)={2,1,2}**	$C(B) = \mathbf{2}*3^2 + \mathbf{1}*3^1 + \mathbf{2}*3^0 = 23$	**C(B)= B b 2**

Tabela 5.29 – Exemplo de parametrização da variável *A*.

Procedimento:

Deseja-se obter a expressão *C(A,B)* e não um conjunto de três expressões de uma variável. O procedimento é mostrado na tabela 5.31, para a parametrização em função da variável *B*, sobre a função genérica da tabela 5.30. O operador (*op*) pode ser qualquer um dos três (*a,b,c*). Os valores representados pelo símbolo "?" dependem de qual conectivo foi escolhido. Nesta dedução, é escolhido o conectivo "a". Para os demais conectivos, a idéia é análoga. Para que a função MVL seja formada, o valor representado por "?" deve ser o indiferente para o conectivo escolhido (tabela 5.32). São três funções de duas entradas. É mostrada a dedução de *Cpar1*. Para *Cpar2* e *Cpar3*, a idéia é análoga. Pode-se subdividi-la em duas partes (tabela 5.33). Cpar1a = **`C(B=0)`**. Para que a função desejada seja obtida, deve-se escolher a operação para a qual o valor "2" seja o dominante, ou seja, "c". O valor representado por "?" deve ser o indiferente do operador escolhido, ou seja, o valor "1". Determina-se, então, que Cpar1b = **`/B b 2`** (tabela 5.34).

$C_{(1)}$	$C_{(2)}$	$C_{(3)}$
$C_{(4)}$	$C_{(5)}$	$C_{(6)}$
$C_{(7)}$	$C_{(8)}$	$C_{(9)}$

Tabela 5.30 – Função genérica.

Cpar1				Cpar2				Cpar3				C		
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$		?	?	?		?	?	?		$C_{(1)}$	$C_{(2)}$	$C_{(3)}$
?	?	?	op	$C_{(4)}$	$C_{(5)}$	$C_{(6)}$	op	?	?	?	=	$C_{(4)}$	$C_{(5)}$	$C_{(6)}$
?	?	?		?	?	?		$C_{(7)}$	$C_{(8)}$	$C_{(9)}$		$C_{(7)}$	$C_{(8)}$	$C_{(9)}$

Tabela 5.31 – Formação da função MVL.

Cpar1				Cpar2				Cpar3				C		
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$		2	2	2		2	2	2		$C_{(1)}$	$C_{(2)}$	$C_{(3)}$
2	2	2	a	$C_{(4)}$	$C_{(5)}$	$C_{(6)}$	a	2	2	2	=	$C_{(4)}$	$C_{(5)}$	$C_{(6)}$
2	2	2		2	2	2		$C_{(7)}$	$C_{(8)}$	$C_{(9)}$		$C_{(7)}$	$C_{(8)}$	$C_{(9)}$

Tabela 5.32 – Formação da função MVL – 1ª etapa.

Cpar1a				Cpar1b				Cpar1		
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$		?	?	?		$C_{(1)}$	$C_{(2)}$	$C_{(3)}$
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	op	2	2	2	=	2	2	2
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$		2	2	2		2	2	2

Tabela 5.33 – Formação da função MVL – 2ª etapa.

`C(B=0)`				`/B b 2`				Cpar1		
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$		1	1	1		$C_{(1)}$	$C_{(2)}$	$C_{(3)}$
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	c	2	2	2	=	2	2	2
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$		2	2	2		2	2	2

Tabela 5.34 – Formação da função MVL – 3ª etapa.

Pode-se ver que a 1ª função (*Cpar1a*) é função de "A" e a 2ª função (*Cpar1b*) é função de "B", fornecendo, como resultado, (*Cpar1*), uma função de {A,B}. A mesma idéia pode ser aplicada para *Cpar2* e *Cpar3*. Finalmente, juntando as três funções por meio do operador "a", obtém-se a função MVL completa:

Genérico		Para o exemplo (tabela 5.25):	
`Cpar1 = C(B=0) c (/B b 2)`	(7a)	`Cpar1 = (A c 0) c (/B b 2)`	(7d)
`Cpar2 = C(B=1) c (B b 2)`	(7b)	`Cpar2 = (A a 1) c (B b 2)`	(7e)
`Cpar3 = C(B=2) c (B/ b 2)`	(7c)	`Cpar3 = (A) c (B/ b 2)`	(7f)

```
C1 = Cpar1 a Cpar2 a Cpar3 (8a)
C1 = [C(B=0) c (/B b 2)] a [C(B=1) c (B b 2)] a [C(B=2) c (B/ b 2)] (8b)
```

A mesma idéia pode ser aplicada utilizando os operadores *b* e *c*. O mesmo processo é aplicado na parametrização em função da variável *A*. Desse modo, obtém-se seis formas diferentes. A união das três expressões parametrizadas aumenta o tamanho da expressão, diminuindo a probabilidade, num caso aleatório, da forma paramétrica ser menor do que a convencional.

```
C1 = [C(B=0) c (/B b 2)] a [C(B=1) c (B b 2)] a [C(B=2) c (B/ b 2)] (9a)
C2 = [C(B=0) a (B/ c 0)] b [C(B=1) a (/B c 0)] b [C(B=1) a (B c 0)] (9b)
C3 = [C(B=0) b (B a 1)] c [C(B=1) b (B/ a 1)] c [C(B=1) b (/B a 1)] (9c)
C4 = [C(A=0) c (/A b 2)] a [C(A=1) c (A b 2)] a [C(A=2) c (A/ b 2)] (9d)
C5 = [C(A=0) a (A/ c 0)] b [C(A=1) a (/A c 0)] b [C(A=2) a (A c 0)] (9e)
C6 = [C(A=0) b (A a 1)] c [C(A=1) b (A/ a 1)] c [C(A=2) b (/A a 1)] (9f)
```

Para o exemplo dado, as expressões equivalentes são:

```
C1 = [(A c 0) c (/B b 2)] a [(A a 1) c (B b 2)] a [(A) c (B/ b 2)] (9g)
C2 = [(A c 0) a (B/ c 0)] b [(A a 1) a (/B c 0)] b [(A) a (B c 0)] (9h)
C3 = [(A c 0) b (B a 1)] c [(A a 1) b (B/ a 1)] c [(A) b (/B a 1)] (9i)
C4 = [(0) c (/A b 2)] a [(B a 1) c (A b 2)] a [(B b 2) c (A/ b 2)] (9j)
C5 = [(0) a (A/ c 0)] b [(B a 1) a (/A c 0)] b [(B b 2) a (A c 0)] (9k)
C6 = [(0) b (A a 1)] c [(B a 1) b (A/ a 1)] c [(B b 2) b (/A a 1)] (9l)
```

O método paramétrico não leva em consideração a formação de linhas e colunas completas nem faz uso de valores irrelevantes. Também não faz distinção da quantidade de valores fornecidos pela função na hora em que determina a estrutura da expressão. As três expressões parametrizadas em função de “B” têm o mesmo tamanho, ocorrendo o mesmo para “A”, mas, geralmente, expressões parametrizadas em função da variável “A” são maiores ou menores do que aquelas em função de “B”. No exemplo (tabela 5.25), coincidentemente, os dois tipos têm o mesmo tamanho. O exemplo, apesar de didático, não demonstra vantagem para a forma paramétrica, pois, pela forma convencional (`C1 = [ A a B ] c [ /A b A b B ]`), a expressão é menor. Um exemplo de vantagem é apresentado na tabela 5.35.

0	1	2
1	2	0
2	0	1

Tabela 5.35 – Exemplo de vantagem.

Forma paramétrica: `C1 = [(A) c (/B b 2)] a [(/A) c (B b 2)] a [(A/) c (B/ b 2)]`
Forma convencional: `C1 = [ (A b B) a (/A b B/) a (A/ b /B) ] c [ (/A c /B) b (A c /B) b (/A c B) b (A/ c B) b (A c B/) b (A/ c B/) ]`

5.7 Forma paramétrica – Quatro valores

Existem $4^4 = 256$ funções de quatro valores e uma entrada. Todas elas podem ser tabeladas e indexadas, tal como foi feito para três valores. Utiliza-se o mesmo processo de formação da tabela a partir dos padrões mais simples, obtendo-se, com isso, expressões menores, para uma entrada. Ou seja, pode-se afirmar que, se a tabela de indexação obtida por inspeção contiver as expressões absolutamente mínimas para todas as 256 combinações, este método fornecerá a melhor forma de síntese para funções MVL de uma entrada.

Exemplo - Método convencional: `B={0,3,0,1}  B = [ //A d 0 ] b [ A a A/ a //A ]`
Exemplo - Por inspeção: `B={0,3,0,1}  B = ( A a 1 )d ( //A b 0 )`

A dificuldade, no entanto, está em formar a tabela de indexação. Pode-se partir das expressões mais simples e, gradualmente, ir aumentando a complexidade das expressões, até que se preencha toda a tabela. Sempre que uma combinação é preenchida, ela é marcada para ser ignorada quando a complexidade for aumentada, de modo a ter-se sempre a forma mínima. Em muitos casos, têm-se diversas formas mínimas, todas com o mesmo tamanho. Essa idéia permite a implementação de um algoritmo para a criação da tabela. São nove padrões de complexidade, que abrangem todas as 256 funções. O padrão zero é o mais simples, o padrão oito é o mais complexo.

A tabela 5.36 mostra a quantidade de combinações para as quais cada padrão fornece a expressão mínima. Os padrões quatro e cinco podem fornecer várias possibilidades para cada combinação, são os padrões mais usados na criação da tabela.

Por causa do algoritmo de obtenção das combinações para as quais cada padrão fornece a expressão mínima, nos padrões 5 e 6 são analisadas 4^7=16384 combinações cada um, das quais somente algumas são selecionadas, pois muitas delas são fornecidas por padrões mais simples.

Padrão	Combinações	%
0	4	1,5625
1	4	1,5625
2	32	12,5
3	24	9,375
4	36	14,0625
5	72	28,125
6	64	25
7	16	6,25
8	4	1,5625

Tabela 5.36 – Uso dos padrões.

Dentre as selecionadas, muitas correspondem às mesmas funções, ou seja, possuem várias opções de expressão. Nestes padrões, a quantidade de opções de expressão para cada função é irregular, algumas têm mais opções, outras, menos. Esses dois padrões (5 e 5), juntos, cobrem mais da metade das funções.

A tabela 5.37 apresenta os cinco primeiros padrões, que fornecem apenas 100 das 256 combinações. A tabela 5.38 mostra os padrões sete e oito. Cada função do padrão oito possui quatro possibilidades de expressão, todas com o mesmo tamanho. O padrão oito é o único caso em que as expressões não foram obtidas por inspeção; foram obtidas pelo método tradicional, corresponde à função "*máximo*" e seus deslocamentos.

```
C1 = [C(B=0) d (//B c 3)] a [C(B=1) d (/B c 3)] a [C(B=2) d (B c 3)] a [C(B=3) d (B/ c 3)] (10a)
C2 = [C(B=0) a (B/ d 0)] b [C(B=1) a (//B d 0)] b [C(B=2) a (/B d 0)] b [C(B=3) a (B d 0)] (10b)
C3 = [C(B=0) b (B a 1)] c [C(B=1) b (B/ a 1)] c [C(B=2) b (//B a 1)] c [C(B=3) b (/B a 1)] (10c)
C4 = [C(B=0) c (/B b 2)] d [C(B=1) c (B b 2)] d [C(B=2) c (B/ b 2)] d [C(B=3) c (//B b 2)] (10d)
C5 = [C(A=0) d (//A c 3)] a [C(A=1) d (/A c 3)] a [C(A=2) d (A c 3)] a [C(A=3) d (A/ c 3)] (10e)
C6 = [C(A=0) a (A/ d 0)] b [C(A=1) a (//A d 0)] b [C(A=2) a (/A d 0)] b [C(A=3) a (A d 0)] (10f)
C7 = [C(A=0) b (A a 1)] c [C(A=1) b (A/ a 1)] c [C(A=2) b (//A a 1)] c [C(A=3) b (/A a 1)] (10g)
C8 = [C(A=0) c (/A b 2)] d [C(A=1) c (A b 2)] d [C(A=2) c (A/ b 2)] d [C(A=3) c (//A b 2)] (10h)
```

Padrão 0	Padrão 1	Padrão 2	Padrão 3	Padrão 4
0002 B= 0	0123 B= A	0111 A a 1 1110 /A a 1 1101 //A a 1 1011 A/ a 1 0122 A a 2 1220 /A a 2 2201 //A a 2 2012 A/ a 2	0120 A a /A 1200 /A a //A 2001 //A a A/ 0012 A/ a A	0002 (A a A/)c 0 0020 (A a /A)c 0 0200 (/A a //A)c 0 2000 (//A a A/)c 0 0011 (A a A/)d 1 0110 (A a /A)d 1 1100 (/A a //A)d 1 1001 (//A a A/)d 1
1112 B= 1	1230 B= /A	2122 A b 2 1222 /A b 2 2221 //A b 2 2212 A/ b 2 3123 A b 3 1233 /A b 3 2331 //A b 3 3312 A/ b 3	1123 A b /A 1231 /A b //A 2311 //A b A/ 3112 A/ b A	3111 (A b A/)d 1 1113 (A b /A)d 1 1131 (/A b //A)d 1 1311 (//A b A/)d 1 2112 (A b A/)a 2 1122 (A b /A)a 2 1221 (/A b //A)a 2 2211 (//A b A/)a 2
2222 B= 2	2301 B= //A	0020 A c 3 0200 /A c 3 2000 //A c 3 0002 A/ c 3 0023 A c 0 0230 /A c 0 2300 //A c 0 3002 A/ c 0	0223 A c /A 2230 /A c //A 2302 //A c A/ 3022 A/ c A	2022 (A c A/)a 2 0222 (A c /A)a 2 2220 (/A c //A)a 2 2202 (//A c A/)a 2 3322 (A c A/)b 3 3223 (A c /A)b 3 2233 (/A c //A)b 3 2332 (//A c A/)b 3
3333 B= 3	3012 B= A/	0003 A d 0 0030 /A d 0 0300 //A d 0 3000 A/ d 0 0113 A d 1 1130 /A d 1 1301 //A d 1 3011 A/ d 1	0133 A d /A 1330 /A d //A 3301 //A d A/ 3013 A/ d A	3313 (A d A/)b 3 3133 (A d /A)b 3 1333 (/A d //A)b 3 3331 (//A d A/)b 3 3003 (A d A/)c 0 0033 (A d /A)c 0 0330 (/A d //A)c 0 3300 (//A d A/)c 0
			0101 A a //A 2121 A b //A 2323 A c //A 0303 A d //A 1010 /A a A/ 1212 /A b A/ 3232 /A c A/ 3030 /A d A/	3131 (A a //A)b 3 2020 (A b //A)c 0 1313 (A c //A)d 1 0202 (A d //A)a 2 1313 (/A a A/)b 3 0202 (/A b A/)c 0 3131 (/A c A/)d 1 2020 (/A d A/)a 2

Tabela 5.39 – Os cinco primeiros padrões de formação para quatro valores (análogo à tabela 5.21).

Há, ainda, mais dois padrões, mais complexos (5 e 6), que, juntos com os demais, geram todas combinações. A tabela 5.39 mostra a maneira genérica de se definir cada padrão.

Padrão 7	Padrão 8
`0332 (A a A/)c(/A d //A)`	`0321 [(A a 1)d(//A c A/)]c(/A b A b //A)`
`0031 (A a A/)d(/A b //A)`	`[(//A b 2)a(/A d A)]d(//A c A c A/)`
`3320 (A a /A)c(//A d A/)`	`[(A c 3)b(A/ a //A)]a(//A d /A d A)`
`0310 (A a /A)d(//A b A/)`	`[(//A d 0)c(/A b A)]b(A a A/ a //A)`
`3100 (A b A/)d(/A a //A)`	`1032 [(A/ a 1)d(//A c /A)]c(/A b A b A/)`
`2110 (A b A/)a(/A c //A)`	`[(/A b 2)a(A/ d A)]d(//A c /A c A/)`
`1003 (A b /A)d(//A a A/)`	`[(A/ c 3)b(//A a /A)]a(A/ d /A d A)`
`1102 (A b /A)a(//A c A/)`	`[(/A d 0)c(A b A/)]b(A/ a //A a /A)`
`1021 (A c A/)a(/A b //A)`	`2103 [(//A a 1)d(/A c A)]c(A b A/ b //A)`
`1322 (A c A/)b(/A d //A)`	`[(A b 2)a(A/ d //A)]d(//A c /A c A)`
`0211 (A c /A)a(//A b A/)`	`[(//A c 3)b(A a /A)]a(A/ d //A d A)`
`3221 (A c /A)b(//A d A/)`	`[(A d 0)c(A/ b //A)]b(A a //A a /A)`
`2213 (A d A/)b(/A c //A)`	`3210 [(/A a 1)d(A c A/)]c(/A b A/ b //A)`
`3203 (A d A/)c(/A a //A)`	`[(A/ b 2)a(//A d /A)]d(/A c A c A/)`
`2033 (A d /A)c(//A a A/)`	`[(/A c 3)b(A a A/)]a(A/ d //A d /A)`
`2132 (A d /A)b(//A c A/)`	`[(A/ d 0)c(/A b //A)]b(A a A/ a /A)`

Tabela 5.38 – Padrões 7 e 8.

Padrão	Esquema	Combinações
0	*NUM*	$4^1 = 4$
1	*VAR*	$4^1 = 4$
2	*VAR OP NUM*	$4^3 = 64$
3	VAR_1 *OP* VAR_2	$4^3 = 64$
4	$(VAR_1\ OP_1\ VAR_2)\ OP_2\ NUM$	$4^5 = 1024$
5	$(VAR_1\ OP_1\ NUM_1)\ OP_3\ (VAR_2\ OP_2\ NUM_2)$	$4^7 = 16384$
6	$(VAR_1\ OP_1\ NUM_1)\ OP_3\ (VAR_2\ OP_2\ VAR_3)$	$4^7 = 16384$
7	$(VAR_1\ OP_1\ VAR_2)\ OP_3\ (VAR_3\ OP_2\ VAR_4)$	$4^7 = 16384$
8	[(*VAR OP NUM*) *OP* (*VAR OP VAR*)] *OP* (*VAR OP VAR OP VAR*)	-

Tabela 5.37 – Uso dos padrões (análogo à tabela 5.24).

O procedimento é mostrado na tabela 5.42, em função de "B", sobre a função genérica da tabela 5.41. (*op*) pode ser qualquer um dos quatro (*a,b,c,d*). Escolhendo o conectivo "a", "?" deve ser "3" (o valor indiferente para o conectivo "a") (tabela 5.43). São quatro funções de duas entradas; cada uma deve ser dividida em duas partes; é mostrado o procedimento para *Cpar1* (tabela 5.44). Cpar1a = `c(B=0)`. Para que a função desejada seja obtida, deve-se escolher a operação para a qual o valor "3" seja o dominante, ou seja, "d". O valor representado por "?" deve ser substituído por "2" (o valor indiferente do operador "d"). Determina-se, então, que Cpar1b = `//B c 3` (tabela 5.45).

$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$
$C_{(5)}$	$C_{(6)}$	$C_{(7)}$	$C_{(8)}$
$C_{(9)}$	$C_{(10)}$	$C_{(11)}$	$C_{(12)}$
$C_{(13)}$	$C_{(148)}$	$C_{(15)}$	$C_{(16)}$

Tabela 5.41 – Função genérica $C_{(A,B)}$.

Cpar1					Cpar2					Cpar3					Cpar4				
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$		?	?	?	?		?	?	?	?		?	?	?	?	
?	?	?	?	op	$C_{(5)}$	$C_{(6)}$	$C_{(7)}$	$C_{(8)}$	op	?	?	?	?	op	?	?	?	?	= $C_{(A,B)}$
?	?	?	?		?	?	?	?		$C_{(9)}$	$C_{(10)}$	$C_{(11)}$	$C_{(12)}$		?	?	?	?	
?	?	?	?		?	?	?	?		?	?	?	?		$C_{(13)}$	$C_{(148)}$	$C_{(15)}$	$C_{(16)}$	

Tabela 5.42 – Formação da função MVL.

Cpar1					Cpar2					Cpar3					Cpar4				
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$		3	3	3	3		3	3	3	3		3	3	3	3	
3	3	3	3	a	$C_{(5)}$	$C_{(6)}$	$C_{(7)}$	$C_{(8)}$	a	3	3	3	3	a	3	3	3	3	= $C_{(A,B)}$
3	3	3	3		3	3	3	3		$C_{(9)}$	$C_{(10)}$	$C_{(11)}$	$C_{(12)}$		3	3	3	3	
3	3	3	3		3	3	3	3		3	3	3	3		$C_{(13)}$	$C_{(148)}$	$C_{(15)}$	$C_{(16)}$	

Tabela 5.43 – Formação da função MVL – 1ª etapa.

Cpar1a					Cpar1b					Cpar1			
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$		?	?	?	?		$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$	op	3	3	3	3	=	3	3	3	3
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$		3	3	3	3		3	3	3	3
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$		3	3	3	3		3	3	3	3

Tabela 5.44 – Formação da função MVL – 2ª etapa.

Cpar1a					Cpar1b					Cpar1			
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$		2	2	2	2		$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$	d	3	3	3	3	=	3	3	3	3
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$		3	3	3	3		3	3	3	3
$C_{(1)}$	$C_{(2)}$	$C_{(3)}$	$C_{(4)}$		3	3	3	3		3	3	3	3

Tabela 5.45 – Formação da função MVL – 3ª etapa.

A tabela 5.40 mostra uma função para a qual a expressão na forma paramétrica é muito menor do que a da forma convencional. Uma vez, na maioria dos casos, a forma convencional fornece expressões menores do que as da forma paramétrica, é importante obter as duas formas e fazer a verificação. Não foi descoberto nenhum indício que permita prever qual das duas formas fornece a menor expressão. A simetria da matriz parece não influenciar nesta questão; as características especiais de algumas tabelas simétricas são abordadas no apêndice 1. O que se sabe é que quanto maior for a ocorrência de expressões pequenas encontradas nas linhas ou nas colunas, menor é a expressão paramétrica.

0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	2

Tabela 5.46 – Exemplo de síntese para quatro valores.

Forma paramétrica:

```
C1 = [(A) d (//B c 3)] a [(/A) d (/B c 3)] a [(//A) d (B c 3)] a [(A/) d (B/ c 3)]
```

Forma convencional:

```
C1 = { [ (A b B) a (/A b B/) a (//A b //B) a (A/ b /B) ] c [ (/A c /B) b (A c /B) b (/A c B)
b (//A c B) b (A/ c B/) b (//A c B/) b (A c //B) b (A/ c //B) ] } d [ (//A d //B) c (/A d
//B) c (A d //B) c (//A d /B) c (/A d /B) c (A/ d /B) c (//A d B) c (A d B) c (A/ d B) c (/A
d B/) c (A d B/) c (A/ d B/) ]
```

5.8 Conclusão

O método paramétrico pode ser integrado ao método tradicional, pois respeita a mesma estrutura na obtenção das expressões MVL.

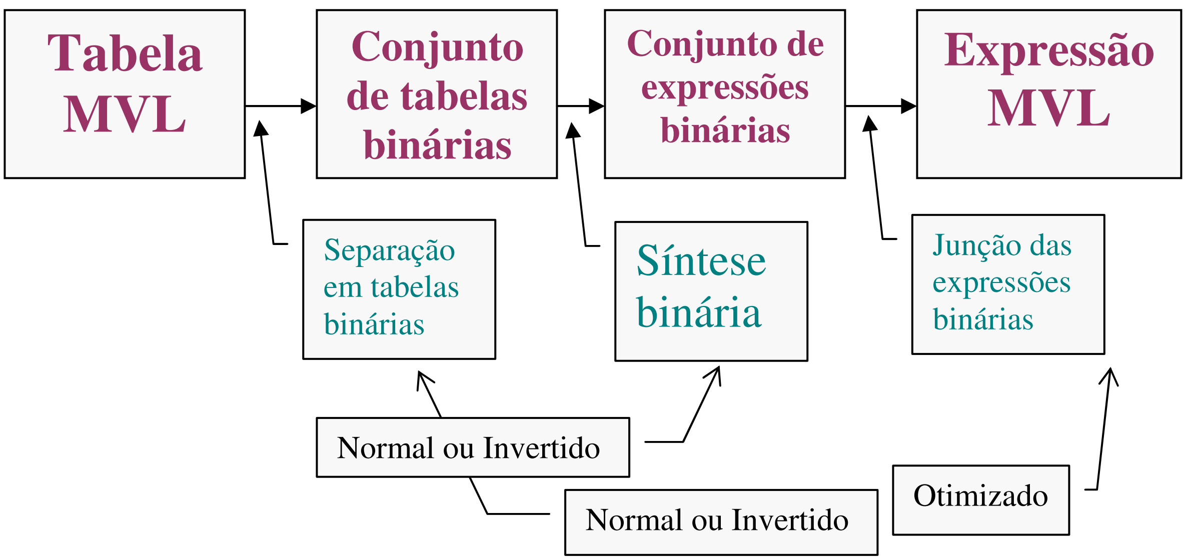


Figura 5.1 – Forma invertida inserida em todas as etapas de síntese.

Se, juntamente com cada posição da tabela de indexação, disponibilizar-se um circuito, pode-se, então, construir circuitos por meio da forma paramétrica de maneira sistematizada (ou até mesmo automatizada). Já foram projetados circuitos eletrônicos para todas as 27 funções ternárias de uma entrada[6]. Construindo-se previamente um circuito para implemetação das funções de agrupamento (fórmulas (9 e 10)), tem-se, então, a disponibilidade de se implementar todas as funções MVL.

Após a explicação da maneira como a expressão minimizada pode ser obtida, apresenta-se, então, uma forma automatizada de realizar essa tarefa, por meio do software LMVS, apresentado no capítulo 6.

Capítulo 6
O aplicativo LMVS

6.1 Aspectos de Desenvolvimento

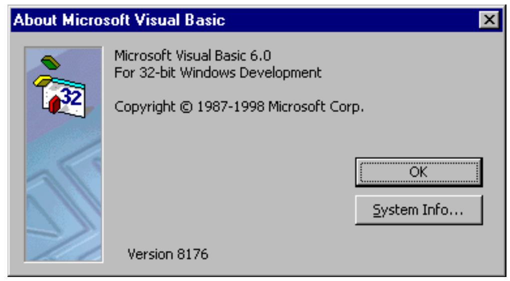


Figura 6.1 – O software utilizado.

O software utilizado no desenvolvimento e na compilação do aplicativo é o Microsoft Visual Basic 6.0 Enterprise Edition. Cada arquivo do tipo "FRM" (*Visual Basic Form File*) consiste em um "*form*" , nome dado pelos aplicativos visuais às janelas.

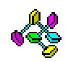
Os arquivos do tipo "BAS" (*Visual* BAS*ic Module*)  consistem em "*module*", conjuntos de procedimentos desvinculados a qualquer *form*. O arquivo principal, que gerencia todos os demais arquivos, é chamado de "*Project*" (*Visual Basic Project*) "VBP".

Figura 6.2

Figura 6.3

6.2 Apresentação do Software LMVS

Figura 6.4 – Logotipo.

Figura 6.5 – Janela de apresentação.

O programa exibe uma janela por vez, não havendo cascateamento. O motivo é proporcionar melhor aproveitamento da área útil da tela, permitindo a visualização de outros softwares ao mesmo tempo. Cada janela apresenta as informações na maneira mais condensada possível. Ao iniciar a execução do programa, uma janela de apresentação é exibida, mostrando informações a respeito do produto. É fornecido o endereço de correio eletrônico do autor do software, a versão, a data da última atualização (não necessariamente a figura 6.5 representa a última atualização) e o logotipo.

6.3 Estrutura do software LMVS

O software LMVS consiste de vários programas menores agrupados, cada um realizando uma conversão diferente. A função mais importante do software é a de fazer a conversão da tabela-verdade para a expressão algébrica, que é chamada de "*tabela -> expressão*" (figura 6.6). Há, também, a opção inversa, ou seja, fazer a conversão da expressão algébrica para a tabela-verdade, que é chamada de "*expressão -> tabela*" (figura 6.7). Esses dois programas se complementam, formando um pacote de manipulação bidirecional de funções MVL (figura 6.8). Quando se comuta do programa "*tabela -> expressão*" para o programa "*expressão -> tabela*", a expressão algébrica gerada no programa "*tabela -> expressão*" é automaticamente enviada para a caixa de entrada do programa "*expressão -> tabela*". Quando se comuta do programa "*expressão -> tabela*" para o programa "*tabela -> expressão*", a tabela-verdade gerada no programa "*expressão -> tabela*" é automaticamente enviada para a matriz de entrada do programa "*tabela -> expressão*".

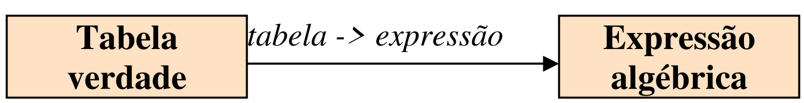


Figura 6.6 – Software *tabela -> expressão*.

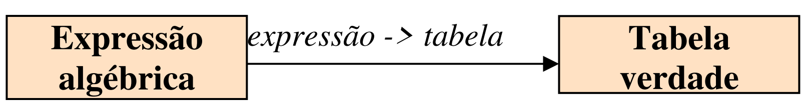


Figura 6.7 – Software *expressão -> tabela*.

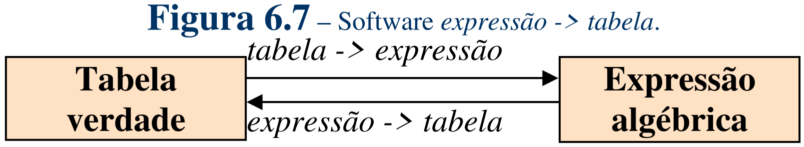


Figura 6.8 – Softwares complementares.

Quando se digita uma expressão algébrica no programa "*expressão -> tabela*" e, a partir da tabela-verdade gerada, calcula-se a expressão algébrica em "*tabela -> expressão*", ela não é, necessariamente, construída da mesma forma que a original, digitada, pois a criação da expressão algébrica em "*tabela -> expressão*" está condicionada a determinados métodos, expostos nos capítulos anteriores. Também se observa que, quando se obtém uma expressão algébrica no programa "*tabela -> expressão*", a expressão algébrica fornecida respeita certos critérios de espaçamento e pode utilizar os caracteres "{","}","[","]"; quando esta expressão algébrica é utilizada no programa "*expressão -> tabela*", outro tipo de espaçamento é utilizado, e os caracteres "{","}","[","]" são substituídos por "(" e ")".

Existem, ainda, duas formas alternativas de se realizar a síntese pela álgebra de Post estendida, chamadas de "*Paramétrico*" e "*Invertido*", vistos nos capítulos 5 (figura 6.9). Ao comutar de um dos três programas de síntese por Post para "*expressão -> tabela*", a expressão algébrica gerada é, automaticamente, enviada para a caixa de entrada do programa "*expressão -> tabela*". Os quatro programas mostrados acima utilizam a lógica de Post estendida, apresentada nos capítulos de 1 a 5.

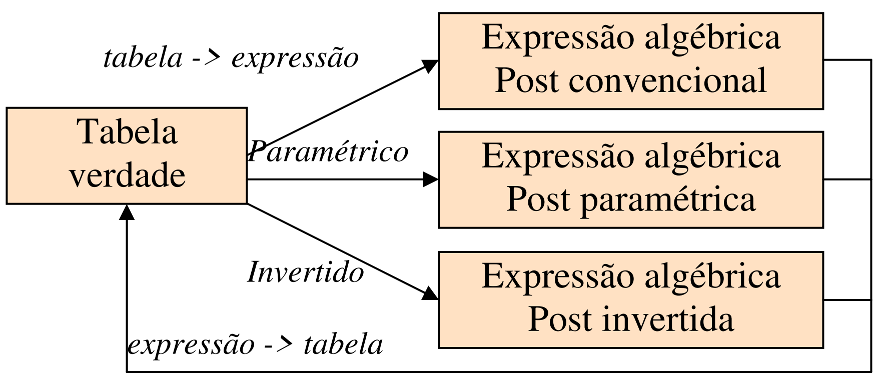


Figura 6.9 – Formas de síntese pela álgebra de Post.

A expressão algébrica, gerada a partir da tabela-verdade, pode ser escrita por meio da síntese de Epstein, através do programa "*MaxMin*" (a síntese MVL clássica[19]). Não foi implementado um software para a conversão da Expressão algébrica Epstein em tabela-verdade porque não há motivação para o emprego desta lógica nos projetos os quais o software LMVS visa auxiliar.

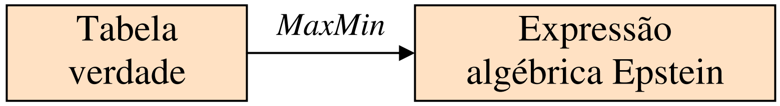


Figura 6.10 – Software *MaxMin*.

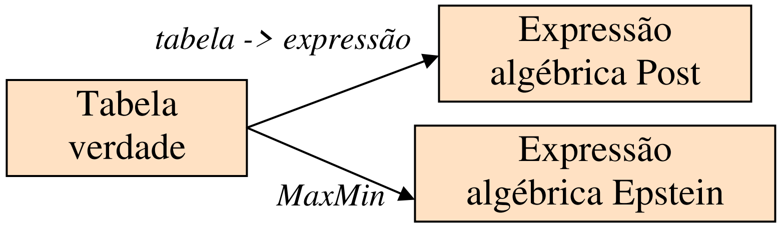


Figura 6.11 – Conversores para expressão.

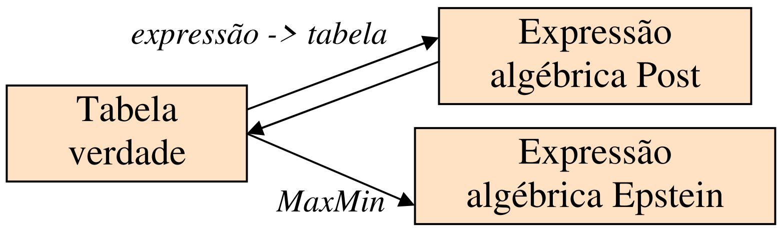


Figura 6.12 – Comparação Post-Epstein.

Não é possível, a partir do software "*MaxMin*", ir para o software "*expressão -> tabela*", pois estes utilizam grafias diferentes. Tal como já foi dito, o software "*expressão -> tabela*", em sua abertura, utiliza a expressão gerada previamente pelo programa que estava sendo utilizado anteriormente, e esta expressão deve ser baseada na lógica de Post estendida. É preciso, primeiro, ir para um dos três programas de síntese por Post, para, depois, ir para "*expressão -> tabela*".Quando se está trabalhando com a síntese de Epstein, o menu de comutação para o modo "*expressão -> tabela*" permanece desabilitado.

Pode-se alternar entre os quatro modos de síntese, como mostrado a seguir. A tabela é transferida automaticamente para a nova janela, e a síntese é feita sobre esta tabela.

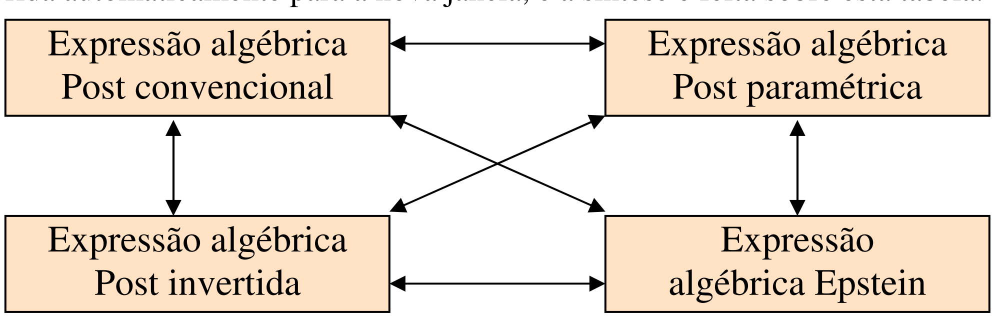


Figura 6.13 – Comutações entre as formas de síntese pela álgebra de Post.

Foi implementada, ainda, uma outra forma de síntese, chamada de "*inspecional*" (apêndice 5), que faz uma varredura das combinações de expressões pequenas em busca de alguma que gere a tabela fornecida. Esta forma não emprega nenhuma técnica de síntese.

Nas doze comutações mostradas na figura 6.13, a tabela-verdade é compartilhada, ou seja, é mantida. Uma mudança na matriz em um dos programas acarreta em mudança automática nas matrizes dos demais programas. É fornecido, ainda, um software que, ao contrário de todos os demais, não realiza conversão entre tabela-verdade e expressão algébrica, apenas converte uma expressão algébrica qualquer em uma que utiliza apenas um conectivo, chamado de "*1 Conectivo*" (apêndice 2).

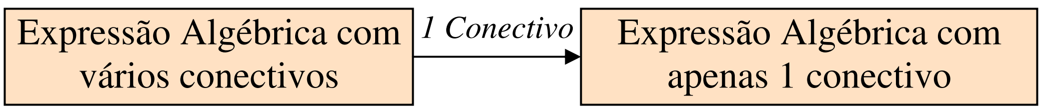


Figura 6.14 – Software *1Conectivo*.

6.4 Menus

Os menus são os mesmos para quase todas as janelas, exceto para janelas especiais de configurações, que não possuem menus. Os menus "Tipos", "Visualizar", "Minimização" e "Conversão" possuem símbolo de ticagem (*check*).

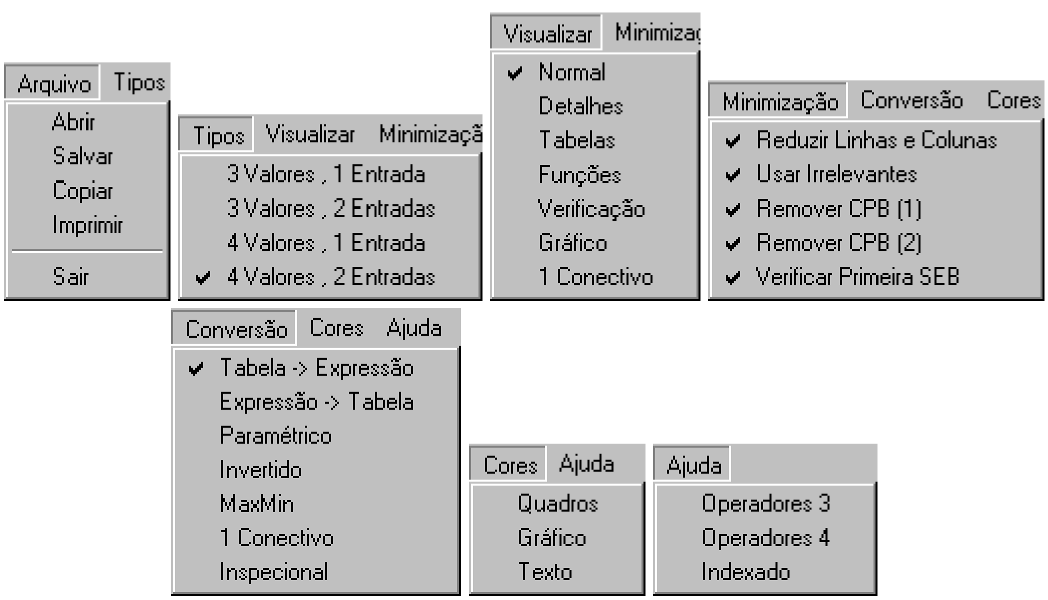


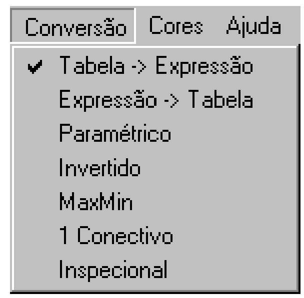
Figura 6.15 – Menus.

6.4.1 – Detalhes sobre os menus:

- **Arquivo**: Menu com opções de manipulação de arquivos ou do programa.
- **Tipos**: Permite escolher a quantidade de valores e de entradas da função.
- **Visualizar**: Mostra as opções de visualização.
- **Minimização**: Mostra as opções de minimização.
- **Conversão**: Mostra as opções de conversão de síntese.
- **Cores**: Menu com opções de cores dos itens das janelas (apêndice 4).
- **Ajuda**: Acessa janelas com ajuda resumida.

O software LMVS é composto de sete programas:

1. Tabela -> Expressão
2. Expressão -> Tabela
3. Paramétrico
4. Invertido
5. MaxMin
6. 1 Conectivo
7. Inspecional



Há quatro possibilidades quanto à quantidade de valores e de entradas:

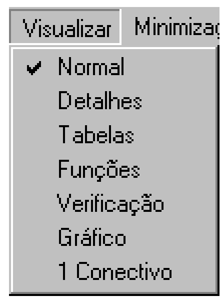
1. 3 valores, 1 entrada
2. 3 valores, 2 entradas
3. 4 valores, 1 entrada
4. 4 valores, 2 entradas

A disponibilidade de manipulação de funções de uma entrada foi implementada apenas como uma facilidade a mais, pois, no campo das funções de duas entradas, pode-se implementar funções de uma entrada. No modo de conversão invertido, a opção de uma entrada foi eliminada, pois, para esta técnica de síntese de funções de uma entrada não é vantajosa, primeiramente porque fornece expressões iguais ou maiores do que o método convencional, e também porque, para funções de uma entrada, o melhor método é o indexado, explicado no capítulo 5. Além disso, a simplificação pelo uso de valores irrelevantes gerados pela inversão de dominâncias pode aumentar o tamanho da expressão, para funções de uma entrada; o menu de comutação para o modo invertido fica desabilitado.

No programa “*Tabela -> Expressão*”, que é o foco principal deste trabalho, há diversos modos de visualização para funções de duas entradas. Para funções de uma entrada só há um modo de visualização, devido à sua pouca utilidade, pois, para funções de uma entrada, o melhor método é o indexado, (capítulo 5). Se o usuário desejar usar os outros modos de visualização para funções de uma entrada, basta transformar a função de duas entradas em função de uma entrada.

Há sete modos de visualização para funções de duas entradas no programa "*Tabela -> Expressão*". O modo de visualização "*Normal*" é suficiente para a elaboração da síntese; os demais modos fornecem informações complementares e mais profundas, facilitando a compreensão, por parte do usuário, do processo de síntese e permitindo a ele optar por outras formas menos minimizadas.

1. Normal
2. Detalhes
3. Tabelas
4. Funções
5. Verificação
6. Listagem
7. Gráfico
8. 1 Conectivo



Os rótulos das janelas são divididos nas seguintes partes:

1. Nome do software: LMVS
2. Conversão
3. Visualização (somente para "*tabela -> expressão*" e duas entradas)
4. Valores x Entradas

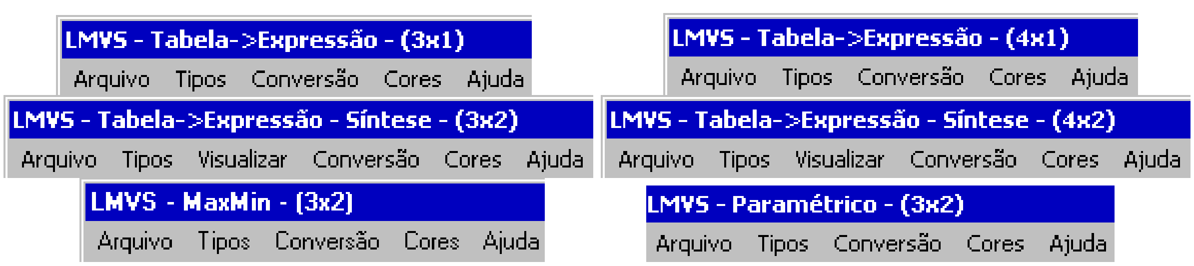
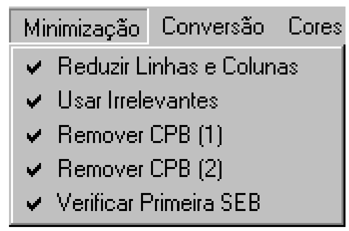


Figura 6.16 – Exemplos de rótulos.

Todas as janelas são apresentadas no modo centralizado. Os comandos maximizar, minimizar, restaurar e fechar, normalmente localizados no canto superior esquerdo das janelas do ambiente Windows, foram retirados, por falta de necessidade.

6.4.2 – Minimização:

Existe a opção para remover algumas opções de minimização para funções de duas entradas no programa "*Tabela -> Expressão*". As cinco opções são escolhidas no menu "*Minimizar*" e são seqüenciais, ou seja, aparecem na ordem como são executadas. Ao se eliminar uma opção, elimina-se, também, as opções seguintes. O exemplo a seguir demonstra como as opções agem.



1. Reduzir linhas e colunas
2. Usar Irrelevantes
3. Remover CPB (1)
4. Remover CPB (2)
5. Verificar Primeira SEB

1	2	0
0	0	0
1	1	0

Tabela 6.1 – Exemplo

```
C1 = [ (A/ b B) a (/A b B) a (A b B/) a (A/ b B/) a (/A b B/) a (/A b /B) a 1 ] c [ (/A c
/B) b (A/ c /B) b (/A c B) b (A c B) b (A/ c B) b (/A c B/) b (A c B/) b (A/ c B/) b 2 ]
```

Reduzir linhas e colunas: Busca por linhas e colunas completas.

```
C1 = [ (A/ b B) a /A a B/ a 1 ] c [ /A b A/ b B b B/ b 2 ]
```

Usar Irrelevantes: Faz a substituição inteligente dos valores irrelevantes.

```
C1 = [ /A a B/ a 1 ] c [ /A b A/ b B b B/ b 2 ]
```

Remover CPB (1): Elimina as desnecessárias conversões para binário.

```
C1 = [ /A a B/ a 1 ] c [ /A b A/ b B b B/ ]
```

Remover CPB (2): Elimina as necessárias conversões para binário.

```
C1 = [ /A a B/ ] c [ /A b A/ b B b B/ ]
```

Verificar primeira SEB: Verifica se a primeira sub-expressão é a função MVL.

```
C = /A a B/
```

Nem todas as funções fazem uso das cinco etapas de minimização. Sempre que se abre o programa, todas as opções de minimização estão selecionadas (ticadas); ao se remover algumas opções, tal configuração não é gravada, de modo que, ao fechar e abrir o programa, a configuração padrão é restabelecida. O motivo disso é que, quase sempre, deseja-se obter a máxima minimização, e, se a configuração das opções de minimização for gravada, há o perigo do usuário esquecer-se disso e passar a usar o programa sem usufruir da máxima minimização.

6.4.3 – Matriz:

O número de dimensões da matriz da tabela-verdade equivale à quantidade de entradas (ou de variáveis). A palavra “entrada” é utilizada em referência às portas lógicas; a palavra “variável” é mais genérica e refere-se apenas à análise algébrica. Para duas entradas, foi feita a escolha (arbitrária) de se utilizar as colunas (eixo *X*) para a variável *A* e as linhas (eixo *Y*) para a variável *B*. Muitos autores preferem inverter a atribuição dos eixos X e Y, mas esta escolha foi feita por causa dos quatro motivos citados no capítulo 1.

A matriz de entrada é composta de um conjunto de caixas de texto (objeto *TextBox*). A síntese é feita à medida que a matriz é digitada, em tempo real. Se for digitado um símbolo inválido, ele é automaticamente convertido para zero.

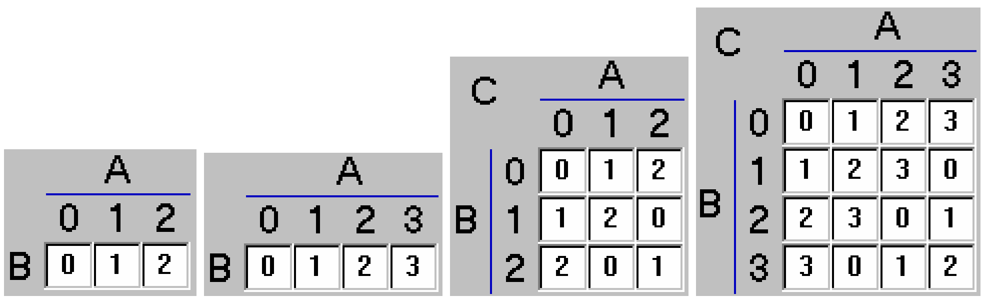


Figura 6.17 – Exemplos para as matrizes de entrada.

Para “*expressão -> tabela*”, por ser a matriz a saída e não a entrada, ela é composta de um conjunto de rótulos (objeto *Label*), não editáveis. A função sintetizada é exibida também em uma caixa de texto. No ambiente visual, a caixa de texto possui uma propriedade (*Property*) chamada *Locked*, para a qual um valor *False* (falso, 0) permite a edição de conteúdo em tempo de execução (*run time*), e um valor *True* (verdadeiro, 1) inibe esta edição. Somente para “*expressão -> tabela*” a caixa de texto é editável, por se tratar da entrada. Nos demais programas, a caixa de texto (quando houver) é selecionável, copiável, mas não editável. A fonte usada na caixa de texto de saída é *Courier New, Bold*. O mesmo vale para a caixa de texto de entrada de “*expressão -> tabela*”.

Entradas	Saída	Entrada(s)
1	*B=B(A)*	*A*
2	*C=C(A,B)*	*A,B*

Tabela 6.2 – O nome da variável de saída.

	Locked	*Apearance*
Matriz	0	1
Função	1	0

Tabela 6.3

Os modos de conversão "*tabela -> expressão*", "*Paramétrico*" e "*Invertido*", por serem processos sofisticados, seu processamento pode ser lento, dependendo da performance do processador utilizado. O fato da síntese ser feita à medida que a matriz é digitada pode causar alguma demora. Para agilizar a digitação em computadores antigos e lentos, é disponibilizado, nestes modos de conversão; a opção "*Não calcular*" (objeto "*CheckBox*"). Quando esta opção é selecionada, a caixa de saída é apagada e os cálculos de síntese não são feitos; somente é feita a manipulação da matriz de entrada. Por não se ter o resultado da síntese; não é possível salvar, copiar nem imprimir. O menu "*Conversão*" também é desabilitado. O software LMVS, em sua primeira versão, começou a ser construído no ano 2000, em um Intel® DX4-100, a questão da performance era perceptível naquelas condições.

Os modos de conversão "*tabela -> expressão*" e "*Invertido*", por usarem a decomposição em SFBs, tornam necessário saber qual a quantidade de valores fornecidos pela função; tal informação é fornecida na janela, à esquerda do *CheckBox* "*Não calcular*".

6.5 Botões

Os botões foram incorporados ao software com o único propósito de facilitar a entrada dos valores da tabela ou da expressão. Quando o *mouse* é apontado sobre um botão, surge um texto explicativo sobre a função do botão.

Figura 6.18 – Exemplo de texto explicativo.

A cada botão está relacionado um comando. Tais comandos podem ser classificados em quatro categorias:

1. Comandos de arquivo
2. Comandos criadores
3. Comandos modificadores
   a. Modificadores de valor
   b. Modificadores de posição

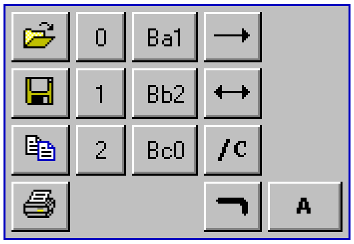


Figura 6.19 – Botões para 3x1.

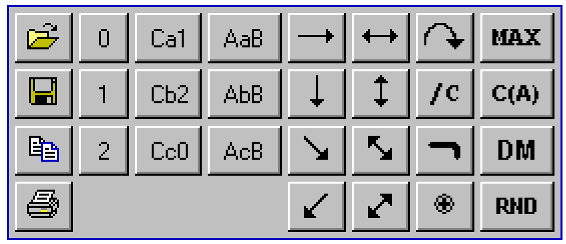


Figura 6.20 – Botões para 3x2.

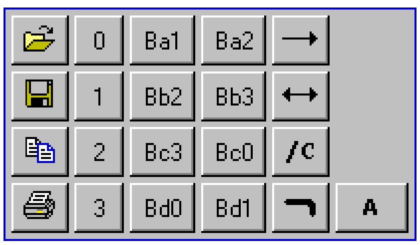


Figura 6.21 – Botões para 4x1.

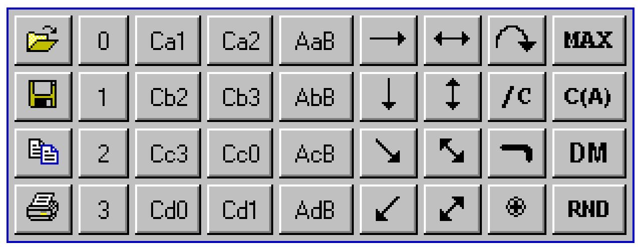
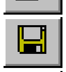


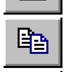
Figura 6.22 – Botões para 4x2.

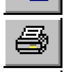
Com exceção dos comandos de arquivo, todos os outros existem apenas para diminuir o trabalho com a digitação da tabela, pois todos eles fornecem um resultado que pode ser obtido manualmente. Eles apenas manipulam a matriz de entrada e nada tem haver com alguma técnica de síntese, não são necessários; somente depois que o comando terminou de manipular a matriz é que a técnica de síntese é aplicada.

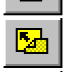
1. **Comandos de arquivo** (também encontrados no menu “Arquivo”:

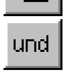
 **Abrir**: Restaura um arquivo salvo anteriormente.

 **Salvar**: Grava a função vigente em um arquivo.

 **Copiar**: Envia para a memória a função vigente.

 **Imprimir**: Envia para a impressora a função vigente.

 **Copiar Texto**: Copia o texto em exibição na janela.



Desfazer (undo): Recupera a tabela anterior.

Os comandos *Abrir* e *Salvar* executam a abertura de outra janela.

Figura 6.23 – Abrir Função.

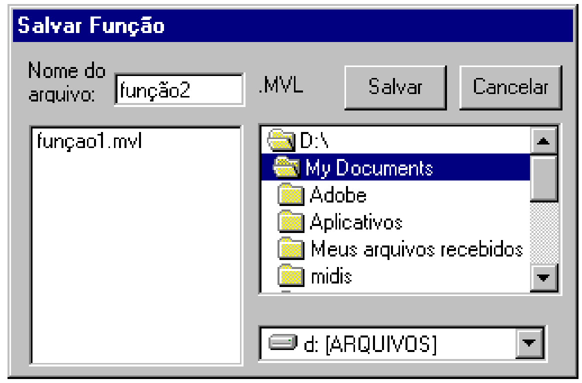
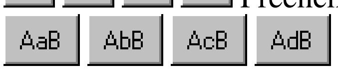


Figura 6.24 – Salvar Função.

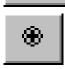
2. Comandos criadores:

Esses comandos não levam em consideração a função que existia antes de serem acionados. As informações anteriores são perdidas. Com exceção do comando RND, todos os demais comandos criadores consistem em funções pré-definidas.

Preenchem a matriz de entrada com o valor mencionado.



Fornecem a função mencionada.

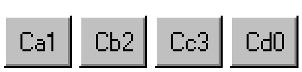

Coloca a função "*ou exclusivo*" [7].

Coloca a função "*máximo*".

Coloca uma função aleatória (*RaNDom*).

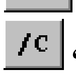
3a. Comandos modificadores de valor:

Esses comandos partem da função vigente e agem baseados no valor de cada combinação, e não em sua posição.

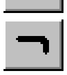


Realizam a conversão para binário.

Realizam a conversão para ternário (para lógica quaternária).



"Logic Shifter": Realiza o deslocamento TOPO.

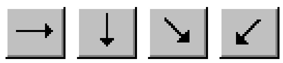

"Logic Fliper": Realiza a inversão de Lukasiewicz, vista no capítulo 1.

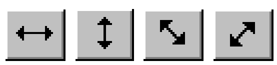
As conversões para binário e para ternário são irreversíveis, ou seja, uma vez executadas, não se pode mais recperar a função original; a operação TOPO acionada tantas vezes quanto for a quantidade de valores retorna a função original; a inversão de

Lukasiewicz retorna a função original quando acionada duas vezes; consiste na inversão tipo espelho, onde o maior valor se torna o menor valor e o menor valor se torna o maior, o secundário se torna o terciário, e o terciário se torna o secundário.

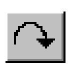
3b. Comandos modificadores de posição:

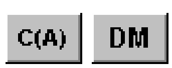
Esses comandos partem da função vigente e alteram a posição de cada combinação.

 **“*Shifters*” ou deslocadores**: Fazem a translação da tabela em uma determinada direção. Os deslocadores, acionados tantas vezes quanto for a base, retornam a função original, similarmente à operação TOPO.

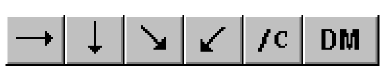
 **“*Fllipers*” ou espelhadores**: A partir de um eixo localizado no plano da tabela, fazem o giro da tabela em 180°. Os espelhadores, acionados duas vezes, retornam a função original, similarmente à inversão de Lukasiewicz.

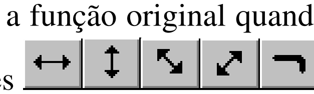
Para os deslocadores e espelhadores, a primeira linha indica ação horizontal; a segunda indica ação vertical, e as duas últimas, as diagonais.

 **Roteador**: Gira a tabela em 90° no sentido horário.



: Estes comandos podem ser classificados tanto como modificadores de valor como modificadores de posição. O primeiro copia a primeira linha da matriz para as demais linhas, tornando a função C(A,B) em C(A). O segundo realiza uma transformação de De Morgan e será abordada mais adiante.

Os botões 

retornam a função original quando acionados tantas vezes quanto for a base (deslocadores). Os botões  retornam a função original quando acionados duas vezes (espelhadores). Cada um deles é explicado, a seguir, por meio da tabela gerada. A tabela original é indexada, onde cada combinação é indicada pelo número cardinal hexadecimal de sua posição (0 a F). Para três valores, o funcionamento é similar.

0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

Tabela 6.4 – A função original.

→

3	0	1	2
7	4	5	6
B	8	9	A
F	C	D	E

↓

C	D	E	F
0	1	2	3
4	5	6	7
8	9	A	B

↘

F	C	D	E
3	0	1	2
7	4	5	6
B	8	9	A

↙

D	E	F	C
1	2	3	0
5	6	7	4
9	A	B	8

↷

C	8	4	0
D	9	5	1
E	A	6	2
F	B	7	3

↔

3	2	1	0
7	6	5	4
B	A	9	8
F	E	D	C

↕

C	D	E	F
8	9	A	B
4	5	6	7
0	1	2	3

⤡

F	B	7	3
E	A	6	2
D	9	5	1
C	8	4	0

⤢

0	4	8	C
1	5	9	D
2	6	A	E
3	7	B	F

C(A)

0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3

DM

/F	/C	/D	/E
/3	/0	/1	/2
/7	/4	/5	/6
/B	/8	/9	/A

Tabela 6.4 – Explicação dos comandos modificadores de posição.

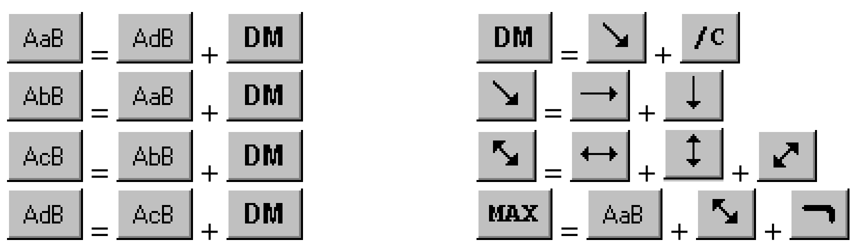
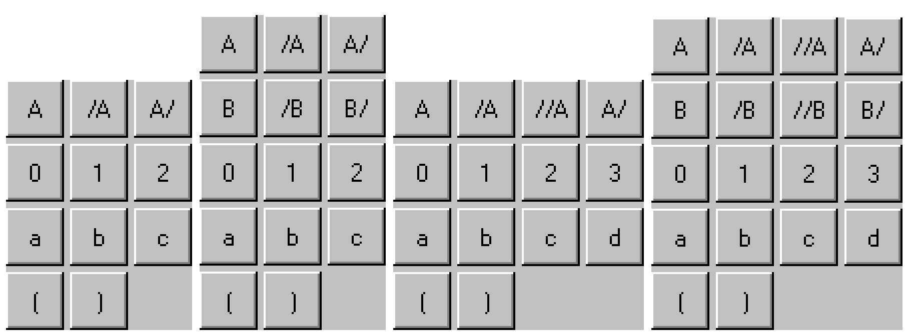


Figura 6.25 – Equivalências.

Os botões já apresentados são comuns para as seguintes opções de conversão:

- Tabela -> Expressão
- Paramétrico
- Invertido
- Inspecional

Nas janelas "*Expressão -> Tabela*" e "*1 Conectivo*", a expressão é editada e a tabela é gerada, os botões são diferentes dos das demais janelas.



3x1 3x2 4x1 4x2

Figura 6.26 – Botões para Expressão -> Tabela.

Os botões para a opção MaxMin são mostrados a seguir.

f0(A), f1(A), f2(A), f3(A), f0(B), f1(B), f2(B), f3(B), g0(A), g1(A), g2(A), g3(A), g0(B), g1(B), g2(B), g3(B): São as funções básicas empregadas na síntese de Epstein.

A*B ; A+B: Conectivos empregados na síntese de Epstein.

C*1 ; C+2: Operações auxiliares. Para três valores, tem-se "C+1" ao invés de "C+2".

A colocação desses botões tem o objetivo de auxiliar o usuário na compreensão do mecanismo por meio do qual a síntese de Epstein é realizada.

Figura 6.27 – Botões para MaxMin 3x1.

Figura 6.28 – Botões para MaxMin 4x1.

Figura 6.29 – Botões para MaxMin 3x2.

Figura 6.30 – Botões para MaxMin 4x2.

6.6 Janelas

As janelas são apresentadas na forma mais compacta possível, com a opção de redimensionamento desabilitados (*sizeable* = *false*). Algumas janelas (aquelas cujo conteúdo apresentado requer a maior área de exibição possível) são apresentadas na forma maximizada, tomando a tela toda. São elas:

- Tabela->Expressão – Detalhes
- Tabela->Expressão – Funções – (4x2)
- Tabela->Expressão – Gráfico
- Tabela->Expressão – 1 Conectivo
- Paramétrico
- Invertido

As imagens mostradas nas janelas dos gráficos se acomodam de acordo com o tamanho da tela, a disposição dos elementos varia de acordo com a resolução adotada.

A definição dos elementos foi testada para as resoluções de 640x480, 800x600, 1024x768, 1152x864 e 1280x960. Todas as janelas foram diagramadas para caber na no tamanho de tela mínimo (648x480) exceto as janelas *Tabela->Expressão – Detalhes* e *Tabela->Expressão – Funções*; elas requerem a resolução mínima de 800x600 para exibir toda a informação.

Algumas janelas não são mostradas neste capítulo, por se tratarem, basicamente, de exibição de texto, e não acrescentarem informação por meio da figura. Tais textos são apresentados na execução do programa, em uma janela de texto. As janelas são:

- Tabela->Expressão – Detalhes
- Tabela->Expressão – 1 Conectivo
- Paramétrico
- Invertido

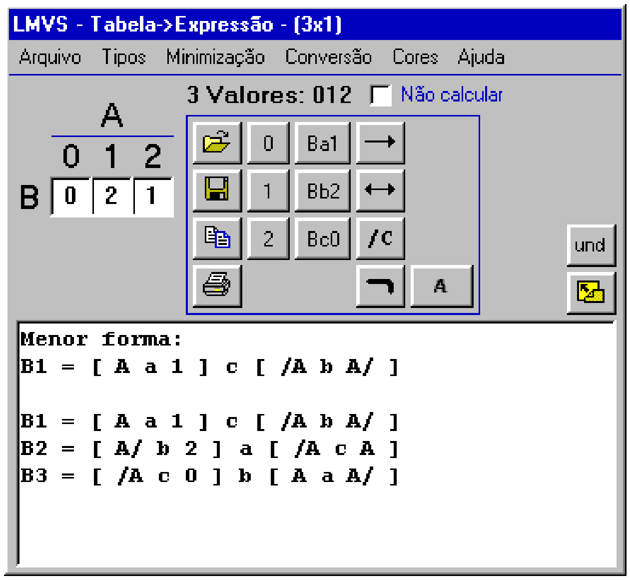


Figura 6.31 – Tabela -> Expressão 3x1.

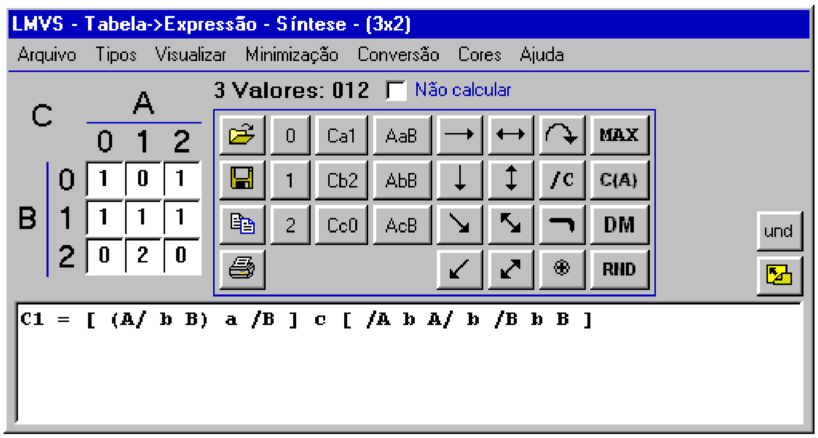


Figura 6.32 – Tabela -> Expressão - Síntese 3x2.

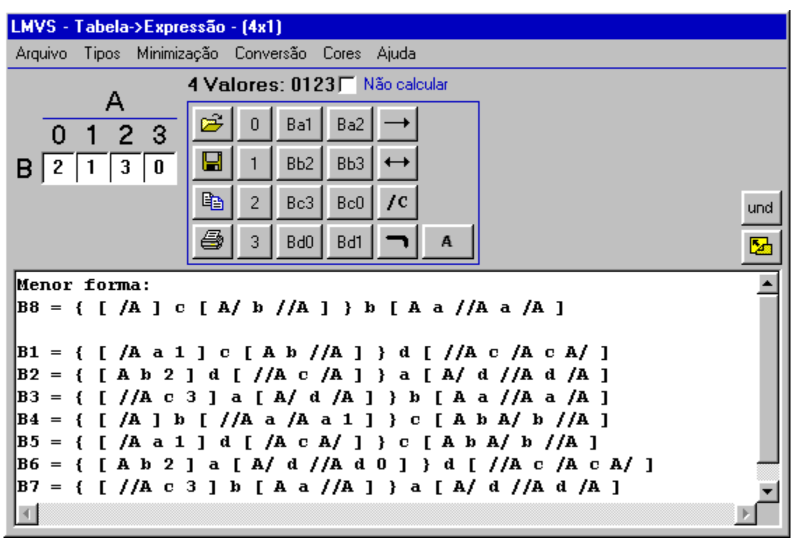


Figura 6.33 – Tabela -> Expressão 4x1.

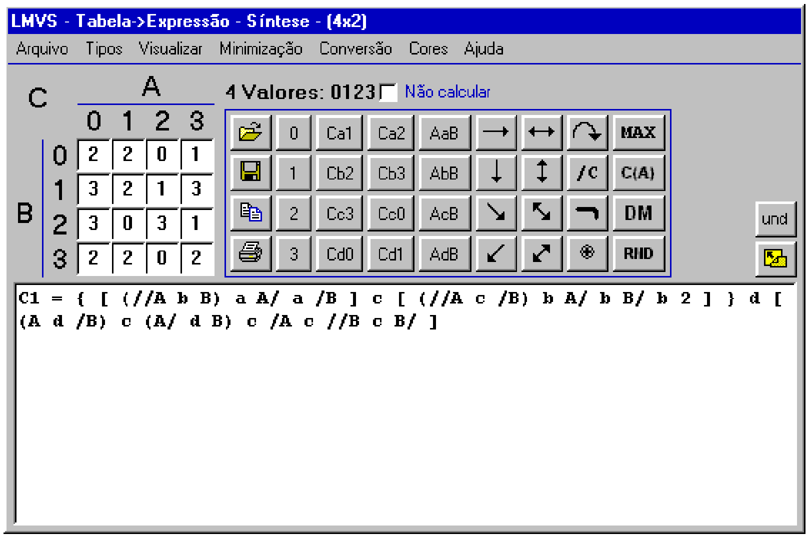


Figura 6.34 – Tabela -> Expressão – Síntese 4x2.

Figura 6.35 – Tabela -> Expressão - Tabelas 3x2.

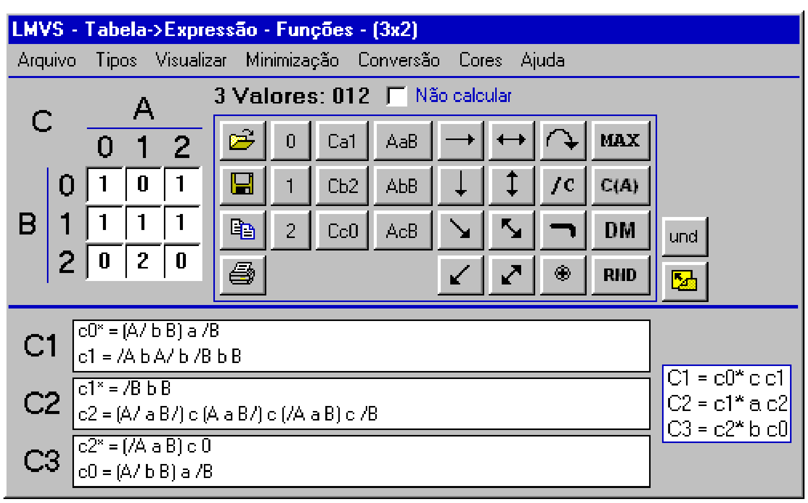


Figura 6.36 – Tabela -> Expressão - Funções 3x2.

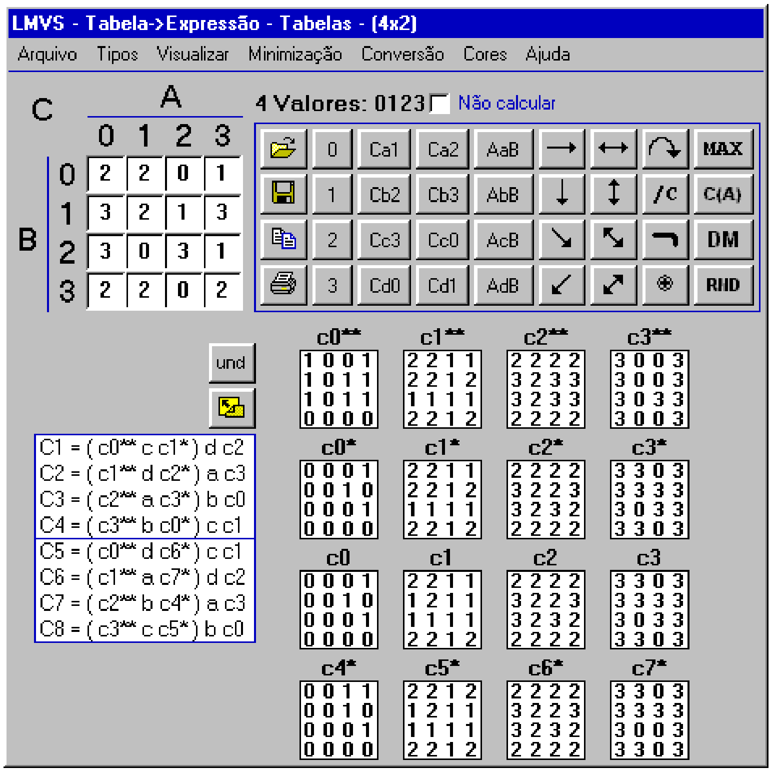


Figura 6.37 – Tabela -> Expressão - Tabelas 4x2.

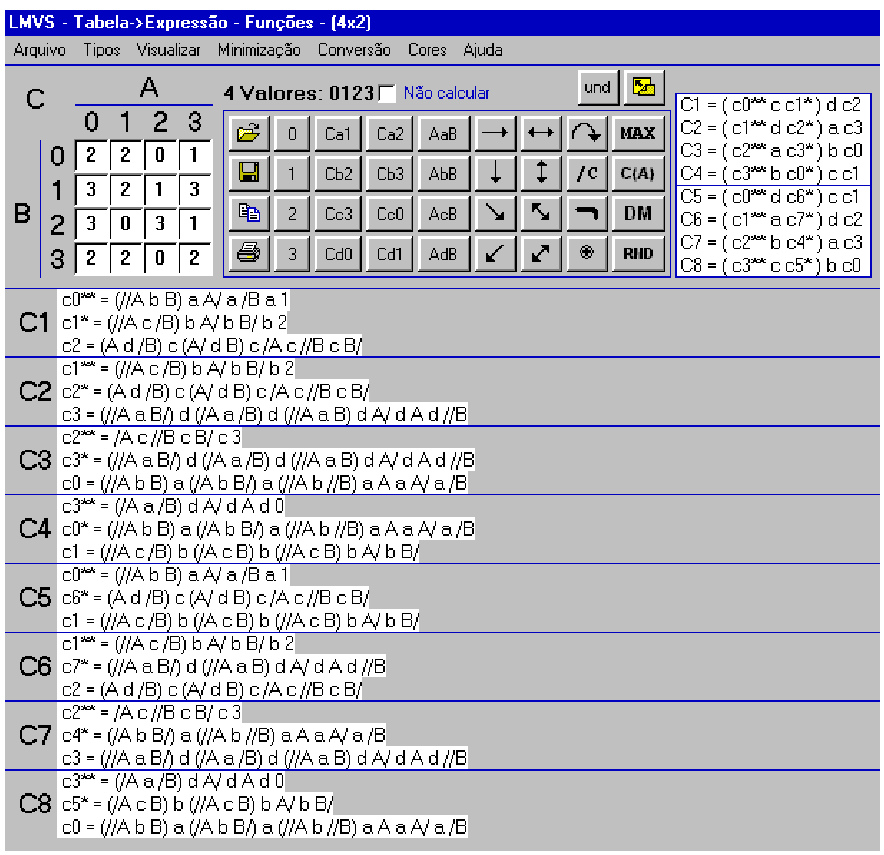


Figura 6.38 – Tabela -> Expressão - Funções 4x2.

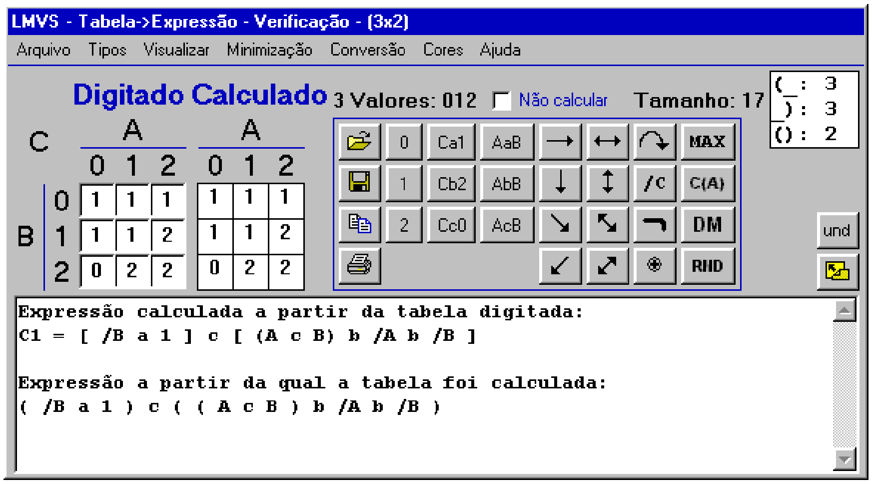


Figura 6.39 – Tabela -> Expressão - Verificação 3x2.

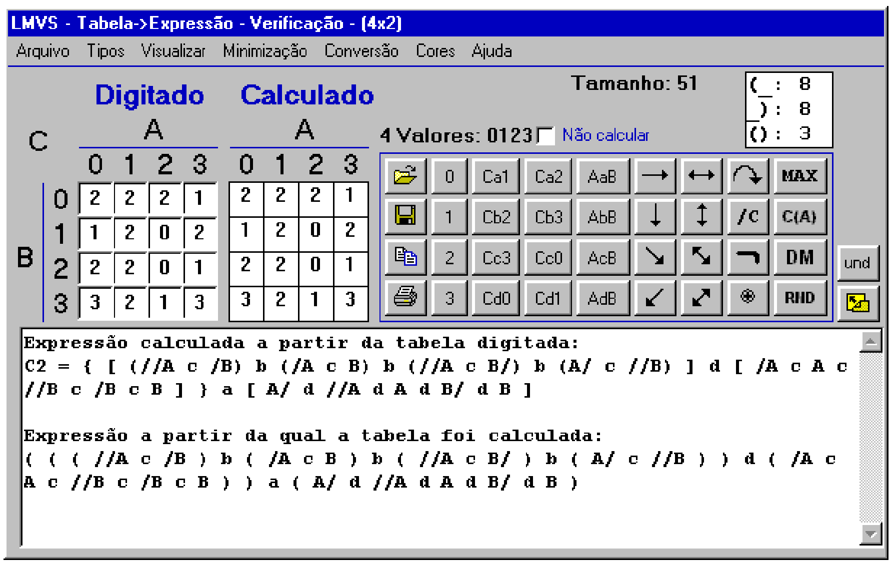


Figura 6.40 – Tabela -> Expressão - Verificação 4x2.

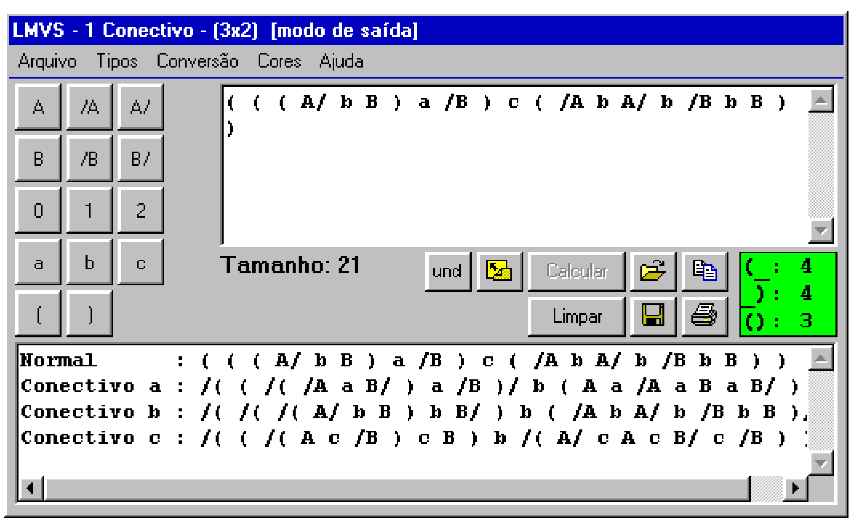


Figura 6.41 – 1 Conectivo 3x2.

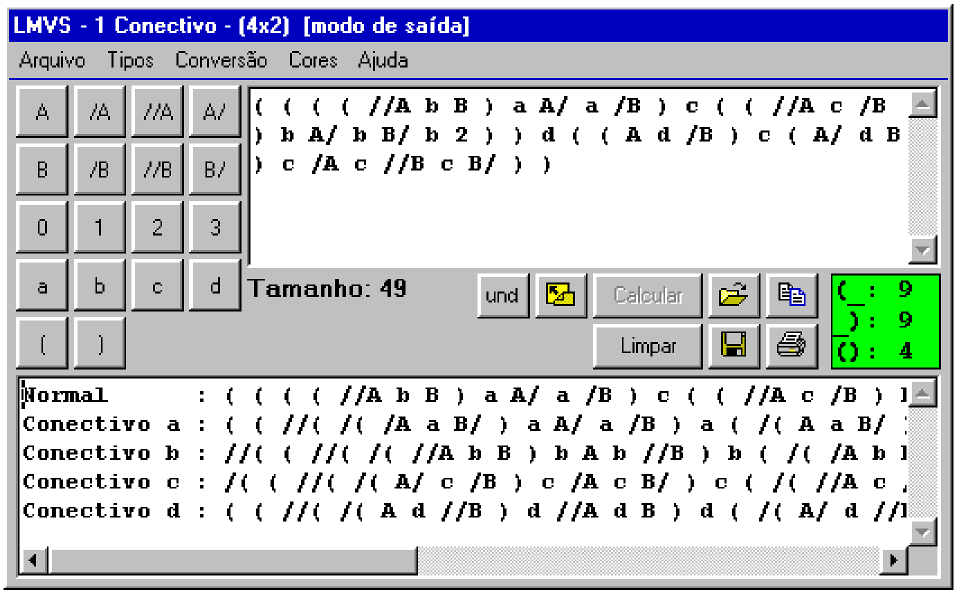


Figura 6.42 – 1 Conectivo 4x2.

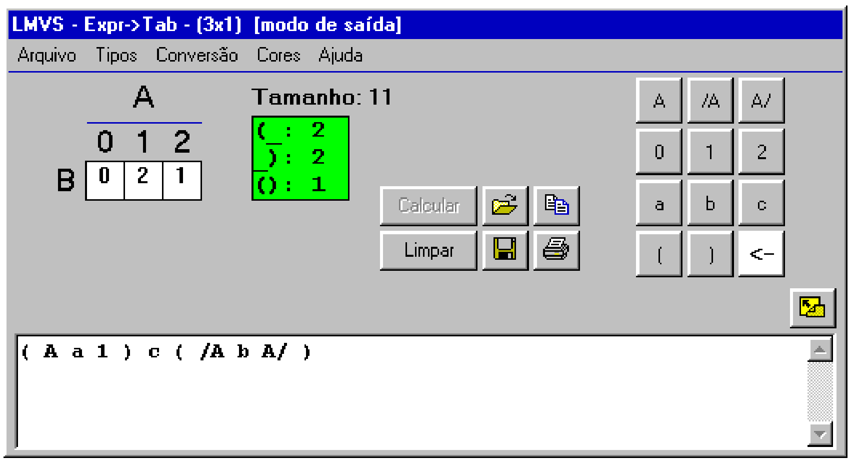


Figura 6.43 – Expressão -> Tabela 3x1.

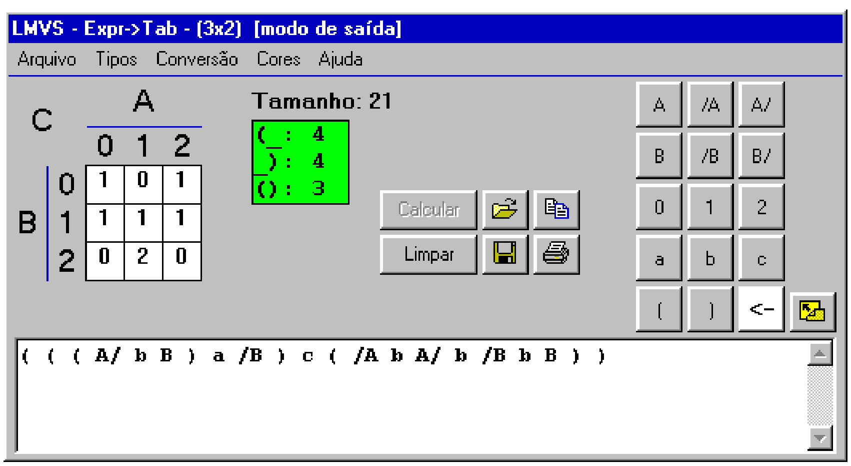


Figura 6.44 – Expressão -> Tabela 3x2.

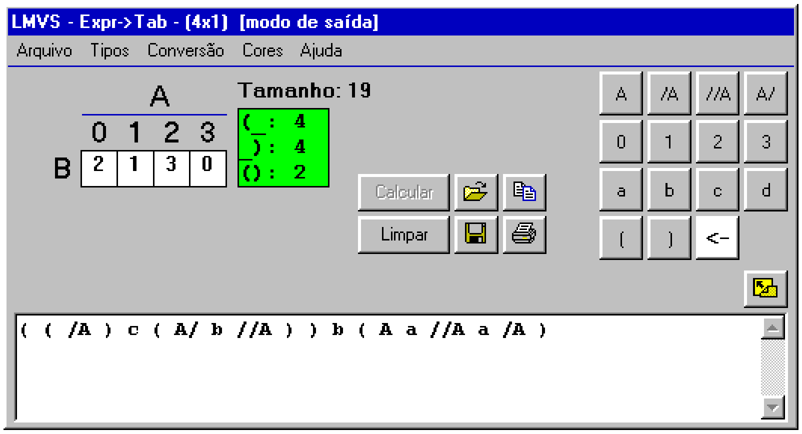


Figura 6.45 – Expressão -> Tabela 4x1.

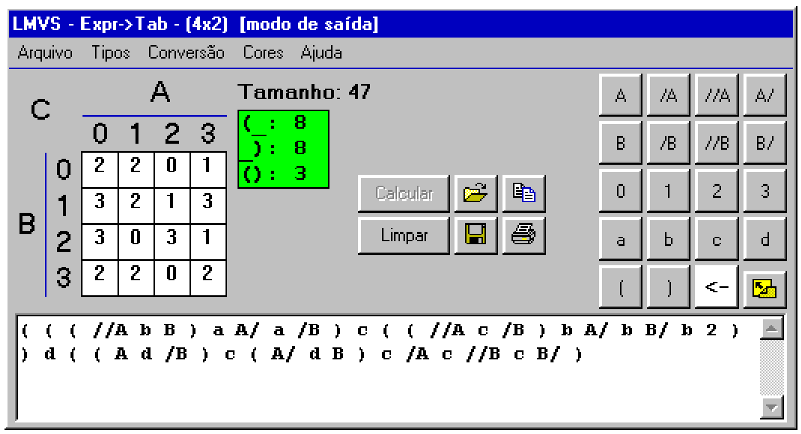


Figura 6.46 – Expressão -> Tabela 4x2.

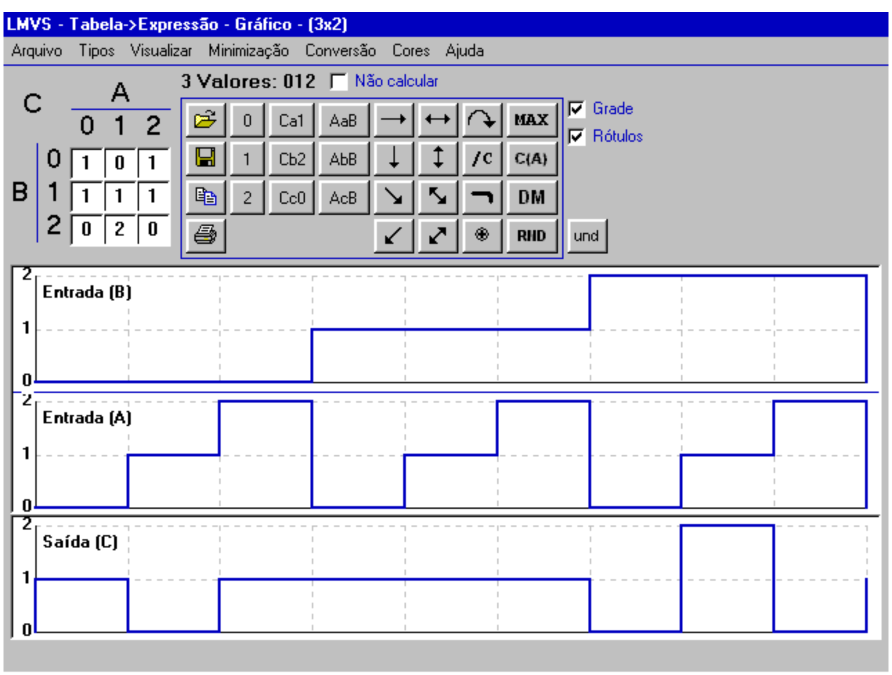


Figura 6.47 – Tabela -> Expressão – Gráfico 3x2.

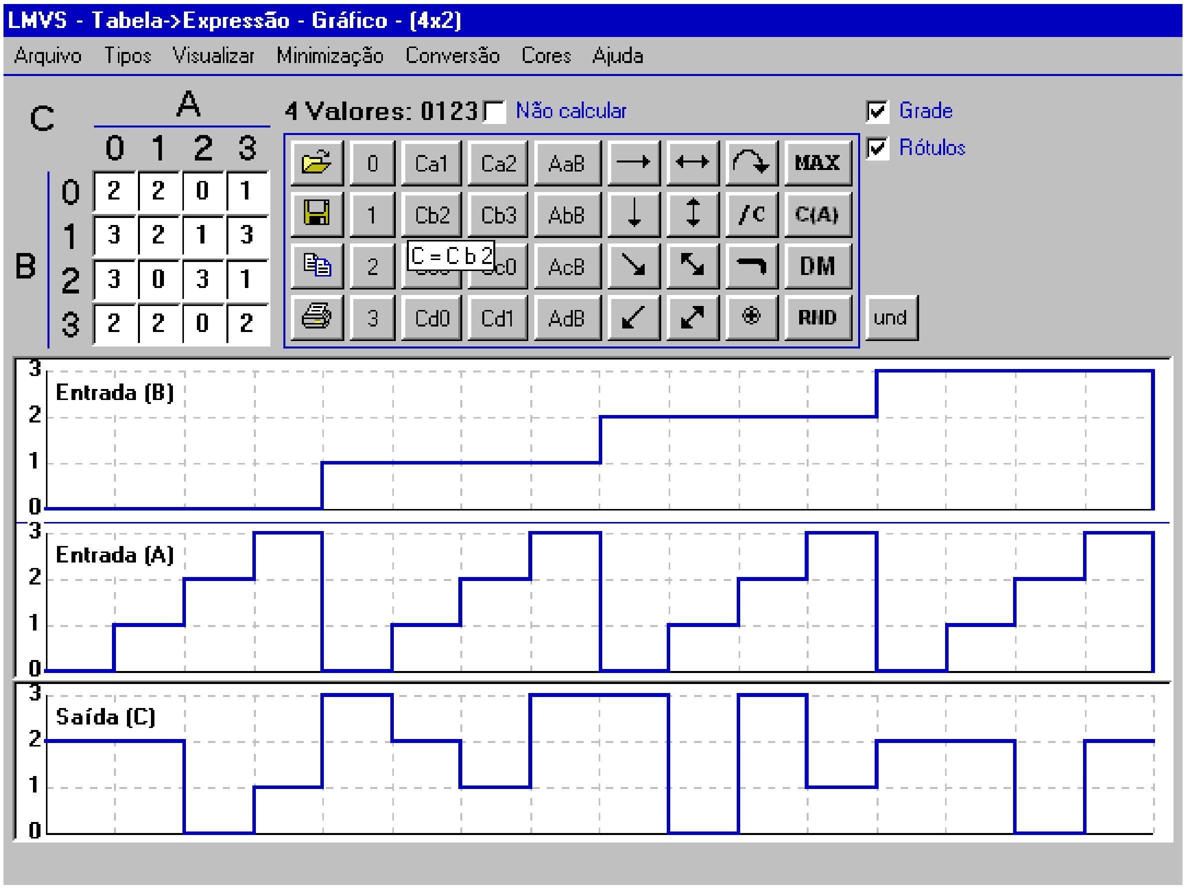


Figura 6.48 – Tabela -> Expressão – Gráfico 4x2.

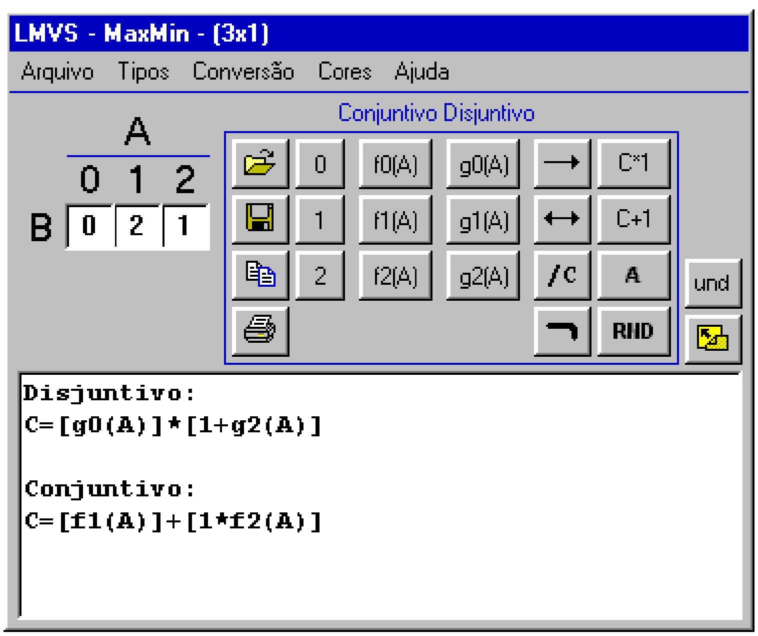


Figura 6.49 – MaxMin 3x1.

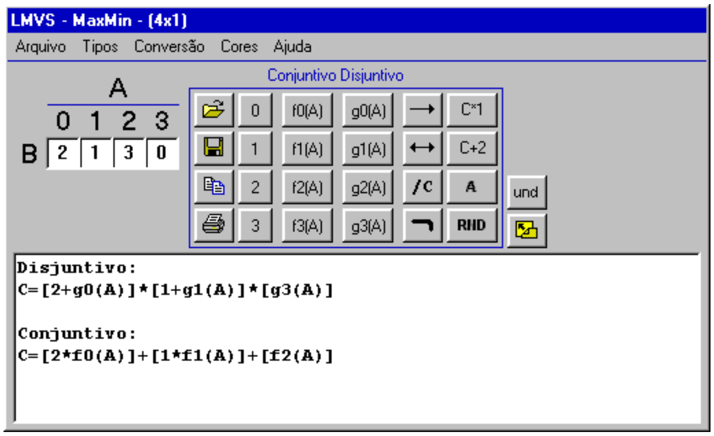


Figura 6.50 – MaxMin 4x1.

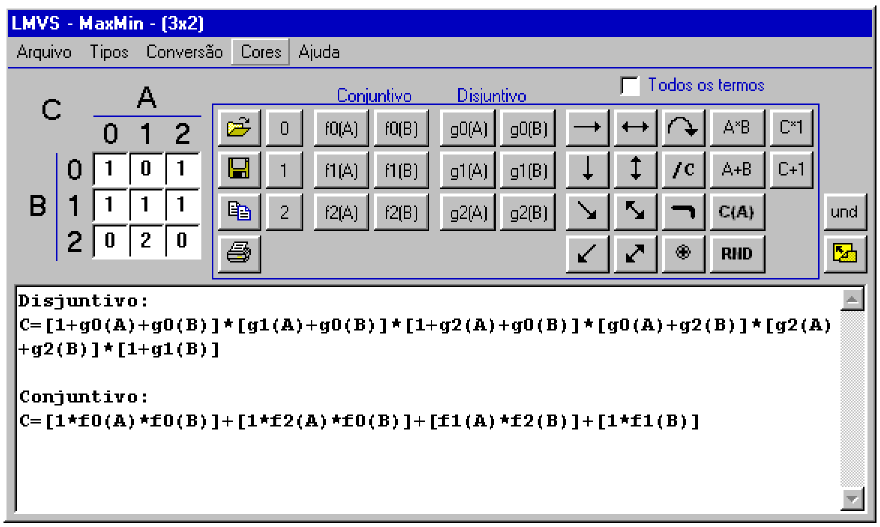


Figura 6.51 – MaxMin 3x2.

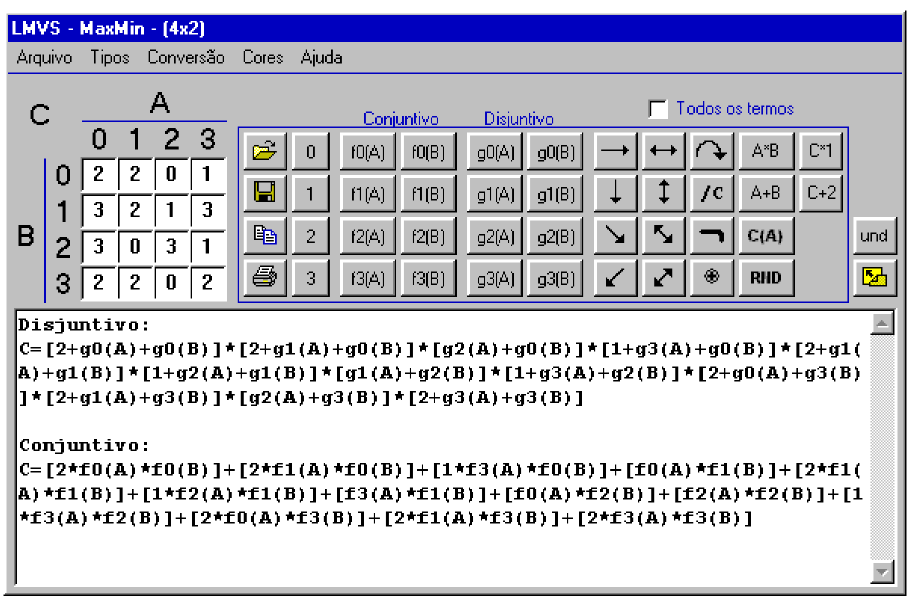


Figura 6.52 – MaxMin 4x2.

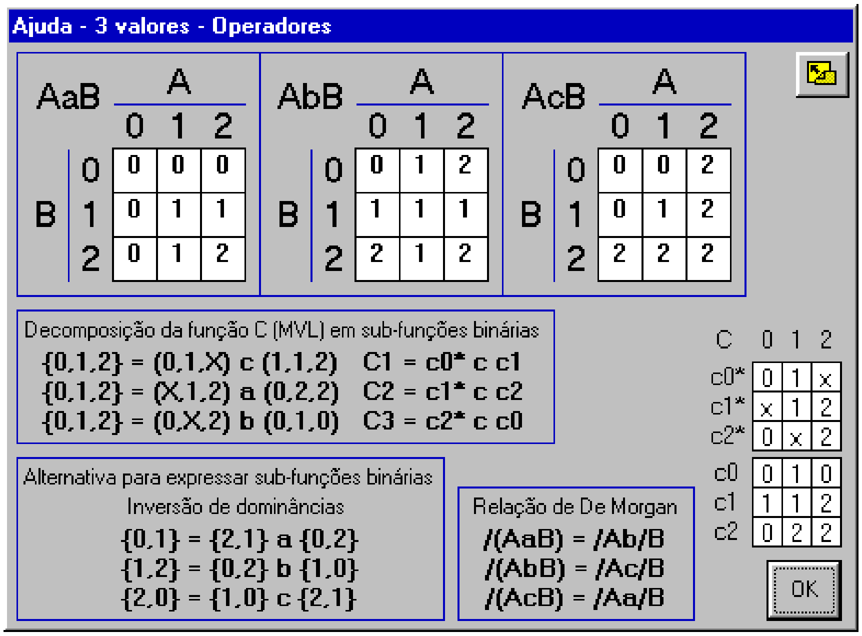


Figura 6.53 – Ajuda 3x2.

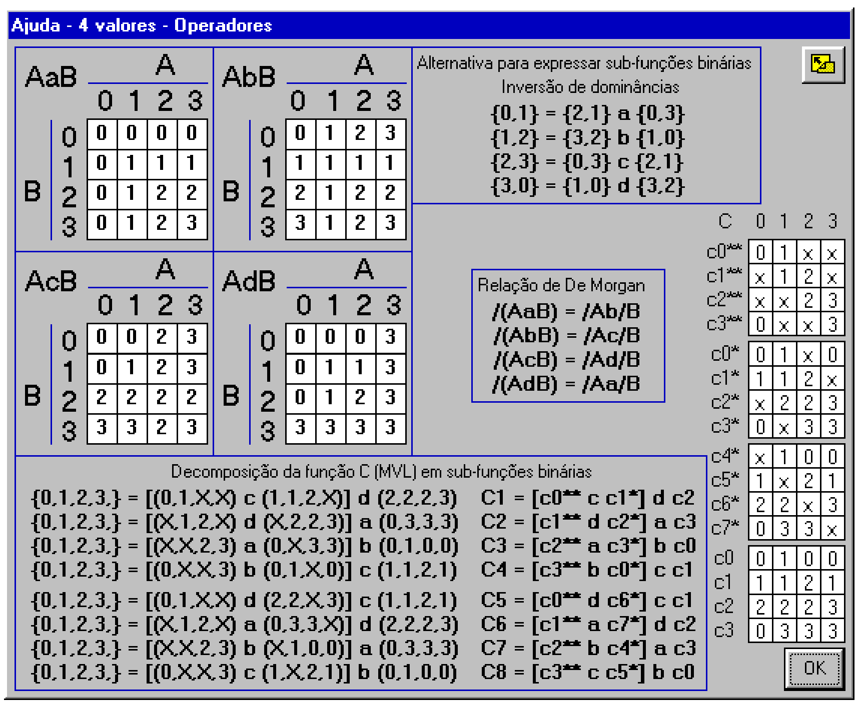


Figura 6.54 – Ajuda 4x2.

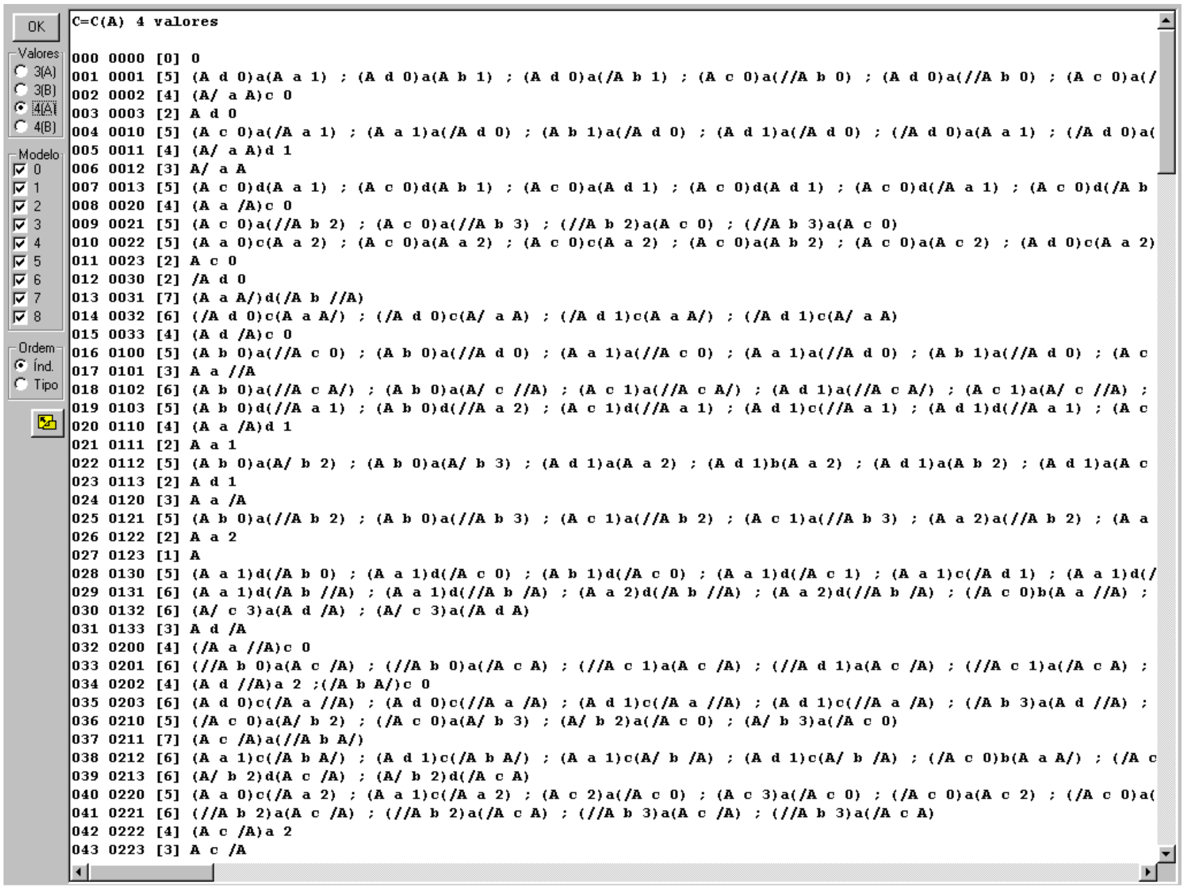


Figura 6.55 – Ajuda Paramétrico.

A síntese de funções lógicas é um processo puramente matemático. Sua implementação física se dá por meio de circuitos lógicos (principalmente elétricos, mas também podem ser mecânicos, hidráulicos, pneumáticos, etc). Uma vez que a aplicação da expressão gerada pelo software LMVS requer a construção de um circuito lógico, é interessante, também, utilizar um software que auxilie a construção deste circuito. Apresenta-se, então, o software LMVP, um simulador de portas e circuitos lógicos baseados na lógica MVL, no capítulo 7.

Capítulo 7
LMVP – Simulador de portas

7.1 Estrutura do software LMVP

A implementação física de uma expressão algébrica digital se dá por meio de circuitos lógicos (principalmente elétricos, mas também podem ser mecânicos, hidráulicos, pneumáticos, etc). Esse fato motivou a criação do software LMVP, simulador de portas e de circuitos lógicos. Ao iniciar a execução do programa, uma janela de apresentação é exibida, mostrando informações a respeito do produto. É fornecido o endereço de correio eletrônico do autor do software, a versão, a data da última atualização e o logotipo. O software consiste de dois programas menores. O primeiro é o simulador de portas lógicas MVL, o segundo programa é o simulador de circuitos lógicos (capítulo 8). O segundo utiliza a porta simulada no primeiro. A comutação entre esses programas pode ser feita a qualquer ir

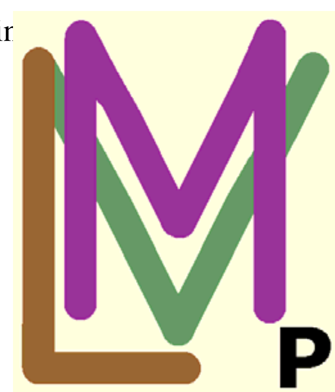


Figura 7.1 – Logotipo.

Figura 7.2 – Janela de apresentação.

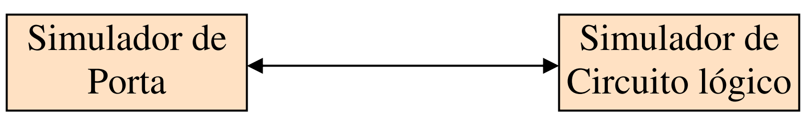


Figura 7.3 – Os dois programas.

As portas simuláveis estão dentro das seguintes categorias:

- Quanto aos valores (**v**), as portas podem ter dois, três ou quatro.
- Quanto às entradas (e), as portas podem ter uma, duas ou três.

- A quantidade de <u>combinações de entrada</u> (**c**) é dada pela fórmula (1) e mostrada na tabela 7.1.
- A <u>quantidade de funções</u> (**f**) permitida para cada combinação “**v**” x “**e**” pode atingir valores muito grandes e é dada pela fórmula (2) e mostrada na tabela 7.2.

$$c = v^{e} \quad (1)$$

$$f = v^{c} \quad (2)$$

Entradas \ Valores	2	3	4
1	2	3	4
2	4	9	16
3	8	27	64

Tabela 7.1 – Combinações de entrada.

Entradas \ Valores	2	3	4
1	2	27	256
2	16	$1{,}96 \cdot 10^{4}$	$4{,}30 \cdot 10^{9}$
3	256	$7{,}62 \cdot 10^{12}$	$3{,}40 \cdot 10^{38}$

Tabela 7.2 – Quantidade de funções.

7.2 Menus e opções

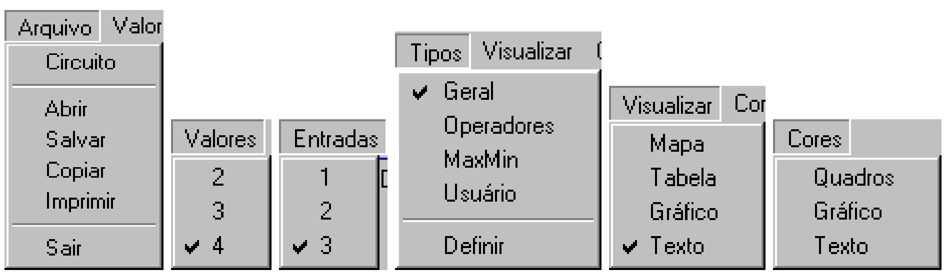


Figura 7.4 – Menus.

- **Arquivo**: Menu com opções de manipulação de arquivos ou do programa.
- **Valores**: Permite escolher a quantidade de valores da função (2,3 e 4).
- **Entradas**: Permite escolher a quantidade de entradas da função (1,2 e 3).
- **Tipos**: Mostra os tipos de lógica a ser empregada.
- **Visualizar**: Mostra as opções de visualização.
- **Cores**: Menu com opções de cores dos itens das janelas (apêndice 3).

Há quatro tipos de portas lógicas selecionáveis no menu “*Tipos*”:

1. Geral
2. Operadores
3. MaxMin
4. Usuário

<u>**1-Geral:**</u>

Este é, dos quatro tipos, o único que pode ser usado na construção de circuitos. Muito embora o simulador de circuitos exija uma definição de porta lógica na forma geral, isso não impossibilita o uso das portas definidas por *Operadores* e *MaxMin*, pois todas as funções definíveis por esses dois tipos podem ser definidas também pelo modo geral. O modo geral compreende todas as funções de *Operadores* e *MaxMin* e, ainda, outras mais. É o tipo de porta mais genérico. Seus parâmetros são:

- **Primeiro**: Define qual é o primeiro valor na hierarquia de dominâncias. Esta opção é desabilitada se houver apenas uma entrada.
- **Deslocamento**: Mostra quantos deslocamentos o valor de saída sofre.
- **Lógica**: Mostra o sentido da hierarquia das dominâncias. Se o valor "0" for selecionado, o valor secundário é o equivalente TOPO do valor dominante. Se o valor "1" for selecionado, o valor secundário é o equivalente BASE do valor dominante. Esta opção é desabilitada se houver apenas uma entrada.

Os dois primeiros parâmetros recebem tantas opções de escolha quanto for a quantidade de valores da base adotada, são parâmetros MVL. O último parâmetro aceita apenas duas opções, é um parâmetro binário.

2-Operadores:

Este tipo de porta consiste nos operadores usados no aplicativo LMVS.

- **Porta**: Permite escolher entre ALFA, BETA, GAMA e DELTA. Esta opção é desabilitada se houver apenas uma entrada.
- **Deslocamento**: Mostra quantos deslocamentos TOPO o valor de saída sofre.

O primeiro parâmetro recebe tantas opções de escolha quanto for a quantidade de valores da base adotada, são parâmetros MVL. O segundo parâmetro aceita apenas duas opções, é um parâmetro binário.

3-MaxMin:

Este tipo de porta consiste nos operadores definidos por Lukasiewicz. Os dois parâmetros são binários.

- **Função**: Permite escolher entre Mínimo e Máximo. Esta opção é desabilitada se houver apenas uma entrada.
- **Inversor**: Permite acionar o inversor de Lukasiewicz.

4-Usuário:

Este tipo de porta não fornece nenhum parâmetro. A cada combinação de valores de entrada é definido um valor de saída. Não há nenhum conceito teórico envolvido neste tipo de definição de porta lógica. A configuração da função é feita clicando-se em "*Definir*".

Figura 7.5 – Parâmetros.

Primeiro	✓		
Porta		✓	
Deslocamento	✓	✓	
Lógica	✓		
Função			✓
Inversor			✓

Tabela 7.3 – Uso dos parâmetros.

Valores	Entradas	Geral	Oprd	MxMn
2	1	2	2	2
3	1	3	3	2
4	1	4	4	2
2	2	8	4	4
3	2	18	9	4
4	2	32	16	4
2	3	8	4	4
3	3	18	9	4
4	3	32	16	4
	Total	125	67	30

Tabela 7.4 – Combinações de opções.

As três primeiras opções (figura 7.5) têm tantos itens quanto for a base adotada, são MVL. As três últimas têm duas opções, são binárias. Os parâmetros genéricos são configurados por meio dos menus e sua configuração pode ser vista por meio dos símbolos de opções que aparecem nos menus. Os parâmetros específicos são configurados dentro das janelas por meio de objetos *OptionButtons* e *CheckBoxes*. Levando-se em conta que as portas têm três possibilidades para os valores, três para as entradas e quatro para os tipos, tem-se 36 tipos de porta.

- Parâmetros
  - Genéricos
    - *Valores*
    - *Entradas*
    - *Tipos*
  - Específicos
    - MVL
      - *Primeiro*
      - *Porta*
      - *Deslocamento*
    - Binários
      - *Lógica*
      - *Função*
      - *Inversor*

Figura 7.6 – Parâmetros de configuração.

Há quatro tipos de visualização: 

1. **<u>Mapa</u>**: A palavra *Mapa* foi escolhida por causa do termo *Mapa de Karnaugh*. A porta é mostrada por meio das tabelas de combinação da entrada.

2. **<u>Tabela</u>**: Embora o modo de visualização por mapa também consista de uma tabela, este modo fornece uma tabela linear. Trata-se de um meio-termo entre a visualização por mapa e a visualização por gráfico. A tabela é disposta nas formas horizontal e vertical.

3. **<u>Gráfico</u>**: Este modo expõe o conteúdo da tabela em forma de gráfico.

4. **<u>Texto</u>**: A principal vantagem deste modo é a possibilidade de copiar o texto e salvar em um documento. Fornece diversas tabelas com formatações diferentes.

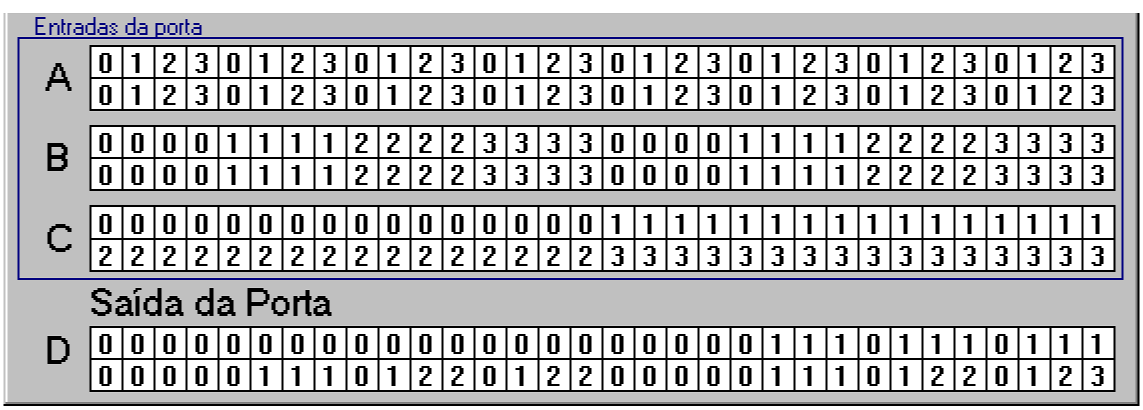
<u>Menu arquivo:</u>

Este é o menu mais genérico do software.

1. **Circuito**: Permite ir para o simulador de circuitos.
2. **Abrir**: Restaura um arquivo salvo anteriormente.
3. **Salvar**: Grava a função vigente em um arquivo.
4. **Copiar**: Envia para a memória a função vigente.
5. **Imprimir**: Envia para a impressora a função vigente.
6. **Sair**: Encerra a execução do programa.

7.3 Mapas e tabelas

Mapas são as matrizes com as combinações de entrada e o valor da saída para a referida combinação. Tabelas são os mapas apresentados na forma linear.



Entradas da porta

A	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
B	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Saída da Porta

D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	1	1	1
	0	0	0	0	0	1	1	1	0	1	2	2	0	1	2	2	0	0	0	0	0	1	1	1	0	1	2	2	0	1	2	3

Figura 7.7 – Tabela para porta Mínimo de quatro valores e três entradas.

2 valores
1 entrada

A	0	1
	0	1

3 valores
1 entrada

A	0	1	2
	0	1	2

4 valores
1 entrada

A	0	1	2	3
	0	1	2	3

2 valores, 3 entradas

C=0

B \ A	0	1
0	0	0
1	0	0

C=1

B \ A	0	1
0	0	0
1	0	1

2 valores
2 entradas

B \ A	0	1
0	0	0
1	0	1

3 valores, 3 entradas

C=0

B \ A	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0

C=1

B \ A	0	1	2
0	0	0	0
1	0	1	1
2	0	1	1

C=2

B \ A	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

4 valores, 3 entradas

C=0

B \ A	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

C=1

B \ A	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	1	1
3	0	1	1	1

C=2

B \ A	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	2

C=3

B \ A	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

3 valores
2 entradas

B \ A	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

4 valores
2 entradas

B \ A	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

Figura 7.8 – Mapas para a porta Mínimo.

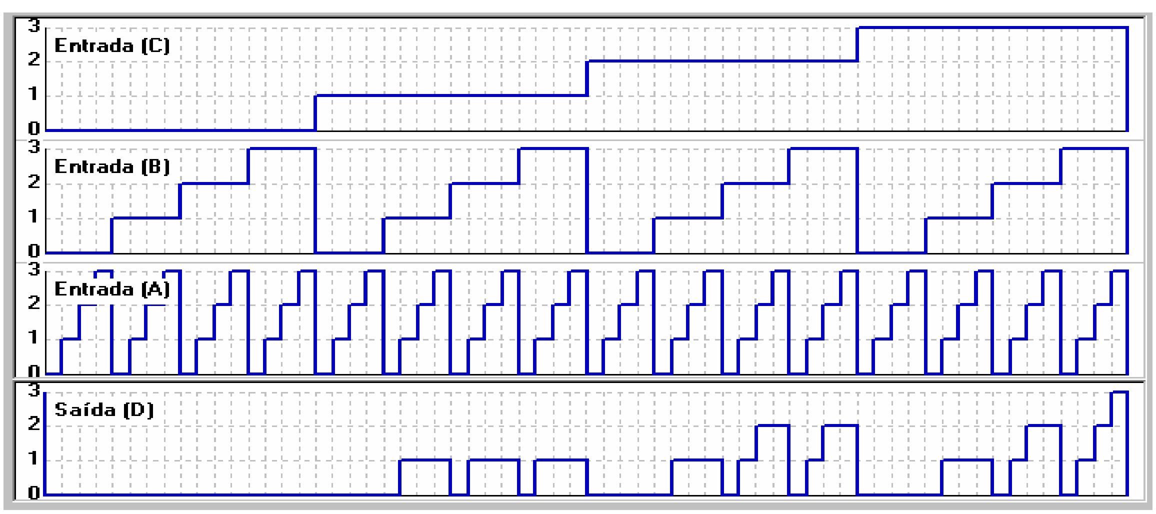
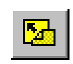


Figura 7.9 – Gráfico para porta Mínimo de quatro valores e três entradas.

Figura 7.10 – Definição da função do usuário.

No menu tipos, existe a opção para se definir a função do usuário, trata-se da única janela do simulador de porta que aceita entrada de valores por meio de caixas de texto. No modo de visualização por texto, o conteúdo apresentado pode ser copiado e editado em outro aplicativo, como, por exemplo, o bloco de notas (*notepad*) do Windows. É a única janela do simulador de porta que possui o botão de copiar texto da jenela . 

O texto pode ser subdividido em três partes:

1. Mapa
2. Tabela horizontal
3. Tabela vertical

A visualização por texto mostra o mesmo conteúdo das janelas dos modos de visualização por mapa e por tabela, com uma vantagem e desvantagem:

Vantagem: Permite que a informação seja transposta para outro documento.
Desvantagem: Visualização mais difícil, sem possibilidade de uso de cores.

É mostrado um exemplo para cada uma das nove possibilidades de valores versus entradas. Para facilitar a comparação e evitar desperdício de espaço, as três partes são apresentadas separadamente. Usa-se a variável "A" como a menos significativa e a "C" como a mais significativa, respeitando a metodologia empregada no software LMVS. A primeira variável é a menos significativa, de modo que uma variável é uma ordem de grandeza maior do que a anterior. Sendo assim, nos mapas, a leitura é feita da maneira como o ser humano da cultura greco-romana escreve: varredura em linhas, da esquerda para a direita.

```
     |--|
     | A|
     |01|
|----+--|
|  B |01|
|----+--|

     |---|
     | A |
     |012|
|----+---|
|  B |012|
|----+---|

     |----|
     | A  |
     |0123|
|----+----|
|  B |0123|
|----+----|

     |--|
     | A|
   C |01|
|----+--|
|   0|00|
|B 1 |01|
|----+--|

     |---|
     | A |
   C |012|
|----+---|
|   0|000|
|B 1 |011|
|   2|012|
|----+---|

     |----|
     | A  |
   C |0123|
|----+----|
|   0|0000|
|B 1 |0111|
|   2|0122|
|   3|0123|
|----+----|
```

Listagem 7.1 – Mapa para funções de uma e duas entradas.

```
     |--|
     | A|
  C=0|01|
|----+--|
|   0|00|
|B 1 |00|
|----+--|

     |--|
     | A|
  C=1|01|
|----+--|
|   0|00|
|B 1 |01|
|----+--|

     |---|
     | A |
  C=0|012|
|----+---|
|   0|000|
|B 1 |000|
|   2|000|
|----+---|

     |---|
     | A |
  C=1|012|
|----+---|
|   0|000|
|B 1 |011|
|   2|011|
|----+---|

     |---|
     | A |
  C=2|012|
|----+---|
|   0|000|
|B 1 |011|
|   2|012|
|----+---|

     |----|
     | A  |
  C=0|0123|
|----+----|
|   0|0000|
|B 1 |0000|
|   2|0000|
|   3|0000|
|----+----|

     |----|
     | A  |
  C=1|0123|
|----+----|
|   0|0000|
|B 1 |0111|
|   2|0111|
|   3|0111|
|----+----|

     |----|
     | A  |
  C=2|0123|
|----+----|
|   0|0000|
|B 1 |0111|
|   2|0122|
|   3|0122|
|----+----|

     |----|
     | A  |
  C=3|0123|
|----+----|
|   0|0000|
|B 1 |0111|
|   2|0122|
|   3|0123|
|----+----|
```

Listagem 7.2 – Mapa para funções de três entradas.

```
A saída
0    0
1    1

A saída
0    0
1    1
2    2

A saída
0    0
1    1
2    2
3    3
```

```
AB  saída
00    0
10    0
01    0
11    1

AB  saída
00    0
10    0
20    0
01    0
11    1
21    1
02    0
12    1
22    2

AB  saída
00    0
10    0
20    0
30    0
01    0
11    1
21    1
31    1
02    0
12    1
22    2
32    2
03    0
13    1
23    2
33    3
```

```
ABC  saída
000    0
100    0
010    0
110    0
001    0
101    0
011    0
111    1
```

```
ABC  saída
000    0
100    0
200    0
010    0
110    0
210    0
020    0
120    0
220    0
001    0
101    0
201    0
011    0
111    1
211    1
021    0
121    1
221    1
002    0
102    0
202    0
012    0
112    1
212    1
022    0
122    1
222    2
```

```
ABC  saída
000    0        002    0
100    0        102    0
200    0        202    0
300    0        302    0
010    0        012    0
110    0        112    1
210    0        212    1
310    0        312    1
020    0        022    0
120    0        122    1
220    0        222    2
320    0        322    2
030    0        032    0
130    0        132    1
230    0        232    2
330    0        332    2
001    0        003    0
101    0        103    0
201    0        203    0
301    0        303    0
011    0        013    0
111    1        113    1
211    1        213    1
311    1        313    1
021    0        023    0
121    1        123    1
221    1        223    2
321    1        323    2
031    0        033    0
131    1        133    1
231    1        233    2
331    1        333    3
```

Listagem 7.3 – Tabela vertical.

```
    A   01                  A   012                 A   0123
saída 01              saída 012               saída 0123
```

Listagem 7.5 – Tabela horizontal para funções de uma entrada.

```
    A   0101              A   012012012         A   0123012301230123
    B   0011              B   000111222         B   0000111122223333
saída 0001            saída 000011012       saída 0000011101220123
```

Listagem 7.6 – Tabela horizontal para funções de duas entradas.

```
    A   01010101                A   012012012012012012012012012
    B   00110011                B   000111222000111222000111222
    C   00001111                C   000000000111111111222222222
saída 00000001              saída 000000000000011011000011012

    A   0123012301230123012301230123012301230123012301230123012301230123
    B   0000111122223333000011112222333300001111222233330000111122223333
    C   0000000000000000111111111111111122222222222222223333333333333333
saída 0000000000000000000011101110111000001110122012200000111011220123
```

Listagem 7.7 – Tabela horizontal para funções de três entradas.

A porta simulada é aquela que é empregada no simulador de circuitos. A simulação de cada porta recebe uma quantidade de passos relativa à quantidade de valores e de entradas. Na definição da porta do usuário, o número de combinações disponíveis é o mesmo relativo à análise combinatória.

Valores	Entradas	Passos	Varredura das combinações
2	1	2	**A** 01
2	2	4	**A** 0101 **B** 0011
2	3	8	**A** 01010101 **B** 00110011 **C** 00001111
3	1	3	**A** 012
3	2	9	**A** 012012012 **B** 000111222
3	3	27	**A** 012012012012012012012012012 **B** 000111222000111222000111222 **C** 000000000111111111222222222
4	1	4	0123
4	2	16	**A** 0123012301230123 **B** 0000111122223333
4	3	64	**A** 0123012301230123012301230123012301230123012301230123012301230123 **B** 0000111122223333000011112222333300001111222233330000111122223333 **C** 0000000000000000111111111111111122222222222222223333333333333333

Tabela 7.5 – Estímulo padrão para cada opção (varredura da análise combinatória das entradas).

Para finalizar a abordagem sobre o simulador de portas, é apresentado um exemplo de porta definida pelo usuário.

```
       |----|       |----|       |----|       |----|
       | A  |       | A  |       | A  |       | A  |
    C=0|0123|    C=1|0123|    C=2|0123|    C=3|0123|
  |----+----|  |----+----|  |----+----|  |----+----|
  |  0 |0223|  |  0 |3133|  |  0 |2110|  |  0 |3110|
  |B 1 |0002|  |B 1 |3222|  |B 1 |1112|  |B 1 |2013|
  |  2 |1211|  |  2 |2011|  |  2 |3032|  |  2 |2331|
  |  3 |1012|  |  3 |3020|  |  3 |2312|  |  3 |3213|
  |----+----|  |----+----|  |----+----|  |----+----|

    A   0123012301230123012301230123012301230123012301230123012301230123
    B   0000111122223333000011112222333300001111222233330000111122223333
    C   0000000000000000111111111111111122222222222222223333333333333333
saída   0223000212111012313332222011302021101112303223123110201323313213
```

Listagem 7.8 – O texto para a porta do usuário (sem a tabela vertical).

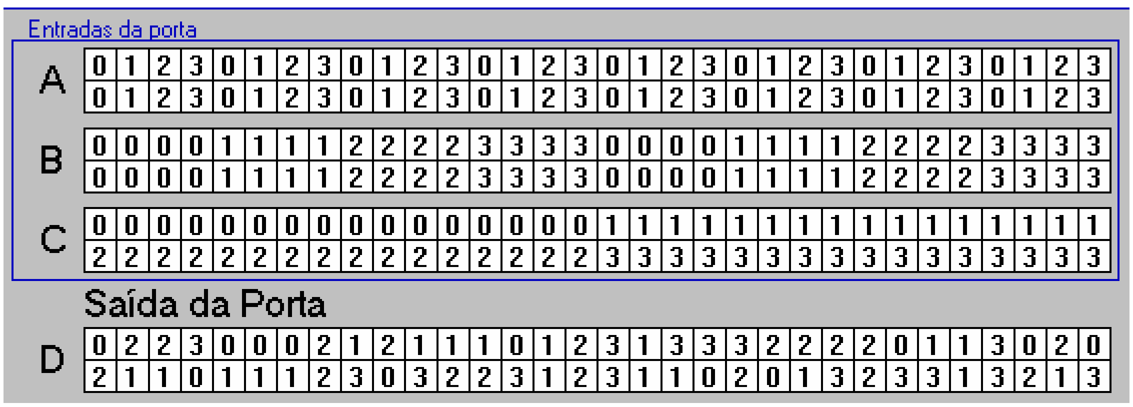


Figura 7.11 – Tabela para a porta do usuário.

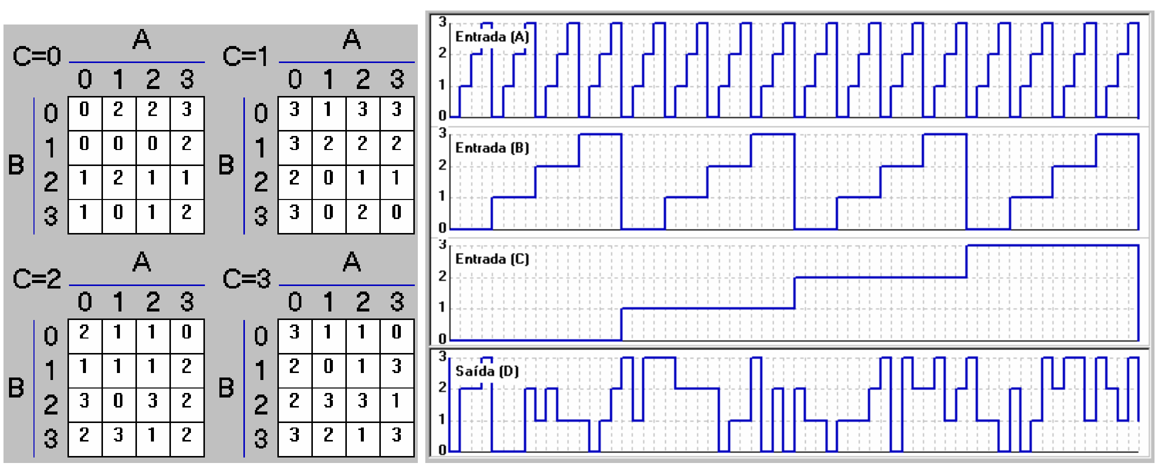


Figura 7.12 – Mapa e gráfico para a porta do usuário.

No capítulo 8, é apresentado o simulador de circuitos, que utiliza os dados provenientes do simulador de portas para construir circuitos MVL.

Capítulo 8
LMVP – Simulador de circuitos lógicos

8.1 Introdução

Ao contrário do software LMVS e do simulador de portas, o simulador de circuitos lógicos é baseado em uma janela principal, onde se pode acessar as janelas secundárias. Esta janela principal mostra as conexões de entrada das portas do circuito. A janela de conexões é onde o circuito é construído. Nesta janela, não é feita nenhuma simulação. Por outro lado, nas janelas de simulação, nenhuma alteração pode ser feita no circuito lógico.

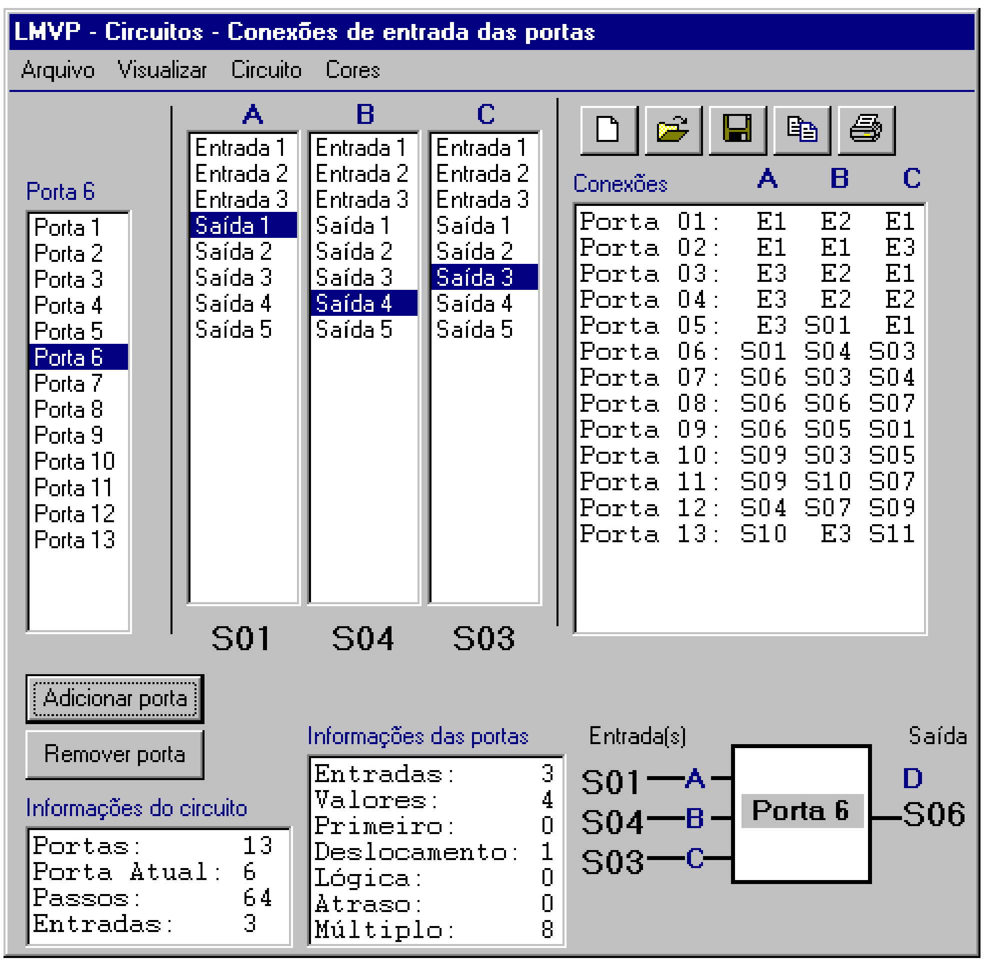


Figura 8.1 – A janela de conexões.

O simulador de circuitos lógicos faz apenas uma análise estática de circuitos combinacionais[19]. Ele não aborda circuitos seqüenciais, com análise de tempo e *clock*, somente circuitos quase-estáticos podem ser simulados. Para evitar que circuitos biestáveis (ou poliestáveis, no caso MVL) sejam construídos, é vetado o uso de realimentações[6]. Esse veto também impede a construção de circuitos astáveis (osciladores de relaxação).

Uma realimentação consiste numa porta cujo sinal em uma de suas entradas é uma função dependente do valor da saída desta porta em um momento anterior. A maneira encontrada para forçar a criação de circuitos seqüenciais foi o estabelecimento de hierarquia de portas, por meio da qual se determina que uma porta só pode receber, em suas entradas, os valores das entradas do circuito e os valores das saídas das portas de hierarquia menor. Se uma porta receber, em uma de suas entradas, o valor de saída de uma porta de hierarquia maior, e esta outra porta tiver, em uma de suas entradas, um sinal que seja diretamente ou indiretamente dependente do sinal de saída da primeira porta, tem-se uma realimentação. O simulador só admite um tipo de porta lógica por circuito. A implementação da possibilidade de se usar mais de um tipo de porta num mesmo circuito não tem incentivo pois os trabalhos voltados para o projeto destes circuitos costumam usar sempre um único tipo de porta. É possível, porém, simular circuitos com portas com número de entradas diferentes, basta, apenas, usar somente portas com o número de entradas referente à porta com maior número de entradas, e, para aquelas que usarem menos entradas, usar uma porta com uma das entradas ligada ao valor de menor dominância para a lógica empregada na porta ou então usar duas entradas ligadas ao mesmo sinal.

A janela de conexões isenta o usuário da tarefa de criar um arquivo texto com a matriz das conexões e impede-o de fornecer uma matriz inválida. Ela incorpora muita informação ao mesmo tempo, todas as informações necessárias para que se saiba como o circuito deve funcionar. Ela não faz qualquer uso da quantidade de valores, pois as conexões das portas do circuito não dizem respeito à base adotada.

8.1.1 – Primeira Parte: Selecionador de porta

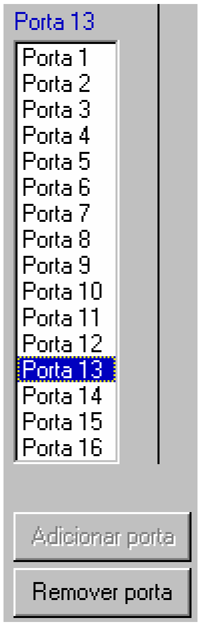


Figura 8.2 – A janela de conexões.

Por meio de um clicar sobre a lista de portas (objeto *ListBox*), a porta vigente pode ser escolhida. A quantidade de portas mostrada na caixa é o número de portas do circuito. Acima, há uma legenda indicando qual é a porta selecionada. Abaixo, há dois botões, um para adicionar porta, outro para remover porta. Quando o limite inferior (1 porta) for atingido, o botão de remoção de porta é desabilitado; quando o limite superior de (16 portas) for atingido, o botão de adição de porta é desabilitado. Ao adicionar a porta, automaticamente a porta selecionada é aquela que foi adicionada. Se, ao remover uma porta, a porta removida for a selecionada, então a porta selecionada passa a ser a de maior hierarquia. Ao se remover uma porta, suas conexões não são perdidas. Ao voltar a adicioná-la, suas informações são recuperadas.

8.1.2 – Segunda Parte: Conexões para a porta selecionada

Essas conexões dependem da hierarquia da porta selecionada, da quantidade de entradas das portas e da quantidade de entradas do circuito. São apresentadas tantas caixas de lista de portas quanto forem as entradas das portas, que podem variar de 1 a 3 (A, B, C). Para uma dada porta selecionada, as opções correspondem às entradas do circuito e as saídas das portas com hierarquia inferior. As saídas das portas são escritas alinhadamente à caixa de seleção de portas. Acima de cada coluna há um rótulo com o nome da entrada da porta (A,B ou C) e abaixo de cada coluna está escrito qual é a conexão feita para a referida entrada da porta. A letra “*S*” indica tratar-se da saída de uma porta com hierarquia inferior, a letra “*E*” indica tratar-se de uma das entradas do circuito. A seguir, são apresentados alguns exemplos. O primeiro é o da porta 16 de um circuito de 3 entradas com portas de 3 entradas. O segundo é o da porta 11 de um circuito de 2 entradas com portas de 3 entradas. O terceiro é o da porta 15 de um circuito de 1 entrada com portas de 2 entradas.

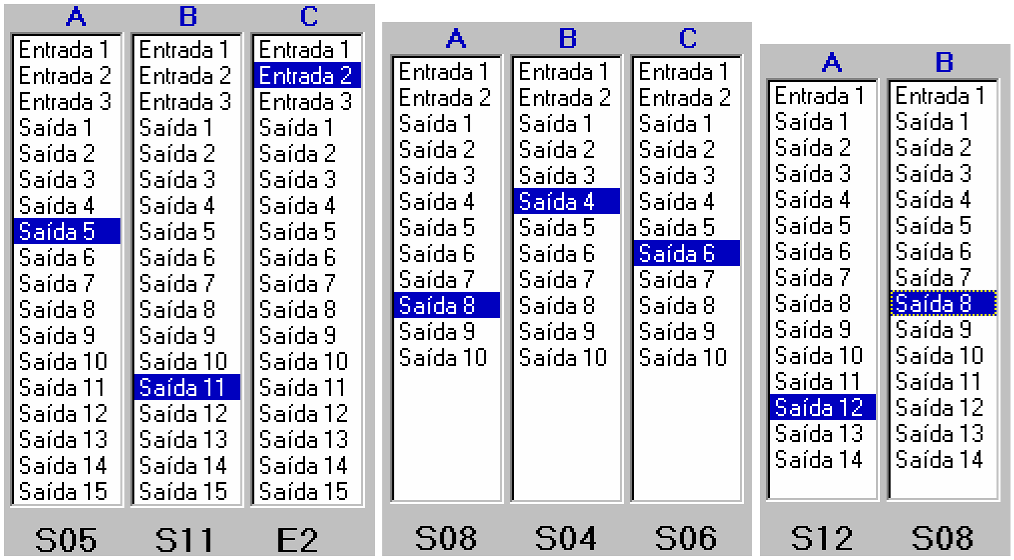


Figura 8.3 – Exemplos das caixas de conexões.

8.1.3 – Terceira Parte: Resumo das conexões

O resumo é apresentado em uma caixa de texto (objeto *TextBox*). É a área mais importante da janela, trata-se do resultado de todo o trabalho de conexões. É a informação que é enviada ao processo de simulação do circuito. As conexões das portas são escritas alinhadamente à caixa de seleção de portas. Cada linha representa as entradas de uma porta. A primeira coluna de conexões corresponde à entrada *A* da porta, a segunda à entrada *B* da porta, e assim por diante. Este texto é selecionável e copiável.

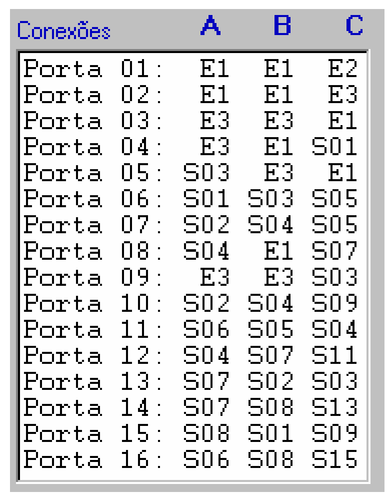


Figura 8.4 – Área de exibição do resumo.

8.1.4 – Quarta Parte: Informações da simulação

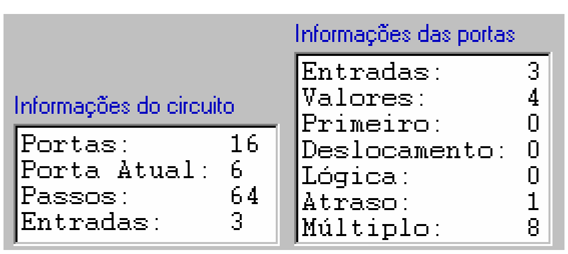


Figura 8.5 – Exemplos das caixas de conexões.

São duas caixas de texto com as informações. A primeira caixa contém as informações do circuito, e a segunda as informações das portas usadas no circuito. Estes textos são selecionáveis e copiáveis.

<u>Informações do circuito:</u>

1. **Portas**: Indica quantas portas tem o circuito.
2. **Porta atual**: Indica qual a porta selecionada.
3. **Passos**: Este valor não é definível pelo usuário, é calculado automaticamente.
4. **Entradas**: Indica quantas entradas tem o circuito.

As portas podem ter dois, três ou quatro valores (v). O circuito pode ter uma, duas ou três entradas (e). A quantia de passos (c) é dado pela fórmula (1).

$$\boxed{c = v^e \quad (1)}$$

Entradas \ Valores	2	3	4
1	2	3	4
2	4	9	16
3	8	27	64

Tabela 8.1 – Passos.

<u>Informações da porta:</u>

Os cinco primeiros parâmetros são configurados no simulador de porta.

1. **Entradas**: Permite escolher a quantidade de entradas da função (1,2 e 3).
2. **Valores**: Permite escolher a quantidade de valores da função (2,3 e 4).
3. **Primeiro**: Define qual é o primeiro valor na hierarquia de dominâncias.
4. **Deslocamento**: Mostra quantos deslocamentos o valor de saída sofre.
5. **Lógica**: Mostra o sentido da hierarquia das dominâncias.
6. **Atraso**: Indica se o simulador deve considerar o atraso das portas.
7. **Múltiplo**: Indica o quanto as portas atrasam.

O parâmetro *Múltiplo* indica quantas vezes a largura de pulso (por nível) do sinal de entrada é maior do que o atraso de cada porta. O atraso corresponde ao inverso do valor dado pelo parâmetro *Múltiplo* que varia entre dois e oito, sendo que o valor dois implica num atraso de meio pulso, e o valor oito implica em um atraso de um oitavo de pulso. Valores de tempo de subida (*Rise Time*) e de descida (*Fall Time*) não são considerados.

Múltiplo	Atraso (%)
2	50
3	33,33
4	25
5	20
6	16,67
7	14,29
8	12,5

Tabela 8.2 – Atraso percentual para cada valor da variável *múltiplo*.

O parâmetro *atraso* recebe os seguintes valores:

0. Sem Atraso
1. Com Atraso
2. Ambos

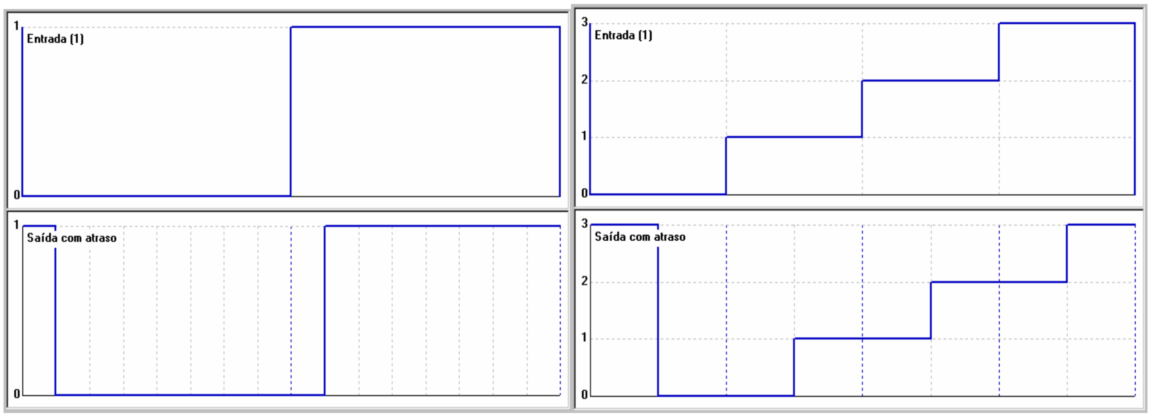


Figura 8.6 – Atraso de um oitavo do pulso (Múltiplo = 8) para dois valores
Atraso de de meio pulso (Múltiplo = 2) para quatro valores.

8.1.5 – Quinta Parte: Desenho da porta selecionada

No lado esquerdo estão as entradas da porta, e no lado direito está a saída. A porta é representada pelo bloco no centro. Sobre a porta, há um rótulo com o nome da porta. São incluídos, também, os nomes das variáveis de entrada e de saída (A,B,C e D). Abaixo, tem-se um exemplo para portas de uma, duas e três entradas.

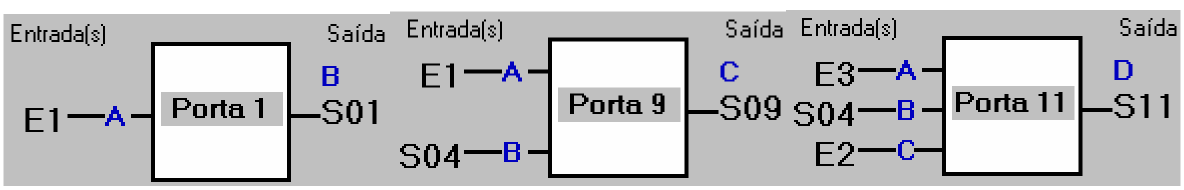


Figura 8.7 – Atraso de meio pulso (Múltiplo = 2).

8.2 Menus

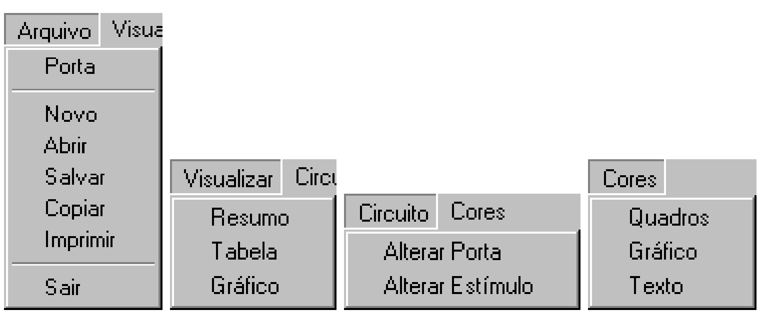
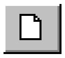


Figura 8.8 – Menus.

- **Arquivo**: Menu com opções de manipulação de arquivos ou do programa.
- **Visualizar**: Permite escolher qual o modo de visualização do circuito.
- **Circuito**: Permite alterar as propriedades do circuito.
- **Cores**: Menu com opções de cores dos itens das janelas (apêndice 4).

Menu *Arquivo*:

Alguns destes comandos são fornecidos, também, por botões.

1. **Porta**: Permite ir para o simulador de portas.
2.  **Novo**: Inicia um novo circuito.
3.  **Abrir**: Restaura um arquivo salvo anteriormente.
4.  **Salvar**: Grava a função vigente em um arquivo.
5.  **Copiar**: Envia para a memória a função vigente.
6.  **Imprimir**: Envia para a impressora a função vigente.
7. **Sair**: Encerra a execução do programa.

Ao criar um novo circuito, todos os contatos recebem a entrada *1* do circuito (E1), mas a quantidade de portas não se altera.

- **Menu Visualizar**: Este menu fornece acesso a três janelas de visualização do circuito. É nelas que a simulação é feita exibida. Essas janelas serão apresentadas mais adiante.
- **Menu Circuito**: Este menu fornece acesso a opções de modificação das características do circuito sem perder as configurações das conexões. Essas opções serão mostradas mais adiante.

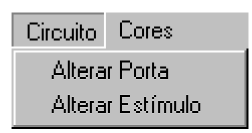
8.3 Janelas

Há três modos de visualização para os circuitos:

1. Resumo
2. Tabela
3. Gráfico

Há duas janelas de configuração para os circuitos:

1. Alterar Porta
2. Alterar Estímulo



A seguir, é feita uma análise mais profunda destas janelas. Ao fazer a simulação do circuito, é criada uma matriz com os valores das saídas de cada porta (linhas) para cada combinação de entrada do circuito (colunas). A quantidade de colunas, ou seja, de combinações de entrada do circuito, é dada pela variável *passos* e é obtida pela fórmula $c=v^e$ mostrada anteriormente. Cada linha representa a saída de uma porta. A matriz da simulação é criada durante a abertura de uma das três janelas de visualização (*Resumo*, *Tabela* ou *Gráfico*), contidas no menu *visualizar*. Se o circuito tiver, por exemplo, três entradas e dezesseis portas de quatro valores, a matriz da simulação tem tamanho de 64 x 16. As matrizes de conexões e de simulação possuem o mesmo número de linhas. Uma vez criada a matriz de simulação, janelas de visualização apenas mostram o conteúdo dessa matriz, de maneiras diferentes, mas não fazem qualquer tipo de manipulação do circuito. Se o atraso das portas for considerado, então são criados, para cada passo, vários sub-passos. A quantidade de sub-passos é fornecida pela variável *múltiplo*. Por exemplo, um atraso de 50% equivale a *múltiplo*=2, são considerados dois sub-passos para cada passo de simulação. Se o circuito tiver, por exemplo, três entradas e dezesseis portas de quatro valores, e *múltiplo*=8, o tamanho da matriz da simulação é de (64×8) x 16 = 8192.

8.3.1 – Resumo

Esta janela apresenta duas caixas de texto, ambas selecionáveis e copiáveis. A da direita contém a matriz das conexões apresentada na janela de conexões. A da esquerda apresenta a matriz da simulação. As linhas da matriz de simulação estão de acordo com as linhas da matriz de contatos, de modo a corresponder as linhas com as portas. Se a opção para a variável *atraso* for 2 (Ambos), ou seja, exposição das simulações com e sem atraso, duas caixas de opção (*OptionButton*) aparecem abaixo da caixa de texto da matriz de simulação, permitindo escolher qual matriz é exposta (com ou sem atraso).

A exposição feita nesta janela permite exportar os dados obtidos para outros aplicativos de plotagem, ou, simplesmente, salvos na forma de texto, afim de serem usados em outros documentos. Dependendo do tamanho da fonte utilizada (escolhida na janela de configuração das cores do texto), o texto pode não caber no quadro de texto e, nesse caso, as barras de rolagem (*scroll*) devem ser utilizadas. Acima da matriz da simulação, há uma matriz de combinações dos valores das entradas do circuito.

Essa matriz não deve ser confundida com a matriz gerada no modo de visualização da porta por texto pois aquela matriz leva em consideração as entradas da porta, não as do circuito. O bot  transfere o texto da simulação do circuito para a memória.

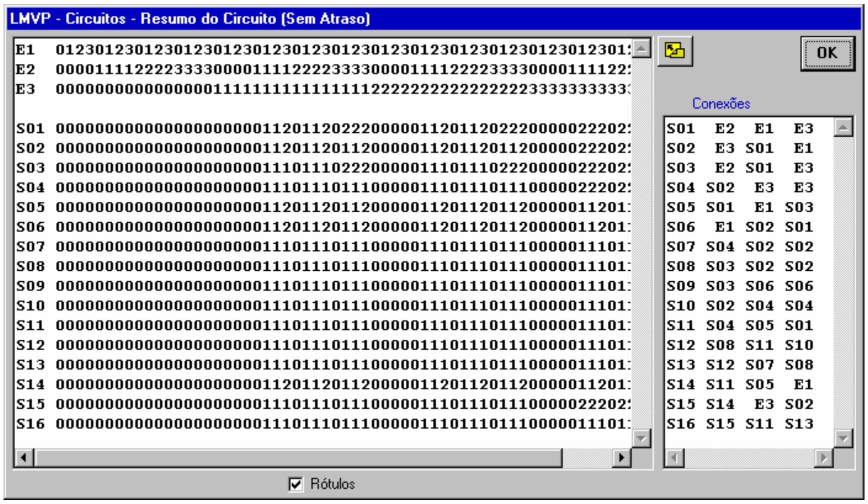


Figura 8.9 – A janela do resumo (com rótulos).

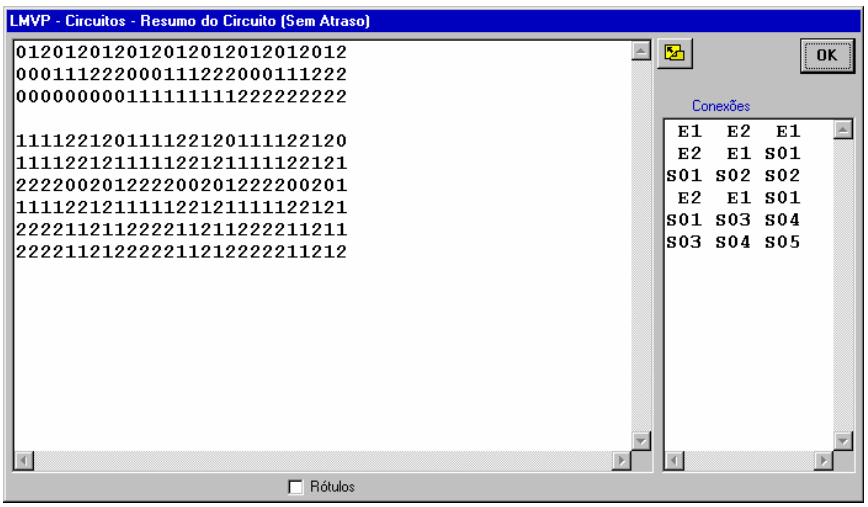


Figura 8.10 – A janela do resumo (sem rótulos).

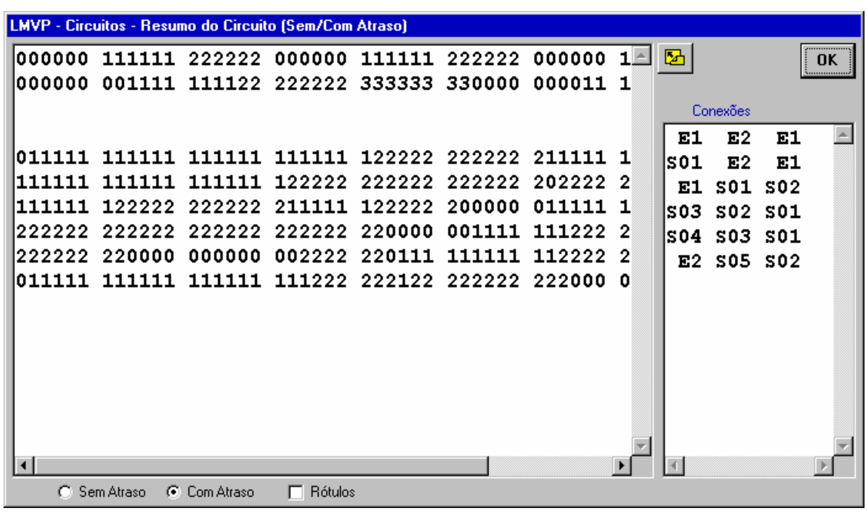
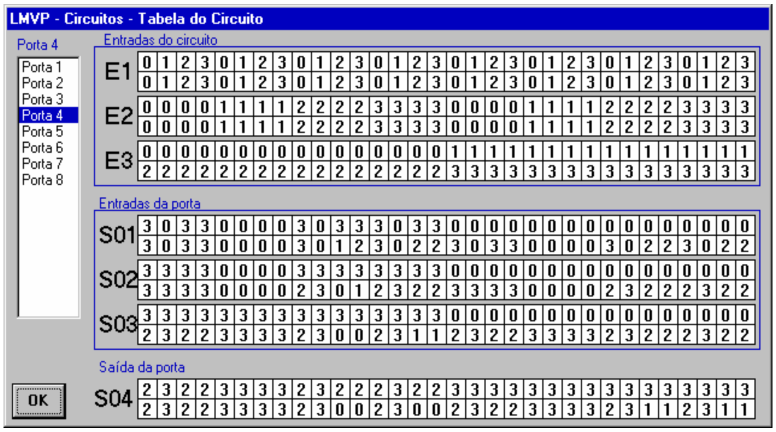


Figura 8.11 – A janela do resumo (com atraso).

8.3.2 – Tabela

Esta janela apresenta diversas tabelas na forma de caixas coloridas (caso as cores estejam ativadas). O conteúdo desta janela pode ser dividido em quatro partes:



1. Caixa de seleção da porta.
2. Entradas do circuito.
3. Entradas da porta.
4. Saída da porta.

Figura 8.12 – A janela da tabela.

Por meio de um clicar sobre a lista de portas (objeto *ListBox*), a porta vigente pode ser escolhida, da mesma forma como na janela de edição de conexões. Somente uma porta pode ser exibida por vez. A parte que mostra as entradas do circuito (E1, E2 e E3) não se altera com a seleção da porta. A quantidade de matrizes da área da entrada do circuito depende da quantidade de entradas do circuito, da mesma forma que a quantidade de matrizes da área da entrada da porta depende da quantidade de entradas da porta.

Os rótulos localizados imediatamente à esquerda das matrizes de entrada da porta são determinados pela matriz de conexões. O modo de visualização por tabela não mostra o efeito do atraso das portas. A quantidade de elementos de cada matriz é determinado pela variável *passos* ($c = v^e$). Ao lado, tem-se um exemplo para circuito de duas entradas e portas de uma entrada e dois valores. O tamanho da janela é ajustado ao tamanho das tabelas.

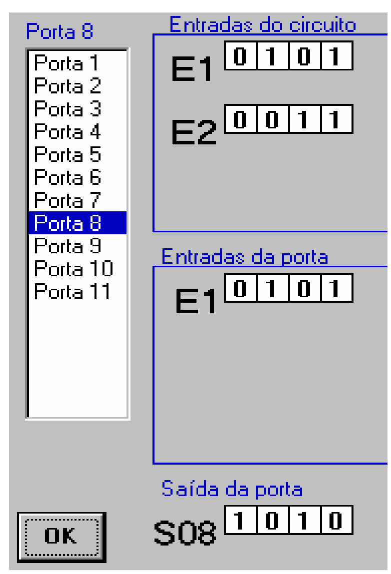


Figura 8.13 – Outro exemplo de tabela.

8.3.3 – Gráfico

Esta janela mostra o mesmo conteúdo da janela da tabela, mas na forma de gráfico. Ao invés de uma caixa de seleção de portas, existe uma barra de rolagem horizontal (objeto HScroll). A amplitude desta caixa corresponde ao número de portas do circuito.

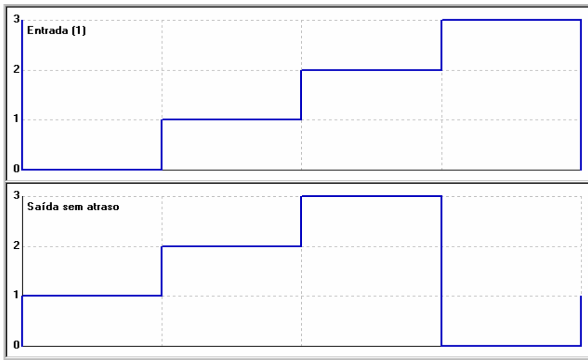


Figura 8.14 – Uma entrada e quatro valores.

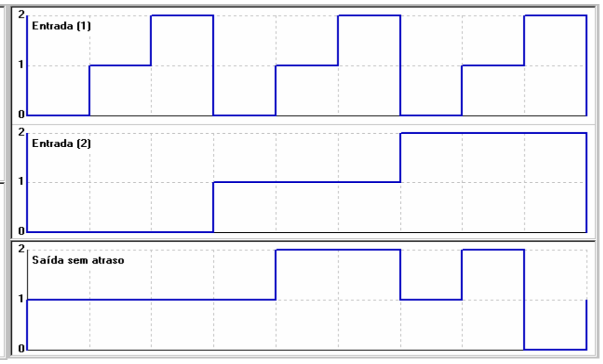


Figura 8.15 – Duas entradas e três valores.

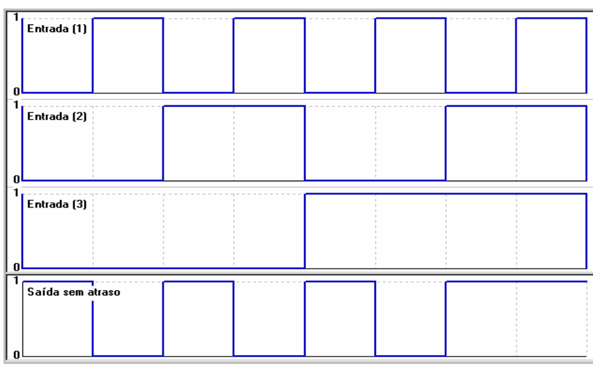


Figura 8.16 – Três entradas e dois valores.

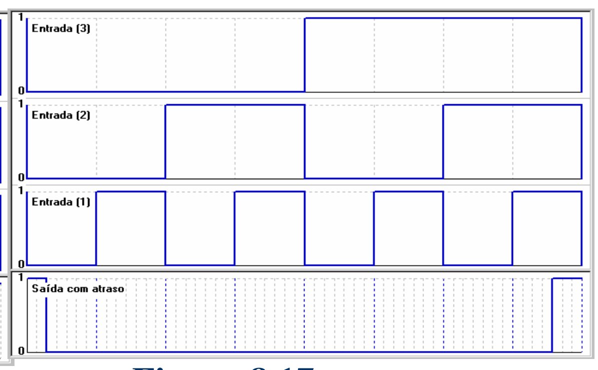


Figura 8.17 – Com atraso.

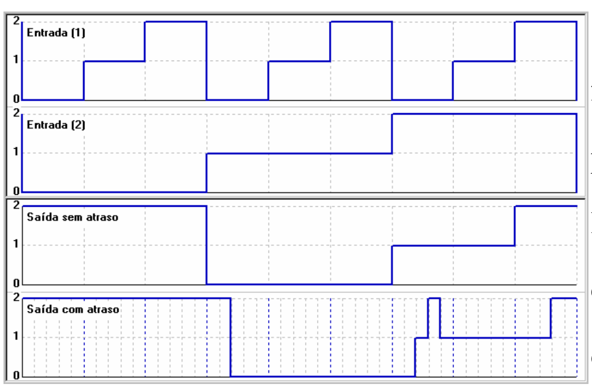


Figura 8.18 Sem e com atraso (ambos).

Por meio da opção ambos, pode-se verificar os problemas causados pelos atrasos das portas. Acima do gráfico, fica a barra de rolagem de escolha da porta, uma caixa de seleção para habilitar ou desabilitar a exposição da grade e outra para os rótulos.

Figura 8.19 – Os controles da janela do gráfico.

8.3.4 – Novo Circuito / Alterar Porta

Estas duas janelas são parecidas. Elas permitem selecionar o tipo de porta quanto ao atraso (sem, com, ambos) e permite definir qual o percentual sobre o atraso (variável *múltiplo*). O múltiplo é configurado por meio de uma barra de rolagem horizontal (*horizontal scroll bar*), que só é habilitada quando a opção de atraso for 1 ou 2.

A opção *Entradas do Circuito* não aparece na janela *Alterar Porta*, trata-se de uma variável utilizada no cálculo do número de passos de simulação, e sua mudança causaria grandes mudanças no circuito e, para fazê-las, torna-se necessário iniciar um novo circuito (matriz de conexões). As mudanças permitidas na janela na janela *Alterar Porta* não alteram a estrutura do circuito nem o número de passos de simulação e podem ser realizadas sem reiniciar a construção do circuito.

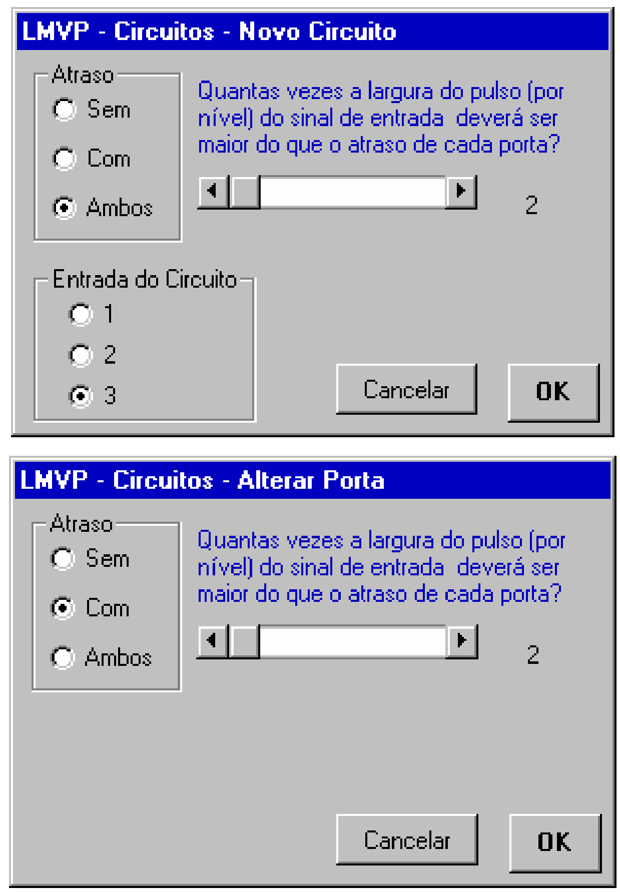


Figura 8.20 – As janela de configuração.

8.3.5 – Alterar Estímulo

Esta janela permite alterar o estímulo com o qual o circuito é simulado. Os estímulos padrão são aqueles que correspondem à análise combinatória das variáveis de entrada do circuito. A quantidade de passos de simulação não pode ser mudada e é determinada pela quantidade de entradas e valores do circuito. Quando se cria um novo circuito, por meio da janela *Novo Circuito*, o estímulo é automaticamente convertido para o padrão. Cada caixa com os valores pode ser alterada com o simples clicar do mouse e a digitação do número.

Um verificador de erros impede que caracteres inválidos sejam colocados.

Figura 8.21 – A janela do configurador do estímulo do circuito.

Valores	Entradas	Passos	Estímulo Padrão
2	1	2	**E1** 01
2	2	4	**E1** 0101 **E2** 0011
2	3	8	**E1** 01010101 **E2** 00110011 **E3** 00001111
3	1	3	012
3	2	9	**E1** 012012012 **E2** 000111222
3	3	27	**E1** 012012012012012012012012012 **E2** 000111222000111222000111222 **E3** 000000000111111111222222222
4	1	4	**E1** 0123
4	2	16	**E1** 0123012301230123 **E2** 0000111122223333
4	3	64	**E1** 0123012301230123012301230123012301230123012301230123012301230123 **E2** 0000111122223333000011112222333300001111222233330000111122223333 **E3** 0000000000000000111111111111111122222222222222223333333333333333

Tabela 8.3 – Estímulo padrão para cada opção (varredura da análise combinatória das entradas).

8.3.6 – Exemplo

A seguir, é mostrado um exemplo de estímulo do usuário, gerado aleatoriamente.

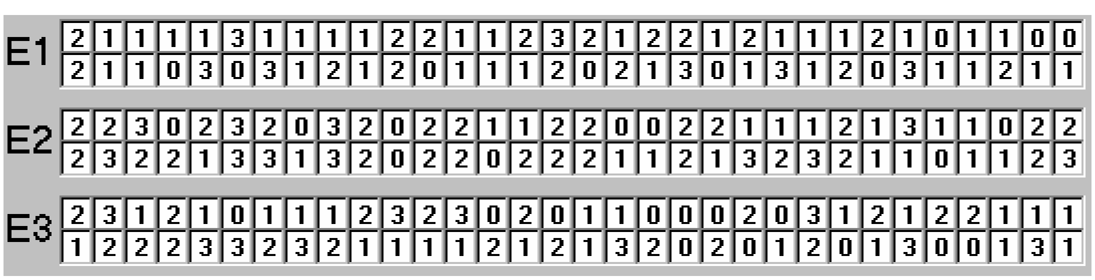


Figura 8.22 – Estímulo do usuário no ambiente de edição.

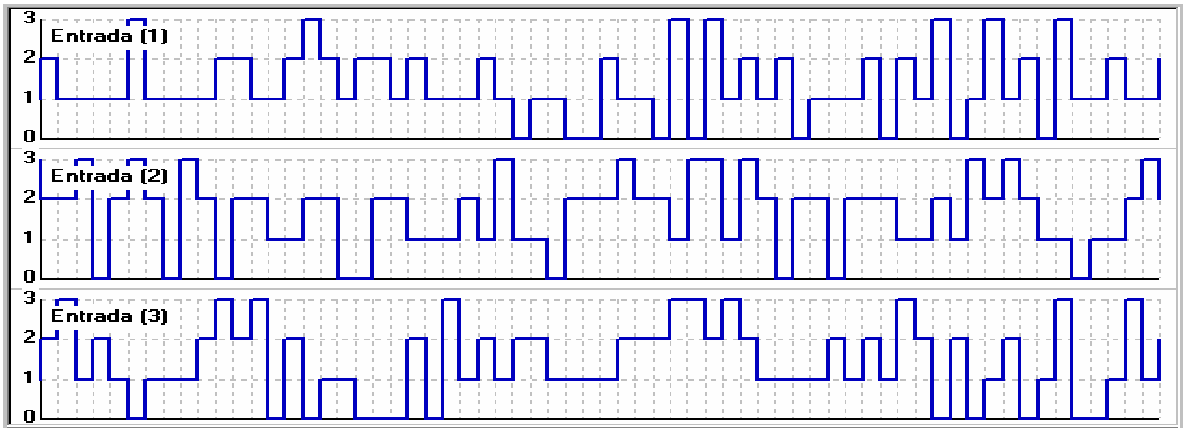


Figura 8.23 – O estímulo visto no gráfico.

```
E1  2111131111221123212212111210110021103031212011120213013120311211
E2  2230232032022112200221112131102223221331320220222112132321101123
E3  2312101112323020110002031212211112223323211112121320201201300131

S01 3221202122132223311322222221211132212102321121231223123231212222
S02 3222212222332131221113122321221122210132322122231321112211011222
S03 3321212123133121211112122222212223332032321221232222121231221123 2
S04 3322212222323313122111313232232222332121332222222323312123121112 32
S05 3022212222323013122111313232332222333212332222222323313123122112 02
S06 3131313133131222311222223232212233322212031231332222222332221221 3
S07 0332323233203232322223233332322233321123032232302332223322122333
S08 3321212123133121211112122222212223321232321221232222121231211123 2
S09 1232323230212232322223233333233300322231323323133323230232222323
S10 2133323331321202332220233032332231121233233233322032223122122313
```

Listagem 8.1 – O resumo.

8.4 Efeitos do atraso

O atraso é um fenômeno inerente a todos os fenômenos físicos. Até mesmo a luz, o fenômeno mais rápido que existe, possui uma velocidade de propagação finita, ou seja, possui um atraso não nulo. O próprio pensamento humano apresenta um tempo de atraso. Nos dispositivos elétricos, os atrasos são provocados, geralmente, por capacitâncias e indutâncias parasitas. Em sistemas mecânicos, por efeitos elásticos e inerciais. Em modelos sociais, o atraso pode se dar pela tendência das pessoas em manterem seus hábitos e costumes.

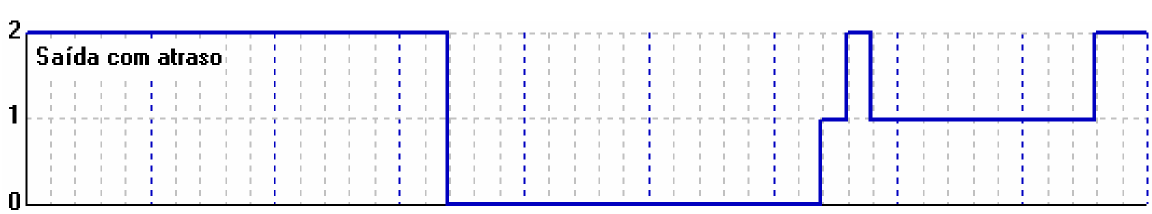


Figura 8.24 – Efeitos do atraso.

O exemplo da figura 8.24 mostra os efeitos causados pelos atrasos das portas. O primeiro efeito é a saída atrasada. O segundo efeito, uma conseqüência do primeiro, é a saída ter transições com atrasos diferentes em virtude das entradas serem provenientes de outras portas tendo, cada entrada, um atraso diferente. No exemplo, a primeira transição 2→0 e a transição 0→1 têm um atraso de 2/5, mas a transição 2→3 tem um atraso de 3/5. O terceiro efeito, o mais grave de todos (e que pode ser considerado um problema) é uma conseqüência do segundo efeito: quando duas entradas têm atrasos diferentes e deveriam, simultaneamente, sofrer uma transição, um estado intermediário aparece, criando uma combinação adicional, levando a saída a um valor espúrio, os chamados *spikes*. Essas transições podem disparar circuitos acoplados, prejudicando o processamento.

Os efeitos, então, são classificados em quatro tipos:

8.4.1 – Atraso

Esse efeito ocorre quando o valor incorreto tem, como causa, o simples atraso gerado pela porta ou uma porta anterior. Para uma mesma porta, as transições podem ter atrasos diferentes, decorrentes dos atrasos das entradas da porta, por esse motivo, o sinal de saída da porta não sofre apenas um atraso, mas uma deformação, dificultando a determinação do instante no qual o valor da saída deve ser lido, pois, pode existir algum passo no qual o atraso seja tão grande que o valor lido seja o do passo anterior. O circuito de leitura do sinal fornecido deve ser ajustado para suportar esse atraso. Na figura 8.25, há erros nos passos de número 2 e 4; no passo 2, o valor correto deveria ser 1 e é 0; no passo 4 era para ser 2 e é 1.

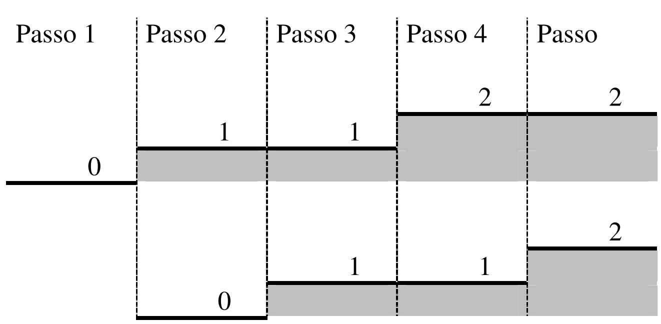


Figura 8.25 – Erros por falta de ajuste ao atraso por parte da leitura dos dados.

8.4.2 – Erro por transição indevida

Este é o pior tipo de erro; ocorre quando as entradas de uma porta sofrem atrasos diferentes, por ligarem-se a subcircuitos com níveis de atrasos diferentes. Essa diferença causa as deformações mencionadas no tratamento do erro por atraso, mas também pode causar transições indevidas, gerando valores espúrios nas saídas (terceiro problema comentado anteriormente). Na porta U_2, por exemplo, tem-se este problema. A primeira entrada recebe o atraso da porta 1, mas a segunda entrada recebe, diretamente, um sinal da entrada do circuito, ou seja, sem atraso.

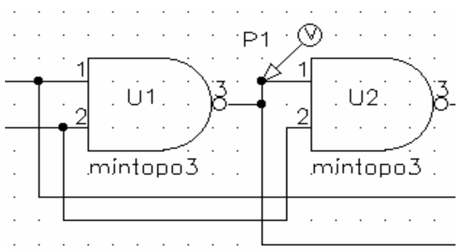


Figura 8.26 – Erro por transição indevida.

Se, por exemplo, em dois passos (pulsos de *clock*), a saída da porta U_1 receber a seqüência "03", e a segunda entrada da porta U_2 receber a seqüência "30", e o valor dominante for o "3", tem-se, na saída da porta U_2, a seqüência "33".

Por outro lado, se as portas apresentam atraso, há um instante em que o sinal da segunda entrada de U_2 já mudou para "0", mas a saída da porta U_1 ainda continua em "0", atrasada, dando, na saída de U_2, o valor "0". Esse valor indevido tende a durar pouco tempo, mas esse pulso pode surgir no exato momento em que os sinais são lidos (em circuitos síncronos), ou então podem causar um chaveamento ou disparo indevido (*trigger* em circuitos assíncronos). Para contornar este problema, deve-se adicionar portas atrasadoras, cuja única função é atrasar uma das entradas para que fique sincronizada com a(s) outra(s) entradas da porta. Outra alternativa é evitar combinações que provoquem esses erros.

8.4.3 – Erro por excesso de atraso

Ainda que não ocorram erros do tipo 2, por causa da adição de portas atrasadoras, o atraso vai se acumulando porta por porta. Pode acontecer de o atraso ser tão grande que extrapole a quantidade de subdivisões dos sub-passos (variável *múltiplo*), atingindo o passo seguinte. Em outras palavras, isso acontece quando o atraso absoluto for numericamente maior ou igual ao valor do múltiplo. Quando erros deste tipo acontecem, é mais difícil recuperar o sinal original (figura 8.27).

8.4.4 – Erro por perda de passo

Esse tipo de erro, quando acontece, é uma conseqüência do erro por excesso de atraso. Se, para uma porta, uma das entradas está atrasada mais de um passo (*múltiplo* × subpassos) em relação à outra entrada, o sinal de saída pode assumir um comportamento indevido. Ainda que o circuito seja projetado para não produzir sinais espúrios, o sinal de saída pode estar atrasado em mais de um passo, ou seja, entrada(s) e saída não estão sincronizados.

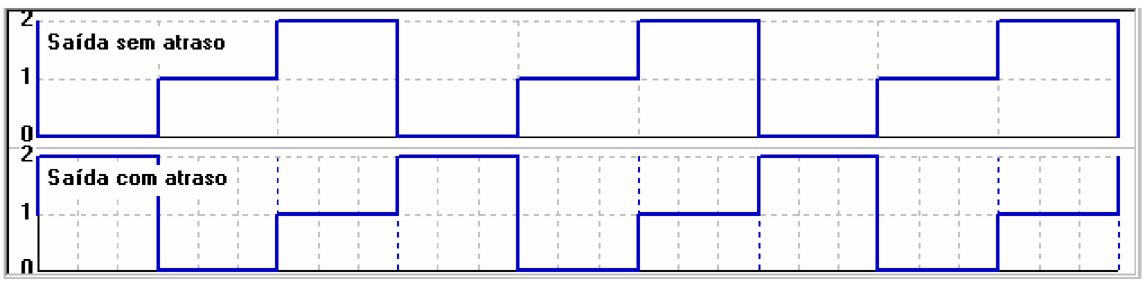


Figura 8.27 – Erro por excesso de atraso.

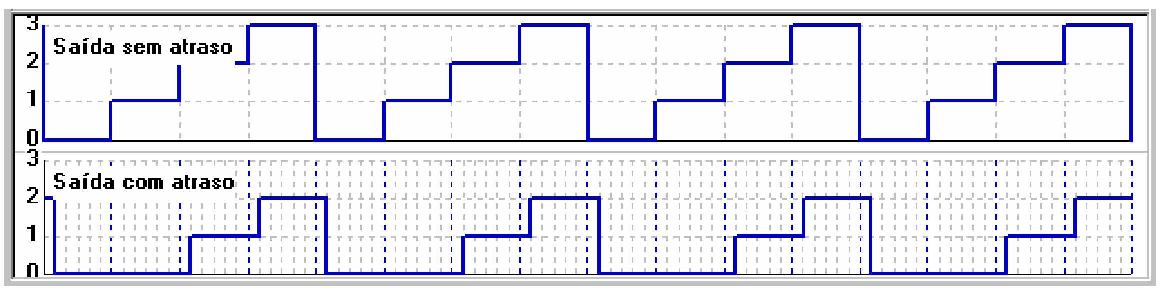


Figura 8.28 – Erro por perda de passo.

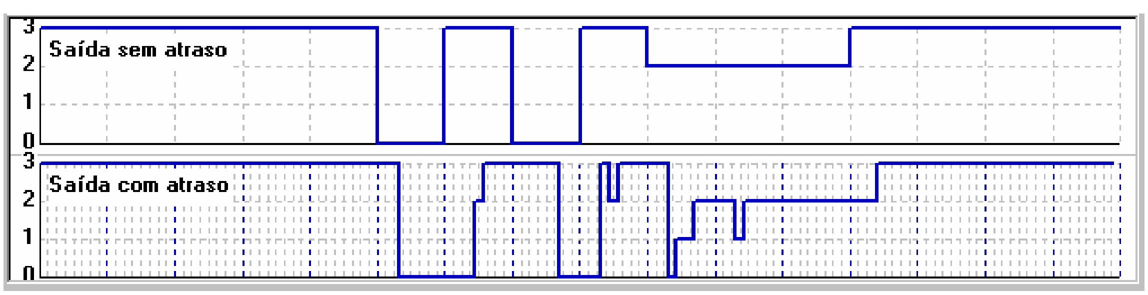


Figura 8.29 – Exemplos de danos causados pelos atrasos.

8.4.5 – Valor inicial

Um problema gerado pelo atraso incorporado a cada porta consiste no valor de saída inicial para as portas, pois, no instante inicial, não se pode saber qual era o valor anterior, valor este que gerou o valor de saída inicial na porta. Para contornar este problema, supõe-se que o circuito vem recebendo a seqüência de estímulos em suas entradas repetitivamente, e que esta seqüência já foi realizada muitas vezes, de modo que os valores precedentes ao inicial correspondem aos últimos valores da seqüência. Por exemplo, numa cadeia de 16 portas, com atraso de 50% (múltiplo = 2), se todas as portas forem ligadas em cascata, uma atrás da outra, a última porta fornece um valor com 16 atrasos, ou seja, relativo a 8 passos de atraso. Se houver apenas uma entrada com dois valores, isso corresponde a um atraso de 4 ciclos de repetição de todas as combinações de entrada.

$$c = v^e \quad (1)$$

$$sp = m * c \quad (2)$$

Múltiplo

Valores x Entradas	1	2	3	4	5	6	7	8
2x1	2	4	6	8	10	12	14	16
2x2	4	8	12	16	20	24	28	32
2x3	8	16	24	32	40	48	56	64
3x1	3	6	9	12	15	18	21	24
3x2	9	18	27	36	45	54	63	72
3x3	27	54	81	108	135	162	189	216
4x1	4	8	12	16	20	24	28	32
4x2	16	32	48	64	80	96	112	128
4x3	64	128	192	256	320	384	448	512

Tabela 8.4 – Quantidade de sub-passos.

Dependendo do valor da variável *múltiplo*, a quantidade de subpassos de simulação pode ser bastante grande. Essa quantidade é dada pela fórmula (1). Na fórmula (2), usa-se os dados do circuito, e não da porta. Se a quantidade de sub-passos for muito grande e a resolução do modo de vídeo for baixa, pode não ser possível exibir todos os detalhes do gráfico. Na resolução mínima (640x480), a quantidade máxima arbitrária de passos que pode ser exibida na tela é de 72. O gráfico precisa ser dividido em páginas.

Por meio de dois botões, pode-se selecionar a página anterior ou a próxima página.

Figura 8.30 – Os controles da janela do gráfico com atraso.

Quando a quantidade superar esse limite, o gráfico é dividido em páginas. Para três valores, são três páginas, e, para quatro valores, são quatro páginas.

	1	2	3	4	5	6	7	8
2x1	1	1	1	1	1	1	1	1
2x2	1	1	1	1	1	1	1	1
2x3	1	1	1	1	1	1	1	1
3x1	1	1	1	1	1	1	1	1
3x2	1	1	1	1	1	1	1	1
3x3	1	1	3	3	3	3	3	3
4x1	1	1	1	1	1	1	1	1
4x2	1	1	1	1	4	4	4	4
4x3	4	4	4	4	4	4	4	4

	1	2	3	4	5	6	7	8
2x1	2	4	6	8	10	12	14	16
2x2	4	8	12	16	20	24	28	32
2x3	8	16	24	32	40	48	56	64
3x1	3	6	9	12	15	18	21	24
3x2	9	18	27	36	45	54	63	72
3x3	27	54	27	36	45	54	63	72
4x1	4	8	12	16	20	24	28	32
4x2	16	32	48	64	20	24	28	32
4x3	16	32	48	64	80	96	112	128

Tabela 8.5 – Quantidade de páginas e de subpassos por página.

8.5 Exemplo de simulação

Supondo que se deseja simular o seguinte circuito, com as seguintes características:

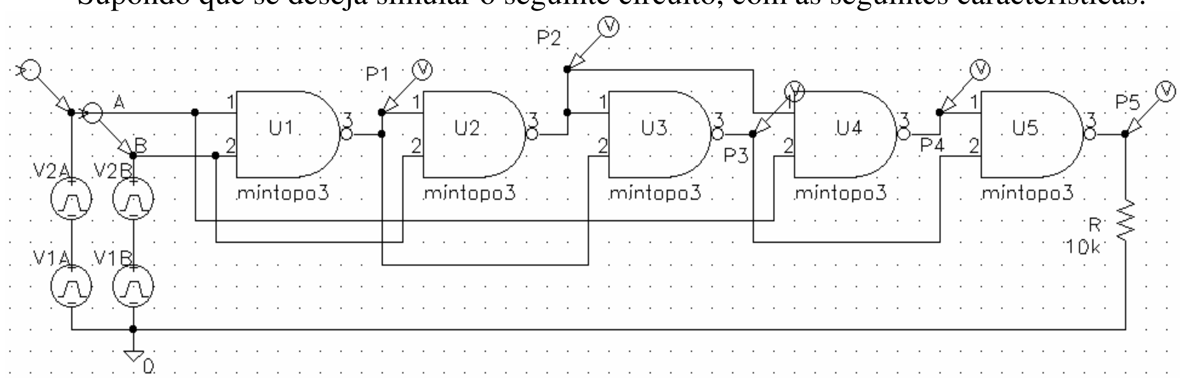


Figura 8.31 – O circuito.

A porta utilizada neste circuito é a “mintopo3” ou “mínimo-TOPO de três valores” e foi construída no SPICE. Detalhes sobre este circuito e sobre o subcircuito equivalente à porta lógica não fazem parte deste trabalho pois a idéia, aqui, é utilizar, somente, conceitos aplicáveis a todas as áreas do conhecimento humano e não aprofundar a abordagem para a engenharia elétrica.

- Três valores
- Um deslocamento
- Primeiro: 0
- Lógica:0
- Entradas da porta: 2
- Entradas do circuito: 2
- Quantidade de portas: 5

```
      |---|
      | A |                          Entradas:     2
    C |012|   Portas:        5       Valores:      3
 |----+---|   Porta Atual:   1       Primeiro:     0
 |  0 |111|   Passos:        9       Deslocamento: 1
 |B 1 |122|   Entradas:      2       Lógica:       0
 |  2 |120|                          Atraso:       0
 |----+---|                          Múltiplo:     8
```

Listagem 8.2 – Parâmetros de simulação.

Figura 8.32 – Parâmetros do simulador de porta.

Observando-se a figura 8.31 com o desenho do circuito, pode-se criar a matriz de conexões. Como o simulador só utiliza portas simétricas, a ordem como as entradas das portas são colocadas não altera o resultado da simulação.

Porta	Possibilidades
1	E1, E2
2	E1, E2, S1
3	E1, E2, S1, S2
4	E1, E2, S1, S2, S3
5	E1, E2, S1, S2, S3, S4

Tabela 8.6 – Opções de conexão para as portas.

```
              E2          012012012
              E1          000111222

Porta 01:   E1   E2   111122120
Porta 02:  S01   E2   111222201
Porta 03:  S02  S01   222200211
Porta 04:  S02   E1   122120112
Porta 05:  S04  S03   200211222
```

Tabela 8.7 – Matriz de conexões e matriz de simulação.

A matriz de simulação mostrada acima pode ser obtida manualmente.

```
    A   012012012
    B   000111222
saída   111122120
```

Listagem 8.3 – Análise individual da porta.

Porta	01	02	03	04	05
Entrada A	E1 012012012	S01 111122120	S02 111222201	S02 111222201	S04 122120112
Entrada B	E2 000111222	E2 000111222	S01 111122120	E1 012012012	S03 222200211
Saída	S01 111122120	S02 111222201	S03 222200211	S04 122120112	S05 200211222

Tabela 8.8 – Formação da matriz de simulação.

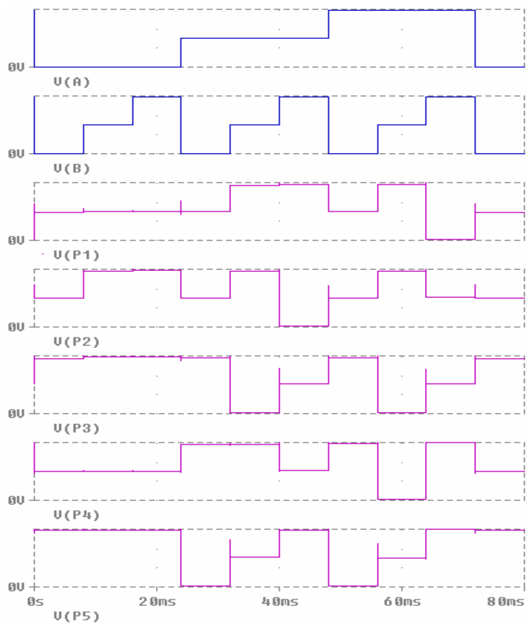


Figura 8.33 – Os resultados da simulação no SPICE.

Observando-se a figura 8.33 ao lado, percebe-se que o simulador do circuito forneceu um resultado coerente com a simulação realizada no SPICE. A seguir, tem-se a matriz de saída para alguns valores de "múltiplo":

As listagens a seguir mostram que, para os valores 2 e 3 de *múltiplo*, há portas cuja saída sofre erros irreparáveis, como, por exemplo, a porta U_5 no quinto e no oitavo passo, e a porta U_3 no oitavo passo. Por esse motivo, o *múltiplo* mínimo deve ser de 4, ou seja, um atraso máximo de ¼.

Por meio desta análise, pode-se determinar qual o máximo atraso suportável pelo circuito. Com base nessa informação, se o resultado for insatisfatório, pode-se optar por aperfeiçoar as portas lógicas a fim de apresentarem menor atraso ou modificar o circuito a fim de torna-lo imune ao atraso das portas.

```
Sem atraso
111122120
111222201
222200211
122120112
200211222

Múltiplo = 2
01 11 11 11 12 22 21 12 20
11 11 11 12 22 22 20 22 00
11 22 22 22 22 00 00 12 01
11 12 22 21 12 20 01 12 21
22 22 00 00 22 01 11 12 01

Múltiplo = 3
011 111 111 111 122 222 211 122 200
111 111 111 122 222 222 202 220 001
112 222 222 222 220 000 001 220 111
211 122 222 211 122 200 011 122 111
222 220 000 002 220 111 111 220 122

Múltiplo = 4
0111 1111 1111 1111 1222 2222 2111 1222 2000
1111 1111 1111 1222 2222 2222 2022 2200 0011
1122 2222 2222 2222 2200 0000 0012 2201 1111
2111 1222 2222 2111 1222 2000 0111 1221 1112
2222 2200 0000 0022 2201 1111 1112 2201 2222

Múltiplo = 5
01111 11111 11111 11111 12222 22222 21111 12222 20000
11111 11111 11111 12222 22222 22222 20222 22000 00111
11222 22222 22222 22222 22000 00000 00122 22011 11111
21111 12222 22222 21111 12222 20000 01111 12211 11122
22222 22000 00000 00222 22011 11111 11122 22012 22222

Múltiplo = 6
011111 111111 111111 111111 122222 222222 211111 122222 200000
111111 111111 111111 122222 222222 222222 202222 220000 001111
112222 222222 222222 222222 220000 000000 001222 220111 111111
211111 122222 222222 211111 122222 200000 011111 122111 111222
222222 220000 000000 002222 220111 111111 111222 220122 222222

Múltiplo = 7
0111111 1111111 1111111 1111111 1222222 2222222 2111111 1222222 2000000
1111111 1111111 1111111 1222222 2222222 2222222 2022222 2200000 0011111
1122222 2222222 2222222 2222222 2200000 0000000 0012222 2201111 1111111
2111111 1222222 2222222 2111111 1222222 2000000 0111111 1221111 1112222
2222222 2200000 0000000 0022222 2201111 1111111 1112222 2201222 2222222

Múltiplo = 8
01111111 11111111 11111111 11111111 12222222 22222222 21111111 12222222 20000000
11111111 11111111 11111111 12222222 22222222 22222222 20222222 22000000 00111111
11222222 22222222 22222222 22222222 22000000 00000000 00122222 22011111 11111111
21111111 12222222 22222222 21111111 12222222 20000000 01111111 12211111 11122222
22222222 22000000 00000000 00222222 22011111 11111111 11122222 22012222 22222222
```

Listagem 8.4 – Matriz do circuito, para cada opção de atraso.

```
1        1        1        1        2        2        1        2        0
01       11       11       11       12       22       21       12       20
011      111      111      111      122      222      211      122      200
0111     1111     1111     1111     1222     2222     2111     1222     2000
01111    11111    11111    11111    12222    22222    21111    12222    20000
011111   111111   111111   111111   122222   222222   211111   122222   200000
0111111  1111111  1111111  1111111  1222222  2222222  2111111  1222222  2000000
01111111 11111111 11111111 11111111 12222222 22222222 21111111 12222222 20000000

1        1        1        2        2        2        2        0        1
11       11       11       12       22       22       20       22       00
111      111      111      122      222      222      202      220      001
1111     1111     1111     1222     2222     2222     2022     2200     0011
11111    11111    11111    12222    22222    22222    20222    22000    00111
111111   111111   111111   122222   222222   222222   202222   220000   001111
1111111  1111111  1111111  1222222  2222222  2222222  2022222  2200000  0011111
11111111 11111111 11111111 12222222 22222222 22222222 20222222 22000000 00111111

2        2        2        2        0        0        2        1        1
11       22       22       22       22       00       00       12       01
112      222      222      222      220      000      001      220      111
1122     2222     2222     2222     2200     0000     0012     2201     1111
11222    22222    22222    22222    22000    00000    00122    22011    11111
112222   222222   222222   222222   220000   000000   001222   220111   111111
1122222  2222222  2222222  2222222  2200000  0000000  0012222  2201111  1111111
11222222 22222222 22222222 22222222 22000000 00000000 00122222 22011111 11111111

1        2        2        1        2        0        1        1        2
11       12       22       21       12       20       01       12       21
211      122      222      211      122      200      011      122      111
2111     1222     2222     2111     1222     2000     0111     1221     1112
21111    12222    22222    21111    12222    20000    01111    12211    11122
211111   122222   222222   211111   122222   200000   011111   122111   111222
2111111  1222222  2222222  2111111  1222222  2000000  0111111  1221111  1112222
21111111 12222222 22222222 21111111 12222222 20000000 01111111 12211111 11122222

2        0        0        2        1        1        2        2        2
22       22       00       00       22       01       11       12       01
222      220      000      002      220      111      111      220      122
2222     2200     0000     0022     2201     1111     1112     2201     2222
22222    22000    00000    00222    22011    11111    11122    22012    22222
222222   220000   000000   002222   220111   111111   111222   220122   222222
2222222  2200000  0000000  0022222  2201111  1111111  1112222  2201222  2222222
22222222 22000000 00000000 00222222 22011111 11111111 11122222 22012222 22222222
```

Listagem 8.5 – Comparação dos erros causados por cada opção de atraso.

Como se pode observar, o programa permite a determinação do atraso máximo permitido às portas, ou, então, com base no atraso das portas, a determinação da máxima frequência de operação, uma informação bastante útil aos projetistas.

Conclusão

Não há como encerrar este assunto com apenas um (ou dois) trabalhos. A lógica (booleana, MVL ou qualquer outra) é uma ciência vasta; ela está disponível para ser aplicada, bastando, apenas, ser descoberta. E muito ainda está por ser descoberto, principalmente na lógica MVL.

A teoria sobre a lógica apresentada é extensamente empregada no software LMVS. Partindo-se de um mapa de uma função, obtém-se a expressão algébrica equivalente. Simulando-se o circuito construído a partir da referida expressão algébrica no software LMVP, obtém-se a tabela original, mostrando que os dois softwares são coerentes entre si. Simulando-se, no SPICE (os circuitos não são apresentados neste trabalho), o mesmo circuito, obtém-se o mesmo gráfico, mostrando que a teoria de construção das portas lógicas está de acordo com a lógica empregada.

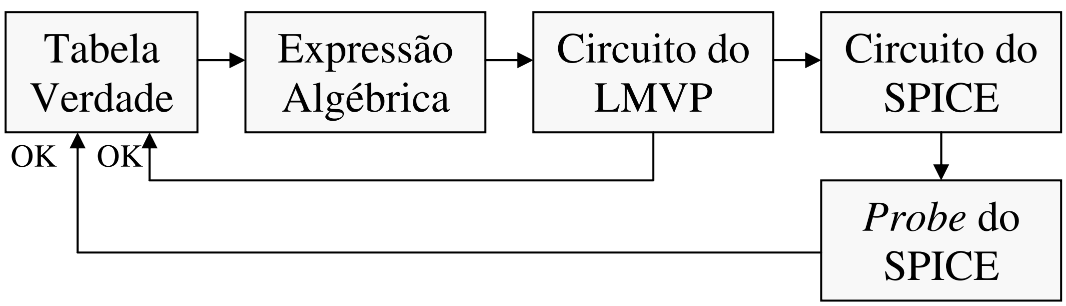


Figura C.1 – Roteiro de projeto.

O comportamento dos circuitos simulados é coerente com o resultado obtido com o software que, por sua vez, também é coerente com os resultados fornecidos com a aplicação dos conceitos lógicos apresentados. Tem-se, então, uma uniformidade conveniente entre os três trabalhos realizados (exame dos postulados, criação do software e construção dos circuitos no SPICE (este último não faz parte desta tese de doutorado)).

A primeira parte do trabalho (aplicação das regras), apesar de apresentar conceitos elaborados, procura apresentar a teoria de maneira clara e aprofundada e permite que o conhecimento seja aplicado de diversas maneiras (tal como já foi dito na introdução, este é um trabalho desvinculado de aplicações), possibilitando a origem de variadas implementações computacionais e, possivelmente, novas descobertas e aperfeiçoamentos.

A segunda parte do trabalho (criação do software), apresenta um ambiente de trabalho amigável, intuitivo, fácil de usar; o software é leve, pequeno, rápido e confiável. Também é possível que outro programador produza outro software com base na estrutura do LVMS e nos conceitos apresentados nos capítulos 1 a 5.

A versão atual (isto é, na data em que este texto foi escrito) dos softwares LMVS e LMVP ocupam 1004KB e 520KB, respectivamente. Podem ser executados em sistemas operacionais desde Windows 95 a Windows XP SP2. Versões mais recentes do Windows são posteriores à publicação deste trabalho e a compatibilidade deve ser verificada. Algumas versões do Windows poderão requerer a instalação do DCOM98 (*Distributed Component Object Model*), encontrado em diversos sites da Internet.

Muitos avanços neste trabalho ainda podem ser feitos. Segue, abaixo, uma lista de sugestões para trabalhos futuros que também podem ser realizados por outros pesquisadores.

1. Formalização de uma técnica de síntese não inspecional puramente MVL, ou seja, que não requeira a conversão em SFBs, que, como se pôde observar neste trabalho, é a responsável pelo grande tamanho das expressões algébricas da maioria das funções sintetizadas por meio da técnica aqui apresentada. O método paramétrico não resolve esse problema, pois também aumenta o tamanho das expressões e requer o uso de uma tabela de busca gerada por método inspecional.
2. Aperfeiçoamento da síntese por método inspecional para diminuir o tempo de execução e para cobrir mais funções (se possível, todas). Esta técnica é um excelente meio para a verificação da eficiência do método proposto no item anterior.
3. Inclusão, no simulador de circuitos lógicos MVL, os parâmetros do tempo de subida (*rise time*) e de descida (*fall time*). Tais parâmetros, tal como os demais parâmetros do

software LMVP, não são específicos de nenhuma área de atuação e não tiram a desvinculação do software às áreas de aplicação.

4. Aperfeiçoamento do software LMVP para que este suporte:
   a. Circuitos seqüenciais.
   b. Mais de um tipo de porta por circuito.
   c. Mais de 16 portas por circuito e mais subdivisões de atraso.
5. Criação de um software derivado do LMVP específico para circuitos elétricos com parâmetros como tensão, corrente, potência, impedância de entrada e de saída das portas, capacitâncias parasitas, etc.
6. Criação de uma versão do LMVP e do LMVS para Linux.

Um dos grandes problemas pelos quais a humanidade tem passado desde a sua criação é a "binarização" do pensamento. Não há como negar que algumas realidades sejam, de fato, binárias, como, por exemplo, vida e morte, céu e inferno, etc. Também é inegável que a exigência de que todas as decisões sejam tomadas de forma binária têm feito o homem se tornar menos humano: "tudo" deve ser "tudo ou nada", "nada" pode ser o "meio termo", decide-se por isso "ou" aquilo, agrada-se a gregos "e" troianos. Pode-se optar por "ambos" (os dois); por que não se pode optar por "antros" (os três – palavra inventada) ? Talvez, quando o homem deixar de "binarizar" o que não é binário ele finalmente terá encontrado o caminho para, de fato, expandir sua mente rumo à sabedoria plena.

Referências

1. S.L. Hurst, *Multi- Valued Logic – Its Status and Its Future*, p.1160-1175, IEEE transactions on computers, vol. C-33, no. 12 (1984).
2. G. Malinowski, *Many-Valued Logics*, University of Lodz (1993).
3. N. V. Serran, *Circuitos digitais ternários baseados na álgebra de Post*, P.hD. Thesis. State University of Campinas (1996.)
4. A. J. Biazon Filho, *Projeto e Construção de uma Porta Universal CMOS em Lógica Ternária*, M.Sc.. State University of Campinas (2000).
5. K.C. Smith, *The Prospects for Multivalued Logic: A Thechnology and Applications View*, p.619-634, IEEE transactions on computers, vol. c-30, no.9 (1981).
6. M. N. R. D. Yacoub, *Proposta de implementação de uma lógica ternária em tecnologia CMOS*. P.hD. Thesis. State University of Campinas (2000).
7. K.C. Smith, *Multiple-Valued Logic: A tutorial and Appreciations*, p.17-27, IEEE, (1988).
8. C. R. Mingoto Jr., *Circuitos Quaternários: Somador e Multiplicador*, M.Sc.. State University of Campinas (2005).
9. O. H. Bertone, *Proposta De Um Registrador Cíclico Para Lógica Multi-Valores E Aplicação Em Um Multiplicador Quaternário*, M.Sc.. State University of Campinas (2005).
10. B. Mates, *Elementary Logic.*, second edition, p. 142, Oxford University Press, New York (1972).
11. L. P. Nascimento, *Uma Ferramenta Automatizada para Análise e Projeto de Circuitos Digitais Multi-Valores*, M.Sc.. State University of Campinas (2001).
12. **12** N. V. Serran, A. M. Jorge, J. A. S. Dias, *A proposal for the implementation of ternary digital circuits*, p.533, Microelectronics Journal (1997).
13. A. M. Jorge, *A universal CMOS gate for multi-valued logic (MVL) circuits*, p.120, XV International Conference on Microelectronics and Packaging, Manaus, Brazil (2000).
14. A. J. Biazon Filho, *Multi-Valued Half Adder circuit using a CMOS Universal gate*, p.125, XV International Conference on Microelectronics and Packaging, Manaus, Brazil (2000).
15. L. P. Nascimento, *An Automated Tool for Analysis and Design of MVL Digital Circuits*, p.52, XV SBMicro, Brasilia, Brazil, (2001).
16. C. R. Mingoto Jr., *A Quaternary Logic Gate Using Current-Mode Operation with Bipolar Transistors Equivalent to an Exclusive-OR Binary Gate*, Proceedings of IEEE - ICECS 2005, Gammarth, Tunisia, (2005).
17. C. R. Mingoto Jr., *A Quaternary Half-Adder Using Current-Mode Operation with Bipolar Transistors*, Proceedings of IEEE - ISMVL 2006, Singapore, (2006).

18. M. A. S. Fregonezi, *Projeto de um Aplicativo para Síntese de Funções Multi-Valores em Ambiente Windows*, M.Sc.. State University of Campinas (2001).

19. G. Epstein, G. Frieder, D. C. Rine, *The development of multiple-valued logic as related to computer science*, p.20-31, Computer, (1974).

20. I. Thoidis, D. Soudris, I. Karafyllidis, S. Christoforidis, A. Thanailakis, *Quaternary voltage-mode CMOS circuits for multiple-valued logic*, p.71-77, IEE Proc – Circuits Devices Syst. Vol 145, No. 2, (1998).

21. I. Bonatti, M. Madureira, *Introdução à análise e síntese de circuitos lógicos*, Editora da Unicamp, (1995).

Apêndice 1
Outra técnica de síntese

A1.1 Três valores

Diversas outras técnicas de síntese podem ser empregadas. Neste apêndice, é apresentada uma delas. A função abaixo pode ser separada em três partes:

C

0	1	2
1	2	0
2	0	1

c0

0	1	2
1	X	X
2	X	X

c1

X	1	X
1	2	0
X	0	X

c2

X	X	2
X	X	0
2	0	1

Tabela A1.1 – A função e sua subdivisões.

Quais são as funções mais simples que podem ser atribuídas a *c0*,*c1* e *c2*?

Por inspeção, determina-se -se a tabela A1.2. De que maneira *c0*, *c1* e *c2* podem ser agrupados, de forma a gerar a função *C*? Há três possibilidades (fórmulas (1)):

`c0 = A b B`

0	1	2
1	1	1
2	1	2

`c1 = /A a /B`

1	1	0
1	2	0
0	0	0

`c2 = A/ c B/`

2	2	2
2	0	0
2	0	1

Tabela A1.2 – Sub-funções definidas c0,c1,c2.

```
C0 : (c0,c1),c2
```
(1a)
```
C1 : (c1,c2),c0
```
(1b)
```
C2 : (c2,c0),c1
```
(1c)

Analisando, primeiramente, as duas sub-expressões entre parênteses, deve-se descobrir qual o operador a ser utilizado, de modo a se obter as seguintes funções:

`c0,c1`

0	1	2
1	2	0
2	0	X

`c1,c2`

X	1	2
1	2	0
2	0	1

`c2,c0`

0	1	2
1	X	0
2	0	1

Tabela A1.3 – As duas primeiras sub-funções, indefinidas.

Por inspeção, deduz-se a tabela A1.4. Para o segundo operador, é mostrada a dedução apenas para a primeira forma, C_0; para as demais formas, a idéia é a mesma. Precisa-se encontrar o conectivo que faça a transformação mostrada na tabela A1.5.

C0

c0 c c1

0	1	2
1	2	0
2	0	2

C1

c1 b c2

1	1	2
1	2	0
2	0	1

C2

c2 a c0

0	1	2
1	0	0
2	0	1

Tabela A1.4 – As duas primeiras sub-funções, definidas.

c0 c c1

0	1	2
1	2	0
2	0	2

op

c2

2	2	2
2	0	0
2	0	1

=

C0

0	1	2
1	2	0
2	0	1

Tabela A1.5 – Dedução do segundo operador para a primeira forma.

Para substituir o valor errado "2" pelo valor certo "1", em **c0 c c1**, pode-se usar os operadores "a" ou "b". Ambas possibilidades provocam o surgimento de erros.

(c0 c c1) a c2

0	1	2
1	0	0
2	0	1

(c0 c c1) b c2

2	1	2
1	2	0
2	0	1

Tabela A1.6 –
As duas possibilidades de escolha do operador.

No primeiro caso, onde se utiliza o operador "a", onde há "0", o valor correto é "2", que é o valor indiferente para o operador "a", ou seja, deve-se substituir, na combinação onde ocorre o erro, o valor "0" por "2", o que significa fazer uma conversão para binário. A CPB que transforma "0" em "2" é **Cb2**. Fazendo tal conversão, tem-se:

C0 = ((A b B) c (/A a /B)) a ((A/ c B/) b 2)

c0 c c1

0	1	2
1	2	0
2	0	2

a

c2 b 2

2	2	2
2	2	2
2	2	1

=

C0

0	1	2
1	2	0
2	0	1

Tabela A1.7 – Primeira opção.

No segundo caso, onde se utiliza o operador "b", o valor correto é "0", que é o valor indiferente para o operador "b", ou seja, deve-se substituir, na combinação onde ocorre o erro, o valor "2" por "0". A CPB que transforma "2" em "0" é **Ca/C**. Fazendo tal conversão, tem-se:

C0 = ((A b B) c (/A a /B)) b ((A/ c B/) a (A a B))

c0 c c1

0	1	2
1	2	0
2	0	2

b

c2 a /c2

0	0	0
0	0	0
0	0	1

=

C0

0	1	2
1	2	0
2	0	1

Tabela A1.8 – Segunda opção.

O segundo caso gera uma expressão maior, escolhe-se o primeiro caso.

Seguindo um procedimento análogo, deduz-se as demais formas:

```
C0 = ( c0 c c1 ) a ( c2 b 2 ) (2a)
C1 = ( c1 b c2 ) c ( c0 a 1 ) (2b)
C2 = ( c2 a c0 ) b ( c1 c 0 ) (2c)
```

```
C0 = ( ( A b B ) c ( /A a /B ) ) a ( ( A/ c B/ ) b 2 )
C1 = ( ( /A a /B ) b ( A/ c B/ ) ) c ( ( A b B ) a 1 )
C2 = ( ( A/ c B/ ) a ( A b B ) ) b ( ( /A a /B ) c 0 )
```

Pelo método tradicional, obtém-se:

```
    C1 = [ (A b B) a (/A b B/) a (A/ b /B) ] c [ (/A c /B) b (A c /B) b
(/A c B) b (A/ c B) b (A c B/) b (A/ c B/) ]
    C2 = [ (A c /B) b (/A c B) b (A/ c B/) ] a [ (/A a B/) c (A a B/) c
(A/ a /B) c (/A a /B) c (A/ a B) c (A a B) ]
    C3 = [ (A a B/) c (/A a /B) c (A/ a B) ] b [ (A b B) a (/A b B) a (A/
b B/) a (/A b B/) a (A b /B) a (A/ b /B) ]
```

Percebe-se que, para este exemplo, esta nova técnica de síntese produz uma minimização muito mais eficiente, para este exemplo.

Seguindo um procedimento análogo, deduz-se as funções deslocadas:

1	2	0
2	0	1
0	1	2

2	0	1
0	1	2
1	2	0

Tabela A1.9 – As funções /C e C/.

```
/C0 = ( ( /A c /B ) a ( A/ b B/ ) ) b ( ( A a B ) c 0 )
/C1 = ( ( A/ b B/ ) c ( A a B ) ) a ( ( /A c /B ) b 2 )
/C2 = ( ( A a B ) b ( /A c /B ) ) c ( ( A/ b B/ ) a 1 )
```

```
C0/ = ( ( A/ a B/ ) b ( A c B ) ) c ( ( /A b /B ) a 1 )
C1/ = ( ( A c B ) a ( /A b /B ) ) b ( ( A/ a B/ ) c 0 )
C2/ = ( ( /A b /B ) c ( A/ a B/ ) ) a ( ( A c B ) b 2 )
```

Pode-se, também, analisar colunas e linhas de forma não simétrica. Tem-se três colunas e três linhas, obtendo-se, então, nove possibilidades. São, também, nove sub-funções. Para a função utilizada anteriormente, tem-se:

c0

0	1	2
1	X	X
2	X	X

c1

0	1	2
X	2	X
X	0	X

c2

0	1	2
X	X	0
X	X	1

c3

0	X	X
1	2	0
2	X	X

c4

X	1	X
1	2	0
X	0	X

c5

X	X	2
1	2	0
X	X	1

c6

0	X	X
1	X	X
2	0	1

c7

X	1	X
X	2	X
2	0	1

c8

X	X	2
X	X	0
2	0	1

Tabela A1.10 – As nove sub-funções.

Substituindo-se os valores irrelevantes adequadamente, tem-se:

```
c0 = A b B;          c1 = A c /B;         c2 = A a B/
c3 = /A c B;         c4 = /A a /B;        c5 = /A b B/
c6 = A/ a B;         c7 = A/ b /B;        c8 = A/ c B/
```

```
C0 = C0(c0,c4,c8)     C1 = C1(c1,c5,c6)    C2 = C2(c2,c3,c7)
C3 = C3(c3,c7,c2)     C4 = C4(c4,c8,c0)    C5 = C5(c5,c6,c1)
C6 = C6(c6,c1,c5)     C7 = C7(c7,c2,c3)    C8 = C8(c8,c0,c4)
```

```
C0 = ( c0 c c4 ) a ( c8 b 2 ) (3a)
C1 = ( c1 b c5 ) c ( c6 a 1 ) (3b)
C2 = ( c2 a c3 ) b ( c7 c 0 ) (3c)
C3 = ( c3 c c7 ) a ( c2 b 2 ) (3d)
C4 = ( c4 b c8 ) c ( c0 a 1 ) (3e)
C5 = ( c5 a c6 ) b ( c1 c 0 ) (3f)
C6 = ( c6 c c1 ) a ( c5 b 2 ) (3g)
C7 = ( c7 b c2 ) c ( c3 a 1 ) (3h)
C8 = ( c8 a c0 ) b ( c4 c 0 ) (3i)
```

```
C0 = ( ( A b B ) c ( /A a /B ) ) a ( ( A/ c B/ ) b 2 )
C1 = ( ( A c /B ) a ( /A b B/ ) ) b ( ( A/ a B ) c 0 )
C2 = ( ( A a B/ ) b ( /A c B ) ) c ( ( A/ b /B ) a 1 )
C3 = ( ( /A c B ) a ( A/ b /B ) ) b ( ( A a B/ ) c 0 )
C4 = ( ( /A a /B ) b ( A/ c B/ ) ) c ( ( A b B ) a 1 )
C5 = ( ( /A b B/ ) c ( A/ a B ) ) a ( ( A c /B ) b 2 )
C6 = ( ( A/ a B ) b ( A c /B ) ) c ( ( /A b B/ ) a 1 )
C7 = ( ( A/ b /B ) c ( A a B/ ) ) a ( ( /A c B ) b 2 )
C8 = ( ( A/ c B/ ) a ( A b B ) ) b ( ( /A a /B ) c 0 )
```

Esta técnica de síntese, além de proporcionar uma minimização mais eficiente para este exemplo, ainda fornece mais opções de formas de expressão, oito, em relação à técnica tradicional, que fornece, apenas, três.

A1.2 Quatro valores

A mesma técnica aplicada para três valores pode ser empregada para quatro valores.

C

0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	2

c0

0	1	2	3
1	X	X	X
2	X	X	X
3	X	X	X

c1

X	1	X	X
1	2	3	0
X	3	X	X
X	0	X	X

c2

X	X	2	X
X	X	3	X
2	3	0	1
X	X	1	X

c3

X	X	X	3
X	X	X	0
X	X	X	1
3	0	1	2

Tabela A1.11 – Exemplo.

`c0 = A b B`

0	1	2	3
1	1	1	1
2	1	2	2
3	1	2	3

`c1 = /A d /B`

1	1	3	0
1	2	3	0
3	3	3	3
0	0	3	0

`c2 = //A b //B`

2	2	2	1
2	3	3	1
2	3	0	1
1	1	1	1

`c3 = A/ d B/`

3	3	3	3
3	0	0	0
3	0	1	1
3	0	1	2

Tabela A1.12 – Sub-funções definidas.

`c0 c c1`

0	1	2	3
1	2	3	0
2	3	2	2
3	0	2	3

`c1 a c2`

1	1	2	0
1	2	3	0
2	3	0	1
0	0	1	0

`c2 c c3`

2	2	2	3
2	3	3	0
2	3	0	1
3	0	1	2

`c3 a c0`

0	1	2	3
1	0	0	0
2	0	1	1
3	0	1	2

Tabela A1.13 – Sub-funções operadas entre si.

Para o segundo operador, é mostrada a dedução apenas para a primeira forma; para as demais formas, a idéia é a mesma. Precisa-se encontrar o operador que faça:

`c0 c c1`					`c2`								
0	1	2	3		2	2	2	1		0	1	2	3
1	2	3	0	op	2	3	3	1	=	1	2	3	0
2	3	2	2		2	3	0	1		2	3	0	1
3	0	2	3		1	1	1	1		3	0	1	X

Tabela A1.14 – Dedução do segundo operador para a primeira forma.

Para substituir o valor errado "2" por "0" e "1", em (`c0 c c1`), pode-se usar os operadores "a" ou "d". Ambas possibilidades provocam o surgimento de erros.

`( c0 c c1 ) a c2`

0	1	2	1
1	2	3	0
2	3	0	1
1	0	1	1

`( c0 c c1 ) d c2`

0	1	2	3
1	3	3	0
2	3	0	1
3	0	1	1

Tabela A1.15 – As duas possibilidades de escolha do operador.

No primeiro caso, onde se utiliza o operador "a", o valor correto é "3" nas combinações (3,0) e (0,3) e “2” na combinação (3,3). “3” é o valor indiferente para o operador “a”, ou seja, deve-se substituir, na combinação onde ocorre o erro, o valor “0” por “3”, o que significa fazer uma conversão para binário. A CPB que transforma “0” em “3” é `Cc3`, mas esse procedimento elimina a ocorrência do valor “0” na combinação (2,2).

c0 c c1					**c2 c 3**								
0	1	2	3		2	2	2	3		0	1	2	3
1	2	3	0	a	2	3	3	3	=	1	2	3	0
2	3	2	2		2	3	3	3		2	3	2	2
3	0	2	3		3	3	3	3		3	0	2	3

Tabela A1.19 – Primeira opção com CPB.

No segundo caso, onde se utiliza o operador “d”, o valor correto é “2”, que é o valor indiferente para o operador “d”, ou seja, deve-se substituir, na combinação onde ocorre o erro, o valor “3” por “2”, o que significa fazer uma CPB. A CPB que transforma “3” em “2” é `Cb2`, mas esse procedimento elimina a ocorrência do valor “0” na combinação (2,2).

c0 c c1					**c2 b 2**								
0	1	2	3		2	2	2	1		0	1	2	3
1	2	3	0	d	2	2	2	1	=	1	2	3	0
2	3	2	2		2	2	2	1		2	3	2	1
3	0	2	3		1	1	1	1		3	0	1	1

Tabela A1.20 – Segunda opção com CPB.

A solução é buscar uma conversão que substitua “3” por “2” sem também substituir “0” por “2”. Isso é realizado pela conversão para ternário `Ca2`.

```
( ( A b B ) c ( /A d /B ) ) d ( ( //A b //B ) a 2 )
```

c0 c c1					**c2 a 2**								
0	1	2	3		2	2	2	1		0	1	2	3
1	2	3	0	d	2	2	2	1	=	1	2	3	0
2	3	2	2		2	2	0	1		2	3	0	1
3	0	2	3		1	1	1	1		3	0	1	3

Tabela A1.21 – Segunda opção com conversão para ternário.

Por fim, basta apenas determinar, agora, a última operação. Para substituir o valor errado “3” pelo valor certo “2”, pode-se usar os operadores “a”,”b” ou “c” (tabela A1.22).

(c0 c c1) d (c2 a 2)

0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	3

op

c3

3	3	3	**3**
3	0	0	**0**
3	0	1	**1**
3	**0**	**1**	**2**

=

C0

0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	2

Tabela A1.22 – Dedução do terceiro operador para a primeira forma.

a

0	1	2	3
1	0	0	0
2	0	0	1
3	0	1	2

b

3	1	2	3
1	2	3	0
2	3	1	1
3	0	1	2

c

3	3	2	3
3	2	3	0
2	3	0	1
3	0	1	2

Tabela A1.23 – As três possibilidades de escolha do operador.

Na primeira possibilidade, usando o operador “a”, o valor errado é “0”. Deve-se fazer uma substituição que mude o “0” para “3”, nas coordenadas (3,2) e (2,3), em c3, o valor indiferente do operador “a”. A CPB que transforma “0” em “3” é **Cc3**.

((c0 c c1) d (c2 a 2)) a (c3 c 3)
c3 c 3
(c0 c c1) d (c2 a 2)

0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	3

3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	2

0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	2

Tabela A1.24 – Primeiro caso.

Por meio de raciocínios parecidos, ou por inspeção, pode-se obter todas as possibilidades mostradas na tabela A1.23. Não há diferença substancial nestas três formas.

```
C0 = (( c0 c c1 ) d ( c2 a 2 )) a ( c3 c 3 ) (4a)
C0 = (( c0 c c1 ) d ( c2 a 2 )) a ( c3 b 3 ) (4b)
C0 = (( c0 c c1 ) d ( c2 a 2 )) b ( c3 c 0 ) (4c)
```

Analisando-se a primeira possibilidade, nas formas C1, C2, C3, tem-se:

```
C0 = (( c0 c c1 ) d ( c2 a 2 )) a ( c3 c 3 ) (5a)
C1 = (( c1 a c2 ) b ( c3 c 0 )) c ( c0 a 1 ) (5b)
C2 = (( c2 a c3 ) b ( c0 a 2 )) c ( c1 c 3 ) (5c)
C3 = (( c3 a c0 ) b ( c1 c 0 )) c ( c2 a 1 ) (5d)
```

```
C0 = ( ( ( A b B ) c ( /A d /B ) ) d ( ( //A b //B ) a 2 ) ) a ( ( A/ d B/ ) c 3 )
C1 = ( ( ( /A d /B ) a ( //A b //B ) ) b ( ( A/ d B/ ) c 0 ) ) c ( ( A b B ) a 1 )
C2 = ( ( ( //A b //B ) c ( A/ d B/ ) ) d ( ( A b B ) a 2 ) ) a ( ( /A d /B ) c 3 )
C3 = ( ( ( A/ d B/ ) a ( A b B ) ) b ( ( /A d /B ) c 0 ) ) c ( ( //A b //B ) a 1 )
```

Pelo método tradicional, para C1, obtém-se:

```
C1 = { [ (A b B) a (/A b B/) a (//A b //B) a (A/ b /B) ] c [ (/A c /B) b (A c /B) b (/A c B) b (//A c B) b (A/ c B/) b (//A c B/) b (A c //B) b (A/ c //B) ] } d [ (//A d //B) c (/A d //B) c (A d //B) c (//A d /B) c (/A d /B) c (A/ d /B) c (//A d B) c (A d B) c (A/ d B) c (/A d B/) c (A d B/) c (A/ d B/) ]
```

As formas C_1, C_2, C_3, C_4, C_5, C_6, C_7 e C_8 do método tradicional, possuem, todas, o mesmo tamanho. Para este exemplo, esta nova técnica de síntese produziu uma minimização muito mais eficiente.

Seguindo um procedimento análogo, deduz-se as funções deslocadas:

1	2	3	0
2	3	0	1
3	0	1	2
0	1	2	3

2	3	0	1
3	0	1	2
0	1	2	3
1	2	3	0

3	0	1	2
0	1	2	3
1	2	3	0
2	3	0	1

Tabela A1.25 – As funções /C, //C e C/..

```
C0 = ( ( ( A b B ) c ( /A d /B ) ) d ( ( //A b //B ) a 2 ) ) a ( ( A/ d B/ ) c 3 )
C1 = ( ( ( /A d /B ) a ( //A b //B ) ) b ( ( A/ d B/ ) c 0 ) ) c ( ( A b B ) a 1 )
C2 = ( ( ( //A b //B ) c ( A/ d B/ ) ) d ( ( A b B ) a 2 ) ) a ( ( /A d /B ) c 3 )
C3 = ( ( ( A/ d B/ ) a ( A b B ) ) b ( ( /A d /B ) c 0 ) ) c ( ( //A b //B ) a 1 )

/C1 = ( ( ( /A c /B ) d ( //A a //B ) ) a ( ( A/ c B/ ) b 3 ) ) b ( ( A a B ) d 0 )
/C2 = ( ( ( //A a //B ) b ( A/ c B/ ) ) c ( ( A a B ) d 1 ) ) d ( ( /A c /B ) b 2 )
/C3 = ( ( ( A/ c B/ ) d ( A a B ) ) a ( ( /A c /B ) b 3 ) ) b ( ( //A a //B ) d 0 )
/C4 = ( ( ( A a B ) b ( /A c /B ) ) c ( ( //A a //B ) d 1 ) ) d ( ( A/ c B/ ) b 2 )

//C1 = ( ( ( //A d //B ) a ( A/ b B/ ) ) b ( ( A d B ) c 0 ) ) c ( ( /A b /B ) a 1 )
//C2 = ( ( ( A/ b B/ ) c ( A d B ) ) d ( ( /A b /B ) a 2 ) ) a ( ( //A d //B ) c 3 )
//C3 = ( ( ( A d B ) a ( /A b /B ) ) b ( ( //A d //B ) c 0 ) ) c ( ( A/ b B/ ) a 1 )
//C4 = ( ( ( /A b /B ) c ( //A d //B ) ) d ( ( A/ b B/ ) a 2 ) ) a ( ( A d B ) c 3 )

C1/ = ( ( ( A/ a B/ ) b ( A c B ) ) c ( ( /A a /B ) d 1 ) ) d ( ( //A c //B ) b 2 )
C2/ = ( ( ( A c B ) d ( /A a /B ) ) a ( ( //A c //B ) b 3 ) ) b ( ( A/ a B/ ) d 0 )
C3/ = ( ( ( /A a /B ) b ( //A c //B ) ) c ( ( A/ a B/ ) d 1 ) ) d ( ( A c B ) b 2 )
C4/ = ( ( ( //A c //B ) d ( A/ a B/ ) ) a ( ( A c B ) b 3 ) ) b ( ( /A a /B ) d 0 )
```

Da mesma forma que, para três valores, há nove possibilidades de formas, baseado na coluna e na linha a partir das quais a síntese é iniciada, há dezesseis possibilidades para quatro valores.

c0

0	1	2	3
1	X	X	X
2	X	X	X
3	X	X	X

c1

0	1	2	3
X	2	X	X
X	3	X	X
X	0	X	X

c2

0	1	2	3
X	X	3	X
X	X	0	X
X	X	1	X

c3

0	1	2	3
X	X	X	0
X	X	X	1
X	X	X	2

c4

0	X	X	X
1	2	3	0
2	X	X	X
3	X	X	X

c5

X	1	X	X
1	2	3	0
X	3	X	X
X	0	X	X

c6

X	X	2	X
1	2	3	0
X	X	0	X
X	X	1	X

c7

X	X	X	3
1	2	3	0
X	X	X	1
X	X	X	2

c8

0	X	X	X
1	X	X	X
2	3	0	1
3	X	X	X

c9

X	1	X	X
X	2	X	X
2	3	0	1
X	0	X	X

c10

X	X	2	X
X	X	3	X
2	3	0	1
X	X	1	X

c11

X	X	X	3
X	X	X	0
2	3	0	1
X	X	X	2

c12

0	X	X	X
1	X	X	X
2	X	X	X
3	0	1	2

c13

X	1	X	X
X	2	X	X
X	3	X	X
3	0	1	2

c14

X	X	2	X
X	X	3	X
X	X	0	X
3	0	1	2

c15

X	X	X	3
X	X	X	0
X	X	X	1
3	0	1	2

Tabela A1.26 – As dezesseis sub-funções.

```
c00 = A b B;        c01 = A c /B;       c02 = A d //B;      c03 = A a B/
c04 = /A c B;       c05 = /A d /B;      c06 = /A a //B;     c07 = /A b B/
c08 = //A d B;      c09 = //A a /B;     c10 = //A b //B;    c11 = //A c B/
c12 = A/ a B;       c13 = A/ b /B;      c14 = A/ c //B;     c15 = A/ d B/
```

```
C00 = ( ( ( A b B ) c ( /A d /B ) ) d ( ( //A b //B ) a 2 ) ) a ( ( A/ d B/ ) c 3 )
C01 = ( ( ( A c /B ) d ( /A a //B ) ) a ( ( //A c B/ ) b 3 ) ) b ( ( A/ a B ) d 0 )
C02 = ( ( ( A d //B ) a ( /A b B/ ) ) b ( ( //A d B ) c 0 ) ) c ( ( A/ b /B ) a 1 )
C03 = ( ( ( A a B/ ) b ( /A c B ) ) c ( ( //A a /B ) d 1 ) ) d ( ( A/ c //B ) b 2 )
C04 = ( ( ( /A c B ) d ( //A a /B ) ) a ( ( A/ c //B ) b 3 ) ) b ( ( A a B/ ) d 0 )
C05 = ( ( ( /A d /B ) a ( //A b //B ) ) b ( ( A/ d B/ ) c 0 ) ) c ( ( A b B ) a 1 )
C06 = ( ( ( /A a //B ) b ( //A c B/ ) ) c ( ( A/ a B ) d 1 ) ) d ( ( A c /B ) b 2 )
C07 = ( ( ( /A b B/ ) c ( //A d B ) ) d ( ( A/ b /B ) a 2 ) ) a ( ( A d //B ) c 3 )
C08 = ( ( ( //A d B ) a ( A/ b /B ) ) b ( ( A d //B ) c 0 ) ) c ( ( /A b B/ ) a 1 )
C09 = ( ( ( //A a /B ) b ( A/ c //B ) ) c ( ( A a B/ ) d 1 ) ) d ( ( /A c B ) b 2 )
C10 = ( ( ( //A b //B ) c ( A/ d B/ ) ) d ( ( A b B ) a 2 ) ) a ( ( /A d /B ) c 3 )
C11 = ( ( ( //A c B/ ) d ( A/ a B ) ) a ( ( A c /B ) b 3 ) ) b ( ( /A a //B ) d 0 )
C12 = ( ( ( A/ a B ) b ( A c /B ) ) c ( ( /A a //B ) d 1 ) ) d ( ( //A c B/ ) b 2 )
C13 = ( ( ( A/ b /B ) c ( A d //B ) ) d ( ( /A b B/ ) a 2 ) ) a ( ( //A d B ) c 3 )
C14 = ( ( ( A/ c //B ) d ( A a B/ ) ) a ( ( /A c B ) b 3 ) ) b ( ( //A a /B ) d 0 )
C15 = ( ( ( A/ d B/ ) a ( A b B ) ) b ( ( /A d /B ) c 0 ) ) c ( ( //A b //B ) a 1 )
```

Este método não é abordado no software LMVS porque ainda não foi elaborado um procedimento iterativo que possa ser implementado e que realize os passos apresentados neste apêndice. Também se pode observar que esta técnica de síntese requer a observação de propriedades mais sofisticadas da lógica MVL.

Apêndice 2
Um conectivo

A2.1 Introdução

Funções booleanas podem ser construídas de três formas:

- Disjuntiva
- Conjuntiva
- Um conectivo

As formas disjuntiva e conjuntiva permitem formar expressões mais simples. Outra forma consiste no uso de, apenas, um deslocador e um conectivo e, também, pode ser empregada em lógica MVL[3,12,13]. Nos capítulos anteriores, as expressões geradas pelas metodologias apresentadas empregam mais de um conectivo. Algumas aplicações para funções MVL podem exigir que as expressões sejam construídas com o uso de um conectivo apenas tal como acontece em circuitos digitais integrados; os conectivos são representados por portas lógicas e é interessante que se utilize, apenas, um tipo de porta porque isso, além de simplificar a criação do *lay-out* das máscaras do processo de fabricação do circuito integrado por aumentar a repetição de padrões únicos, referentes à porta lógica utilizada, também permite minimizar diversos efeitos indesejáveis que surgem quando portas lógicas apresentam diferenças, como, por exemplo, tempo de atraso, tensão máxima e mínima para cada valor lógico, efeitos de temperatura, etc. Problemas dessa natureza justificam a síntese utilizando-se, apenas, um tipo de conectivo. O conectivo a ser escolhido deve ser aquele que ofereça maior vantagem na implementação.

As tabelas A2.1 e A2.2 apresentam algumas propriedades da lógica MVL necessárias para a transformação de uma expressão multi-conectivo para uma expressão uni-conectivo. A primeira linha consiste na mudança de valor; a segunda consiste no agrupamento de deslocadores, e, na terceira, em expansões do teorema de De Morgan para a lógica MVL.

`/( 0 ) = 1`	(1a)	`( 0 )/ = 2`	(1d)
`/( 1 ) = 2`	(1b)	`( 1 )/ = 0`	(1e)
`/( 2 ) = 0`	(1c)	`( 2 )/ = 1`	(1f)
`/( A ) = /A`	(1g)	`( A )/ = A/`	(1j)
`/( /A ) = A/`	(1h)	`( /A )/ = A`	(1k)
`/( A/ ) = A`	(1i)	`( A/ )/ = /A`	(1l)
`(/A a /B) = /A c B`	(1m)	`/(A/ a B/) = A b B`	(1p)
`(/A b /B) = /A a B`	(1n)	`/(A/ b B/) = A c B`	(1q)
`(/A c /B) = /A b B`	(1o)	`/(A/ c B/) = A a B`	(1r)

Tabela A2.1 – Propriedades para três valores.

`/( 0 ) = 1`	(2a)	`//( 0 ) = 2`	(2f)	`( 0 )/ = 3`	(2j)
`/( 1 ) = 2`	(2b)	`//( 1 ) = 3`	(2g)	`( 1 )/ = 0`	(2k)
`/( 2 ) = 3`	(2c)	`//( 2 ) = 0`	(2h)	`( 2 )/ = 1`	(2l)
`/( 3 ) = 0`	(2d)	`//( 3 ) = 1`	(2i)	`( 3 )/ = 2`	(2m)
`/( A ) = /A`	(2n)	`//( A ) = //A`	(2r)	`( A )/ = A/`	(2v)
`/( /A ) = //A`	(2o)	`//( /A ) = A/`	(2s)	`( /A )/ = A`	(2w)
`/( //A ) = A/`	(2p)	`//( //A ) = A`	(2t)	`( //A )/ = /A`	(2x)
`/( A/ ) = A`	(2q)	`//( A/ ) = /A`	(2u)	`( A/ )/ = //A`	(2y)
`/(A/ a B/)= A b B`	(2z)	`//(//A a //B) = A c B`	(2ã)	`(/A a /B)/ = A d B`	(2ê)
`/(A/ b B/)= A c B`	(2á)	`//(//A b //B) = A d B`	(2â)	`(/A b /B)/ = A a B`	(2ë)
`/(A/ c B/)= A d B`	(2à)	`//(//A c //B) = A a B`	(2é)	`(/A c /B)/ = A b B`	(2í)
`/(A/ d B/)= A a B`	(2ä)	`//(//A d //B) = A b B`	(2è)	`(/A d /B)/ = A c B`	(2ì)

Tabela A2.2 – Propriedades para quatro valores.

A2.2 Três valores

Partindo de um exemplo (tabela A2.3), é mais fácil compreender como a transformação é feita. A menor forma é C_2, mas a análise apresentada é sobre C_1.

```
C1 = [ (A/ b /B) a 1 ] c [ (A c /B) b (/A c B) b (A c B/) b A/]
```

2	1	1
1	2	1
2	0	2

Tabela A2.3 – Exemplo.

São definidos dois conceitos:

- Hierarquia: Número natural que determina a quantidade de parênteses abertos e não fechados quando a literal é escrita.
- Literal: Qualquer uma das entidades **0, 1, 2, 3, A, /A, //A, A/, B, /B, //B, B/, a, b, c, d, (,).**

Pode-se obter a expressão para os conectivos “a”, “b” e “c”. Esta análise cobre, apenas, a obtenção da expressão para o conectivo “a”, mas o mesmo procedimento pode ser empregado na obtenção das expressões para os demais conectivos. O primeiro passo é fazer a determinação da hierarquia de cada literal. A literal "(" aumenta a hierarquia, incluindo a sua própria hierarquia. A literal ")" diminui a hierarquia, excluindo a sua própria hierarquia. Fazendo, então, a determinação da hierarquia de cada literal da expressão fornecida, obtém-se o esquema a seguir.

```
[ (A/ b /B) a 1 ] c [ (A c /B) b (/A c B) b (A c B/) b A/ ]
1 2 2 2 2 2 1 1 1 0 1 22 2 2 2 1 2 2 2 22 1 22 2 2 2 1 1  1
```

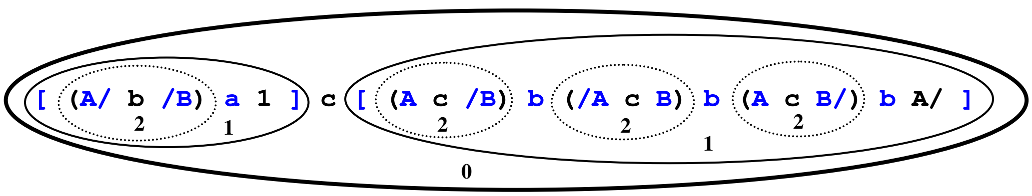


Figura A2.1 – Conjuntos de hierarquia.

A conversão pode ser feita por meio de dois tipos de análise:

- Top Down: Parte-se da hierarquia maior para a hierarquia menor.
- Button Up: Parte-se da hierarquia menor para a hierarquia maior.

A2.2.1 – Análise Top Down:

A maior hierarquia é ❷. As expressões nesta hierarquia são:

```
(A/ b /B) ; (A c /B) ; (/A c B) ; (A c B/)
```

Convertendo cada uma para o conectivo "a":

```
/(/A a B) ; (/A a B/)/ ; (A/ a /B)/ ; (/A a B)/
```

Reconstruindo-se a expressão:

```
C1 = [/(/A a B) a 1 ] c [ (/A a B/)/ b (A/ a /B)/ b (/A a B)/ b A/ ]
```

Para a hierarquia ❶, tem-se:

```
[ /(/A a B) a 1 ] ;     [ (/A a B/)/ b (A/ a /B)/ b (/A a B)/ b A/ ]
```

A primeira expressão já está na forma de um conectivo.

Convertendo a segunda expressão para o conectivo "a":

```
/[ /(/A a B/) a /(A/ a /B) a /(/A a B) a /A ]
```

Reconstruindo-se a expressão:

```
C1 = [ /(/A a B) a 1 ] c /[ /(/A a B/) a /(A/ a /B) a /(/A a B) a /A ]
```

Para a hierarquia ⓿, tem-se a expressão toda. Convertendo para o conectivo "a":

```
C1 = ( /( /(/A a B) a 1 ) a ( /(/A a B/) a /(A/ a /B) a /(/A a B) a /A )/ )/
```

A2.2.2 – Análise BottomUp:

A menor hierarquia é ⓿. Convertendo para o conectivo "a":

C1 = (/[(A/ b /B) a 1] a /[(A c /B) b (/A c B) b (A c B/) b A/])/

Para a hierarquia ❶, tem-se:

/[(A/ b /B) a 1] ; /[(A c /B) b (/A c B) b (A c B/) b A/]

A primeira expressão já está na forma de um conectivo.

Convertendo a segunda expressão para o conectivo "a":

[(A c /B)/ a (/A c B)/ a (A c B/)/ a /A]/

Reconstruindo-se a expressão:

C1 = (/[(A/ b /B) a 1] a [(A c /B)/ a (/A c B)/ a (A c B/)/ a /A]/)/

Para a hierarquia❷, tem-se:

(A/ b /B) ; (A c /B)/ ; (/A c B)/ ; (A c B/)/

Convertendo cada uma para o conectivo "a":

/(/A a B) ; /(/A a B/) ; /(A/ a /B) ; /(/A a B)

Reconstruindo-se a expressão:

C1 = (/[/(/A a B) a 1] a [/(/A a B/) a /(A/ a /B) a /(/A a B) a /A]/)/

As duas análises geram a mesma expressão. A transformação para uma expressão de um conectivo aumenta seu tamanho, mas não acrescenta nenhuma literal. Para os demais conectivos, tem-se:

a. **C1 = (/(/(/A a B) a 1) a (/(/A a B/) a /(A/ a /B) a /(/A a B) a /A)/)/**
b. **C1 = /(/(/(A/ b /B) b 2) b (/(A/ b B) b /(A b B/) b /(A/ b /B) b A/)/)**
c. **C1 = (/(/(A c B/) c 0) c (/(A c /B) c /(/A c B) c /(A c B/) c A)/)**

Também se pode obter as expressões de C2 e C3 para "a", "b", "c":

C1 = (((A/ b /B) a 1) c ((A c /B) b (/A c B) b (A c B/) b A/))
a. C1 = (/(/(/A a B) a 1) a (/(/A a B/) a /(A/ a /B) a /(/A a B) a /A)/)/
b. C1 = /(/(/(A/ b /B) b 2) b (/(A/ b B) b /(A b B/) b /(A/ b /B) b A/)/)
c. C1 = (/(/(A c B/) c 0) c (/(A c /B) c /(/A c B) c /(A c B/) c A)/)

C2 = (((A c /B) b (/A c B) b A/) a (A/ c A c B/ c /B))
a. C2 = (/(/(/A a B/) a /(A/ a /B) a /A) a (A a /A a B a B/)/)
b. C2 = (/(/(A/ b B) b /(A b B/) b A/) b (/A b A/ b /B b B)/)/
c. C2 = /(/(/(A c /B) c /(/A c B) c A) c (A/ c A c B/ c /B)/)

C3 = ((A/ c /B) b ((A b B) a (A/ b B/) a (A b /B) a (A/ b /B)))
a. C3 = /(/(A a B/) a (/(A/ a B/) a /(/A a /B) a /(A/ a B) a /(/A a B))/)
b. C3 = (/(/A b B) b (/(A b B) b /(A/ b B/) b /(A b /B) b /(A/ b /B))/)
c. C3 = (/(A/ c /B) c (/(/A c /B) c /(A c B) c /(/A c B/) c /(A c B/))/)/

A2.3 Quatro valores

Para quatro valores, o processo é o mesmo. Segue um exemplo didático.

0	1	1	0
1	3	1	1
2	2	3	2
0	2	2	1

Tabela A2.4 – Exemplo.

A menor forma é C2, mas o exemplo é feito para C1.

```
C1 = { [ (A b B) a (/A b B) a (A b /B) a 1 ] c [
(/A c //B) b (//A c //B) b /B b B b 2 ] } d [ (A d
/B) c (/A d B) c //A c A/ c //B c B/ ]
```

Fazendo, a determinação da hierarquia de cada literal, obtém-se:

```
C1 = { [ (A b B) a (/A b B) a (A b /B) a 1 ] c [ (/A c //B) b (//A c
     1 2 33 3 33 2 3 3 3 33 2 33 3 3 3 2 2 2 1 2 3 3 3  3 3 2 3 3  3

//B) b /B b B b 2 ] } d [ (A d /B) c (/A d B) c //A c A/ c //B c B/ ]
 3 3 2 2  2 2 2 2 2 1 0 1 22 2 2 2 1 2 2 2 22 1  1  1 1  1  1  1  1 1
```

A2.3.1 – Análise Top Down:

Hierarquia ❸:

```
(A b B)      →   /(A/ a B/)
(/A b B)     →   /(A a B/)
(A b /B)     →   /(A/ a B)
(/A c //B)   →   //(A/ a B)
(//A c //B)  →   //(A a B)
```

```
C1 = { [ /(A/ a B/) a /(A a B/) a /(A/ a B) a 1 ] c [ //(A/ a B) b //(A a
B) b /B b B b 2 ] } d [ (A d /B) c (/A d B) c //A c A/ c //B c B/ ]
```

Hierarquia ❷:

```
[ /(A/ a B/) a /(A a B/) a /(A/ a B) a 1 ] → [ /(A/ a B/) a /(A a B/) a /(A/ a B) a 1 ]
[ //(A/ a B) b //(A a B) b /B b B b 2 ] → /[ /(A/ a B) a /(A a B) a B a B/ a 1 ]
    (A d /B) → (/A a //B)/
    (/A d B) → (//A a /B)/
```

```
C1 = { [ /(A/ a B/) a /(A a B/) a /(A/ a B) a 1 ] c /[ /(A/ a B) a /(A a B)
a B a B/ a 1 ] } d [ (/A a //B)/ c (//A a /B)/ c //A c A/ c //B c B/ ]
```

Hierarquia ❶:

```
{ [ /(A/ a B/) a /(A a B/) a /(A/ a B) a 1 ] c /[ /(A/ a B) a /(A a B) a B a B/ a
1 ] } → //{ //[ /(A/ a B/) a /(A a B/) a /(A/ a B) a 1 ] a [ /(A/ a B) a /(A a B)
a B a B/ a 1 ]/ }
[ (/A a //B)/ c (//A a /B)/ c //A c A/ c //B c B/ ] → //[ /(/A a //B) a /(//A a
/B) a A a /A a B a /B ]
```

```
C1 = //{ //[ /(A/ a B/) a /(A a B/) a /(A/ a B) a 1 ] a [ /(A/ a B) a /(A a
B) a B a B/ a 1 ]/ } d //[ /(/A a //B) a /(//A a /B) a A a /A a B a /B ]
```

Hierarquia ⓿:

```
C1 = ( { //[ /(A/ a B/) a /(A a B/) a /(A/ a B) a 1 ] a [ /(A/ a B) a /(A a
B) a B a B/ a 1 ]/ }/ a [ /(/A a //B) a /(//A a /B) a A a /A a B a /B ]/ )/
```

A2.3.2 – Análise Bottom Up:

Hierarquia ⓿:

C1 = (/{ [(A b B) a (/A b B) a (A b /B) a 1] c [(/A c //B) b (//A c //B) b /B b B b 2] } a /[(A d /B) c (/A d B) c //A c A/ c //B c B/])/

Hierarquia ❶:

C1 = ({ //[(A b B) a (/A b B) a (A b /B) a 1] a //[(/A c //B) b (//A c //B) b /B b B b 2] }/ a [//(A d /B) a //(/A d B) a A a /A a B a /B]/)/

Hierarquia ❷:

C1 = ({ //[(A b B) a (/A b B) a (A b /B) a 1] a [(/A c //B)/ a (//A c //B)/ a B a B/ a 1]/ }/ a [/(/A a //B) a /(//A a /B) a A a /A a B a /B]/)/

Hierarquia ❸:

C1 = ({ //[/(A/ a B/) a /(A a B/) a /(A/ a B) a 1] a [/(A/ a B) a /(A a B) a B a B/ a 1]/ }/ a [/(/A a //B) a /(//A a /B) a A a /A a B a /B]/)/

Fazendo-se a análise para os demais conectivos, tem-se:

a. **C1 = ((//(/(A/ a B/) a /(A a B/) a /(A/ a B) a 1) a (/(A/ a B) a /(A a B) a B a B/ a 1)/)/ a (/(/A a //B) a /(//A a /B) a A a /A a B a /B)/)/**

b. **C1 = //((//(/(A b B) b /(/A b B) b /(A b /B) b 2) b (/(A b /B) b /(/A b /B) b /B b B b 2)/)/ b (/(//A b B/) b /(A/ b //B) b /A b //A b /B b //B)/)**

c. **C1 = /((//(/(/A c /B) c /(//A c /B) c /(/A c //B) c 3) c (/(/A c //B) c /(//A c //B) c //B c /B c 3)/)/ c (/(A/ c B) c /(A c B/) c //A c A/ c //B c B/)/)**

d. **C1 = ((//(/(//A d //B) d /(A/ d //B) d /(//A d B/) d 0) d (/(//A d B/) d /(A/ d B/) d B/ d //B d 0)/)/ d (/(A d /B) d /(/A d B) d A/ d A d B/ d B)/)**

As expressões utilizando os conectivos “a”,“b”,“c” e “d” podem ser construídas para as oito formas (C1, ..., C7), gerando 32 expressões. Deve-se escolher o conectivo que melhor satisfaça os requisitos da implementação; deve-se escolher a forma que proporcione a menor expressão, pois isso implica em um circuito menor, mais rápido e menos sujeito a falhas de fabricação e de operação.

Apêndice 3
Método Inspecional

A3.1 Introdução

O método inspecional não utiliza sistematicamente os postulados da lógica MVL; é um método de tentativa e erro. Tal como foi pode ser observado, nos capítulos 3, 4 e 5, ainda não foram descobertas regras de síntese que permitam obter minimização 100% eficiente, ou seja, encontrar a expressão com o menor tamanho possível.

O método inspecional cria expressões algébricas em ordem de complexidade e verifica, uma a uma, se a expressão gera a tabela fornecida, aquela cuja expressão algébrica se deseja obter. Se a tabela for obtida por meio de alguma expressão, tal expressão é fornecida como resposta. Se nenhuma expressão testada fornecer a tabela digitada, então a expressão não foi encontrada.

O processo é segmentado em tamanhos de expressão. Primeiramente, são tentadas as expressões com, apenas, uma literal (primeira etapa). Se a expressão desejada for obtida, o processo continua até que todas as opções de expressão com este tamanho tenham sido analisadas; pode ser que mais de uma opção tenha sido obtida, neste caso todas elas devem ser apresentadas, pois a decisão sobre qual delas escolher cabe ao usuário. Uma vez que a(s) expressão(ões) já foi(ram) obtida(s), o processo termina, não há razão para inspecionar expressões de tamanhos maiores.

Se a expressão desejada não for encontrada, então se parte para expressões de duas literais e um conectivo (segunda etapa). Seguem-se os mesmos passos da etapa anterior.

Se, ainda assim, a expressão não foi encontrada, agrupa-se as duas literais e o conectivo em parênteses e adiciona-se mais uma literal e um conectivo (terceira etapa). Amplia-se, assim, o tamanho da expressão progressivamente, num processo iterativo.

Uma vez que se deseja obter a expressão com tamanho mínimo, quando a expressão é encontrada, não se analisa mais as expressões de tamanhos maiores, porém, continua-se a análise para o tamanho para o qual a primeira expressão foi encontrada, a fim de se obter todas as expressões com o mesmo tamanho para a tabela dada, permitindo, ao usuário, determinar qual a expressão que melhor se encaixa à sua aplicação, seguindo um determinado critério de escolha.

0	A	B
1	/A	/B
2	A/	B/

0	A	B
1	/A	/B
2	//A	B//
3	A/	B/

Tabela A3.1 – Expressões mínimas (uma literal) para três e quatro valores.

Para três valores, são 9 funções ternárias de uma literais. Para quatro valores, 12.

As expressões mínimas também são as literais disponíveis para a formação de expressões maiores. As expressões de duas literais e um operador podem ser listadas.

A a 1	B a 1
A b 2	B b 2
A c 0	B c 0
/A a 1	/B a 1
/A b 2	/B b 2
/A c 0	/B c 0
A/ a 1	B/ a 1
A/ b 2	B/ b 2
A/ c 0	B/ c 0

A a /A	B a /B
A b /A	B b /B
A c /A	B c /B
/A a A/	/B a B/
/A b A/	/B b B/
/A c A/	/B c B/
A/ a A	B/ a B
A/ b A	B/ b B
A/ c A	B/ c B

A a B	/A a B	A/ a B
A b B	/A b B	A/ b B
A c B	/A c B	A/ c B
A a /B	/A a /B	A/ a /B
A b /B	/A b /B	A/ b /B
A c /B	/A c /B	A/ c /B
A a B/	/A a B/	A/ a B/
A b B/	/A b B/	A/ b B/
A c B/	/A c B/	A/ c B/

Tabela A3.2 – Expressões de duas literais para três valores.

A a 1	B a 1
A b 2	B b 2
A c 3	B c 3
A d 0	B d 0
/A a 1	/B a 1
/A b 2	/B b 2
/A c 3	/B c 3
/A d 0	B d 0
//A a 1	//B a 1
//A b 2	//B b 2
//A c 3	//B c 3
//A d 0	//B d 0
A/ a 1	B/ a 1
A/ b 2	B/ b 2
A/ c 3	B/ c 3
A/ d 0	B d 0

A a /A	B a /B	A a //A
A b /A	B b /B	A b //A
A c /A	B c /B	A c //A
A d /A	B d /B	A d //A
/A a //A	/B a //B	/A a A/
/A b //A	/B b //B	/A b A/
/A c //A	/B c //B	/A c A/
/A d //A	/B d //B	/A d A/
//A a A/	//B a B/	B a //B
//A b A/	//B b B/	B b //B
//A c A/	//B c B/	B c //B
//A d A/	//B d B/	B d //B
A/ a A	B/ a B	/B a B/
A/ b A	B/ b B	/B b B/
A/ c A	B/ c B	/B c B/
A/ d A	B/ d B	/B d B/

A a B	/A a B	//A a B	A/ a B
A b B	/A b B	//A b B	A/ b B
A c B	/A c B	//A c B	A/ c B
A d B	/A d B	//A d B	A/ d B
A a /B	/A a /B	//A a /B	A/ a /B
A b /B	/A b /B	//A b /B	A/ b /B
A c /B	/A c /B	//A c /B	A/ c /B
A d /B	/A d /B	//A d /B	A/ d /B
A a //B	/A a //B	//A a //B	A/ a //B
A b //B	/A b //B	//A b //B	A/ b //B
A c //B	/A c //B	//A c //B	A/ c //B
A d //B	/A d //B	//A d //B	A/ d //B
A a B/	/A a B/	//A a B/	A/ a B/
A b B/	/A b B/	//A b B/	A/ b B/
A c B/	/A c B/	//A c B/	A/ c B/
A d B/	/A d B/	//A d B/	A/ d B/

Tabela A3.3 – Expressões de duas literais para quatro valores.

No primeiro grupo, estão aquelas formadas por uma variável e um número. No segundo grupo, estão as formadas por uma variável operada com esta mesma variável deslocada. No terceiro, duas variáveis.

Para três valores, são 63 funções ternárias de duas literais. Para quatro valores, 144. Essas expressões, listadas e armazenadas, podem ser usadas para, juntamente com as expressões mínimas, formar todas as demais expressões.

As funções pré-definidas (tipo 1) são aquelas de uma e duas literais, totalizando 72 para três valores (9 + 63) e 156 para quatro valores (12 + 144). Levando-se que existem 19683 funções ternárias de duas entradas e mais de quatro bilhões quaternárias de duas entradas, as funções pré-definidas cobrem uma parcela desprezível de funções.

Chamando-se de “*li*” as literais (tabela A3.1), de “*op*” o conectivo e de “*()*” as expressões de duas literais, pode-se fazer a seguinte progressão:

- <u>Tipo 1</u>: “*li*” e “*()*” – as expressões catalogadas.
- <u>Tipo 2</u>: *() op li*
- <u>Tipo 3</u>: *() op ()*
- <u>Tipo 4</u>: (*() op ()*) *op li*
- <u>Tipo 5</u>: (*() op ()*) *op ()*

Há uma progressão. Os tipos podem ser escritos, genericamente, como:

- <u>Tipo n (par)</u>: (tipo n-1) *op li*
- <u>Tipo n+1 (ímpar)</u>: (tipo n-1) *op ()*

Para três valores, “*li*” compõe 9 combinações, e, para quatro valores, 12. “*()*” compõe 63 funções ternárias e 144 quaternárias. Sendo assim, pode-se estipular a quantidade de combinações que cada tipo abrange.

Para cada aumento do tipo, a quantidade de combinações de funções aumenta exponencialmente, o tempo de processamento também aumenta exponencialmente, o software torna-se mais lento. Porém, como as combinações aumentam, aumenta, também, a probabilidade da função desejada ser encontrada. No software LMVS, foi incluído até o tipo 3, uma sugestão para trabalhos futuros é o aumento do número de tipos.

Tipo	Ternário	Quaternário
1	$9 + 63 = 72$	$12 + 144 = 156$
2	$63 \cdot 9 = 567$	$144 \cdot 12 = 1728$
3	$63^2 = 3969$	$144^2 = 20K$
4	$63^2 \cdot 9 = 36K$	$144^2 \cdot 12 = 248K$
5	$63^3 = 250K$	$144^3 = 3M$

Tabela A3.4 – Quantidade de combinações.

O principal causador de demora no processamento está no fato de que, para cada tentativa, é preciso construir uma tabela a partir de uma expressão. Um refinamento no algoritmo utilizado no programa "*expressão -> tabela*" certamente melhorará bastante o desempenho do programa "*inspecional*".

A3.2 Exemplos

Os exemplos a seguir ilustram o poder do método inspecional e o quanto as técnicas de minimização MVL precisam ser aperfeiçoadas.

Inspecional	Método convencional
C = (A b /A) b B/ C = (A b B/) b /A C = (/A b B/) b A	C2 = /A b A b B/
C = (/A a 1) b B/	C2 = [/A b A b B/] a [A/ c /A c B/ c B]
C = (A b B/) a 1	C1 = (A b B/) a 1

Tabela A3.4 – Tipo 2 (três valores).

Inspecional	Método convencional
C = (A/ b B/) a (A b 2) C = (A/ b B/) a (A/ b A) C = (A/ b B/) b (A a B/)	C1 = [(A/ b B/) a 1] c [A b A/ b B/]
C = (/A b A/) b (B a 1) C = (B/ b B) b (A/ a 1) C = (B/ b B) a (/A b A/) C = (B/ b B) b (/A b A/) C = (A a B) b (/A b A/) C = (A/ b B) b (/A a B/) C = (A/ b B) b (/A b B/) C = (A/ a /B) b (B/ b B) C = (A/ b B/) b (/A b B)	C2 = /A b A/ b B b B/
C = (A/ a B) b (/A a B/) C = (A/ b B/) a (/A b B)	C = (/A b B) a (A/ b B/)
C = (A b /B) a (A/ a 1)	C1 = (A b /B) a A/ a 1
C = (A/ b B/) a (A b B/)	C = (A b B/) a (A/ b B/)
C = (/A a B/) b (B a /B)	C2 = [/A b B b B/] a [A/ c /A c /B]
C = (/A c B) a (A/ b 2)	C2 = [(/A c B) b A/] a [(A/ a /B) c /A c B]
C = (A/ c B/) a (B b /B)	C2 = [A/ b /B b B] a [(A a B) c A/ c B/]

Tabela A3.5 – Tipo 3 (três valores).

Inspecional	Método convencional
C = (A a 1) a B C = (B a 1) a A C = (A a B) a 1	C1 = A a B a 1

C = (A d 0) a 1
C = (A d 0) a //A
C = (//A a A/) a A
C = (A/ a A) a A//
	C1 = A a A/ a //A

C = (A a B) a 2	C1 = [//A a B] c [/A b A b /B b B b 2]
C = (B b /B) a A	C1 = { [A a B] c [/A b A b //A b /B b B] } d [//A c /A c A c //B c /B]
C = (A c B) a 1	C1 = (A b B) a (A/ b B) a (A b B/) a 1
C = (A a B) d 1	C4 = [A a B] b [(/A b /B) a A a B a 1]

Tabela A3.4 – Tipo 2 (quatro valores).

Inspecional	Método convencional
C = (A/ d B/) a (/A b /B)	C1 = (A/ b B/) a (//A b B/) a (/A b B/) a (A/ b //B) a (A/ b /B) a (/A b /B)
C = (/A c B) b (A c B/)	C2 = [(/A c B) b (A c B/)] d [(//A d /B) c /A c A c B c B/]
C = (/A d B/) a (A c /B)	C3 = { [A] a [(A a B/) d /A d 0] } b [(A b B) a //A a /A a B/ a /B]

Tabela A3.6 – Tipo 3 (quatro valores).

Como se pode observar, quando a função é encontrada pelo método inspecional, ela é fornecida em uma forma quase sempre mais compacta do que aquela proveniente do método convencional, exceto para as funções "*li*" e "*()*", para as quais os dois métodos fornecem o mesmo resultado.

Menos da metade das funções de três e quatro valores são cobertas pelos tipos 1,2 e 3, implementados. Para as funções não cobertas, é necessário recorrer ao método tradicional.

Apêndice 4
Cores

As janelas de configuração de cores são acessadas por meio do menu "Cores".

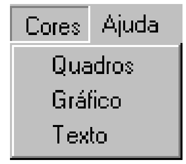


Figura A4.1 – Menu Cores.

1. Quadros: Mudar as cores das caixas que contém os números das matrizes, e botões.
2. Gráfico: Permite mudar as cores do gráfico da matriz de entrada.
3. Texto: Permite mudar as cores dos textos e do fundo dos textos e o tamanho da fonte.

As janelas de configuração de cores são divididas em duas áreas, a de edição de cores e a de armazenamento. A janela de configuração das cores dos quadros (figura A4.2) armazena quatro cores, a do gráfico (figura A4.3), seis, e a do texto (figura A4.4), duas.

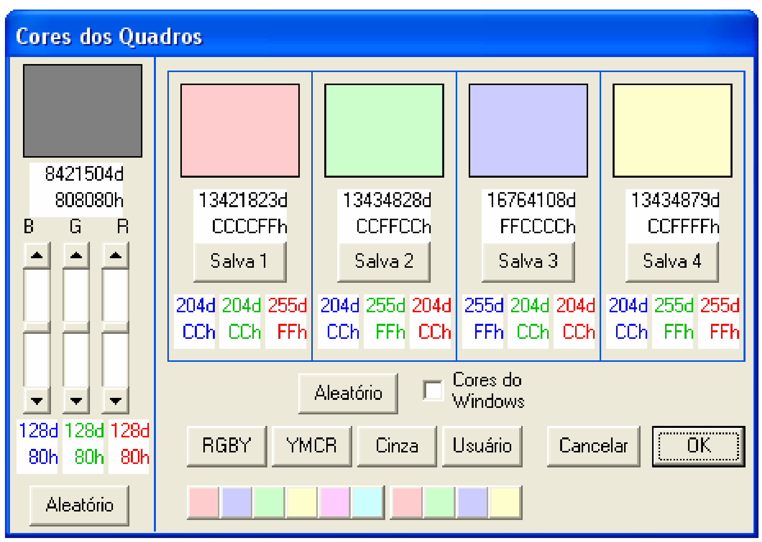


Figura A4.2 – Janela das cores dos quadros.

Grava as alterações e retorna à janela anterior.

Volta à janela anterior sem gravar as alterações.

Coloca, nas cores individuais, valores aleatórios.

Coloca, nas cores individuais, os valores vigentes.

Clicando na caixa de demonstração da cor armazenada (figura A4.5), a cor é transferida para a área de edição. Alguns botões colocam valores padronizados para as cores individuais. Outros botões pequenos e coloridos colocam valores individuais na área de edição. Uma caixa de seleção com rótulo 

"**Cores do Windows**" permite que, ao invés das cores mostradas, sejam usadas as cores do Windows. Com exceção das figuras deste tópico, todas as demais foram obtidas com esta opção selecionada.

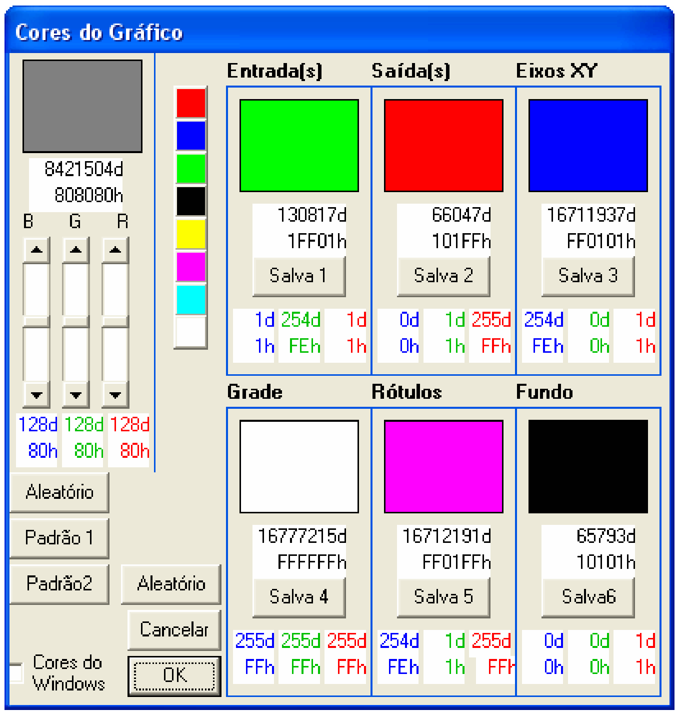


Figura A4.3 – Janela das cores do gráfico.

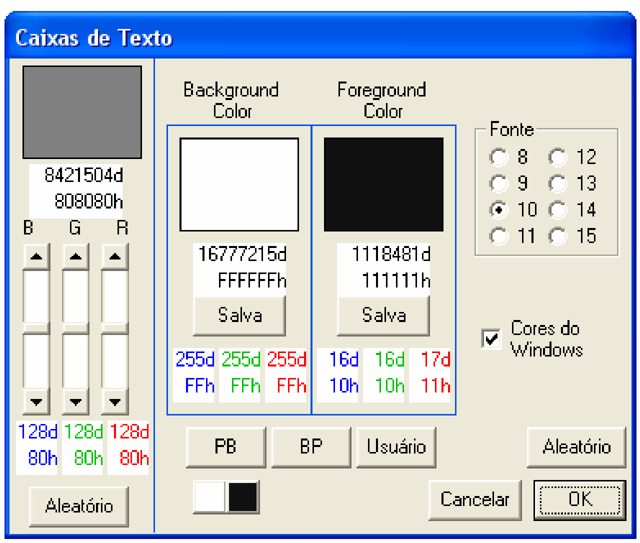


Figura A4.4 – Janela das cores e tamanho dos textos.

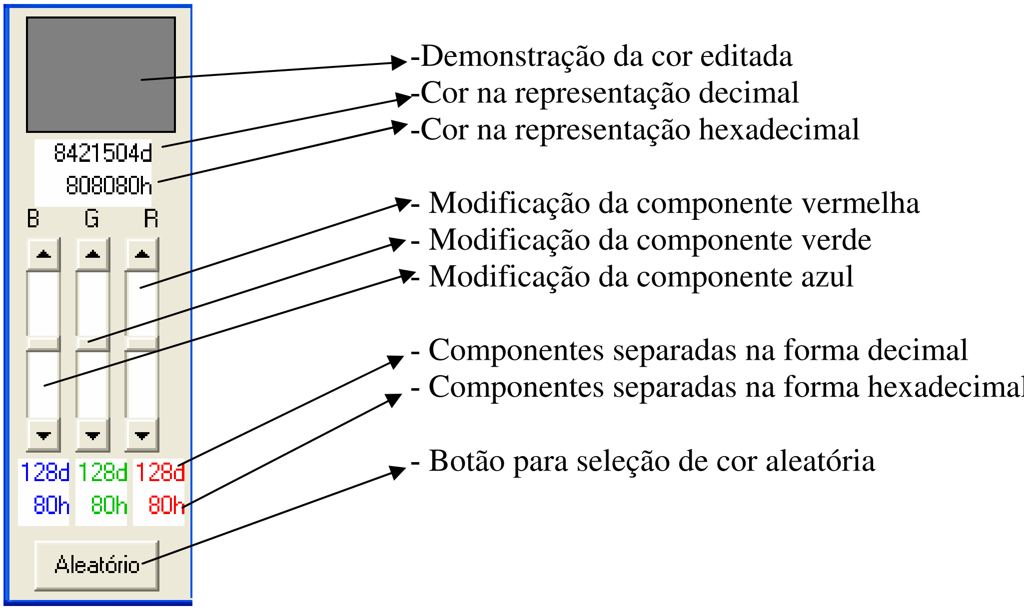


Figura A4.5 – Área de edição de cores.

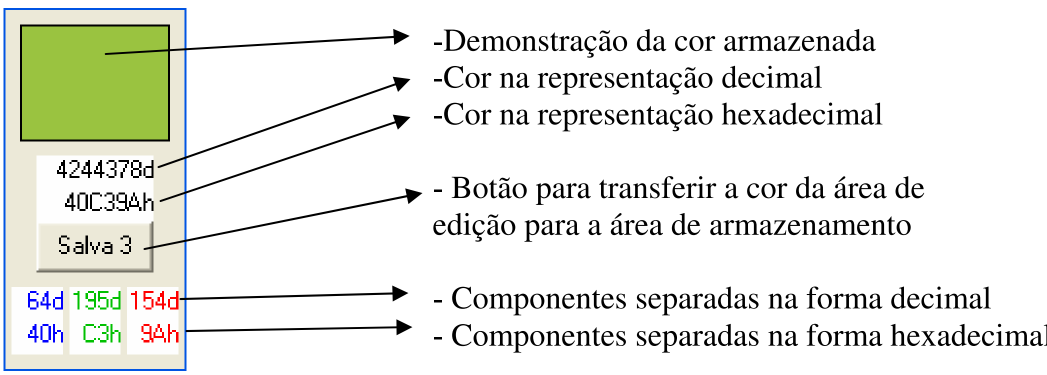


Figura A4.6 – Área de armazenamento de cor individual.

Quadros:

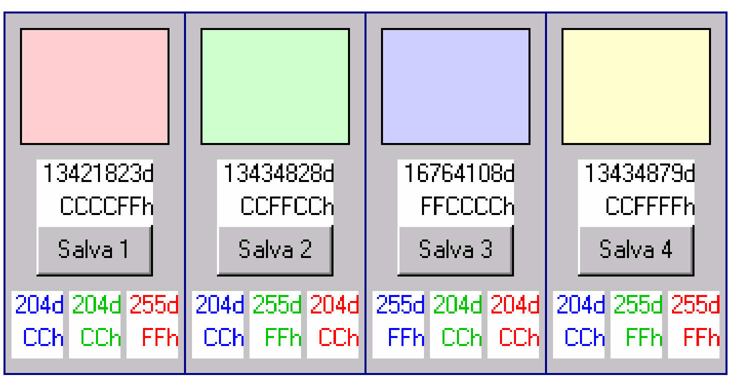


Figura A4.7 – As cores selecionadas para os quadros.

Figura A4.8 – Matriz e botões usando as cores definidas.

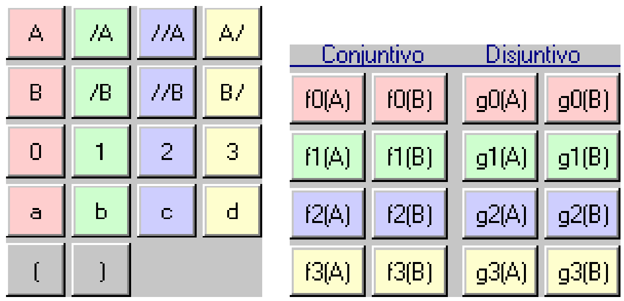


Figura A4.9 – Botões para os modos *Expressão->Tabela*, *1 Conectivo* e *Epstein*.

Texto:

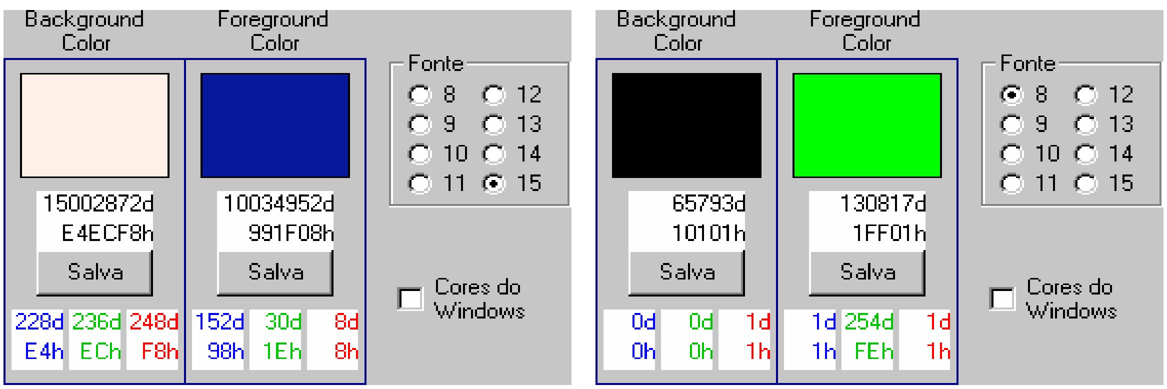


Figura A4.10 – Comparação entre duas formatações de texto diferentes.

Na janela das cores e tamanho dos textos, é possível escolher o tamanho da fonte, entre 8 e 16. A fonte usada é *Courier New Bold*.

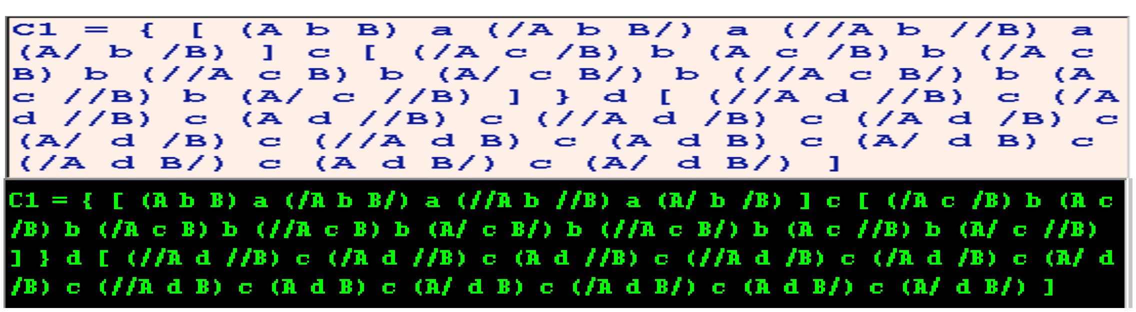


Figura A4.11 – As aplicações das formatações escolhidas.

Apêndice 5
Glossário

A: Variável de entrada menos significativa.

ALFA (*a*)(α): Operador *mínimo*, seleciona o menor número entre dois ou mais.

B: Variável de saída das funções de uma entrada B(A) ou a variável mais significativa das funções de duas entradas C(A,B).

b0, b1, b2, b3, b4, b5, b6, b7, b0*, b1*, b2*, b3*, b0, b1**, b2**, b3****: As SFBs para funções de uma entrada.

B1, B2, B3, B4, B5, B6, B7, B8: Todas as formas de construção de funções MVL de uma entrada a partir de SFBs

Base: Número natural que indica a quantidade de valores.

BASE: Deslocamento para a esquerda ou para baixo, positivação.

BETA (*b*)(β): Operador ALFA com um deslocamento TOPO das variáveis (A,B,C).

C: Variável de saída das funções de duas entradas C(A,B).

CV: Conjunto Verdade

c0, c1, c2, c3, c4, c5, c6, c7, c0*, c1*, c2*, c3*, c0, c1**, c2**, c3****: As SFBs para funções de duas entradas.

C1, C2, C3, C4, C5, C6, C7, C8: Todas as formas de construção de funções MVL de duas entradas a partir de SFBs

Comparador de dominância: Compara duas ou mais variáveis, selecionando a menor delas (operador *mínimo*) ou a maior delas (operador *máximo*).

Conectivo: Conecta duas ou mais variáveis. Os comparadores são um subconjunto dos conectivos. Os conectivos são um subconjunto dos operadores.

CPB: Conversão Para Binário; pode ser tanto o processo de conversão de uma função MVL para binário, como a expressão adicional que realiza essa atividade (*a 1*), por exemplo.

DELTA (*d*): Operador ALFA com três deslocamentos TOPO das variáveis (A,B,C).

Dominante: Valor mais forte de todos, segundo algum critério de comparação.

Expressão algébrica: Função definida por meio das variáveis de entrada.

F: Valor lógico FALSO.

Função: Lei de formação da variável de saída a partir da(s) variável(is) de entrada.

GAMA (*c*): Operador ALFA com dois deslocamentos TOPO das variáveis (A,B,C).

Indiferente: Valor mais fraco de todos, segundo algum critério de comparação.

Irrelevante (*X*): Valor lógico que pode ser substituído por outros.

Irrelevante fixo: Valor irrelevante gerado pela lei de formação das funções MVL a partir de SFBs.

Irrelevante móvel: Valor irrelevante gerado pela eliminação da CPB (2º critério). Pode ser total ou parcial.

Lógica aberta: Define-se valor falso, verdadeiro, e a graduação intermediária de valores.

Lógica fechada: Não há falso nem verdadeiro, há um círculo de valores.

Mapa: Função definida por uma tabela cuja dimensão é dada pela quantidade de variáveis de entrada.

MOD: Operação *Módulo* que fornece o resto da divisão inteira, cerne da lógica fechada.

Operador: Transforma uma ou mais variáveis (inclui os conectivos).

Porta lógica: Uma unidade dos circuitos lógicos, que realiza um deslocamento, uma inversão, uma comparação, ou uma mescla.

Secundário: Valor imediatamente mais fraco que o dominante.

SF: Sub-função genérica, pode ser binária ou MVL.

SFB: Sub-Função Binária; fornece apenas dois valores consecutivos ou vizinhos.

Tabela verdade: Função definida por meio da varredura das combinações da(s) variável(is) de entrada.

Terciário: Valor imediatamente mais fraco que o secundário.

TOPO: Deslocamento para a direita ou para cima, negação.

V: Valor lógico VERDADEIRO ou DOMINANTE.