UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ELETRÔNICA E MICROELETRÔNICA

Estudo do Fluxo de Projeto de Circuitos Integrados Digitais de Aplicação Específica (ASICS) Aplicado a um CI Monitor de Velocidade

WELLINGTON ROMEIRO DE MELO

ORIENTADOR: Prof. Dr. JOSÉ ANTONIO SIQUEIRA DIAS

Dissertação apresentada como parte dos pré-requisitos exigidos para obtenção do TÍTULO DE MESTRE EM ENGENHARIA ELÉTRICA

Banca Examinadora

Prof. Dr. José Antonio Siqueira Dias – (DEMIC/FEEC/UNICAMP)
Prof. Dr. Saulo Finco - (DCSH/CenPRA)
Prof. Dr. Wilmar Bueno de Moraes - (DEMIC/FEEC/UNICAMP)
Prof. Dr. Alberto Martins Jorge - (DEMIC/FEEC/UNICAMP)

Campinas
18 de junho de 2004

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

M491e

Melo, Wellington Romeiro de
Estudo do fluxo de projeto de circuitos integrados digitais de aplicação específica (ASICS) aplicado a um CI monitor de velocidade / Wellington Romeiro de Melo. --Campinas, SP: [s.n.], 2004.

Orientador: José Antonio Siqueira Dias.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Projetos. 2. Circuitos integrados digitais. 3. Circuitos integrados – Simulação por computador. 4. Layout. I. Dias, José Antonio Siqueira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Este trabalho apresenta um estudo do fluxo de projeto de circuitos integrados digitais de aplicação especifica, voltado a um circuito integrado Monitor de Velocidade. O objetivo deste estudo foi o de percorrer as etapas de realização de sistemas digitais utilizando ferramentas EDA profissionais, partindo de uma concepção inicial e chegando ao protótipo de um produto, um circuito monitor de velocidade. As ferramentas possibilitam a descrição de projetos digitais em forma de esquemáticos, tabela verdade, máquina de estados e HDLs. O resultado desta descrição são HDLs simuláveis e sintetizáveis em uma tecnologia especifica. O circuito Monitor de Velocidade foi sintetizado para um SDRAM configurável que integra uma placa de desenvolvimento educacional, onde os testes elétricos funcionais foram realizados. O Circuito Monitor de Velocidade atendeu as especificações tanto na simulação quanto no teste elétrico funcional, e com esta validação, encerra-se o fluxo com o layout e verificação física. O estudo teórico das técnicas digitais de projetos de ASICs, a execução de um projeto, a sua concretização (na forma de um protótipo baseado em FPGA) e os respectivos testes funcionais sobre este protótipo possibilitaram o domínio tecnológico do processo de projeto em circuitos digitais.

Abstract

This work presents a study of the digital design flow in application specific integrated circuits. A case study of a speed monitor integrated circuit is reported. The purpose of this study is to show all the steps of the design of digital systems (using professional EDA tools). This was done starting from an initial specification and finishing with a prototype of a product, the aforementioned “speed monitor circuit”. This tools permit to describe digital designs in schematics, truth table, finite state machine and HDLs. The HDLs are simulated and synthesized in a specific technology.

The Speed Monitor Circuit was synthesized in a configurable SDRAM which is part of an educational PC board. The functional electric tests were realized using this kit. The measured results of the prototype of the Speed Monitor Circuit agreed with the specifications, being very close to the results obtained in simulation. After this validation of the design in the SDRAM educational PC board, the layout and physical verifications (Design Rule Check – DRC and Layout versus Schematic – LVS) of the IC are realized.

The description of the theoretical study of the digital techniques employed to design ASICS followed by the design and functional tests of a prototype of a Speed Monitor Circuit based in FPGA makes this document a valuable educational source for all those who are beginning to work with the design of ASICs digital integrated circuits.

Dedicatória:

Dedico este trabalho a minha esposa Patrícia Serazzi, aos meus pais Francisco Canindé Soares de Melo e Maria Inês Romeiro, meus sogro e sogra, Ary Serazzi e Ivani Veronezzi Serazzi pelo apoio e carinho em todos os momentos importantes de minha vida.

Agradecimentos

A Deus e ao Senhor Bom Jesus de Iguape a quem ofereço, todo o meu dia, meu trabalho, minhas lutas, minhas alegrias e minhas dores. Concedei a mim, minha família e todos a quem agradeço neste trabalho, a vossa benção e uma vida feliz.

Aos meus pais Maria Inês e Francisco, aos meus irmãos Francis e Cíntia, pelo apoio, incentivo, infra-estrutura e base que sempre me proporcionaram.

A minha amada esposa Patrícia Serazzi, seus pais Ary e Ivani e sua irmã, cunhado e sobrinho Ivânia, Dagoberto e Felipe pelo apoio e motivação.

Ao Prof. Dr. José Antônio Siqueira Dias pela orientação, compreensão, idéias, incentivo e tempo despendidos para o término desta dissertação.

Ao Dr. Saulo Finco, do CenPRA/DCSH, pela amizade, ensinamentos, idéias, incentivo e tempo despendidos ao término desta dissertação.

Ao meu amigo Marcos Henrique Mamoru Otsuka Hamanaka pelo incentivo e colaboração com este projeto.

Aos meus amigos, colegas e profissionais que colaboraram de alguma forma com este projeto Luis Eduardo Seixas Jr., Antonio Carlos Fiore de-Mattos, Paulo José Martins Tavares, Diogo Rosário, Carlos Roberto Mendes de Oliveira, José Carlos da Silva e Noêmia.

Ao Centro de Pesquisas Renato Archer (CenPRA), em especial a Divisão de Tecnologia de Concepção de Sistemas de Hardware (DCSH) pela colaboração e disponibilização das ferramentas necessárias para desenvolvimento do projeto.

Ao CNPQ e a Capes pelo apoio financeiro.

Epigrafe: “Obstáculos são coisas que a pessoa vê quando tira os olhos dos objetivos”.

Anônimo

Lista de Ilustrações

Lista de figuras

Lista de Tabelas

Lista de abreviaturas e símbolos

AMS	- Analog Mixed Signal	- Sinais Analógicos e Mistos
ASCII	- American standard Code for Information Interchange	
ASIC	- Application Specific Integrated Circuit	- Circuito Integrado de Aplicação Especifica
ASM	- Algorithmic State Machine	- Algoritmo de Máquina de Estados
BCD	- Binary Coded Decimal	- Binário Codificado para Decimal
CAD	- Computer Aided Design	- Projeto Auxiliado pelo Computador
CD	- Compact Disc	- Disco compacto
CI	- Circuito Integrado	
CIF	- Caltech Intermediate Form	
CIMV	- Circuito Integrado Monitor de Velocidade	
CPLD	- Complex Programming Logic Device	- Dispositivo lógico com programação complexa
DLP	- Dispositivo Lógico Programável	
DRC	- Design Rule Check	- Cheque de Regras de Projeto
DVD	- Digital Versatile Disc	- Disco Digital Versátil
EDA	- Electronic Design Automation	- Automatização de Projetos Eletrônicos
FPGA	- Field Programmable Gate Array	- Matriz de Portas Lógicas Programáveis por Campo
FSM	- Finite State Machine	- Máquinas de Estado Finitas
GDS	- Graphics Design System	
HDL	- Hardware Description Language	- Linguagem de Descrição de Hardware
IC	- Integrated Circuit	- Circuito Integrado
LSI	- Large Scale Integration	- Larga Escala de Intergração
LVS	- Layout versus Schematic	- Leiaute versus Esquemático
MSI	- Medium Scale Integration	- Média Escala de Integração
NRE	- Nonrecurring Engineering	
PC	- Personal Computer	- Computador Pessoal
PEX	- Parasitic Extraction	- Extração de Parasitas
PLA	- Programming Logic Arrays	- Matriz Lógica Programável
PLD	- Programmable Logic Device	- Dispositivo Lógico Programável
PLI	- Programming Language Interface	- Interface de Linguagem de Programação
RTL	- Register Transfer Level	- Nível de Transferência de Registros
UCE	- Unidade de Comando Eletrônico	
VHDL	- Very Hardware Description Language	- Linguagem de Descrição de Hardware
VHSIC	- Very High Speed Integrated Circuit	- Circuito Integrado de Alta Velocidade

Sumário

Introdução Geral

Introdução

A revolução eletrônica iniciou-se com a evolução da física do estado sólido e o desenvolvimento comercial do transistor na década de 50, notadamente com a produção dos rádios a transistor. Hoje temos a revolução digital, que liderada pelos microprocessadores, faz com que empresas como Intel, Motorola, Advanced Micro Devices, IBM, Sun Microsystems e Hewlett-Packard invistam bilhões de dólares por ano competindo intensamente para produzir processadores cada vez melhores, sendo que a cada um ano e meio há um aumento de 100% na performance dos processadores sem custo adicional. Hoje a tecnologia digital possui um mercado de cerca de $40 bilhões de dólares por ano, com aplicações em diversas áreas como: automotivos; processamento de sinais digitais; telefones celulares; DVDs; produtos para lazer; etc. [1], [2], [3], [4]

O sucesso e o desenvolvimento de circuitos digitais deve-se muito a algumas facilidades de projeto e fabricação, como: *Reprodutibilidade* – dado uma entrada, provavelmente o circuito digital produzirá na saída o mesmo resultado sempre; *Fácil de Projetar* – dispensa equações complicadas e o procedimento de projetar pequenos circuitos lógicos pode ser feito mentalmente sem conhecer operação de capacitores, transistores ou dispositivos que requerem modelos de cálculos; *Flexível e Funcional* – sendo o problema reduzido à forma digital, pode ser resolvido usando alguns passos lógicos; *Programável* – estamos familiarizados com computadores digitais e a facilidade que se pode projetar, escrever e depurar programas. Muitos projetos digitais são carregados por programas escritos em *hardware description language* (HDLs). Esta linguagem segue estrutura e função dos circuitos digitais para ser especificada ou modelada. Compiladores típicos de HDLs, incorporam ferramentas de simulação e programas de síntese. Estas ferramentas de software são usadas para testar modelos de hardware antes da fabricação do hardware real; *Velocidade* – atualmente dispositivos digitais são muito rápidos. Transistores num circuito integrado rápido podem chavear em menos de 10 ps, e, dispositivos complexos construídos com este transistor podem analisar a entrada e produzir um efeito na saída em menos de 2 ns. Isto significa que cada dispositivo pode produzir 500 milhões ou mais de resultados por segundo; *Economia* – circuitos digitais podem realizar muitas funções num pequeno espaço. Circuitos usados repetidamente podem ser integrados num simples chip e ser

produzido com baixo custo, tornando possível fabricar itens como: calculadoras, relógios, e cartões de aniversário sonoros, etc. *Tecnologia em constante avanço* - projetando um sistema digital, você saberá que em poucos anos haverá um mais rápido, barato ou melhor tecnologicamente. Projetistas experientes podem se adequar a estes avanços durante o projeto inicial do sistema, para prevenir a obsolescência do sistema. Por exemplo, computadores desktop freqüentemente têm soquetes de expansão para acomodar processadores mais rápidos ou memórias com mais capacidade que não estão disponíveis quando da fabricação do computador. [4]

Uma vez que possuem as vantagens dos circuitos digitais, pode-se entender o porque da grande variedade existente de circuitos integrados com funções lógicas programadas. A maioria destes dispositivos usa tecnologias que também permitem as funções serem reprogramadas, significando que ao se encontrar um erro de projeto, pode-se resolvê-lo utilizando o mesmo dispositivo. Estes dispositivos são genericamente chamados de Dispositivos Lógicos de Programação (PLD) e são MSI (Medium-Scale Integration) na industria de lógica programável.

Talvez, o mais interessante no desenvolvimento de tecnologias de circuitos integrados para uma parte dos projetistas digitais não seja o aumento do tamanho dos chips e sua densidade, mas o aumento de oportunidades “projete seu próprio chip”. Chips projetados pelo próprio usuário ou para produtos e/ou aplicações particulares são chamados de *custom* ICs ou *application-specific* ICs (ASICs). ASICs geralmente reduzem o número total de componentes e custo de fabricação de um produto por reduzir a quantidade de CI’s, o tamanho físico, a potência consumida e, freqüentemente, consegue-se alta performace.

Os custos de NRE (*nonrecurring engineering*) para projetar um ASIC podem exceder o custo de um projeto discreto de US$5.000,00 à US$250.000,00. As taxas NRE são pagas para o fabricante do CI e outros responsáveis pelo projeto das estruturas internas do chip, que desenvolvem ferramentas para fabricação e desenvolvimento dos testes do chip fabricado.

O custo NRE para um ASIC típico de média complexidade, com cerca de 100.000 portas, é de US$30.000,00-US$50.000,00.

Um projeto de ASIC normalmente faz sentido quando o custo NRE é adequado ao volume de vendas esperado do produto. O custo NRE para um projeto LSI *custom* – um

chip em que as funções, arquitetura interna e projeto detalhado em nível de transistor são feitos para um cliente específico – é muito alto, tipicamente da ordem de US$250.000,00 ou mais. Deste modo, um projeto *full custom* LSI é feito só para chips que tenham aplicação comercial geral ou que tenham um volume de vendas muito alto em uma aplicação específica (por exemplo, um chip para um relógio digital, circuitos de interface para PC, etc.). Para reduzir as taxas NRE, fabricantes de CIs têm desenvolvido bibliotecas de células (*standart cells*) que incluem comumente funções usadas em MSI, tais como decodificadores, registradores e contadores, e também funções comumente usadas em LSI tais como memórias, matrizes programáveis logicamente e microprocessadores. Em um projeto *standart cell*, o projetista lógico interconecta as funções de maneira a torná-lo um projeto multichip MSI/LSI. Células personalizadas são criadas (aumentando o custo) somente se absolutamente necessário. Todas as células são dispostas em ordem no chip, otimizando o layout, para reduzir a propagação de atrasos e minimizar o tamanho do chip. Minimizando o tamanho do chip reduz-se o custo por unidade por aumentar o número de chips que podem ser fabricados em uma única lâmina. O custo NRE para um projeto *standart-cell* é tipicamente da ordem de US$150.000,00 [4].

Como US$150.000,00 é uma soma muito elevada os fabricantes de CI, para tornar os ASICs acessíveis a mais pessoas, utilizam os *gate arrays*,que são CIs nos quais as estruturas internas estão em matrizes de portas, e sua interconexão não é especificada inicialmente. O projetista digital especifica o tipo de portas e interconexões. Embora o projeto do chip seja especificado no final em termos de portas lógicas o projetista trabalha com macrocélulas, funções de alto nível usadas em multichips MSI/LSI e projetos *standart cells*.

A principal diferença entre *standart cells* e *gate arrays* é que as macrocélulas e o layout de um *gate array*, não são tão otimizados quanto às de um *standart cell.* Então, o chip pode ser maior e, portanto, custar mais. Também, não há possibilidade de criar células personalizadas em *gate array*. Por outro lado, seu projeto pode ser terminado mais rápido e com menor custo NRE, variando em torno de US$5.000,00 à US$ 75.000,00 [4].

Objetivos

O objetivo deste trabalho é projetar um circuito integrado implementado em PLD (FPGA) e *Full Custom*, detalhando todas as etapas e/ou fluxo do projeto de sistemas digitais, implementando uma aplicação específica a uma necessidade real de um nicho de mercado. O projeto escolhido para percorrer as etapas foi um monitor de velocidade.

No decorrer do trabalho foram percorridas as principais etapas de realização de sistemas digitais utilizando as ferramentas EDA de alta performance tais como Mentor Graphics e Cadence, partindo de uma idéia inicial e chegando a um produto.

Mesmo utilizando toda a tecnologia disponível atualmente, existem alguns pontos fundamentais que um projetista digital deve considerar para fazer um bom projeto:

- Sempre documentar seu projeto para fazê-lo inteligível para você e para os outros;
- Entender e usar a construção padrão de blocos funcionais;
- Pensar em reduzir custos e deixando-o aberto para atualizações tecnológicas;
- Evitar projetos assíncronos e praticar projetos síncronos até ter uma boa metodologia adquirindo experiência.
- Boas ferramentas não garantem um bom projeto, apenas ajudam o projetista.

Organização

É apresentada uma introdução geral sobre a evolução da eletrônica digital e a importância do mercado de ASICs no que diz respeito ao aumento de projetos e sistemas particulares devido a redução dos custos e avanços tecnológicos.

O primeiro capítulo descreve de forma mais detalhada o fluxo típico de projetos digitais e as ferramentas utilizadas para realização das diversas etapas, bem como as diferentes formas de descrição de circuitos: as linguagens HDL utilizadas, simulação e síntese lógica. No segundo capítulo apresentada a especificação do circuito monitor de velocidade e dá-se início ao projeto, utilizando o fluxo de projetos descrito no primeiro capítulo. É descrita as atividades de captura dos esquemáticos, descrição de blocos por meio de tabela verdade e máquina de estados. Os principais blocos são simulados com o método de validação da especificação inicial realizando a simulação global.

No terceiro capítulo é apresentado a ferramenta utilizada, a síntese final, o veículo utilizado para realização dos testes elétricos e os resultados experimentais funcionais. Validada a especificação inicial do circuito nos testes no quarto capítulo é descrita a realização do layout e verificação física do circuito.

Capítulo 1 – Fluxo de Projetos de Circuitos Digitais

1.1 Introdução

Fluxo de projeto é um termo usado para descrever as várias fases do projeto de um Circuito Integrado. A primeira coisa necessária para o início de um projeto é a especificação das funções e das condições de operação do circuito. A importância desta fase não deve ser nunca subestimada, pois uma especificação mal executada poderá implicar em refazer o projeto, com apreciável aumento do custo global do produto final (RNE). [5]

Dada uma especificação, um fluxograma de projetos digitais genérico é exemplificado na Figura 1.

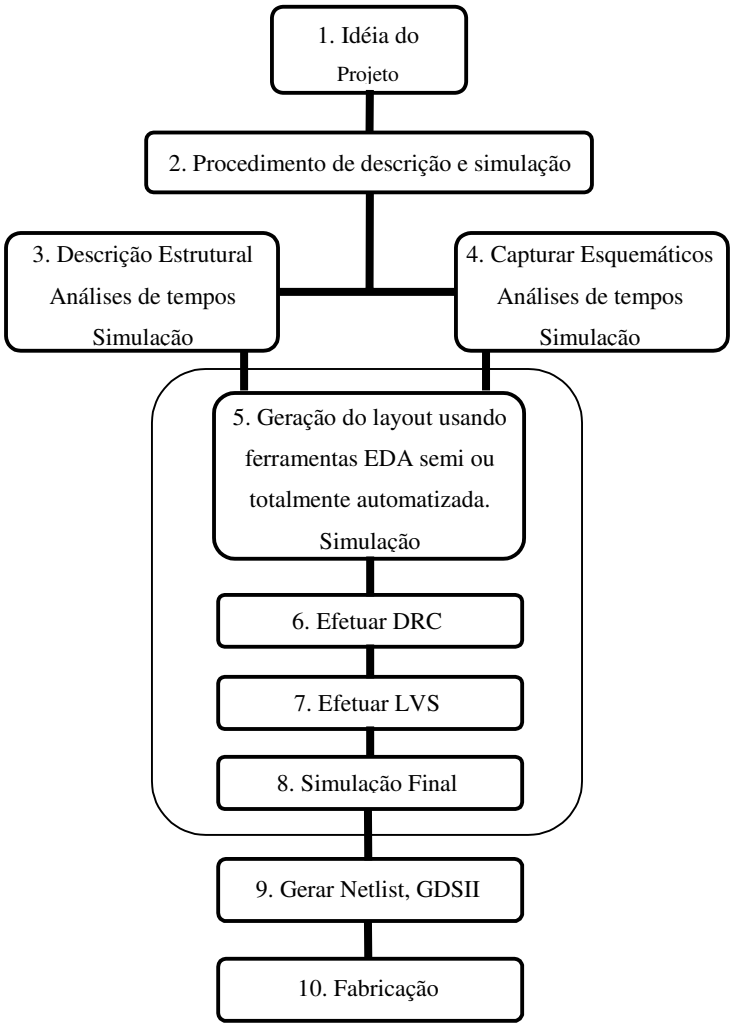


Figura 1: Exemplo de um fluxo de projeto digital genérico.

Baseado numa dada especificação, o projetista forma uma idéia geral com relação a solução para o problema. Neste estágio um diagrama de blocos geral é a solução para o projeto. Nesta etapa, geralmente, ferramentas de CAD não são necessárias. Este estágio costuma ser definido com o conceito do projeto em "Papel de Pão" [6], [7].

Estando a idéia geral definida, os próximos passos para a continuação da execução do projeto são:

- Descrição funcional: Hardware Description Languages (HDLs) são usados para modelar a idéia do projeto (Diagramas de Blocos). Detalhes do circuito e componentes elétricos não são especificados. Ao invés disto, a funcionalidade de cada bloco é modelada. Simulações são então necessárias para verificar se os blocos executam a função esperada. A descrição funcional é importante para confirmar a integridade da idéia do projeto.

- Descrição estrutural: HDLs são usados para descrever a construção do projeto em outro nível de abstração. Nestes estágios vários componentes eletrônicos e detalhes dos blocos de um sistema são modelados e pode incluir portas lógicas ou standard cells. Simulações para cada bloco separadamente são realizadas para testar a operação. Uma vez completados todos os blocos são interconectados e uma nova simulação é realizada. O sucesso desta simulação assegura que um circuito eletrônico pode ser construído de acordo com a descrição funcional.

- Descrição esquemática: Uma descrição esquemática é similar à descrição estrutural porém, usando símbolos de componentes eletrônicos. Na maioria dos casos, ferramentas gráficas são usadas para construir este ambiente. Vários componentes eletrônicos como portas lógicas e *standard cells* são inseridos e interconectados de modo a gerar os blocos do circuito. Esta é uma forma alternativa de descrição estrutural. Uma simulação com sucesso assegura que os circuitos eletrônicos, também, estão de acordo com a descrição funcional.

- Descrição de Layout: Este é o nível mais baixo de abstração possível em descrição de circuitos. Neste estágio, o circuito é descrito com relação ao silício e outros materiais que fazem parte do circuito integrado. Deve ser dedicada uma atenção especial à geometria dos componentes. Elementos parasitas precisam ser extraídos neste estágio. A colocação e interconexões dos blocos individuais de estágios anteriores precisam ser especificadas, planejando e organizando os blocos a serem roteados.

- Verificação do Layout versus Esquemático: Depois de completar a descrição do layout, uma verificação layout versus esquemático (LVS) é realizada. Isto assegura que o layout está em conformidade com o esquemático. Nesta etapa se dá um "*loop*" entre layout, LVS e DRC (Design Rule Check), até que haja total conformidade.

- Design Rule Check – DRC: Este estágio é dependente da tecnologia de processo que é usada para fabricar o chip. A verificação das regras de projeto assegura que as regras do processo de fabricação não são violadas. O layout deve ser desenhado baseado nas limitações da regra de projeto. Como já foi dito anteriormente, nesta etapa o projeto comuta entre layout, LVS e DRC até que haja conformidade. Sendo verificadas regras de *Electrical Rule Check* (ERC), Layout, *Electrical Static Discharge* (ESD) e Antena.

- Preparação da base de dados geração do CIF e GDSII: Este é o ultimo estágio antes da fabricação. A maioria das fabricas aceita as submissões nestes formatos. Estes são arquivos que descrevem o layout do CI. Eles incorporam todos os detalhes necessários para a fabricação do *chip* [8].

1.2 Especificação de um projeto digital

Esta provavelmente é a única fase do fluxo de projetos que não há a necessidade de um computador. Nesta fase, engenharia do projeto toma decisões com relação a aspectos gerais na construção do chip. Experiência é recomendada nesta fase do projeto.

Como exemplo, abaixo são apresentados alguns itens da especificação de um projeto digital de um contador de 8 bits. O nome dado a este contador foi bloco1_sc, e é parte integrante do CI monitor de velocidade.

1. Contador crescente digital de 8 bit;
2. Contar os pulsos de uma determinada entrada SS_in;
3. Contar até um determinado valor especificado externamente – velo_ref;
4. Atingindo o valor Velo_ref o contador reinicia a contagem – RN_sin;
5. Sinal VC, pulso ativo em nível alto com duração de um número de pulsos determinados por Velo_ref;
6. Entrada para clock de 1200 Hz, que será o valor a ser contado;

Dada a idéia inicial, pode-se representá-la em um diagrama de blocos como uma caixa preta relacionando a interface da idéia (chip) com o mundo externo. A Figura 2 ilustra essa idéia:

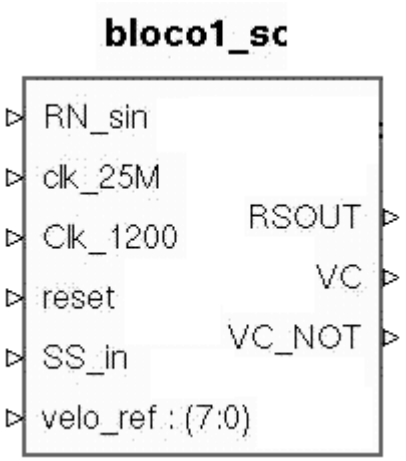


Figura 2: Caixa preta como diagrama de blocos.

Depois desta etapa, um diagrama de bloco mais detalhado (mostrando os detalhes dos blocos intermediários) pode ser desenhado, como ilustra a Figura 3:

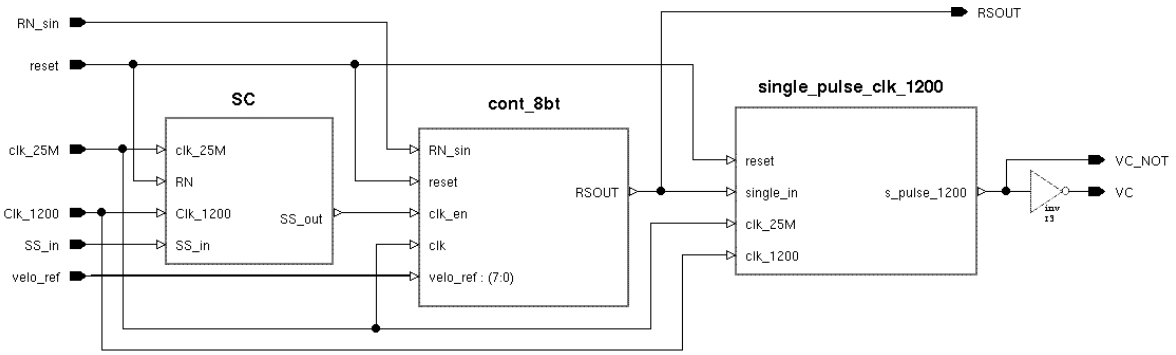


Figura 3: Diagrama de blocos internos ao bloco1_sc

A seguir a descrição de um dos blocos do circuito detalhada. O bloco cont_8bt é descrito em forma de células básicas na Figura 4.

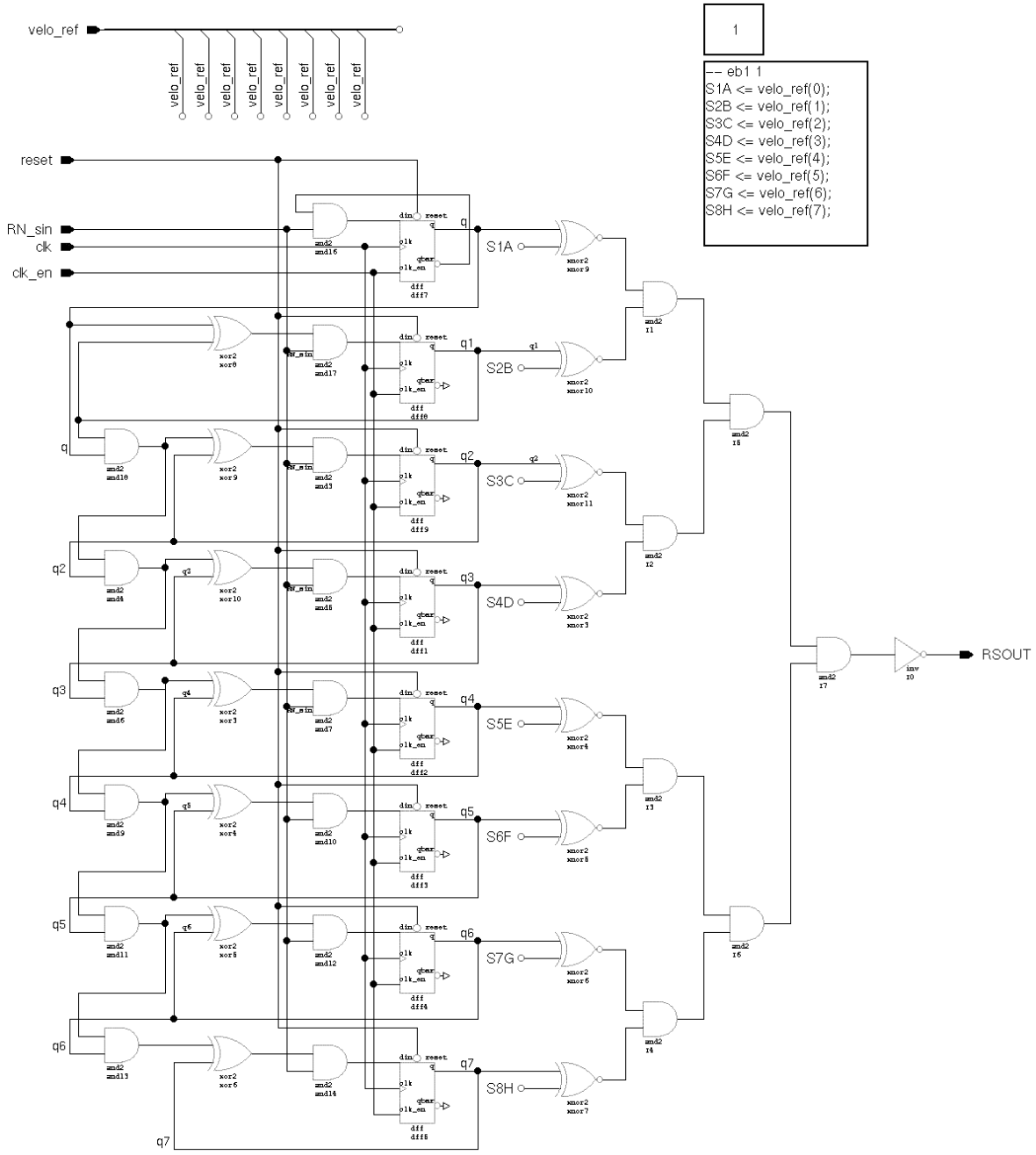


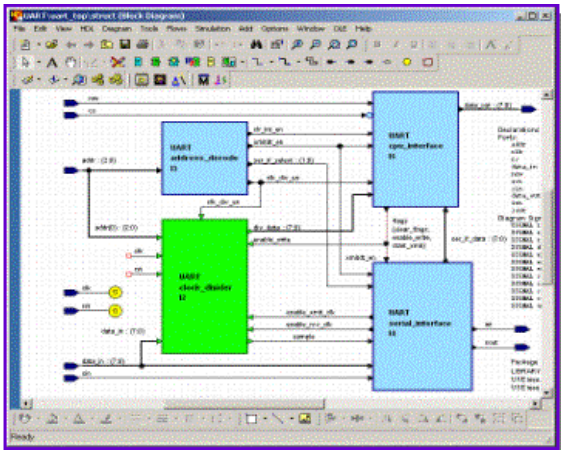
Figura 4: Descrição esquemática das células básicas do circuito cont_8bt.

O esquemático da Figura 4 precisa ser simulado para comprovar que a funcionalidade especificada foi devidamente implementada.

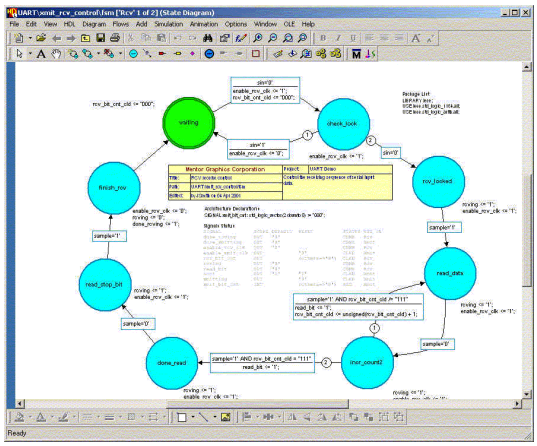
1.3 Descrição de Circuitos

Circuitos digitais podem ser descritos em diferentes representações com a mesma funcionalidade. Pode-se descrever um circuito eletrônico na forma de diagrama de blocos ou esquemático, representando assim as interligações e funcionalidade em nível de células

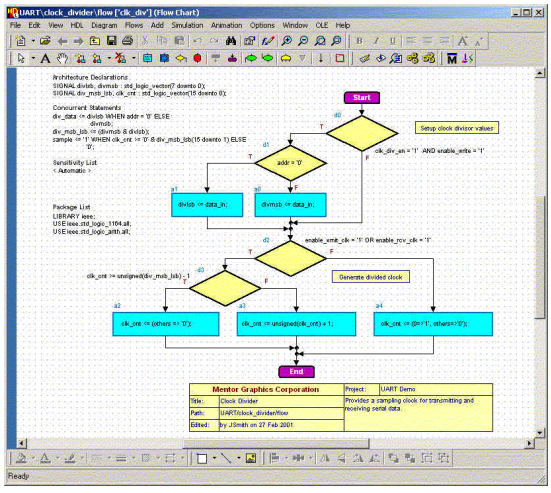
básicas; ou, representar, ainda, a funcionalidade do circuito, na forma de diagramas de estados, tabela verdade, fluxograma e descrição textual utilizando HDLs (VHDL ou Verilog). A melhor representação da funcionalidade do circuito vai depender de sua aplicação. Por exemplo, pode-se descrever um circuito com relação a diagramas de estados quando se deseja a definição de controle lógico, fluxograma para lógica seqüencial e *test benches*, tabela verdade para multiplexadores, *encoders*, *decoders*, etc. Existem ferramentas de software que oferecem a descrição dos circuitos nos diferentes modos. Mas, na prática, estes editores orientados a gráficos quando compilados, geram um código HDL, VHDL ou Verilog. A Figura 5 ilustra o aspecto de alguns editores de diagrama de blocos, máquinas de estados, fluxograma e tabela verdade da ferramenta de projetos HDL Designer Series [6], [7].



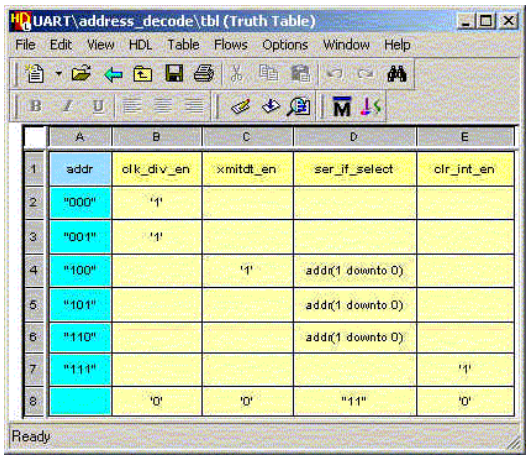
a)



b)



c)



d)

Figura 5: Ilustração dos editores de: a) diagramas de blocos, b) máquinas de estados, c) fluxograma e d) tabela verdade.

A seguir serão descritas formas utilizadas para descrição de circuitos digitais:

1.3.1 Esquemáticos

As atividades de edição de esquemáticos, simulação, síntese, verificação, e ferramentas de layout, normalmente precisam trabalhar juntas em um ambiente de apoio ao projeto para alcançar a máxima produtividade.

A abordagem baseada em captura de esquemáticos para projetos de circuitos integrados foi muito adotada. A partir dos anos 90, com um maior avanço nas ferramentas de automação, tem-se utilizado na prática uma abordagem de especificação de projeto, baseada no uso de linguagens de descrição de hardware (HDL), como uma etapa precedente a utilização de ferramentas de edição de esquemáticos no projeto de CIs.

Atualmente, a função principal dos editores de esquemáticos deixou de ser a especificação e tornou-se a função de interface gráfica ou visualização. Assim, esta ferramenta fornece ao projetista uma visão gráfica do circuito em diferentes níveis de abstração, utilizando o princípio de hierarquia. Deste modo, o editor pode ser utilizado tanto para a especificação de projetos quanto para a visualização gráfica dos resultados obtidos nos processos de síntese.

O editor de esquemáticos, portanto, pode ser utilizado na especificação do circuito a ser projetado, ou servindo como interface entre o projetista e as ferramentas que fazem parte do ambiente de apoio ao projeto. Observa-se que, neste contexto, é imprescindível que esta ferramenta de edição de esquemáticos seja integrável com outras ferramentas.

Deste modo, o editor de esquemáticos serve de interface entre o projetista e as ferramentas de síntese, permitindo ao projetista visualizar o circuito de forma hierárquica e em diferentes níveis de abstração.

1.3.2 Circuitos Lógicos

Circuitos lógicos são classificados em dois tipos: combinacionais e seqüenciais.

Circuitos Combinacionais

Em um circuito lógico combinacional a sua saída depende somente de sua entrada atual. Um circuito combinacional pode conter um número arbitrário de portas lógicas e

inversores, mas não pode haver "*loops*" de retorno que levem o sinal de saída ao sinal de entrada, já que este procedimento pode gerar um comportamento seqüencial no circuito [9].

A análise de um circuito combinacional inicia com o diagrama lógico ou esquemático e caminha para uma descrição formal das funções realizadas pelo circuito na forma de tabelas verdade e expressões lógicas. Na simplificação, é realizado o caminho inverso, inicia-se com a descrição formal para chegar ao diagrama lógico.

A Figura 6 ilustra um circuito lógico combinacional de 3 entradas e 1 saída, os sinais das saídas a partir de sinais de entrada, a expressão lógica e sua tabela verdade [9].

Fila	X	Y	Z	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

F = ((X + Y)*Z)+(X'*Y*Z')

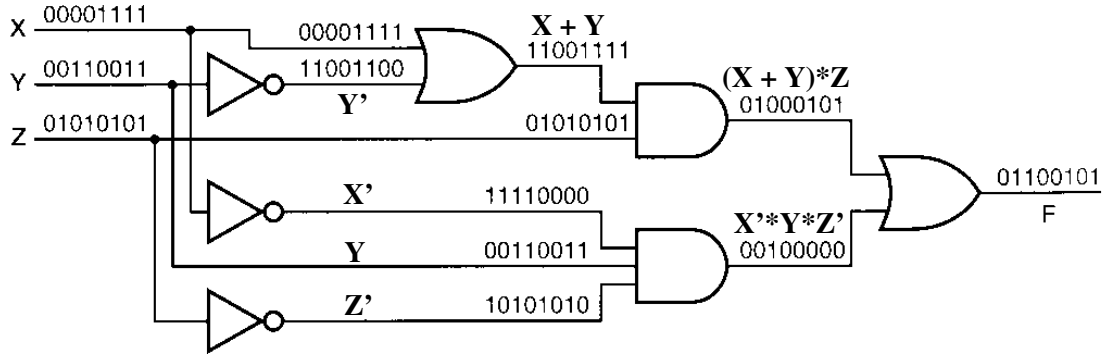


Figura 6: Circuito lógico combinacional de 3 entradas e 1 saída, sinais de saída gerados a partir de sinais de entrada, a expressão lógica e tabela verdade.

Circuitos Seqüenciais

Uma classe de circuitos que a saída depende do estado do circuito anterior, bem como dos valores presentes em sua entrada são chamados circuitos seqüenciais. Na maioria dos casos o funcionamento é cadenciado por um sinal periódico de relógio e são chamados de circuitos seqüenciais síncronos. Uma alternativa, que pela definição anterior ficam associados os circuitos que a transição de estado não é cadenciada por um sinal de relógio, ou onde o sinal de relógio não é periódico, são chamados circuitos seqüenciais assíncronos.

Circuitos síncronos são mais fáceis de projetar e são usados na grande maioria de aplicações práticas.

Os circuitos seqüenciais são também chamados de FSMs (finite state machines), um nome mais formal freqüentemente encontrado na literatura especializada. Este nome deriva do fato de o comportamento funcional destes circuitos poder ser representado usando um número finito de estados [10].

Circuitos Síncronos

Circuitos seqüenciais síncronos usam lógica combinacional e um ou mais flip-flops controlados por um *clock* que definem em qualquer instante o estado do sistema. Em qualquer circuito síncrono é identificável uma unidade de memória, um circuito combinacional que gera as entradas da unidade de memória e um circuito combinacional responsável pela implementação das saídas do sistema. O "*clock*" é um sinal periódico que consiste de pulsos onde as mudanças de estado podem ocorrer na borda positiva ou negativa de cada pulso. A Figura 7 mostra uma estrutura geral de circuitos síncronos [10].

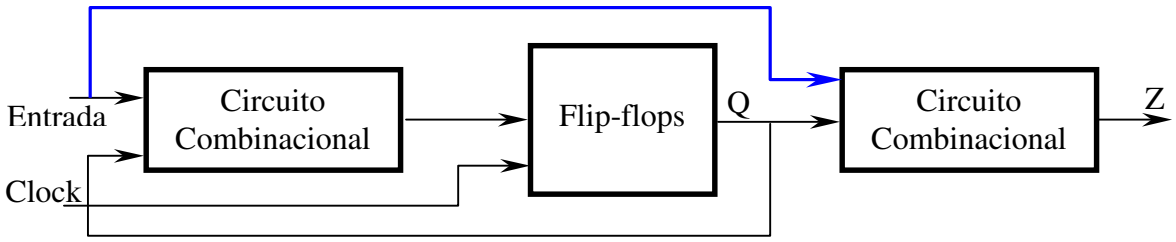


Figura 7: Estrutura geral de circuitos síncronos.

Circuitos Assíncronos

Em circuitos seqüenciais assíncronos as mudanças de estado não são disparadas por um pulso de *clock*. Em vez disto às mudanças de estado, dependem se cada uma das entradas do circuito tem nível lógico 0 ou 1, em qualquer tempo dado. Para ativar a operação as entradas do circuito devem mudar a seu tempo, ou seja, o tempo deve ser suficiente entre as mudanças nos sinais de entrada de modo a permitir que o circuito encontre um estado estável, quando todos os sinais internos param de comutar.

Máquinas de estados

Máquinas de Estados Finitas (FSM) são assim chamadas porque as seqüências lógicas que as implementam é fixada pelo número de estados possíveis. As saídas e o próximo estado de uma FSM são funções lógicas combinacionais de suas estradas e seu estado atual. A escolha do próximo estado pode depender do valor de uma entrada, comandando comportamentos mais complexos do que de contadores. FSM são cruciais para a realização do controle e tomada de decisão de sistemas lógicos digitais [31].

As máquinas de estado podem ser descritas em termos de diagramas de estado e tabelas. No entanto, para máquinas de estado mais complexas, estes modos de descrição podem ser de difícil aplicação. Um modo alternativo de descrição do comportamento de FSMs mais parecido com a descrição de software é o algoritmo de máquinas de estado, ASM, *algorithmic state machine*. O ASM é similar aos fluxogramas de programação, mas possui um conceito de tempos mais rigoroso.

A notação da ASM consiste em três elementos primitivos: caixas de estado, de decisão e de saída, como mostrado na Figura 8.

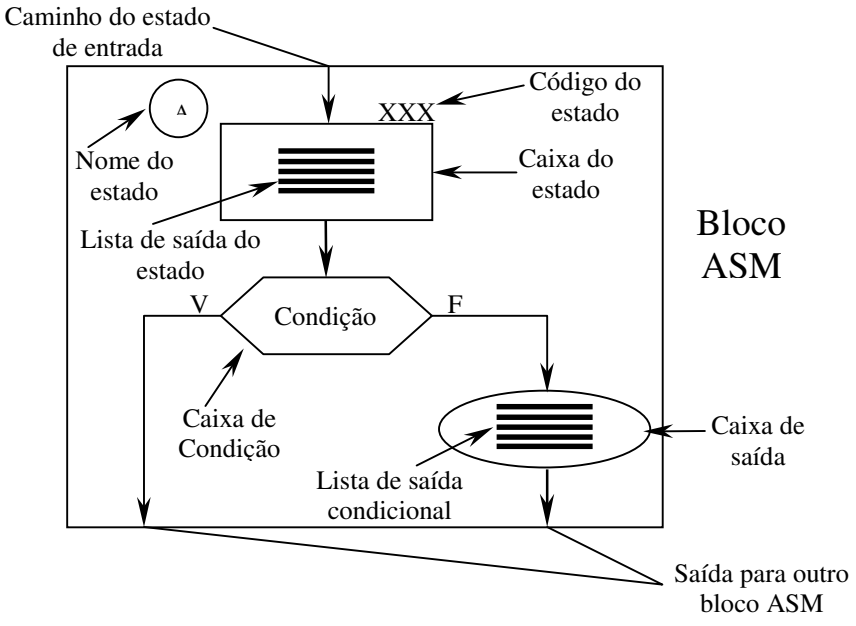


Figura 8: Elementos de uma notação ASM.

A caixa de estado, que contém uma lista de sinal de saída é identificada por um nome de estado simbólico em um circulo e um código de estado binariamente codificado. A

caixa de condição testa uma entrada para determinar um caminho de saída de um bloco ASM atual para o bloco da próxima entrada. A ordem em que as caixas de condição são cascateadas não tem efeito na determinação do próximo bloco ASM. A caixa de saída no caminho de uma caixa de estado contém os sinais que devem ser associados ao sinal mencionado na caixa de estado. A máquina de estados avança de um estado para o próximo em uma taxa discreta e em passos contínuos.

Existem dois modos básicos de organizar máquina de estados: a Máquina de Moore e a Máquina de Mealy. Estes nomes foram dados em homenagem a Edward Moore e George Mealy, que pesquisavam o comportamento deste tipo circuitos na década de 1950. A Figura 9 mostra os modelos gerais das máquinas de Moore e Mealy [9], [31].

Na máquina de Moore, a saída depende somente do estado atual (EA). Neste tipo de máquina apenas as variáveis de entrada estão associadas às transições entre estados. Na máquina de Mealy, a saída depende do estado atual e dos valores presentes na entrada. Neste tipo de máquina as variáveis de entrada e de saída estão associadas às transições entre estados. Uma comparação entre os dois tipos de máquinas, usando diagrama de transição de estados, é mostrada na Figura 10.

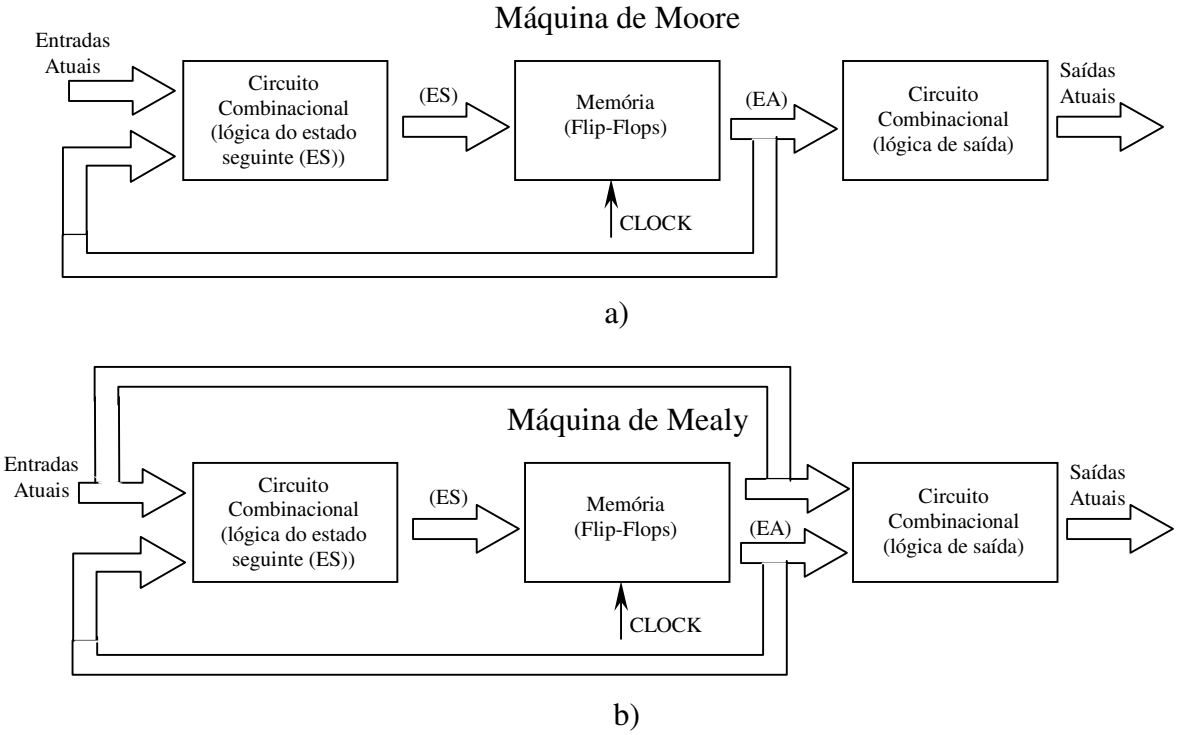


Figura 9: a) Máquina de Moore; b) Máquina de Mealy

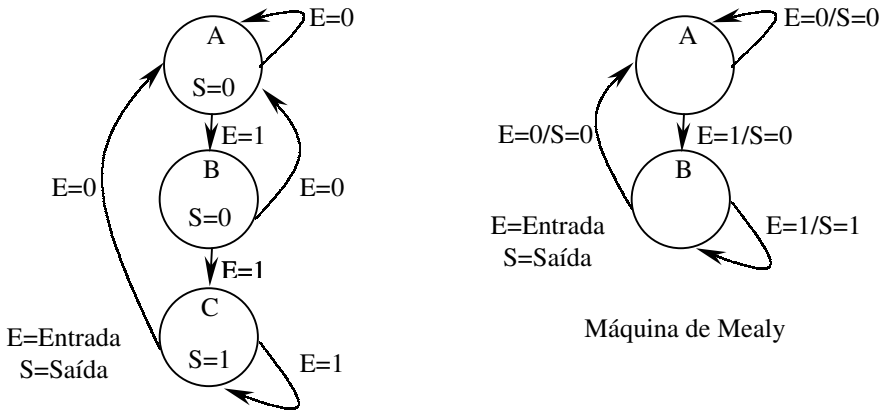


Figura 10: Dois diagramas de transição com as mesmas entradas e saídas representados em Máquina de Morre e Mealy, respectivamente.

Os estados são indicados por círculos e definidos por combinações de valores lógicos presentes nas variáveis de estado (flip-flops que constituem a memória do circuito). O estado seguinte e o valor das saídas são definidos pelo estado atual e pelo valor das entradas, quando ocorre uma transição ativa no sinal de *clock*. Por exemplo, quando o circuito se encontra no estado A, sendo a entrada exterior E=0, a próxima transição ativa no sinal de *clock* provocará a passagem para o estado A (mantém-se o estado atual); se, no entanto, a entrada exterior for E=1, o circuito passará para o estado B e assim sucessivamente, mudando para outros estados sempre de acordo com a entrada exterior e as setas indicadoras de mudança de estado. Devido a Máquina de Mealy associar as saídas com as transições, pode-se gerar a mesma seqüência de saída em menos estados que a máquina de Moore.

1.3.3 Linguagens HDL

Não é necessária a utilização de ferramentas de software para um projeto digital. Nas escolas, era usado um modelo de plástico com os símbolos lógicos no qual eram feitos os diagramas manualmente. Porém, ferramentas de software são uma parte indispensável de projetos digitais. O advento de síntese lógica a partir da década de 1980 mudou a metodologia de projetos radicalmente. Circuitos digitais poderiam ser descritos em RTL (*register transfer level*) por uso de um HDL (*hardware description languages*). Assim, o projetista especifica como os dados fluem entre registradores e como o projeto processa o

dado. A disponibilidade e a praticidade dos HDLs e ferramentas de simulação e síntese têm mudado o panorama dos projetos digitais nos últimos anos e apresentam enorme vantagem comparada com o projetos baseados em esquemáticos tradicionais.

Várias ferramentas de CAD (*computer-aided design*) melhoram a produtividade dos projetistas, ajudando na correção e qualidade dos projetos. Num mundo competitivo se torna obrigatório o uso de ferramentas de software para se obter alta qualidade nos resultados em cronogramas restritos. Exemplos importantes de opções que estas ferramentas normalmente oferecem para facilitar projetos digitais:

Captura Esquemática: Permite diagramas esquemáticos serem capturados "*on line*" ao invés de com papel e lápis. Os mais avançados programas de entrada de esquemáticos também realizam verificações com relação à polarização, tornando fácil a depuração de erros tais como saídas curto-circuitadas, sinais que não levam a lugar nenhum, entre outros.

HDLs: Linguagem de Descrição de Hardware, originalmente desenvolvida para modelamento de circuitos, são agora cada vez mais usadas em projetos de hardware. Elas podem ser usadas para projetar qualquer coisa em funções individuais em módulos extensos, como sistemas digitais mutichip. Projetos podem ser descritos em alto nível de abstração. É possível escrever a descrição RTL sem escolher uma tecnologia de fabricação específica. Ferramentas de síntese lógica permitem a conversão do projeto para outras tecnologias de fabricação compatíveis. No caso de novas tecnologias não é necessário projetar o circuito novamente. Simplesmente entra-se com a descrição RTL na ferramenta de síntese e um novo *netlist* em nível de *gates* usando a nova tecnologia é gerado.

Compiladores HDL, Simuladores e Ferramentas de Síntese: Um pacote típico de software HDL contém vários componentes. Em um ambiente típico, o projetista escreve o programa e o compilador HDL e analisa o programa quanto a erros de sintaxe. Se o projeto for compilado corretamente, o projetista tem a opção de, usando a ferramenta de síntese criar, um projeto de circuito dirigido a uma tecnologia de hardware particular. Antes da síntese, o projetista usará os resultados da compilação como entrada em um simulador para verificar se o procedimento do projeto está correto.

Simuladores: O ciclo de projeto para um circuito integrado dedicado é longo e caro. Uma vez o *chip* processado é muito difícil, muitas vezes impossível depurá-lo por acesso interno a suas conexões, mudando as portas e as interconexões. Usualmente, as mudanças

precisam ser feitas na base de dados do projeto inicial e um novo *chip* precisa ser fabricado para incorporar as mudanças necessárias. A fabricação pode durar meses para ser completada e os projetistas são estimulados a acertar na primeira tentativa. Simuladores ajudam os projetistas a prever características elétricas e funcionais do *chip* sem efetivamente construí-lo, permitindo encontrar a maioria, (senão todas) as falhas antes do *chip* ser fabricado.

Simuladores também são usados em projetos de dispositivos lógicos programáveis – PLD (*programmable logic devices*) - Para projetos de sistemas que incorporam muitos componentes individuais. Eles são menos críticos neste caso porque é mais fácil para o projetista fazer mudanças nos componentes e interconexões. Contudo, com uma simples simulação pode-se ganhar muito tempo encontrando erros simples e grosseiros.

Testes de Bancada: Projetistas digitais têm utilizado facilidades de ferramentas HDL para otimizar a simulação e os testes de circuitos em ambientes de software chamados *test benches*. A idéia é construir uma configuração de programa que automatize testes funcionais e dos tempos do sistema. Isto é especialmente usado quando pequenas mudanças no projeto são feitas. O teste de bancada pode ser realizado para assegurar que a correção de erros fixos, ou melhorias em uma área não afetem nada em outras partes do CI. Os programas podem ser escritos em HDL, como o próprio projeto, em C ou C++, ou em combinação de linguagens incluindo linguagem PERL.

Analisadores de tempo e verificadores: A dimensão do tempo é muito importante em projetos digitais. Todo circuito digital leva um tempo para produzir um novo valor na saída em resposta a uma mudança na entrada e muito do esforço dos projetistas é gasto para assegurar que as mudanças na saída ocorram no tempo previsto. Programas especializados podem automatizar a tarefa de desenhar os diagramas de tempo, especificando e verificando a relação de tempo entre diferentes sinais em um sistema complexo.

Processador de Palavras: Estas ferramentas são obviamente usadas para a criação do código fonte para HDL (RTL), mas eles têm um importante uso em todo o projeto para documentação.

Embora as ferramentas de CAD sejam importantes, elas não fazem um projetista digital. Fazendo uma analogia em outro campo, alguém não pode se considerar um grande escritor só porque digita rápido ou usa muito bem um processador de palavras. Aprender a

usar as ferramentas disponíveis não garante a produção de bons resultados. Deve-se prestar atenção no que se está produzindo [9], [16].

VHDL – Objetivo de descrever e documentar

Very Hardware Description Language (VHDL) é um VHSIC que é a abreviação de *Very High Speed Integrated Circuit*. VHDL é uma linguagem para descrever a estrutura e o comportamento de projetos de hardware em eletrônica digital.

Esta linguagem começou a ser desenvolvida em 1981 pelo departamento de defesa dos Estados Unidos da América com o objetivo de reduzir os custos dos projetos devido aos avanços tecnológicos. À medida que as tecnologias ficavam obsoletas, a mudança para novas tecnologias esbarrou no problema de parte dos blocos funcionais dos projetos não estarem documentados adequadamente, fazendo com que os vários componentes que faziam parte do sistema devessem ser verificados individualmente, usando, em larga escala, linguagem de simulação e ferramentas diferentes e incompatíveis.

A necessidade de uma linguagem com grande capacidade de descrição que trabalhasse ao mesmo tempo com qualquer simulador e fosse independente de tecnologias e metodologias de projeto, tornou o VHDL um padrão internacional regulamentado pela IEEE como uma linguagem sem propriedade em 1987 (VHDL 87) e estendido em 1993 (VHDL 93). Apresenta as seguintes características: os projetos podem ser decompostos hierarquicamente; cada elemento do projeto tem uma interface bem definida para conectá-lo a outros elementos e uma precisa especificação comportamental para simulá-lo. Especificação comportamental pode usar um algoritmo ou uma estrutura de hardware para definir uma operação de um elemento. Por exemplo, um elemento pode ser definido inicialmente por um algoritmo para permitir a verificação de elementos em maior nível que o utilizam e mais tarde a definição do algoritmo pode ser trocada por uma estrutura de hardware. Concorrência, tempos e "*clocks*" podem todos ser modelados. A linguagem VHDL é uma ferramenta de descrição que permite tanto a descrição de circuitos síncronos como assíncronos as operações lógicas e o comportamento no tempo de um projeto podem ser simulados. No entanto, o simulador e a ferramenta de síntese tem dificuldade de trabalhar com assíncronos [9], [12].

Fluxo de Projeto em VHDL

Existem vários passos em um projeto baseado em VHDL. Estes passos são aplicáveis em qualquer projeto baseado em HDLs e são esboçados na Figura 11.

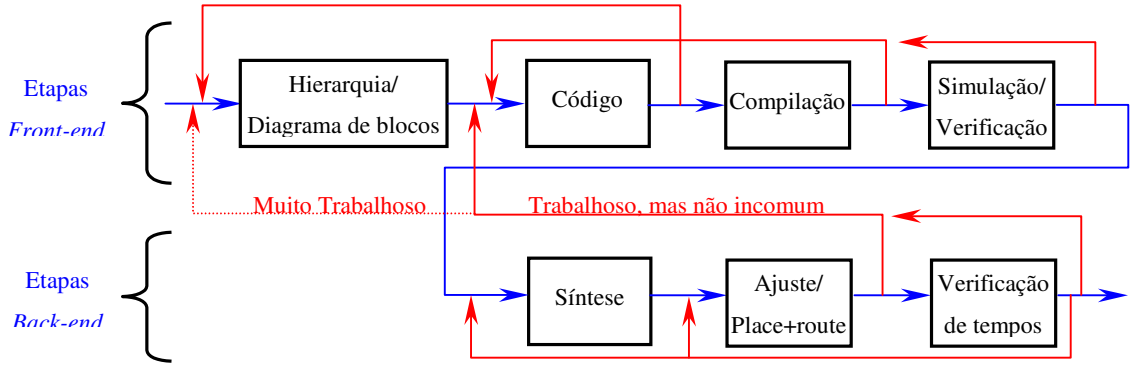


Figura 11: Etapas de um fluxo de projeto baseado em HDLs.

As etapas chamadas "*front end*" iniciam fora da aproximação básica e da construção de blocos em nível de diagramas. O passo seguinte é escrever o código VHDL para os módulos, suas interfaces e detalhes internos. Sendo VHDL uma linguagem baseada em texto, pode ser utilizado qualquer editor de texto. Contudo, os melhores ambientes de projeto incluem editores VHDL tornando o trabalho mais fácil. Estes editores incluem características como destaque de palavras chaves em VHDL, construção de *templates* (estruturas usadas freqüentemente) verificação de sintaxe e acesso ao compilador.

Uma vez escrito o código, ele precisa ser compilado. Um compilador VHDL analisa seu código quanto a erros de sintaxe e compatibilidade com outros módulos que ele se relaciona. Também cria informações internas, que serão utilizadas pelo simulador processar seu projeto. Não se deve esperar para compilar todo o código no final. Compilando em partes, pode-se prevenir a proliferação de erros de sintaxe, nomes inconsistentes e assim por diante.

Um dos passos prazerosos no fluxo é a simulação. Um simulador VHDL permite definir e aplicar entradas no projeto e observar sua saída sem ter que construir um circuito físico. Para projetos pequenos é possível gerar entradas e analisar as saídas manualmente, mas para projetos extensos há a opção de criar *test benches* que automaticamente aplicam sinais nas entradas e comparam com as saídas. Atualmente, simulação é só uma parte de

uma extensa etapa chamada verificação. Em grandes projetos, um esforço substancial é feito durante e depois do estágio de preparação do código para definir os testes que exercitam o circuito sob condições severas de operações lógicas. Encontrar erros de projeto nesta etapa tem um grande valor. Se os erros são encontrados mais tarde, tudo nas etapas chamadas "*back end*" deve ser repetido. Na verificação funcional estuda-se a operação lógica do circuito, independente das considerações de tempo: atrasos nas portas e outras considerações de tempo são consideradas zero. Na verificação dos tempos estuda-se a operação do circuito, incluindo os atrasos estimados, e verificam-se configurações, influências e outros tempos necessários para dispositivos seqüenciais, como flip-flops. A verificação dos tempos nesta etapa é muito limitada, já que os tempos podem ser muito dependentes dos resultados da síntese e ajustes.

Depois da verificação, o próximo passo é a síntese. Converte-se a descrição VHDL em portas primitivas ou componentes. Pode ser gerada uma lista de portas e um *netlist* que especifica como devem ser interconectadas estas portas. Na etapa de ajuste, uma ferramenta de ajuste varre os primitivos ou componentes sintetizados, verificando os recursos disponíveis do dispositivo. Para um PLD ou CPLD, isto pode significar determinar equações para disponibilizar elementos AND-OR. Para um ASIC, isto pode significar colocar o layout de gates individuais em um padrão e encontrar modos de interconectá-los no interior de um chip. Isto é chamado *place-and-route*, ou seja, colocar as células básicas e roteá-las. O passo final é a verificação dos tempos do circuito ajustado. É só nesta etapa que atrasos, medidas elétricas e outros fatores podem ser calculados com razoável precisão. Nesta etapa é usual aplicar testes que foram usados na verificação funcional. Como em qualquer processo criativo pode ocasionalmente ocorrer, de avançarmos, dois passos à diante e termos que voltar para trás. Pode-se encontrar problemas que forcem a voltar e repensar a hierarquia, como erros de compilação e simulação que forcem a reescrever partes do código. Os problemas mais penosos são aqueles encontrados no *back end* do fluxo de projeto. Por exemplo, um projeto sintetizado não ajusta em um FPGA disponível ou não encontra os tempos necessários. É possível que se tenha que voltar e repensar o projeto inteiro. É por isto que excelentes ferramentas não substituem o pensamento cuidadoso no início de um projeto.

Estrutura do programa

VHDL foi projetado com princípios de programação estruturada, usando idéias das linguagens de software de programação do PASCAL e ADA. A idéia chave é definir a interface do módulo de hardware sem mostrar seus detalhes internos. Deste modo, uma *entity* é uma simples declaração dos módulos de entrada e saída, enquanto uma *architecture* é uma descrição detalhada da estrutura dos módulos internos ou comportamento. No arquivo de texto de um programa VHDL a declaração da *entity* e a definição da *architecture* são separadas como mostrado na Figura 12.

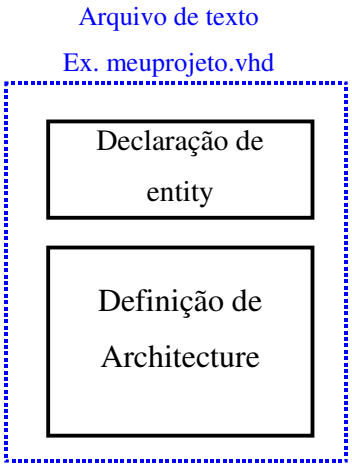


Figura 12: Estrutura de um arquivo de programa VHDL.

Como outras linguagens de programação de alto nível, VHDL geralmente ignora espaços e linhas interrompidas. Comentários iniciam com dois hífens (--) e terminam no final da linha. VHDL define muitos caracteres especiais, chamados palavras reservadas ou *keywords*. Como exemplo temos: entity, port, is, in, out, end, architecture, begin, when, else, and, not. A sintaxe de uma declaração de entity básica é mostrada na Tabela 1.

```
entity entity-name is
      port ( signal-names  :  mode  signal-type;
             signal-names  :  mode  signal-type;
             . . .
             signal-names  :  mode  signal-type;
end entity-name;
```

Tabela 1: Sintaxe de uma declaração de entity VHDL.

Além de nomear a *entity*, o propósito da declaração de *entity* é definir sua interface interna sinais ou *ports* na sua parte de declaração de *port*. Usando as palavras reservadas "*entity, is, port e end*", uma declaração de *entity* tem os seguintes elementos:

- ✓ *entity-name* – selecionado pelo usuário identifica o nome da *entity*;
- ✓ *signal-names* – uma lista de um ou mais identificadores separados por vírgula para nomear sinais de interface externa;
- ✓ *mode* – uma das quatro palavras reservadas especificando a direção do sinal:
  a) *in* – sinal de entrada da *entity*;
  b) *out* – sinal de saída da *entity*, este sinal não pode ser lido pela architecture de sua entity;
  c) *buffer* – sinal de saída de uma *entity* e seu valor pode ser lido dentro da *architecture* de sua *entity*;
  d) inout – este sinal é usado como entrada ou saída de uma entity, este modo é tipicamente usado por pinos "*tri-state*" de entrada/saída em PLDs;
- ✓ *signal-type* – tipo de sinal definido pelo usuário.

Os *ports* das entidades e seus modos e tipos é tudo que é visto por outros módulos que os utilizam. A operação interna de uma *entity* é especificada em sua definição de *architecture,* cuja sintaxe geral é mostrada na Tabela 2.

```
Architecture architecture-name of entity-name is
        type declarations
        signal declarations
        constant declarations
        function definitions
        procedure definitions
        component declarations
begin
        concurrent-statement
        . . .
        concurrent-statement
end architecture-name;
```

Tabela 2: Sintaxe de uma definição de *architecture* VHDL.

O *entity-name* deve ser o mesmo dado previamente na declaração da *entity*. O *architecture-name* é um identificador selecionado pelo usuário, usualmente relacionado com o nome da entity. Os sinais de interface externos são declarados na entity. Uma *architecture* pode também incluir sinais e outras declarações que são apenas locais, similares a outras linguagens de alto nível. As declarações na Tabela 2 podem aparecer em qualquer ordem. As variáveis são similares aos sinais, exceto por não terem significado físico no circuito. De fato, elas não são declaradas na Tabela 2. Variáveis são usadas em

VHDL em "*functions*", "*procedures*" e "*process*", que são recursos usados por linguagens de programação de alto nível [9], [13].

Verilog

Quase ao mesmo tempo em que VHDL estava se desenvolvendo, uma linguagem diferente de projeto de hardware entrou em cena. Verilog HDL, ou simplesmente Verilog foi introduzida em 1984 por *Gateway Design Automation*; a proprietária desta linguagem de descrição de hardware e simulação. Em 1988 foi apresentada pela Synopsys uma ferramenta Verilog baseada na síntese, e em 1989 a aquisição da *Gateway* pela *Cadence Design System* foi uma combinação vencedora, que aumentou a popularidade da linguagem. Hoje, Verilog e VHDL, juntas, usam e compartilham a síntese lógica no mercado em aproximadamente 50/50. Verilog tem sua raiz sintática na linguagem C [9].

Verilog permite que diferentes níveis de abstração sejam misturados num mesmo modelo. Deste modo, o projetista pode definir um modelo de hardware em termos de *switches*, *gates*, RTL ou código comportamental. O projetista precisa aprender somente uma linguagem para estímulos e projetos hierárquicos. As mais populares ferramentas de síntese suportam Verilog, tornando-a uma linguagem muito escolhida pelos projetistas. Todos os fabricantes disponibilizam bibliotecas em Verilog para simulação e síntese pós-lógica. O uso de PLI (Programming Language Interface) é uma poderosa característica que permite ao usuário escrever em um código C comum e interagir com a estrutura de dados internos do Verilog. Projetistas podem personalizar seu simulador Verilog de acordo com suas necessidades do uso de PLI.

Metodologia de Projeto

Há dois tipos básicos de metodologia de projeto: a "*top-down"* e a "*bottom-up"*. Na metodologia de projeto "*top-down"* define-se o bloco principal e depois se identificam os sub-blocos necessários para compor o bloco principal. A divisão dos blocos e sub-blocos é feita até o nível das células básicas. Na metodologia de projeto "*bottom-up"* é realizado o caminho inverso. Identificam-se as células básicas que estão disponíveis, que são organizadas formando um bloco maior. Este procedimento é realizado até chegar a um bloco principal. A Figura 13 mostra as metodologias de projeto "*top-down"* e "*bottom-up"*. Tipicamente, uma combinação das duas metodologias é utilizada. O fluxo encontra um

ponto intermediário, onde uma parte dos projetistas desenvolve uma biblioteca de células básicas e outra parte desenvolve o bloco principal.

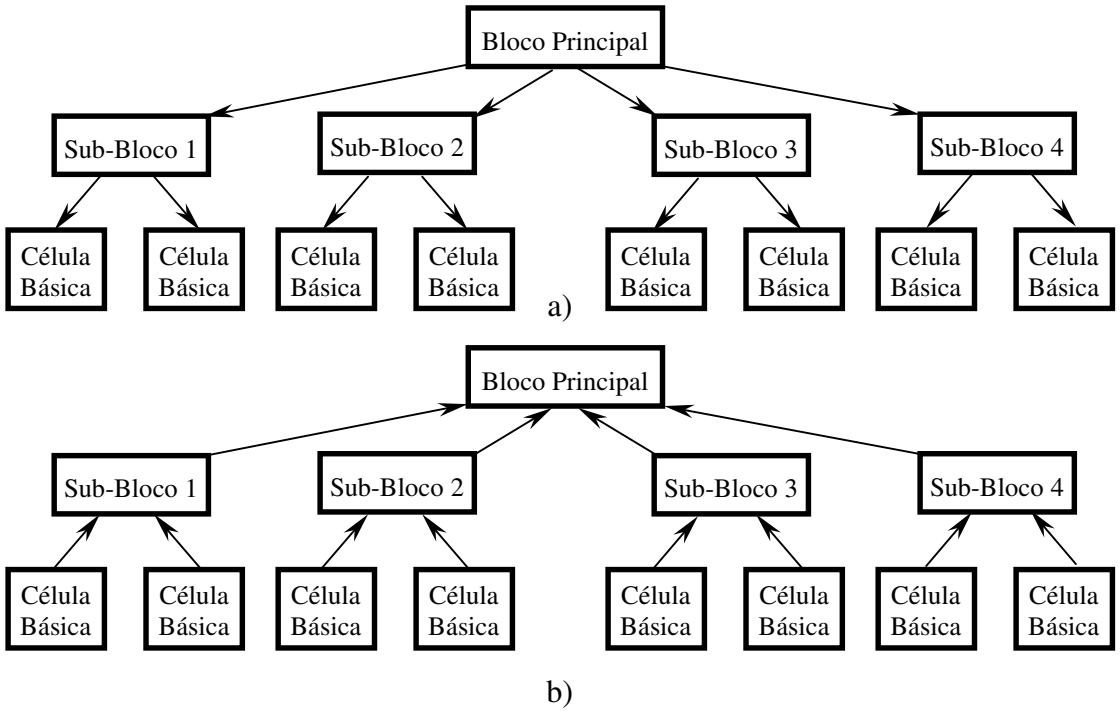


Figura 13: a) Metodologia de projeto top-down. b) Metodologia de projeto bottom-up.

Estrutura do Programa

Em Verilog existe o conceito de "módulo", que é a construção de um bloco básico. Um módulo pode ser apenas um elemento ou uma coleção de elementos em menor nível com uma funcionalidade comum que pode ser usado em vários lugares no projeto. O módulo supre a necessidade funcional de um bloco de maior nível através de seus *ports* de interface (entrada e saída), mas esconde sua implementação interna. Isto permite aos projetistas modificar módulos internos sem afetar o resto do projeto. Em Verilog um módulo é declarado por uma palavra chave *module*. Uma palavra chave correspondente *endmodule* deve aparecer no final da definição do módulo. Cada módulo deve ter um *module_name* que é identificado por um módulo e um *module_terminal_list* que descreve os terminais de entrada e saída do módulo. A Tabela 3 mostra a estrutura do Módulo.

```
Module <module_name> (<module_terminal_list>);

    . . .
    <funcionalidade do módulo>
    . . .

endmodule
```

Tabela 3: Estrutura do módulo em Verilog.

Verilog é uma linguagem estrutural e comportamental onde os módulos podem ser definidos em até quatro níveis de abstração, dependendo das necessidades do projeto. O módulo comporta-se da mesma forma com o ambiente externo, independentemente do nível de abstração que é descrito. Como a descrição interna não aparece para o ambiente externo, a descrição interna pode ser mudada sem qualquer mudança no ambiente. Os níveis de abstração são definidos como:

- ✓ Comportamental ou algorítmico – Este é o mais alto nível de abstração disponibilizado por Verilog. Um módulo pode ser descrito em termos de seu algoritmo sem possuir detalhes de implementação de hardware. Projetar deste modo é muito parecido com programar em C;
- ✓ Fluxo de dados – Neste nível o módulo é projetado especificando-se o fluxo de dados. O projetista informa como os dados fluem entre os registradores de hardware e como os dados são processados no projeto;
- ✓ Portas – Estes módulos são projetados em termos de portas lógicas e interconexões entre suas portas. Projetar neste nível é similar a descrever um projeto em termos de um diagrama lógico de portas;
- ✓ Chaves – Este é o nível mais baixo de abstração em Verilog. Um módulo pode ser implementado em termos de chaves, nós e as interconexões entre eles. Projetar neste nível requer conhecimento detalhado das chaves.

Verilog permite ao projetista misturar e casar todos os quatro níveis de abstração em um projeto. Normalmente, o nível mais alto de abstração é o mais flexível, e o projeto torna-se independente da tecnologia. É possível realizar uma inserção de módulos já criados, denominados instâncias. Quando o módulo é invocado, Verilog cria um "*template*" em que cada objeto tem seu próprio nome, variáveis, parâmetros e interfaces de entrada e saída.

A funcionalidade do projeto pode ser testada pela aplicação de estímulos e verificação de resultados. O bloco que realiza esta tarefa é chamado bloco de estímulos, e pode ser escrito em Verilog. O bloco de estímulos é também comumente chamado de *test bench.* Dois estilos de aplicação de estímulos são possíveis. No primeiro estilo, o bloco de

estímulo insere o bloco do projeto e controla os sinais diretamente no bloco do projeto, como na Figura 14a. O segundo estilo é inserir o bloco de estímulos e de projeto em um módulo principal "*dummy*". Assim o bloco de estímulos interage com o bloco do projeto só através da interface. O módulo de estímulo controla os sinais de entrada do bloco do projeto e verifica o sinal de saída, Figura 14b [11].

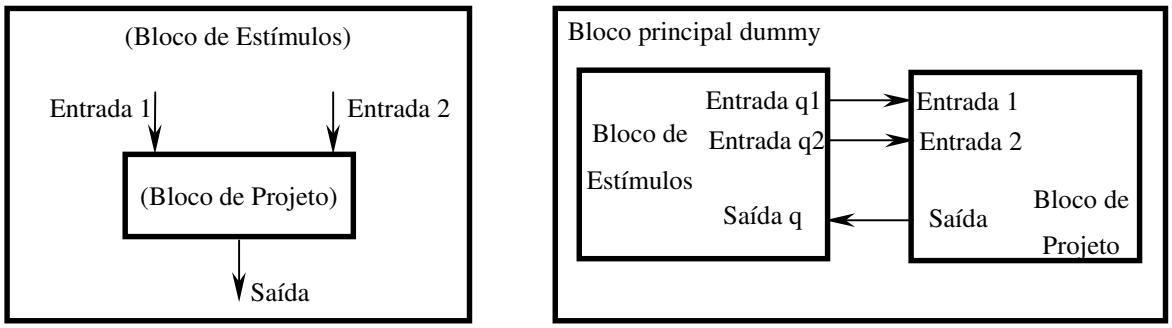


Figura 14: a) Bloco de projeto inserido no bloco de estímulos; b) Blocos de estímulos e projetos inseridos em um módulo *dummy*;

1.4 Síntese Digital

VHDL e Verilog são linguagens de descrição de alto nível para sistemas e projetos de circuitos. Suportam vários níveis de abstração e estilos de descrição, incluindo a descrição estrutural, fluxo de dados e comportamental. Permitem misturar vários níveis de entrada do projeto, sintetizando todos os níveis de abstração e minimizando a lógica necessária, resultando em um *netlist* final em tecnologia a ser escolhida. São altamente simuláveis, mas não totalmente sintetizáveis. Existem muitas construções em ambas linguagens que não têm representação válida em circuitos digitais. O estado da arte dos algoritmos de síntese pode otimizar circuitos de descrição (RTL) objetivando uma tecnologia específica.

Um projeto descrito em VHDL ou Verilog depois de simulado pode ser sintetizado e otimizado, sendo possível ainda misturar descrições HDL com esquemáticos. A hierarquia do projeto e o modo com que este é particionado influenciam no resultado da síntese. O particionamento do projeto pode ser baseado na funcionalidade, mas em alguns casos pode ser feito apenas com propósitos de síntese. Os objetivos de particionamento para síntese são produzir o melhor resultado, melhorar a velocidade, otimizar o tempo e simplificar o processo de síntese. Para tanto, há uma série de medidas que podem ser tomadas no sentido de otimização da síntese: por exemplo, o tamanho dos blocos não deve

exceder 5000 portas, os tempos de execução dos algoritmos de síntese não são lineares, portanto, se um bloco a ser sintetizado é grande demais, sua execução pode demorar um longo tempo.

Registrar todas as entradas e saídas de um bloco. Uma boa prática para simplificar o processo de síntese é separar sub projetos com objetivos diferentes; lógicas com características similares devem ser agrupadas para melhorar a qualidade da síntese. A Figura 15 ilustra o fluxo de um processo de síntese [14].

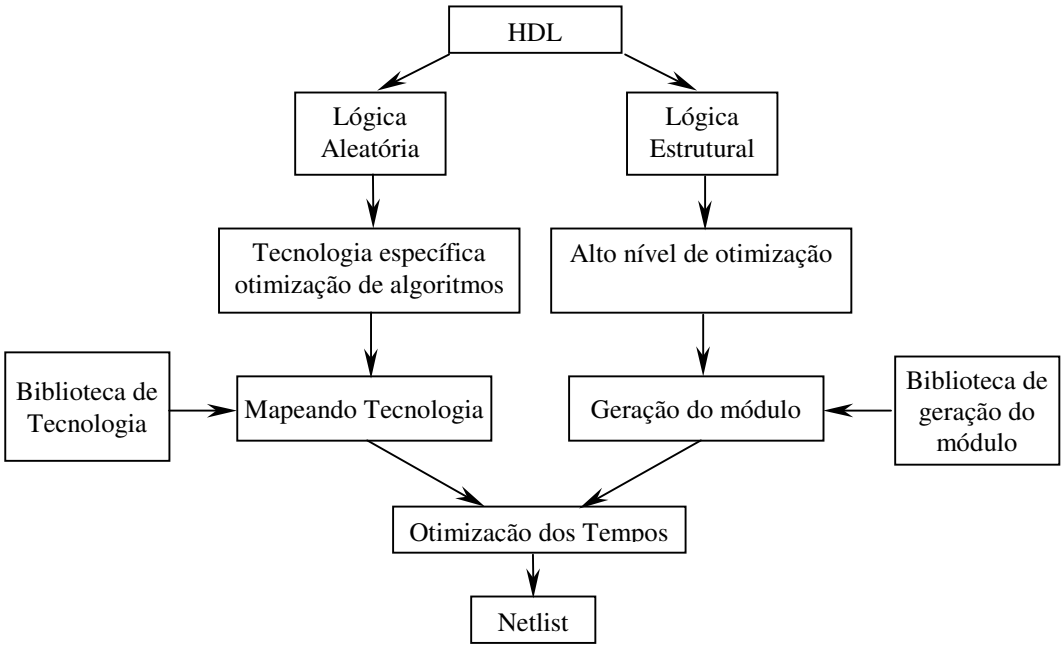


Figura 15: Fluxo de um processo de síntese

1.5 Simulação Lógica

O propósito principal de simulação lógica é o de ajudar na verificação da exatidão de um projeto ASIC. Por definição, simulação digital é processo de exercitar o modelo teórico de um projeto como uma função do tempo, por alguns estímulos na entrada. O modelo teórico consiste na descrição de como os componentes no projeto são conectados, chamado de conectividade, e de modelos de simulação que descrevem a funcionalidade de cada um dos componentes. A seqüência de entrada é referida como estímulos ou vetores. A conectividade é obtida como resultado da captura de esquemáticos ou do processo de síntese. A simulação lógica é usada para verificar se o projeto se encontra dentro das especificações pretendidas. A habilidade do projetista em usar o simulador determina o grau de confiança no projeto, aumentando o nível de qualidade na produção de ASIC.

Há quatro componentes necessários para uma simulação lógica:

- A descrição de como as células em um ASIC são conectadas;
- Os modelos que descrevem a operação de cada célula;
- Estímulos para o teste do projeto e
- Comandos de controle de simulação.

A descrição lógica é normalmente fornecida ao simulador como resultado de uma captura de esquemático. Alguns simuladores necessitam de um preparo dos dados da captura de esquemáticos para o uso do simulador. A descrição lógica é muitas vezes traduzida em um *netlist*. Os simuladores mais avançados já têm a habilidade de ler a conectividade dos dados criados pela captura de esquemáticos.

Os modelos são a chave para a simulação de um ASIC bem sucedida. O grau em que os vários modelos representam a realidade dos componentes físicos é o fator limitante para a confiança nos resultados da simulação. Conceitualmente, os modelos podem ser divididos em funcional e temporal. O componente funcional descreve os valores booleanos que resultam da aplicação de estímulos nas entradas do modelo. A simulação com relação aos tempos descreve o modelo quando os resultados ocorrem no tempo.

Os estímulos são desenvolvidos pelo projetista de modo que aumente a confiabilidade do projeto, cobrindo ao máximo suas especificações. Em geral os estímulos representam os 1's e 0's que são aplicados nas entradas do circuito.

O simulador deve também ter uma série de comandos de simulação que adicionem facilidades como ajustar a resolução dos tempos de simulação, selecionar os sinais para serem mostrados pelo simulador, habilitar o arquivo de log para capturar os resultados do simulador para uso nos testes. Instruções para iniciar uma execução, parar e continuar de um ponto específico são exemplos de comandos de controle que um simulador deve ter [15].

1.6 Layout

Layout é o termo usado para descrever a geometria de várias camadas de materiais que compõe a fabricação de um CI. Anteriormente à fase de layout, o projeto não é submetido ao processamento e antes de começar a fase de layout, é preciso especificar que planta de fabricação o projeto será apontado. A razão é que cada processo de fabricação tem seu próprio conjunto de regras de projeto e layout, e isto é uma coisa que não pode ser

mudada facilmente. Devido ao rigor dos processos de layout, é preciso trabalhar com ferramentas industrialmente aceitas. Existem muitas ferramentas de CAD que são capazes de editar layout sem um preço elevado. Mas o problema com estas ferramentas ocorre quando se precisa associar o projeto a uma tecnologia de fabricação particular. A maioria dos fabricantes oferece kits de projeto personalizados para poucas ferramentas já estabelecidas e caras e outras ferramentas de CAD não podem efetivamente ser usadas para gerar circuitos integrados para a fabrica em questão.

Os fabricantes de CI oferecem diferentes tecnologias com características como tipo, tamanho e custo. Em geral a tecnologia de menor dimensão possui transistores mais rápidos e uma quantidade maior de circuitos podem ajustar-se na mesma área, porém o custo de fabricação também é maior.

As fábricas aceitam submissão dos projetos em formatos de intercâmbio como o *Caltech Intermediate Form* (CIF) e *Graphics Design System* (GDSII). O formato CIF é um formato para descrever a geometria das camadas em ASCII. O mais popular formato de intercâmbio atualmente é o GDSII. Caracteriza-se por ser um formato binário e não podendo ser decifrado como o formato CIF [16]. A Figura 16 ilustra o layout de um circuito integrado digital.

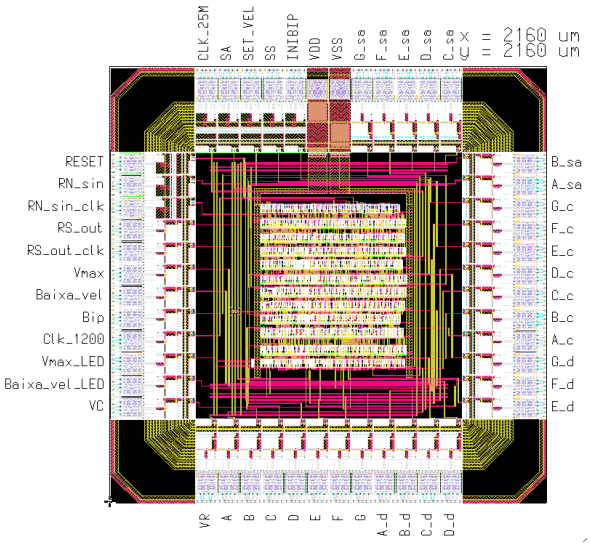


Figura 16: Layout de circuito integrado utilizando uma tecnologia específica.

1.7 Dispositivo de Lógica Programável – DLP

Historicamente, PLAs (Programmable Logic Arrays) foram os primeiros dispositivos lógicos programáveis, com dois níveis de estruturas: as portas AND e OR, e,

conexões programadas pelo usuário. Podem resolver qualquer função lógica a um certo nível de complexidade usando teorias e minimização de síntese.

A estrutura PLA foi aumentada e o custo reduzido com a introdução das matrizes lógicas de programação. Hoje estes dispositivos são genericamente chamados de Dispositivos Lógicos de Programação (PLD) e são MSI (Medium-Scale Integration) na industria de lógica programável.

O constante aumento da capacidade de circuitos integrados gerou oportunidades para a fabricação de circuitos integrados em projetos digitais para muitas aplicações. Contudo, por razões técnicas, a estrutura básica de dois níveis AND-OR de PLDs não pode ser feita muito grande. Ao invés disto, foi desenvolvido PLD Complexo (CPLD) que são múltiplos PLDs em uma estrutura de interconexões programáveis, tudo no mesmo chip. A estrutura de CPLD proporciona uma alta variedade e possibilidades de projetos. Pode ser feita em tamanhos maiores pelo aumento de PLD individuais e o grande número de estruturas individuais de interconexões.

Quase ao mesmo tempo em que os CPLDs estavam sendo inventados, outro circuito integrado recebeu um enfoque diferente para o tamanho da escala dos chips lógicos programáveis. Comparado a um CPLD, uma matriz de portas programadas por campo (FPGA – field-programmable gate array) contem um número muito maior de blocos lógicos individuais (porém menores) e as estruturas de interconexão dominam totalmente o chip. A Figura 17 ilustra a distribuição dos blocos lógicos em CPLD e FPGA.

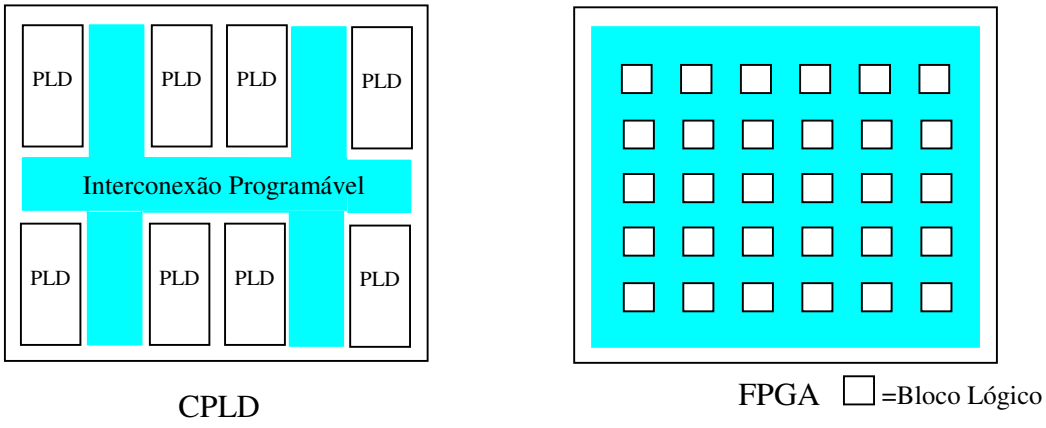


Figura 17: Distribuição dos blocos lógicos em CPLD e FPGA.

Mais importante que a arquitetura dos chips é que ambos suportam um estilo de projeto em que os produtos podem ser mudados de protótipo para produto em tempos reduzidos.

É importante também alcançar curtos *"time-to-market"* ("tempos para chegar ao mercado") para todos os tipos de PLD que usam HDLs em seus projetos. Linguagens como ABEL e VHDL e suas correspondentes ferramentas de software permitem o projeto ser compilado, sintetizado e gravado em PLD, CPLD ou FPGA, literalmente em minutos. A força da linguagem estruturada como VHDL é especialmente importante para ajudar projetistas a utilizar os milhões de portas dos CPLDs e FPGAs [9].

1.8 Full custom

Um projeto de circuito integrado *full custom* é um ASIC projetado sem o uso de silício pré-compilado ou processado. O circuito integrado *custom* é manufaturado ao nível de transistor pelo projetista, para otimizar cada célula quanto à área. Circuitos Integrados *custom* podem ter uma maior densidade de transistores por pastilha que outros tipos de ASIC, porque cada célula é manualmente colocada. Isto faz com que o desenvolvimento de CI *custom* tenha um maior tempo de projeto, sendo que todos os passos de máscaras devem ser completados. O número de passos de máscaras de processos CMOS é normalmente treze, incluindo metalização com uma, duas ou três níveis de interconexões de metais. O maior benefício de um projeto de um CI full custom sobre outros tipos de ASIC são que sua densidade é maior e não há espaço perdido. O custo de produção pode ser muito menor que o de outros tipos de ASICs.

A densidade tende a ser mais alta por causa da manipulação individual dos transistores e das técnicas de compactação de células usadas no desenvolvimento de processos. Todas as células baseadas em transistores são construídas para produzir a máxima densidade possível. Devido a este processo manual, o custo NRE é mais alto para projetos de CI full custom. Porém o custo por unidade pode ser o menor, especialmente se grandes quantidades forem necessárias (usualmente maior que 100.000 unidades).

Se o tempo para o mercado é importante, temos que lembrar que um CI full custom precisará consideravelmente de mais tempo para ser completado que um ASIC semicustom comum [15].

1.9 Verificação Física

Um elemento importante na verificação física do layout é conhecer onde ela se ajusta no fluxo de projetos de circuitos integrados, e que outras ferramentas serão usadas em conjunto para criar os testes e completar o layout do projeto. Os elementos primários do

fluxo de projetos de circuitos integrados são: a captura de esquemáticos, simulação, geração de vetores de teste, a edição do layout manual e/ou automático e a verificação física do layout. A Figura 18 ilustra o fluxo destes elementos primários.

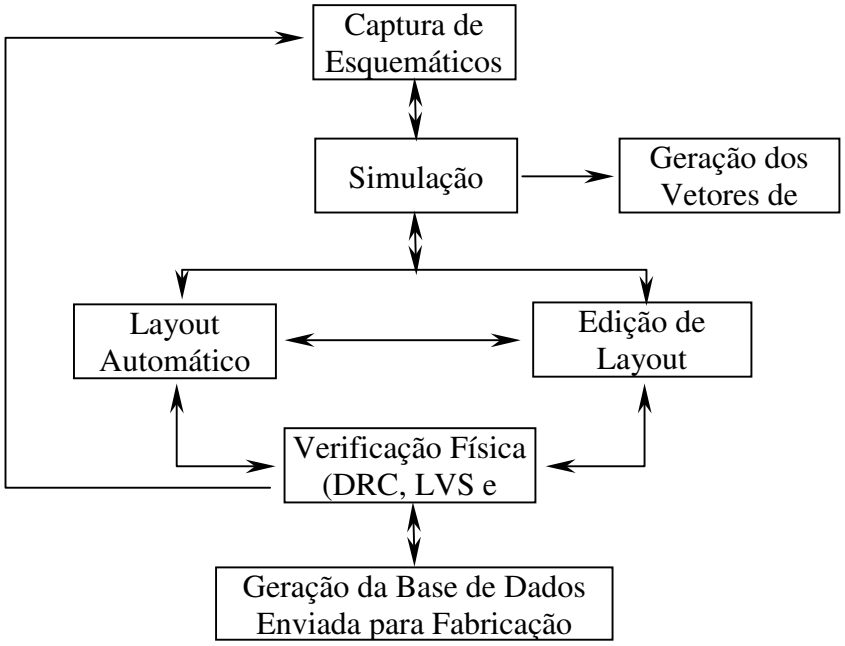


Figura 18: Fluxo de projeto de circuitos integrados dos elementos primários.

A verificação física do layout é dividida em três etapas:

- *Design Rule Checking* (DRC) – Verifica se todos os objetos do layout estão de acordo com as regras de projeto físicas.
- *Layout Versus Schematic* (LVS) – Verifica se o layout concorda com o esquemático original.
- *Parasitic Extraction* (PEX) – Obtém informações do layout com relação às resistências e capacitâncias parasitas. Podem-se também usar estas informações para o *backannotation*, que permite pegar os valores parasitas de interconexões no layout e adicionar nos objetos correspondentes no layout para uma simulação mais precisa. Numa simulação mais precisa é possível otimizar o layout para melhorar a performance do circuito integrado que tenha sido degradada pelos parasitas [17].

1.9.1 DRC – Design Rule Checking

O DRC é inteiramente dependente da tecnologia do processo. As regras de projeto são fornecidas pelo fabricante, através de uma biblioteca com células padrão. Além do arquivo de regras, o DRC necessita de uma entrada de base de dados do layout, que nada

mais é que o próprio projeto a ser verificado. Como resultado de saída do DRC tem-se um arquivo com a base de dados dos erros. Estes erros são derivados de camadas, bordas e polígonos que por não estarem em conformidade com a resolução do processo da tecnologia, podem ser fontes de erros. Esta etapa de verificação deve ser realizada até que todos os erros sejam eliminados [16], [18].

1.9.2 LVS – *Layout Versus Squematic*

O LVS é uma comparação entre o esquemático capturado e o desenho do layout. Todas as "*nets"* de ligação são comparadas e um relatório é gerado em arquivo. Quando o projeto é aprovado nesta verificação significa que o layout está idêntico ao esquemático simulado. Esta etapa de verificação deve ser realizada até que todas as discrepâncias sejam resolvidas e eliminadas.

1.9.3 Extração de parasitas – *backannotation*

Após as verificações de DRC e LVS, um modelo mais preciso de seus sinais de atraso pode ser gerado. Nesta etapa o comprimento das "*nets*" é conhecido e a resistência e capacitância dos fios pode ser calculada. Estes valores são adicionados nas entradas e saídas dos transistores e uma nova simulação pode ser feita usando um modelo que se aproxima mais do circuito real [19], [20].

1.9.4 Simulação considerando os elementos parasitários

A nova simulação utilizando os elementos parasitários considera os atrasos dos transistores e portas lógicas. Estes atrasos ocasionados pelas resistências e capacitâncias geradas pelos diferentes materiais utilizados no processamento, bem como seu comprimento, aproximam o circuito integrado do funcionamento real, tornando esta simulação mais confiável e possibilitando a detecção de falhas. As falhas detectadas podem ser corrigidas antes da base de dados ser enviada para a fabricação, ocasionando em grande economia de tempo e dinheiro. A base de dados é enviada para a fabricação com uma probabilidade muito maior de funcionamento dentro das especificações.

1.10 Exemplo de Fluxo de Projetos Típico

Tudo o que foi descrito nos tópicos anteriores faz parte do fluxo de projetos VLSI seja ele Digital, Analógico, Misto, Automático, Semi ou Manual. A Figura 19 ilustra um exemplo de um processo partindo da idéia até a fabricação.

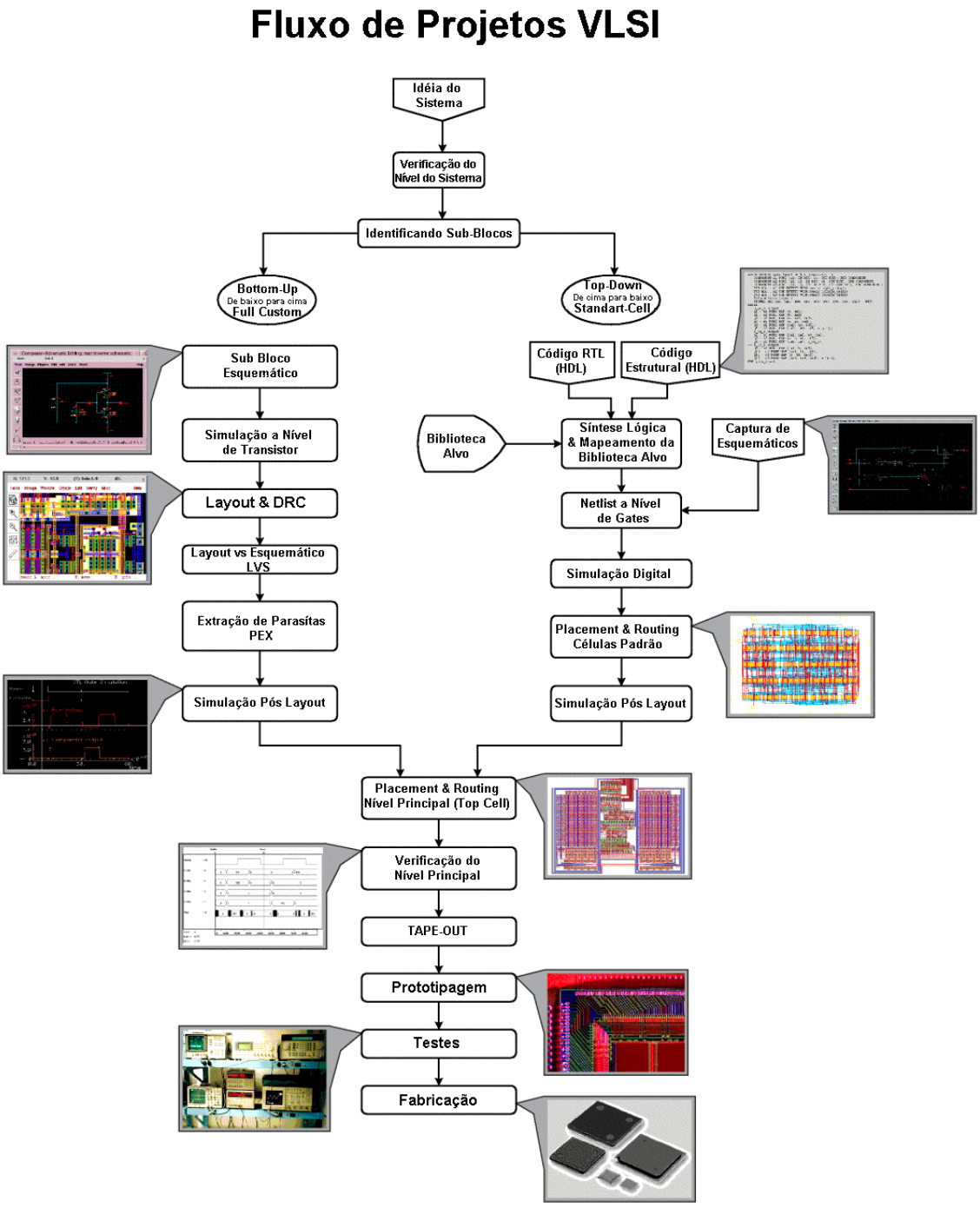


Figura 19: Fluxo de Projetos VLSI

Capítulo 2 – Especificação do CI Monitor de Velocidade - CIMV

2.1 Introdução

Aplicando os conceitos expostos no capítulo anterior, iremos percorrer algumas das etapas do fluxo de projetos digitais, utilizando como veículo o software Mentor Graphics – FPGA Advantage for HDL Design – Release 5.3 [21]. Este software é dividido em 3 aplicativos: HDL Designer Series – versão 2002.1 [22], [23], ModelSim SE versão 5.6a [24], [25], [26] e Leonardo Spectrum – Software Release 2002b [27], [28].

HDL Designer Series é uma família de ferramentas para projetos de sistemas eletrônicos. Todas as ferramentas suportam VHDL e Verilog como linguagem de descrição de hardware – HDL. Os projetos podem ser descritos por diagramas, tabela verdade, máquina de estados e fluxograma. Sendo gerado um arquivo HDL correspondente ao projeto independente da descrição utilizada, o HDL final pode ser compilado e simulado no ModelSim quando utilizado VHDL, e/ou no ModelSim ou Verilog-XL quando utilizado Verilog. O projeto pode ser sintetizado utilizando a ferramenta Leonardo Spectrum.

Para percorrer o fluxo de projetos digitais usando as ferramentas descritas será projetado um Circuito Integrado Monitor de Velocidade – CIMV, cuja especificação será discutida a seguir.

O CIMV analisará a saída de um sensor de velocidade convencional de veículos automotores. O sensor de velocidade do veículo - VSS (Vehicle Speed Sensor), pode ser encontrado em três configurações: sensor magnético ou de relutância variável, sensor de efeito Hall e sensor de efeito óptico.

O sensor VSS gera um sinal que é diretamente proporcional à velocidade do veículo. A unidade de comando eletrônico - UCE utiliza esta informação, principalmente para o controle das condições de marcha lenta e freio motor. Os sensores de efeito Hall vêm instalados nos carros com velocímetro eletrônico e são alimentados com tensão de bateria. Fornecem à UCE um sinal pulsado, cuja amplitude deve ser igual à tensão de alimentação e cuja freqüência é proporcional à velocidade do veículo. Estão comumente instalados no eixo de saída da transmissão, junto ao cabo do velocímetro. O sensor de velocidade tipo Hall é o mais comum no mercado nacional. Normalmente exibe um conector com 3 vias, sendo: 12V, terra e sinal. O sinal de saída representa pulsos proporcionais à rotação do eixo de roda.

Os sensores de efeito óptico possuem comportamento similar aos de efeito Hall. Consistem basicamente de um diodo emissor de luz (LED) e um sensor óptico (fototransistor) separado por um disco giratório com janelas. Toda vez que as janelas permitem que a luz procedente do LED incida no sensor óptico é enviado um sinal (pulso) para a UCE. Estão normalmente instalados junto ao painel de instrumentos e são acionados pelo cabo do velocímetro.

Os sensores magnéticos ou de relutância variável não necessitam de alimentação elétrica. Seu sinal é gerado por indução eletromagnética devido à interação entre o sensor e a roda dentada. Outra particularidade é que ele possui dois fios para o sinal de saída [32], [33], [34], [35], [39]. A Figura 20 representa as partes principais de um sensor de relutância variável e a forma de onda característica na saída.

A Figura 21 representa as formas de onda dos sensores de relutância variável e efeito Hall em diferentes velocidades.

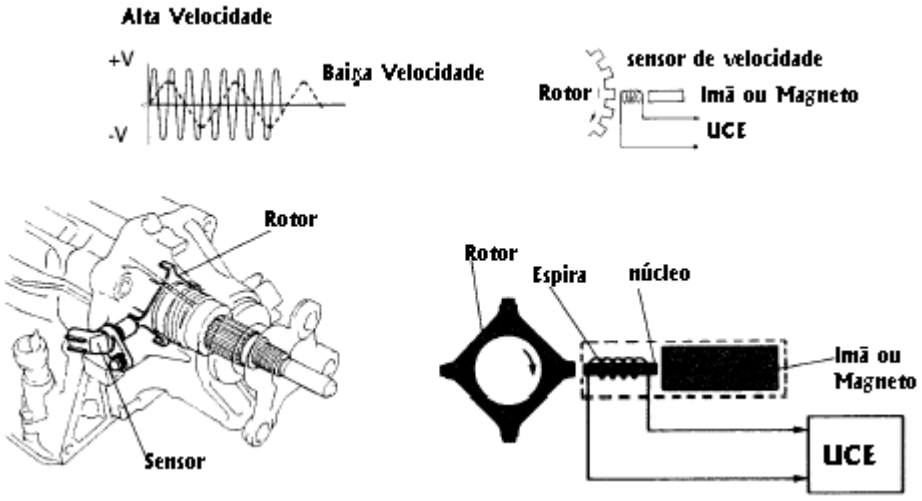


Figura 20: Representa um sensor de relutância variável com suas partes principais.

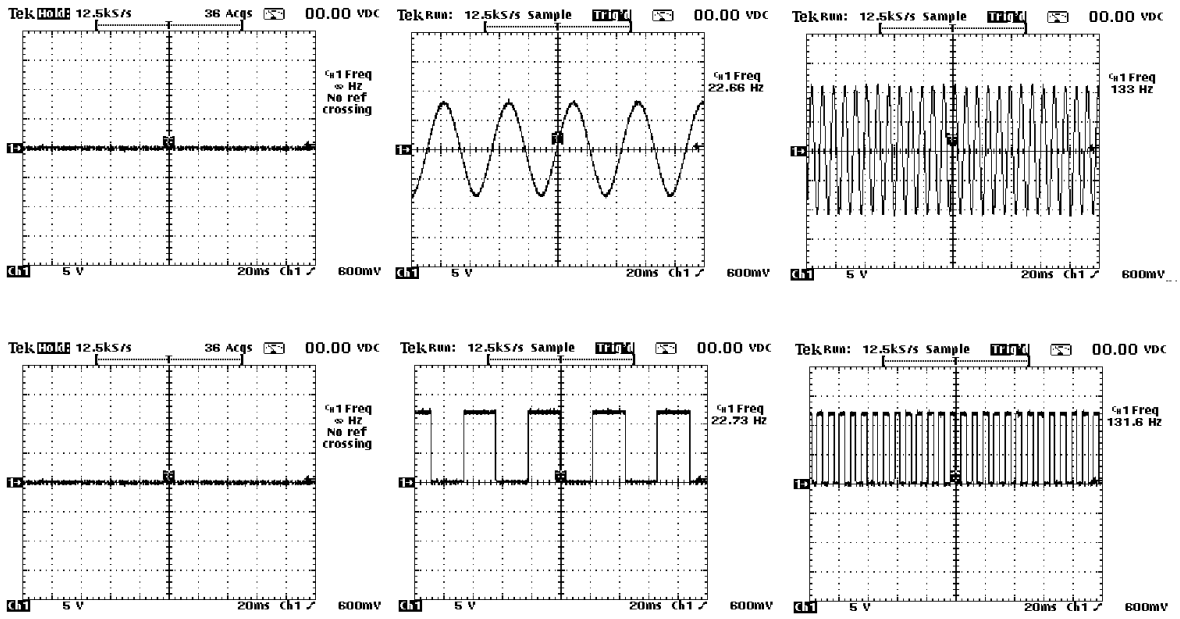


Figura 21: Formas de onda dos sensores de sensores de relutância variável e efeito Hall para as velocidades 0 Km/h, 16 Km/h e 96 Km/h [36], [37], [38].

Os sinais gerados pelos sensores serão analisados pelo CIMV, de forma que este atue convenientemente segundo a especificação.

2.2 Especificação Funcional

O CIMV têm como função básica monitorar a velocidade de veículos automotivos. Para realizar esta função o CI possuí registradores que armazenam valores proporcionais às velocidades definidas pelo usuário. Estes valores são utilizados como parâmetros internos de comparação com a velocidade real do veículo.

A velocidade definida pelo usuário se torna uma referência (Vref) para o Monitor de velocidade sendo o limite para a definição da velocidade máxima (Vmax). Ultrapassado este valor, o usuário é advertido com um sinal luminoso e sonoro. O circuito também é capaz de identificar uma velocidade mínima (Vmin), definida como ¼ da velocidade de referência, evitando, deste modo, que o usuário trafegue com velocidade menor que as mínimas permitidas. Um sinal luminoso será acionado quando este valor for atingido.

Os valores proporcionais à velocidade real devem ser enviados a displays de 7 segmentos. O valor da velocidade de referência é definido chegando à velocidade desejada e apertando um botão (*push button*) de memória.

A velocidade é medida por amostragem, sendo a quantidade de pulsos desta amostragem definida pelo usuário, com três opções de valores selecionados a partir de um botão de seleção.

O CI tem como entradas principais:

- ✓ O sinal de um sensor (SS) que é a velocidade modulada em freqüência;
- ✓ Sinal de memória (SET_VEL) para armazenamento da velocidade de referência;
- ✓ Seleção de amostragem (SA) para definir seu grau de precisão;
- ✓ Um sinal de clock (CLK) de sincronismo para os registradores na freqüência de 25.175 MHz.

As saídas principais são:

- ✓ Indicações de velocidade máxima (Vmax) ultrapassada;
- ✓ Alerta sonoro (BIP) e visual (Vmax_LED);
- ✓ Velocidade inferior a velocidade mínima (Baixa_vel_LED) (alerta visual);
- ✓ Saídas para a indicação na forma de 3 displays de 7 segmentos dos valores proporcionais a velocidade real e um display para mostrar a seleção de amostragem utilizada.

Existem, ainda, entradas e saídas de realimentação e para verificação e testes de sinais internos. A Figura 22 ilustra o símbolo do Circuito Monitor de Velocidade com suas 8 entradas e 40 saídas resultando em 48 pinos no total.

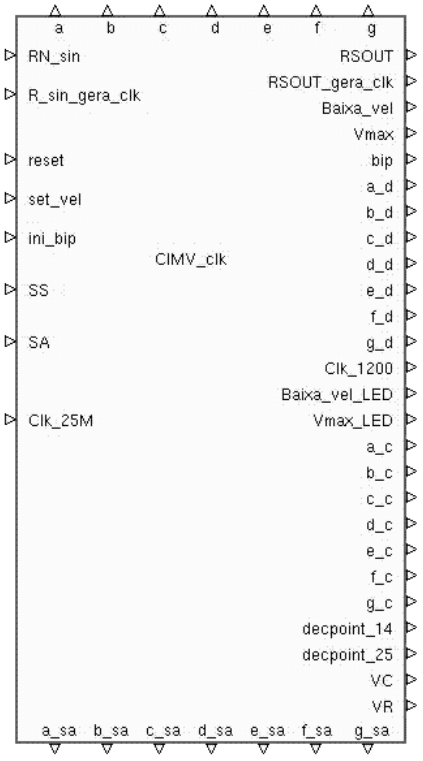


Figura 22: Circuito Monitor de Velocidade: entradas e saídas.

O Circuito é dividido em 5 blocos principais:

- ✓ Bloco de Processamento da Velocidade – PV;
- ✓ Bloco de Memorização da Velocidade de Referência – SM;
- ✓ Bloco do Display – Display;
- ✓ Bloco de geração das freqüências internas – GERA_CLK;
- ✓ Bloco para filtrar os sinais enviados por *push button* de modo a evitar os contatos "Bounce" – FILTRO_PB.

A Figura 23 ilustra os 5 blocos que integram o circuito monitor de velocidade.

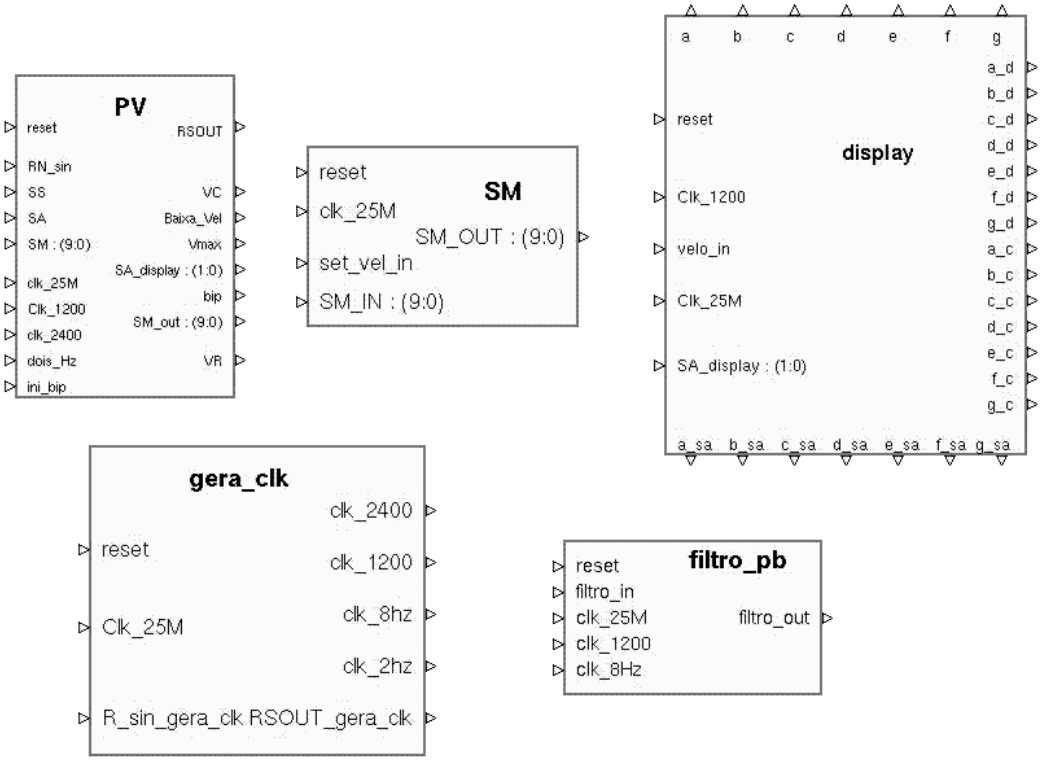


Figura 23: Os 5 Blocos principais que compõe o circuito monitor de velocidade

Bloco de Processamento da Velocidade – PV

Este bloco tem a função de reconhecer um sinal, enviado por um sensor acoplado à transmissão junto ao velocímetro de um veículo, e assim identificar se o mesmo está parado, acima ou abaixo de uma determinada velocidade limite programada e, por meio destas informações, atuar convenientemente, interagindo com os outros blocos para obter o resultado desejado.

Assim sendo, se o veículo estiver parado, ou com uma velocidade menor ou igual a ¼ da velocidade programada, a saída denominada BAIXA_VEL deverá permanecer em

nível lógico 1 (5V) (advertência visual). Caso contrário, a referida saída deve permanecer em nível lógico 0 (0V).

Se, no entanto, o veículo estiver com velocidade igual ou superior à que foi programada, a saída BIP deve apresentar um sinal intermitente composto de uma freqüência de 2400Hz modulada por uma de 2Hz. Esse sinal acionará um dispositivo sonoro e visual cuja função é a de alertar o motorista. Opcionalmente o condutor pode inibir o sinal sonoro deixando apenas o visual (entrada INI_BIP).

Outras características que devem ser mencionadas são: a velocidade de referência é definida pelo condutor chegando à velocidade desejada e acionando um botão, emitindo um pulso para que Vref seja armazenada. Os pulsos correspondentes à velocidade devem ser amostrados, e, os valores de amostragem são definidos pelo condutor acionando um botão. O valor inicial de amostragem SA(0) será de 10 pulsos. Acionando o botão uma vez o condutor muda SA(1) para 15 pulsos, acionando o mesmo botão, novamente, muda SA(2) para 30 pulsos, e retornando a SA(0) ao acionar novamente o botão; A amostragem selecionada será indicada em display de 7 segmentos.

A Tabela 4 os valores de programação para realizar a amostragem.

Amostragem	B A	Número de pulsos amostrados
SA(0)	0 0	10
SA(1)	0 1	15
SA(2)	1 0	30

Tabela 4: Programação dos valores de amostragem.

Toda a lógica para que o circuito desempenhe a funcionalidade descrita está baseada em dois sinais: o sinal VC e o sinal VR. O sinal VC é a velocidade SS amostrada e o sinal VR é a velocidade de referência Vref. Esta inicialmente possui um valor máximo e após um pulso de SET_VEL, VR recebe o número de pulsos de clock de 1200 Hz contidos no sinal VC. A comparação entre estes dois sinais é que determina a velocidade máxima, de cruzeiro, ou mínima verificada no veículo.

Bloco de Memorização da Velocidade de Referência – SM

O Bloco de memorização é um conjunto de *latches* e multiplexadores de 10 bits. Os *latches* têm a função de armazenar o valor recebido após um pulso de SET_VEL. Os multiplexadores iniciam a saída do bloco SM com um valor máximo e, após o pulso de

SET_VEL, são realimentados de modo que o valor armazenado fique "memorizado" até um próximo pulso de SET_VEL. A entrada do Bloco SM recebe o valor do número de pulsos de *clock* de 1200 Hz que estão contidos no sinal VC, valor este obtido na saída de um contador de 10 bits, que é reiniciado sincronamente pelo sinal VC. A saída do Bloco SM alimenta a entrada do Bloco PV com o valor da velocidade de referência escolhida pelo usuário através do botão SET_VEL.

Bloco de Display – Display

O Bloco Display é o responsável pela interface entre o Bloco PV e o usuário. O Bloco Display tem saídas para quatro displays de 7 segmentos. Um deles mostra o valor da seleção de amostragem e os outros três mostram um valor proporcional à velocidade na ordem de unidade, dezena e centena. Devido ao valor da velocidade real estar relacionado a uma série de fatores como diâmetro do pneu, número de pulsos do sensor e a variedade de tipos de sensor, decidiu-se exprimir o valor da velocidade em função do número de pulsos de clock de 1200 Hz contidos no sinal de VC. Para tanto, o sinal VC reinicializa sincronamente três contadores de década de 0 à 9 de 4 bits em código BCD – Binary-Coded Decimal interligados, de forma a contar de 000 à 999. O resultado desta contagem é enviado para um decodificador BCD - Display 7 segmentos sendo mostrado para o meio externo um valor em quantidades de pulsos de clock. Para a codificação do display foi utilizado o padrão segundo a Figura 24.

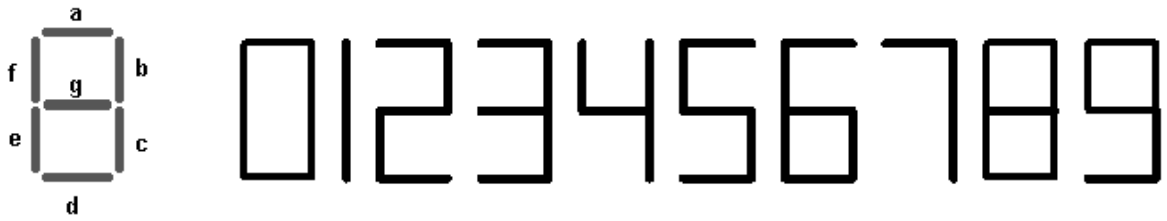


Figura 24: Display de 7 segmentos com a identificação dos segmentos e os dígitos decimais.

Bloco de geração das freqüências internas – GERA_CLK

A freqüência dos cristais utilizados como osciladores é muito alta se comparada com a ordem de grandeza da freqüência da velocidade. Por exemplo, a freqüência de cristal escolhida para ser utilizada no circuito é de 25,175 MHz e a freqüência de saída dos sensores de velocidades gira em torno de dezenas ou centenas de Hertz. Se fosse utilizada a

freqüência em Mega Hertz os contadores utilizados no circuito precisariam de um número muito elevado de bits, aumentando muito o número de portas lógicas e consequentemente o tamanho do circuito, além de dificultar a análise.

Da especificação do circuito têm-se a freqüência do BIP de 2400 Hz modulada por uma freqüência de 2 Hz. Para se chegar a estas freqüências partindo dos 25,175 MHz foi utilizado um contador de pulsos de 14 bits reinicializado sincronamente quando a contagem atingisse o valor pré determinado de 5245 em decimal ou 1010001111101 em binário. Este valor resulta em um sinal com a freqüência de 4800 Hz, que por sua vez entra em um divisor de freqüência por dois de 10 bits, resultando nas freqüências múltiplas de 2400 Hz, 1200Hz, 8 Hz e 2Hz. A freqüência de 1200 Hz é utilizada para a contagem do valor da velocidade amostrada, VC, e a freqüência de 8 Hz é utilizada no bloco de filtro de contatos "Bounce". Para evitar problemas de síntese com relação a freqüências de *clock* assíncronas, decidimos a utilizar apenas FLIP-FLOPS com *enable*, utilizando como *enable* a freqüência de 1200 Hz ou a velocidade SS, dependendo da necessidade de utilização do circuito.

Bloco de filtro de contatos "*Bounce*" ou circuito "*de-bounce*" – FILTRO_PB

Este bloco foi projetado após a síntese e o teste elétrico funcional inicial do CIMV. Utilizando botões para acionar a entrada SET_VEL e a entrada SA verificou-se, principalmente por esta última, que tem uma saída direta para o display, que as oscilações causadas pelo acionamento do botão faziam com que a amostragem da velocidade fosse selecionada de maneira indesejada e aleatória. Analisando esta situação constatou-se que o circuito era sensível ao efeito chamado de contato "*bounce*".

Contato "*bounce*" ocorre em contatos de botões, chaves, chaves comutadoras e reles Eletro-mecânico. Estes contatos são projetados para abrir e fechar rapidamente, com pouca resistência em seu movimento. Devido aos contatos terem massa e elasticidade, com pouca resistência de movimento, ao serem acionados irão trepidar, fazendo com que o sinal oscile até finalmente atingir a posição de repouso.

Se o desejado é utilizar botões ou chaves para acender lâmpadas ou iniciar o motor de um pequeno ventilador o contato "*bounce*" não é um problema. Mas se as chaves ou botões serão usadas como entrada para um contador digital ou um microprocessador, o contato "*bounce*" deve ser considerado. A razão disto é que o tempo medido para um

contato "*bounce*" estabilizar está na ordem de milisegundos e sinais digitais respondem em microsegundos ou menos em muitas vezes.

Existem varias maneiras de resolver o problema do contato "*bounce*", usando os chamados circuitos de "*de-bounce*". Um circuito de "*de-bounce*" pode fazer uso de constante de tempo e usando um buffer, ou ainda, usar flip-flops para filtrar o sinal [40], [41]. A Figura 25 ilustra o sinal de proveniente de uma chave com o efeito de contato "*bounce*".

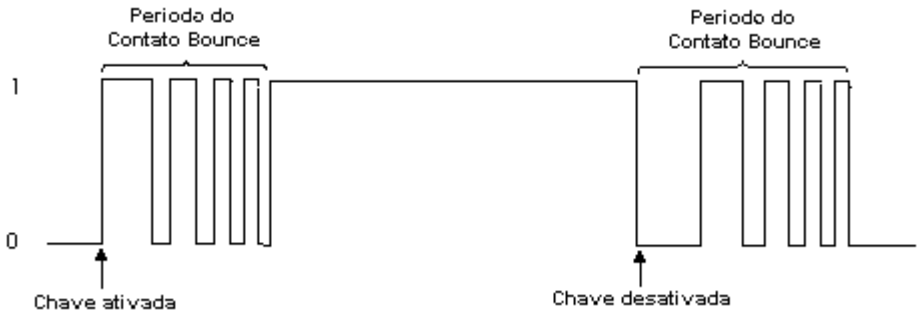


Figura 25: Sinal proveniente de uma chave exemplificando o efeito de contato "*bounce*".

Com esta ultima descrição é finalizada a descrição dos blocos que formam o CIMV. As Tabelas 5 e 6 ilustram os 48 pinos do CIMV com suas entradas e saídas. Estão contidos nesta tabela os pinos essenciais segundo a especificação e ainda alguns pinos de teste para verificar se os sinais internos do circuito estão de acordo com o simulado.

IN/OUT		NOME	DESCRIÇÃO
1	In	INIBIP	Inibe a saída Bip
2	In	SS	Sinal do Sensor
3	In	SET_VEL	Sinal de Memória para armazenar a velocidade de referência
4	In	SA	Seleção de amostragem
5	In	CLK_25M	Sinal de clock de sincronismo em 25.175 MHz
6	In	RESET	Inicializa o circuito assincronamente
7	In	RN_sin	Realimentação externa, entrada do reinicializador síncrono
8	In	R_sin_gera_clk	Realimentação externa, entrada do reinicializador síncrono
9	Out	RSOUT	Realimentação externa, saída do reinicializador síncrono
10	Out	RSOUT_gera_clk	Realimentação externa, saída do reinicializador síncrono
11	Out	Vmax	Sinal de indicação de Velocidade Máxima – teste
12	Out	Baixa_vel	Sinal de indicação de Velocidade Minima – teste

Tabela 5: Pinos de entrada e saída do circuito CIMV com a descrição resumida de sua função – Continua Tabela 6.

13	Out	Bip	Sinal de indicação sonora de Velocidade Máxima atingida
14	Out	Clk_1200	Sinal referência para contagem da Velocidade – 1200 Hz – teste
15	Out	Vmax_LED	Sinal de indicação luminosa de Velocidade Máxima
16	Out	Baixa_vel_LED	Sinal de indicação luminosa de Velocidade Minima
17	Out	VC	Sinal da velocidade SS amostrada – teste
18	Out	VR	Sinal de referencia da velocidade limite – teste
19	Out	A	Segmento ilustrado na Figura 3.2.3 referente ao valor da unidade
20	Out	B	Segmento ilustrado na Figura 3.2.3 referente ao valor da unidade
21	Out	C	Segmento ilustrado na Figura 3.2.3 referente ao valor da unidade
22	Out	D	Segmento ilustrado na Figura 3.2.3 referente ao valor da unidade
23	Out	E	Segmento ilustrado na Figura 3.2.3 referente ao valor da unidade
24	Out	F	Segmento ilustrado na Figura 3.2.3 referente ao valor da unidade
25	Out	G	Segmento ilustrado na Figura 3.2.3 referente ao valor da unidade
26	Out	A_d	Segmento ilustrado na Figura 3.2.3 referente ao valor da dezena
27	Out	B_d	Segmento ilustrado na Figura 3.2.3 referente ao valor da dezena
28	Out	C_d	Segmento ilustrado na Figura 3.2.3 referente ao valor da dezena
29	Out	D_d	Segmento ilustrado na Figura 3.2.3 referente ao valor da dezena
30	Out	E_d	Segmento ilustrado na Figura 3.2.3 referente ao valor da dezena
31	Out	F_d	Segmento ilustrado na Figura 3.2.3 referente ao valor da dezena
32	Out	G_d	Segmento ilustrado na Figura 3.2.3 referente ao valor da dezena
33	Out	A_c	Segmento ilustrado na Figura 3.2.3 referente ao valor da centena
34	Out	B_c	Segmento ilustrado na Figura 3.2.3 referente ao valor da centena
35	Out	C_c	Segmento ilustrado na Figura 3.2.3 referente ao valor da centena
36	Out	D_c	Segmento ilustrado na Figura 3.2.3 referente ao valor da centena
37	Out	E_c	Segmento ilustrado na Figura 3.2.3 referente ao valor da centena
38	Out	F_c	Segmento ilustrado na Figura 3.2.3 referente ao valor da centena
39	Out	G_c	Segmento ilustrado na Figura 3.2.3 referente ao valor da centena
40	Out	A_sa	Segmento ilustrado na Figura 3.2.3 referente à seleção de amostragem
41	Out	B_sa	Segmento ilustrado na Figura 3.2.3 referente à seleção de amostragem
42	Out	C_sa	Segmento ilustrado na Figura 3.2.3 referente à seleção de amostragem
43	Out	D_sa	Segmento ilustrado na Figura 3.2.3 referente à seleção de amostragem
44	Out	E_sa	Segmento ilustrado na Figura 3.2.3 referente à seleção de amostragem
45	Out	F_sa	Segmento ilustrado na Figura 3.2.3 referente à seleção de amostragem
46	Out	G_sa	Segmento ilustrado na Figura 3.2.3 referente à seleção de amostragem
47	Out	Decpoint_14	Segmento do ponto decimal, colocado em estado ‘1’ devido a não utilização
48	Out	Decpoint_25	Segmento do ponto decimal, colocado em estado ‘1’ devido a não utilização

Tabela 6: Pinos de entrada e saída do circuito CIMV com a descrição resumida de sua função.

2.3 Especificação estrutural (diagramas de blocos e suas interligações) e Simulação dos Blocos

Após a especificação dos blocos principais que compõe o CIMV, apresenta-se, neste tópico, como eles estão interligados, e também uma descrição dos sub-blocos. Os blocos serão descritos do topo para o nível mais baixo na hierarquia.

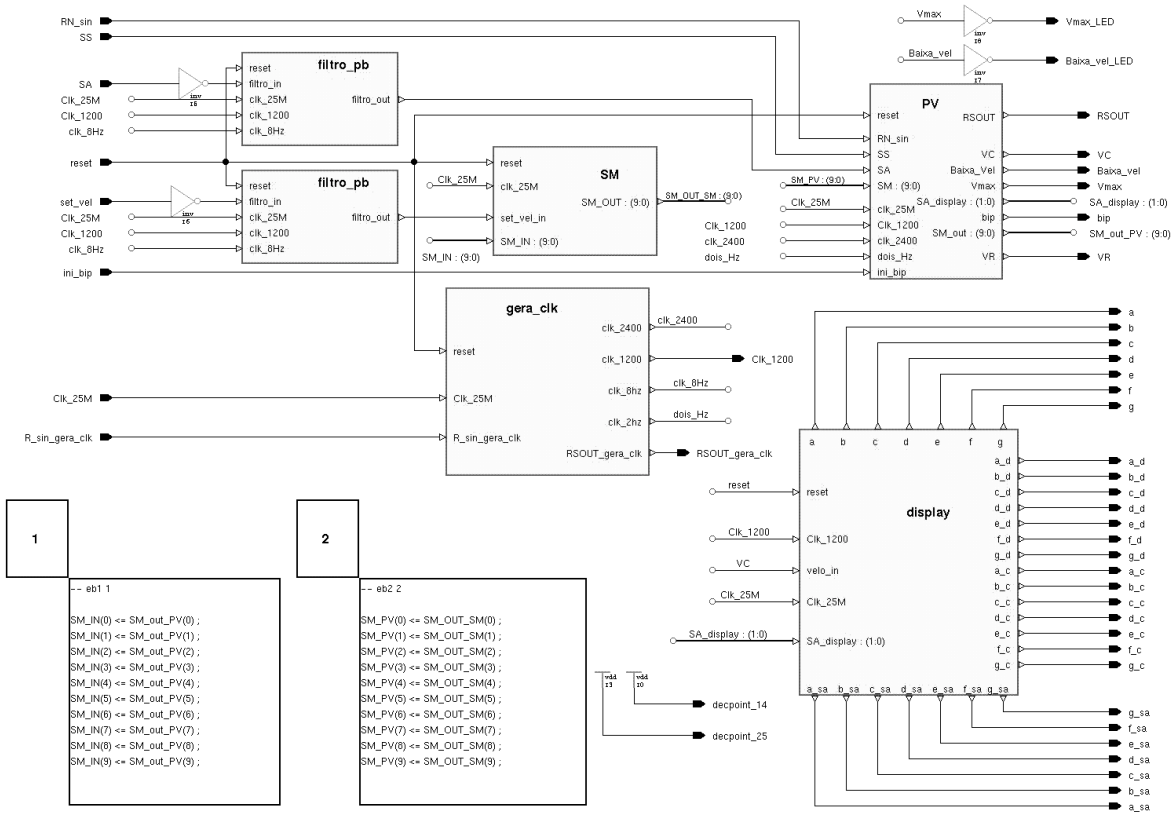


Figura 26: Circuito Monitor de Velocidade com seus cinco blocos principais interligados.

A entrada CLK_25M recebe o sinal do *clock* de 25,175 MHz vindo de um cristal oscilador. Este sinal é o *clock* dos flip-flops de quatro blocos, apenas alguns flip-flops do bloco Display não possuem este sinal em seus *clocks*. Desta forma pode-se dizer que o circuito CIMV é quase que totalmente síncrono. Este sinal também entra no bloco GERA_CLK, gerando 4 sinais de clock de freqüência inferior. Os Sinais gerados CLK_2400 e DOIS_HZ são usados no bloco PV para ativação do sinal sonoro BIP. O sinal CLK_1200 é o sinal de referência utilizado em todos os blocos, controlando a entrada *enable* dos flip-flops utilizados. Ele é o valor unitário utilizado para quantificar o valor do sinal amostrado VC e transformá-lo no sinal de referência VR. O sinal CLK_8Hz é o sinal de *enable* que habilita os FLIP-FLOPs do bloco FILTRO_PB.

Os sinais SET_VEL e SA são gerados por botões, e verificou-se que a lógica utilizada para processar estes sinais é sensível a contato "*bounce*". Isto explica a passagem destes sinais pelo bloco FILTRO_PB. Saindo do bloco FILTRO_PB, o sinal SET_VEL entra no bloco SM para desempenhar a sua função de armazenar a velocidade de referência. O sinal SA ao sair do FILTRO_PB vai par o bloco PV, onde selecionará os valores de amostragem da velocidade SS escolhidos pelo usuário.

O sinal de RESET reinicia assincronamente os flip-flops de todo o circuito, colocando-os em estado lógico baixo. Este sinal é comum a todos os blocos.

O sinal INI_BIP inibe o sinal sonoro indicador de velocidade máxima atingida. Este sinal entra diretamente no bloco PV.

O sinal SS é a velocidade real do veículo vinda do sensor de velocidade. Este sinal entra diretamente no bloco PV, onde é tratado convenientemente.

Os sinais de saída de interesse ao usuário são Vmax_LED, Baixa_vel_LED e BIP. Todos os outros sinais utilizados como saída ou são para controlar os 4 Displays de 7 segmentos ou para verificar sinais internos confirmando a especificação nos testes elétricos.

Os inversores utilizados externamente aos blocos foram colocados para adequar os sinais de entrada dos testes elétricos à lógica utilizada na placa de testes que será usada para o teste funcional.

A seguir todos os blocos serão apresentados em mais detalhes, descrevendo seus sub-blocos até o nível mais baixo de descrição: portas lógicas, tabela verdade ou máquina de estados. Será utilizada a seguinte seqüência de descrição: Bloco PV, Bloco SM, Bloco Display, Bloco Gera_clk e Bloco Filtro_pb.

2.3.1 Bloco PV

O bloco PV é formado por 3 outros blocos: o Sel_amostragem_fsm, o Monitor_SS e o Bloco4. A seguir será descrito cada bloco até que se chegue no nível mais baixo de cada um deles, passando a descrever o bloco seguinte até que todos sejam descritos. Neste ponto será mostrado o Bloco PV.

O Bloco Sel_amostragem_fsm recebe o sinal SA que saí do Bloco Filtro_Pb, conta quantas vezes o sinal SA foi acionado e decodifica esta contagem nos valores de amostragem especificados. A Figura 27 ilustra o Bloco Sel_amostragem_fsm.

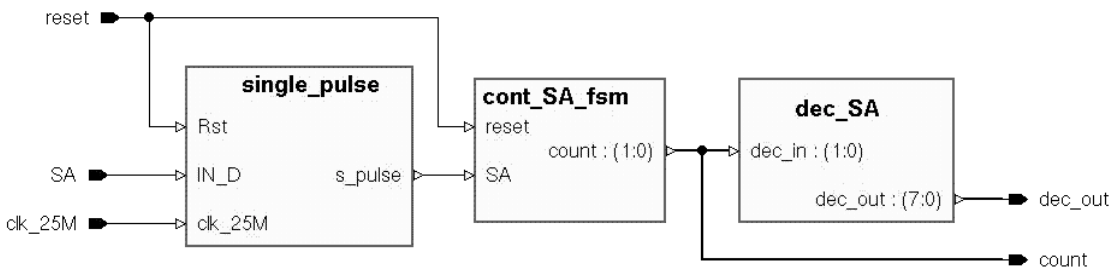
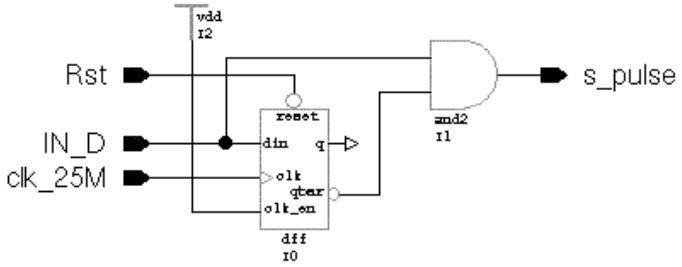
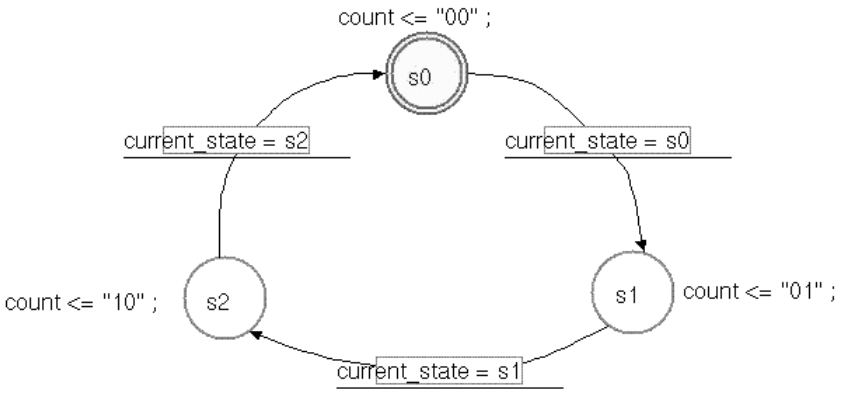


Figura 27: Bloco Sel_amostragem_fsm

O bloco Sel_amostragem_fsm é composto por três outros blocos: o single_pulse, o cont_SA_fsm e o dec_SA. O bloco single_pulse recebe o sinal SA e o transforma em um único pulso com o período de um pulso de *clock* síncrono ao *clock* de 25.175 MHz. O bloco cont_SA_fsm é um contador de 2 bits que conta de 0 a 10 em binário ou de 0 à 2 em decimal. Ele é descrito na forma de máquinas de estados, que muda de estado a cada evento do sinal SA. O Bloco dec_SA é o decodificador que recebe o valor da contagem do bloco cont_SA_fsm e decodifica para o valor especificado. Este bloco é descrito na forma de tabela verdade. A Figura 28 ilustra os três blocos descritos em seus níveis mais básicos. E a Figura 29 ilustra a simulação do bloco single_pulse e Sel_amostragem_fsm.



a)

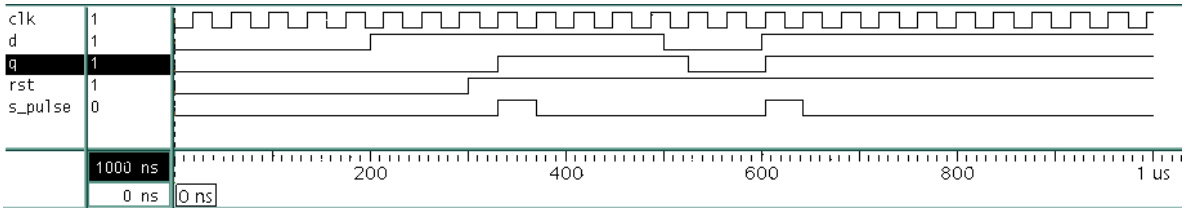


b)

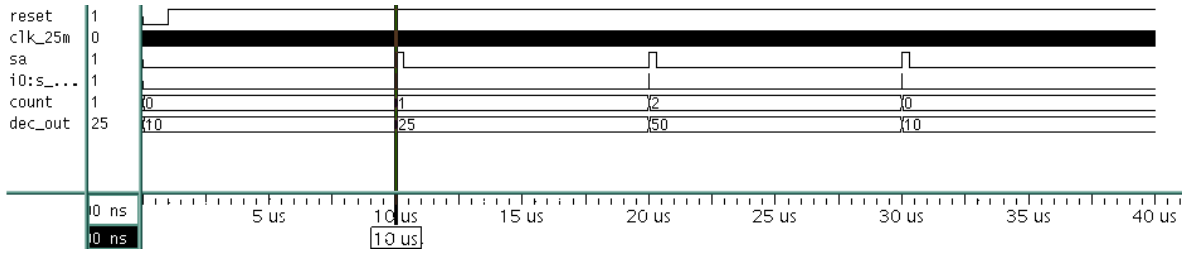
	A	B	C	D	E	F	G	H	I	J
1	dec_in(1)	dec_in(0)	dec_out(7)	dec_out(6)	dec_out(5)	dec_out(4)	dec_out(3)	dec_out(2)	dec_out(1)	dec_out(0)
2	'0'	'0'	'0'	'0'	'0'	'0'	'1'	'0'	'1'	'0'
3	'0'	'1'	'0'	'0'	'0'	'0'	'1'	'1'	'1'	'0'
4	'1'	'0'	'0'	'0'	'0'	'1'	'1'	'1'	'0'	'1'

c)

Figura 28: a) Diagrama esquemático do bloco single_pulse. b) Descrição do bloco cont_sa_fsm na forma de Máquina de Estados. c) Descrição do bloco decodificador da seleção de amostragem (dec_SA) na forma de tabela verdade.



a)



b)

Figura 29: a) Simulação do bloco single_pulse. b) Simulação do bloco Sel_amostragem_fsm.

O Bloco Monitor_SS recebe o sinal da velocidade SS diretamente do sensor do veículo. Este bloco recebe o valor da amostragem e da velocidade de referência selecionados pelo usuário, valores estes que vem a compor a geração dos sinais VC e VR que comparados segundo a especificação geram os sinais Vmax e Baixa_vel_freq. A Figura 30 ilustra o bloco Monitor_SS.

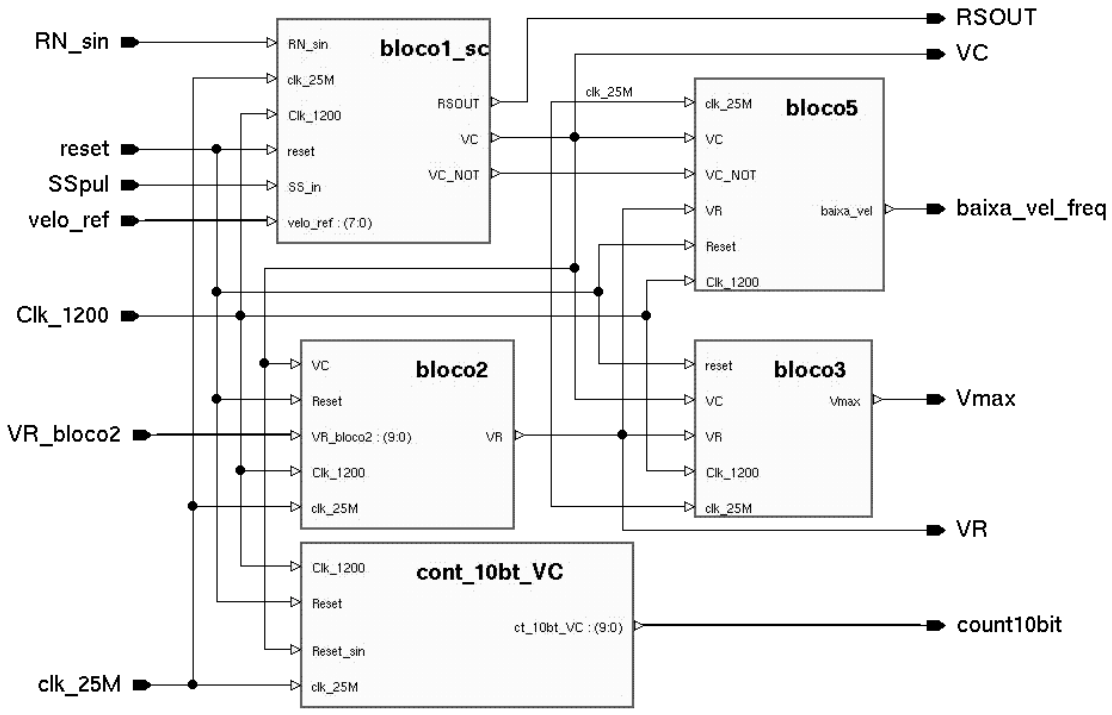
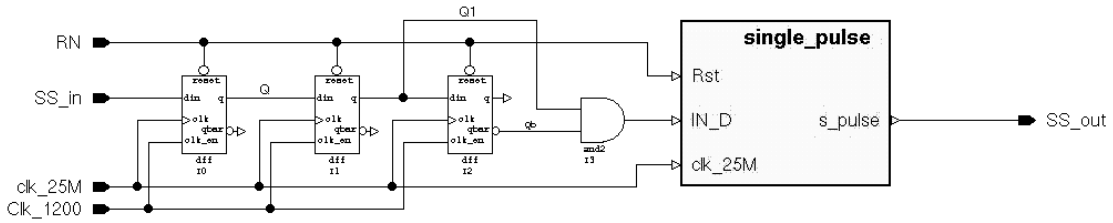
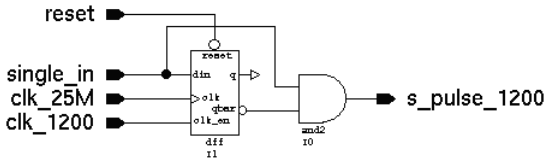


Figura 30: Bloco Monitor_SS.

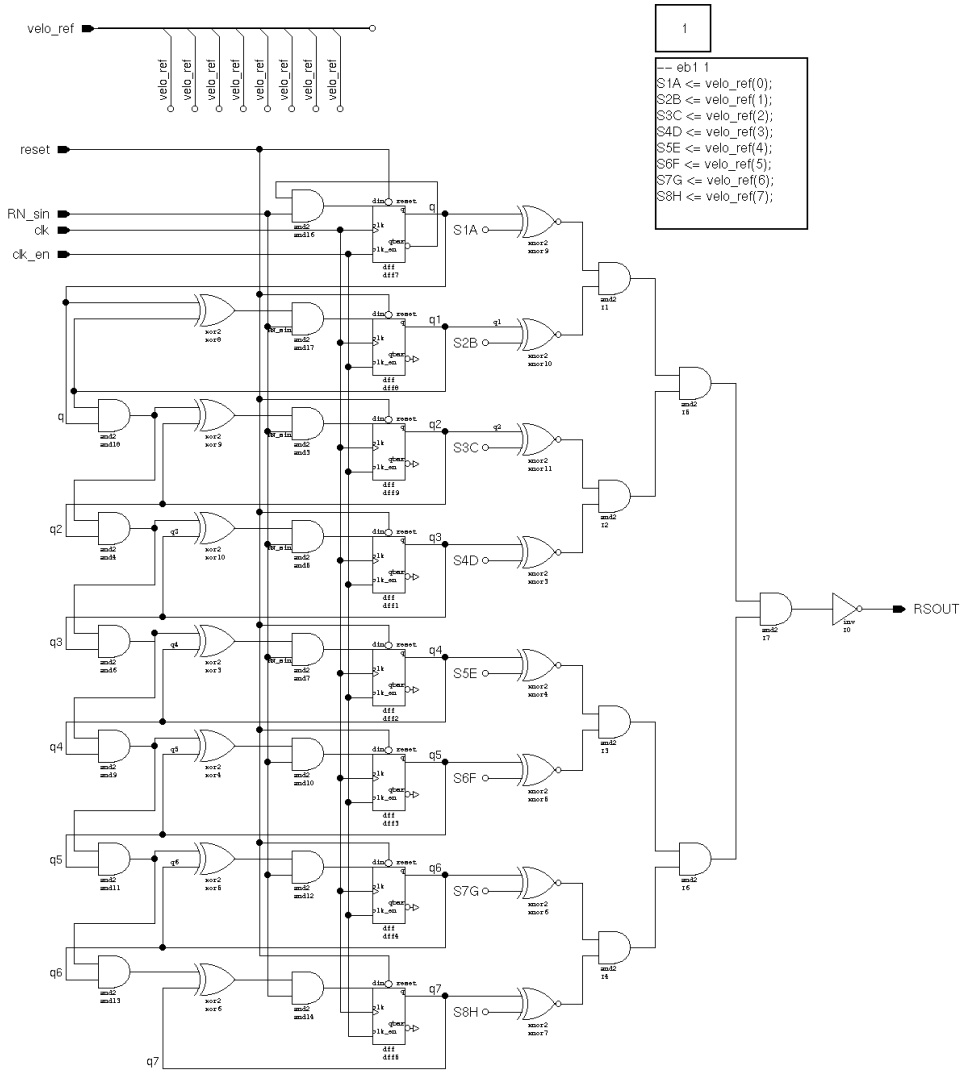
O Bloco Monitor_SS é composto por cinco outros blocos: Bloco1_sc, bloco2, bloco3, bloco 5 e cont_10bt_VC. O Bloco1_sc recebe o sinal SS que é estabilizado e sincronizado aos *clock's* 1200 Hz e 25.175 MHz, e os transforma em um pulso de uma unidade do período do *clock* de 25.175 MHz. Este processamento é realizado no bloco SC, que está contido no Bloco1_sc juntamente com os blocos Cont_8bt e single_pulse_clk_1200. Destes dois blocos, o Cont_8bt recebe o sinal da velocidade processado no Bloco SC e conta os pulsos de velocidade até o valor determinado pela seleção de amostragem. Este contador é feito com flip-flops com *enable*, e é reinicializado sincronamente, com sua realimentação feita externamente (com a saída RSOUT entrando na entrada RN_sin). O sinal de reinicialização síncrona, que possui o valor da amostragem selecionada (RSOUT) entra no bloco single_pulse_clk_1200 para que o sinal se torne um pulso com período unitário de um clock de 1200 Hz. O sinal que sai deste bloco é o sinal VC. A Figura 31 ilustra os blocos SC, single_pulse_clk_1200 e Cont_8bt e a Figura 32 ilustra a interligação destes blocos formando o bloco1_sc. A Figura 33 ilustra a simulação do bloco1_sc: com um valor de amostragem colocado em sua entrada, o contador conta estes pulsos e o contador é reinicializado quando este valor é atingido, resultando no sinal VC.



a)



b)



c)

Figura 31: a) Bloco SC. b) Bloco single_pulse_clk_1200. c) Bloco Cont_8bt.

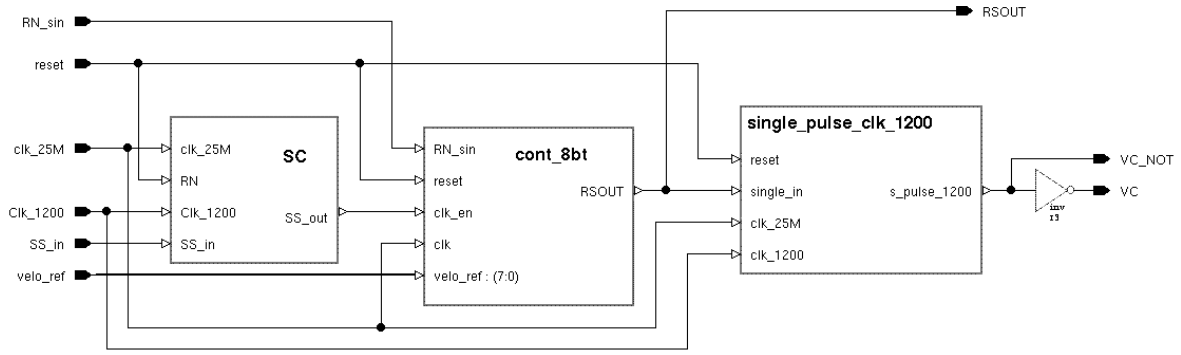
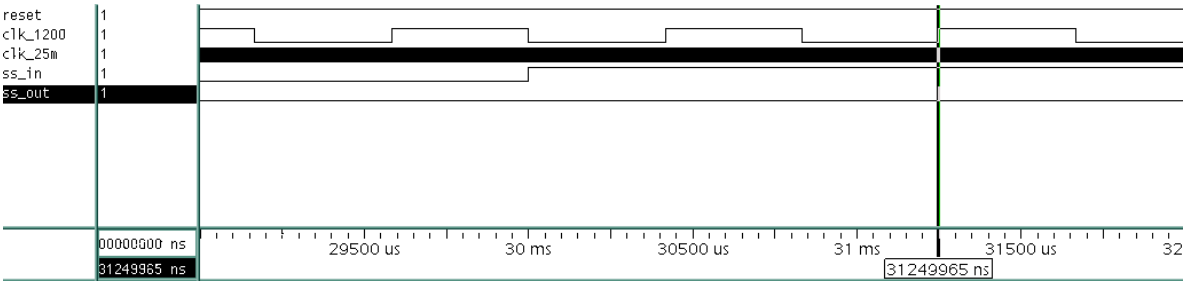
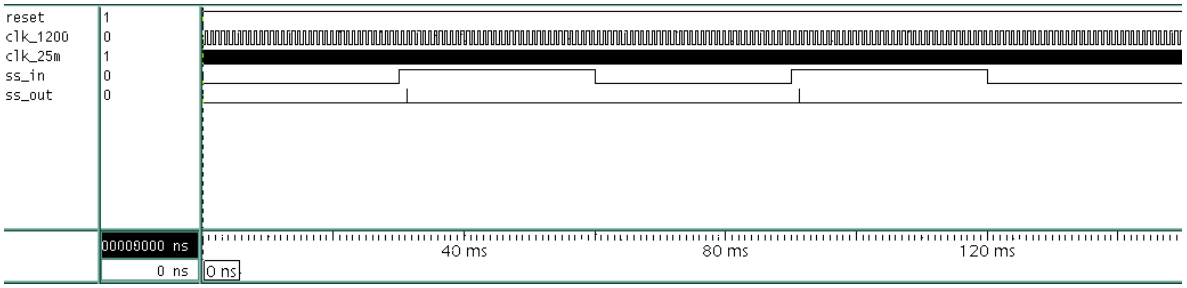


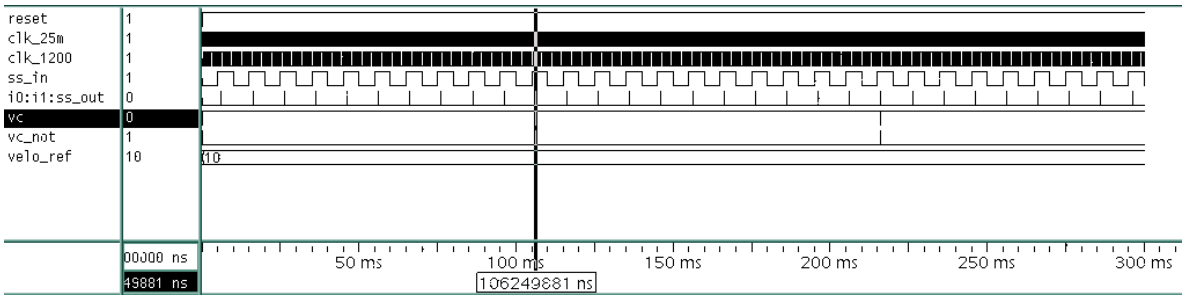
Figura 32: Bloco1_SC



a)



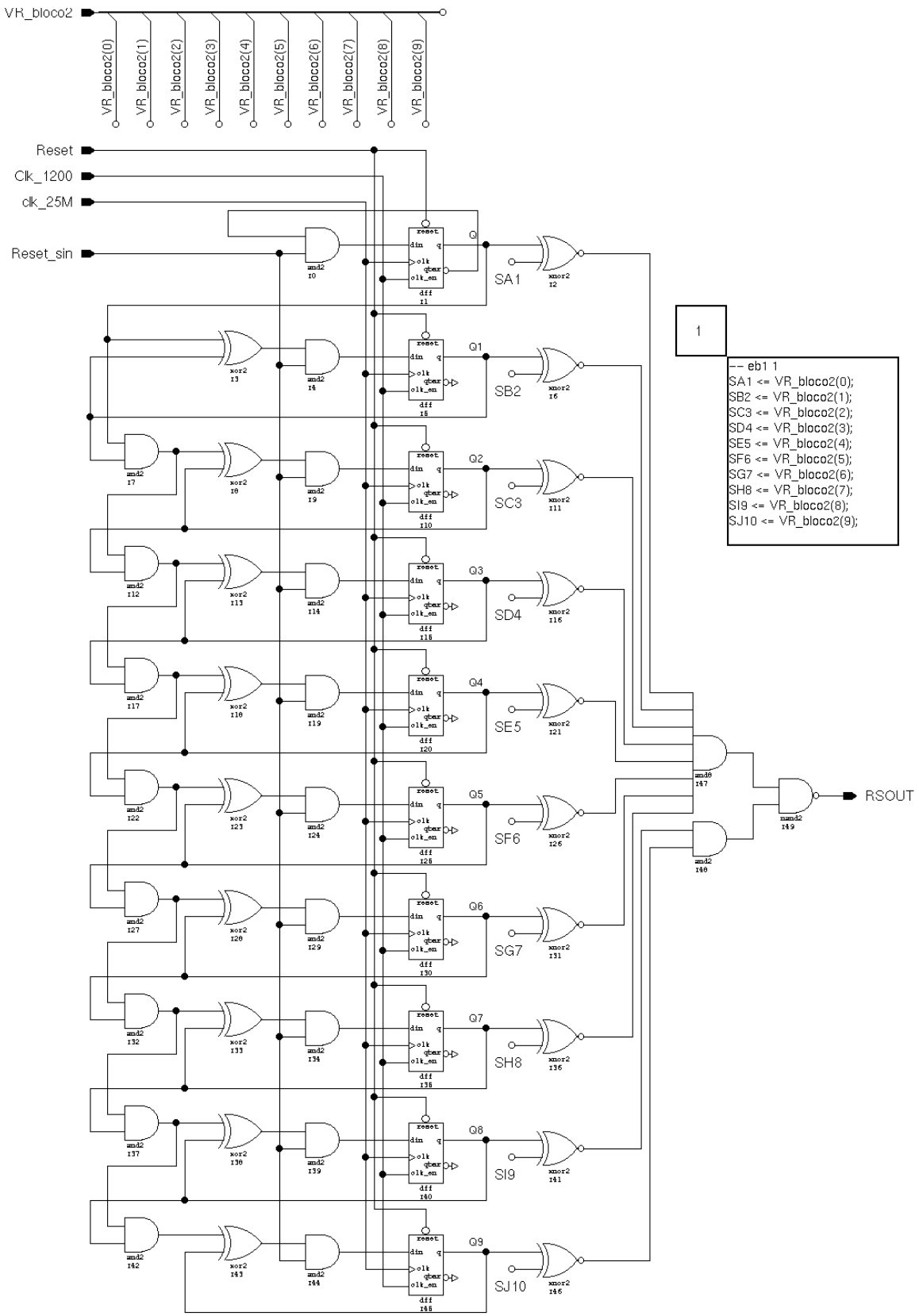
b)



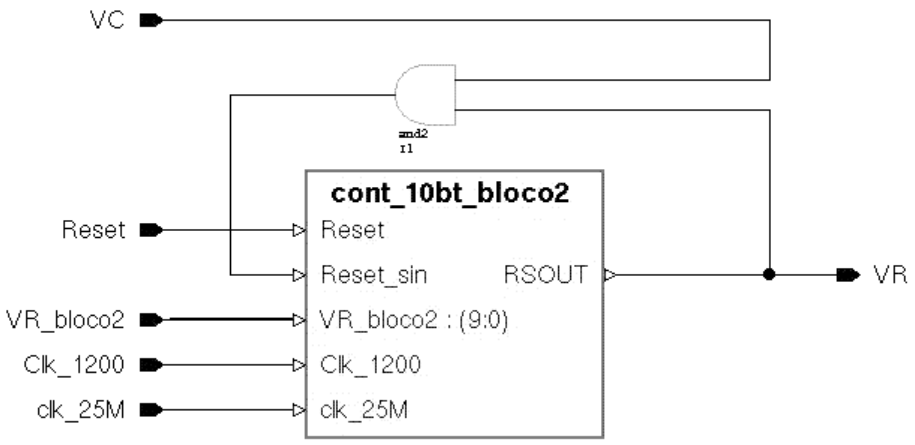
c)

Figura 33: a) Simulação do bloco_SC mostrando o pulso SS após dois pulsos de clock 1200Hz. b) Simulação do bloco_SC ilustrando o sincronismo da velocidade SS vinda do sensor com os sinais de clock existentes no circuito. c) Simulação do Bloco1_SC.

O Bloco2 recebe os sinais de referência vindos do Bloco SM, onde ficam memorizados os valores da velocidade de referência e o sinal VC. Estes dois sinais reinicializam sincronamente um contador de 10 bits, que é o bloco principal do bloco2. O Bloco Cont_10bt_bloco2 é um contador síncrono utilizando flip-flops com *enable.* Estes flip-flops trabalham com *clock* de 25,175 MHz e seu *enable* recebe o número de pulsos do *clock* 1200 Hz. O contador de 10 bits conta o valor dos pulsos de 1200 Hz até o valor pré-definido na entrada VR_bloco2, reinicializando sincronamente quando este valor é atingido. Esse sinal é denominado VR. Se o contador é reinicializado sincronamente por VR, significa que VC tem mais pulsos de *clock* 1200 que VR, ou seja, a freqüência é menor, e a velocidade a ser comparada está abaixo da velocidade de referência. Se o contrário acontecer e o contador for reinicializado por VC, significa que VC tem uma freqüência maior que VR, e o veículo está em velocidade maior que a de referência. A Figura 34 ilustra o Bloco Cont_10bt_bloco2 que faz parte do Bloco2.


VR_bloco2
VR_bloco2(0)
VR_bloco2(1)
VR_bloco2(2)
VR_bloco2(3)
VR_bloco2(4)
VR_bloco2(5)
VR_bloco2(6)
VR_bloco2(7)
VR_bloco2(8)
VR_bloco2(9)
Reset
Clk_1200
clk_25M
Reset_sin
SA1
SB2
SC3
SD4
SE5
SF6
SG7
SH8
SI9
SJ10
RSOUT
1
-- eb1 1
SA1 <= VR_bloco2(0);
SB2 <= VR_bloco2(1);
SC3 <= VR_bloco2(2);
SD4 <= VR_bloco2(3);
SE5 <= VR_bloco2(4);
SF6 <= VR_bloco2(5);
SG7 <= VR_bloco2(6);
SH8 <= VR_bloco2(7);
SI9 <= VR_bloco2(8);
SJ10 <= VR_bloco2(9);

a)



b)

Figura 34: a) Bloco Cont_10bt_bloco2. b) Bloco2.

O bloco3 faz uma comparação entre os sinais VC e VR, para verificar a velocidade máxima. Quando o sinal VR é reinicializado sincronamente, como foi explicado no bloco2, antes do sinal VC, significa que a freqüência do sinal VC é menor que a do sinal de referência e o veículo está com velocidade abaixo do valor de referência. Nesta condição a saída Vmáx recebe um valor nulo (0 lógico). Quando o sinal VC tem uma freqüência maior que o sinal VR o bloco2 é reinicializado por VC e VR recebe um valor constante igual a 1 lógico. Nesta condição o sinal de saída Vmáx recebe um valor igual a 1 lógico. O bloco3 apresenta a saída Vmáx igual a 1 lógico para sinais de VC iguais a VR. A Figura 35 ilustra o bloco3.

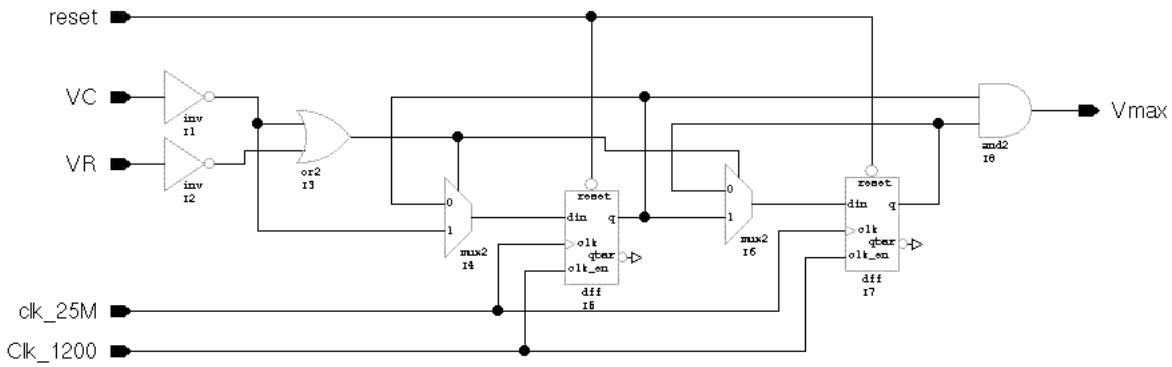


Figura 35: Bloco3

O bloco5 compara os sinais VC e VR para verificar se VC está 4 vezes menor que o sinal de referencia, VR. Caso haja quatro pulsos ou mais de VR contidos no intervalo de VC, a saída BAIXA_VEL vai para 1 lógico. Caso contrário à saída BAIXA_VEL se mantém em 0 lógico. Se não há pulso em SS, não há velocidade. Então, após o quarto pulso de VR, VEL_BAIXA vai para 1 lógico e isto significa que o veículo está parado. A Figura 36 ilustra o bloco5.

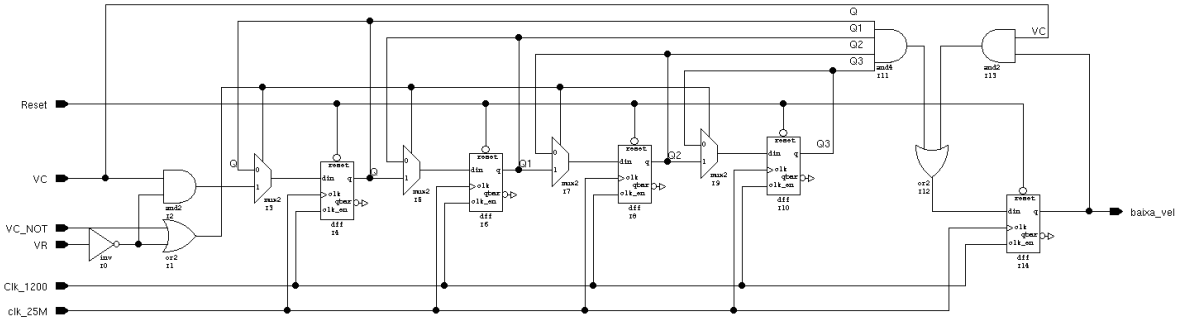


Figura 36: Bloco5.

O Bloco Cont_10bt_VC é um contador síncrono de 10 bits com freqüência de *clock* 25.175 MHz e sinal de *clock* 1200 Hz habilitando a entrada *enable*. Recebe o sinal VC, que é responsável pela reinicialização síncrona do contador. Resumindo, este bloco conta quantos pulsos de *clock* 1200 Hz estão contidos no sinal VC. Sua saída realimenta o Bloco SM.

A função do Bloco Cont_10bt_VC é monitorar o sinal VC, para que quando houver um pulso de SET_VEL acionado pelo usuário, o valor de VC neste momento seja transportado para a saída do bloco SM, alimentando a entrada SM do bloco PV e tornando-se o novo valor da velocidade de referência. A Figura 37 ilustra o diagrama esquemático do Bloco Cont_10b_VC quanto a suas células básicas. A Figura 38 ilustra a simulação do bloco Cont_10bt_VC.

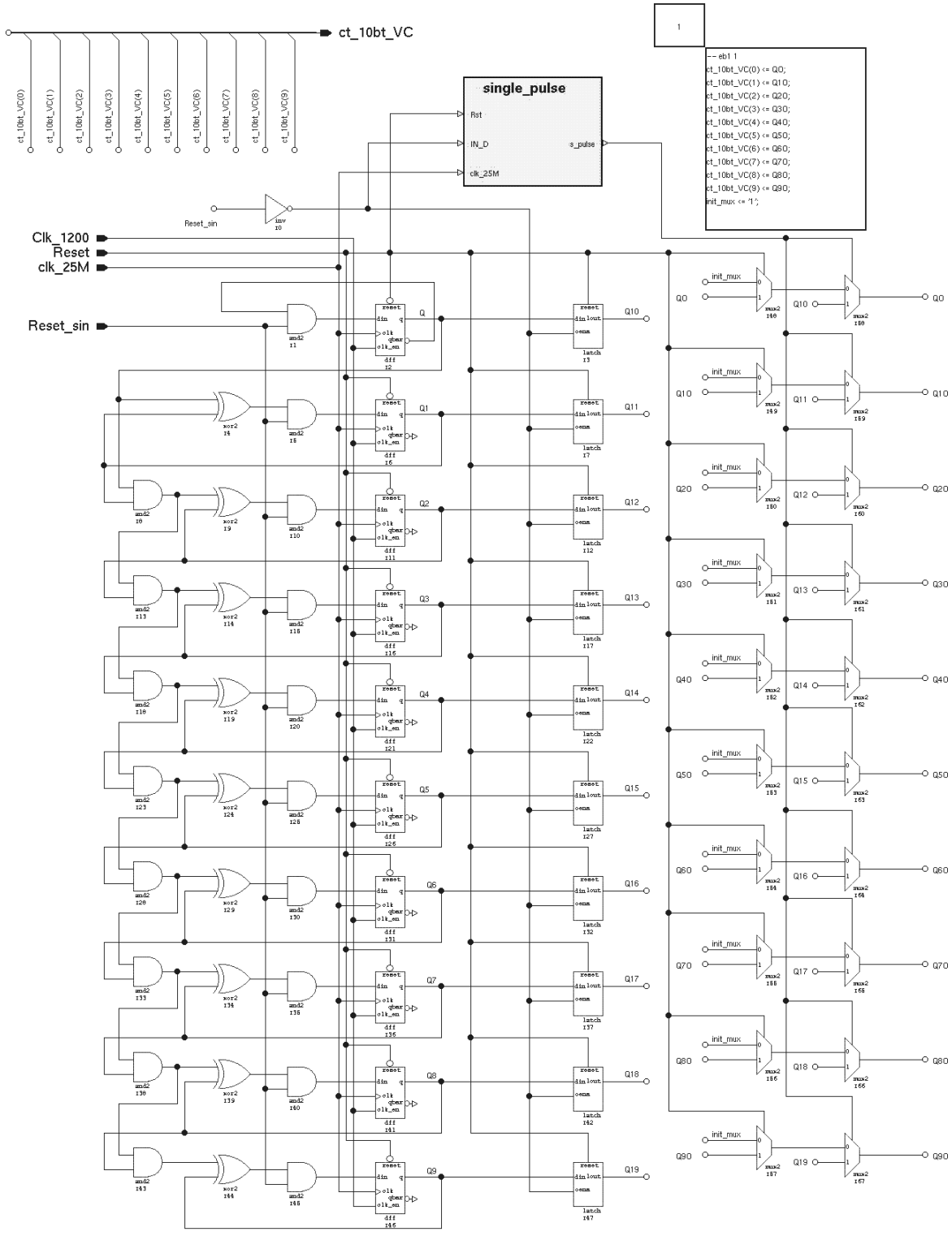


Figura 37: Diagrama esquemático do Bloco Cont_10bt_VC.

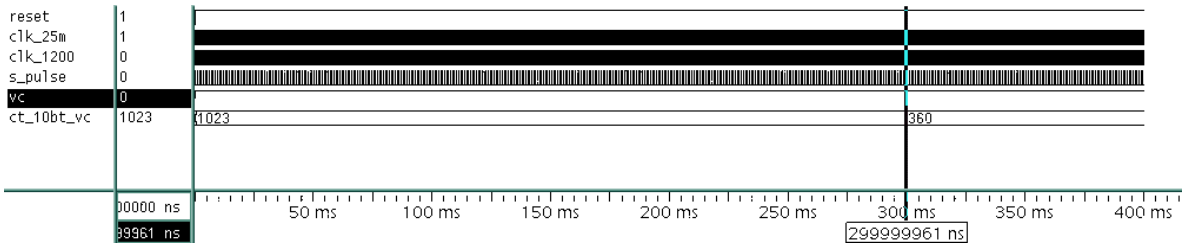
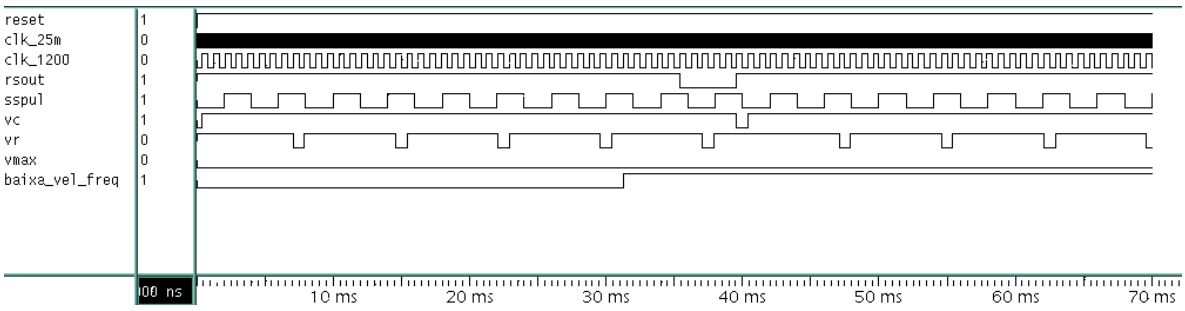
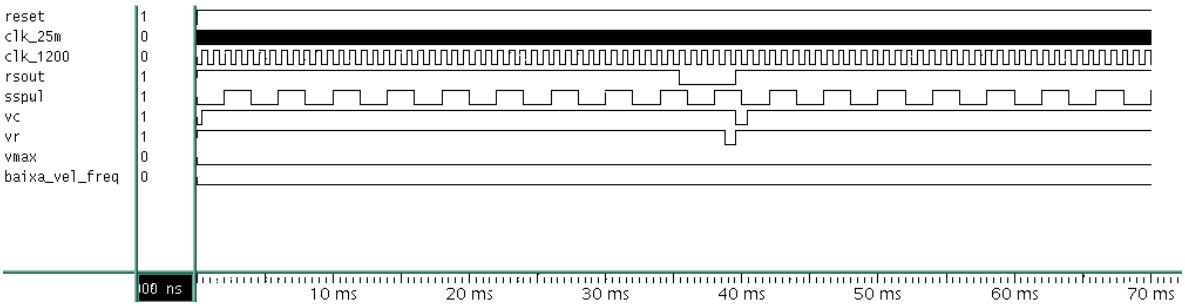


Figura 38: Simulação do Bloco Cont_10bt_VC

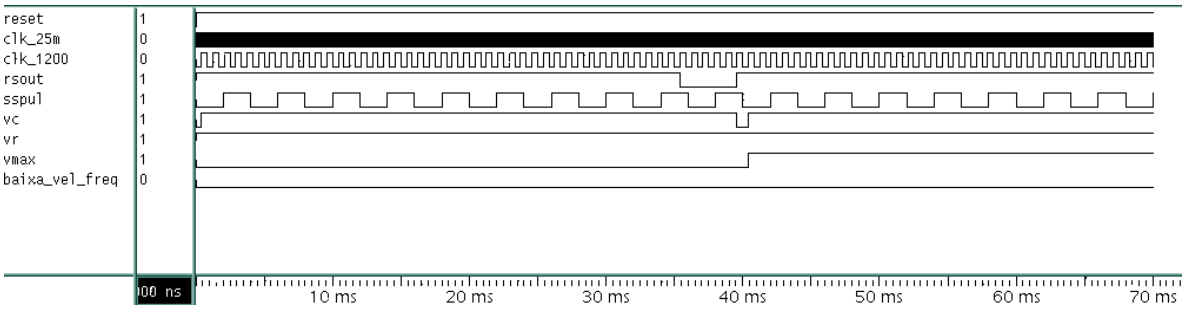
Com a descrição do Bloco Cont_10bt_VC é encerrada a descrição dos blocos que formam o Monitor_SS. As simulações que resumem o funcionamento deste bloco estão ilustradas na Figura 39.



a)



b)



c)

Figura 39: a) Simulação representando o sinal VR contido em mais de 4 vezes no sinal VC resultando no sinal baixa_vel acionado. b) Simulação do sinal VR reinicializado momentos antes do sinal VC representando que o veículo está na iminência de atingir a velocidade máxima mas ainda está em velocidade normal, os sinais baixa_vel e Vmáx estão desabilitados. c) Simulação do sinal VC reinicializando antes do Sinal VR. O Sinal VR permanece em nível lógico alto e o sinal de Vmáx é habilitado.

O último Bloco que forma o bloco PV é o BLOCO4. Este bloco é o responsável pelo controle do sinal sonoro do circuito CIMV. Dependendo da saída Vmáx e do sinal seletor Ini_bip, o circuito aciona uma sirene com um sinal intermitente composto de uma freqüência de 2400Hz modulada por uma de 2Hz. Se o sinal Inib_bip estiver em nível lógico 1, o sinal bip pode ser habilitado. Se o Inib_bip estiver em nível lógico 0, o sinal bip não é habilitado, independente do sinal Vmáx. A Figura 40 ilustra o diagrama esquemático do circuito do Bloco4.

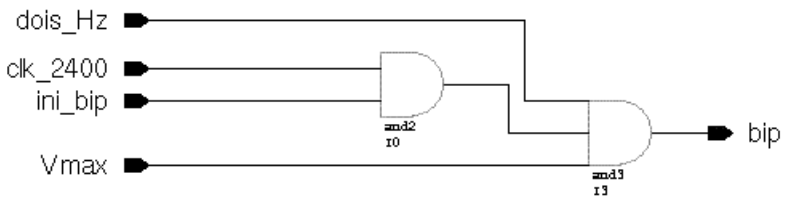


Figura 40: Diagrama esquemático do circuito do Bloco4

O bloco PV com seus três blocos principais é ilustrado na Figura 41.

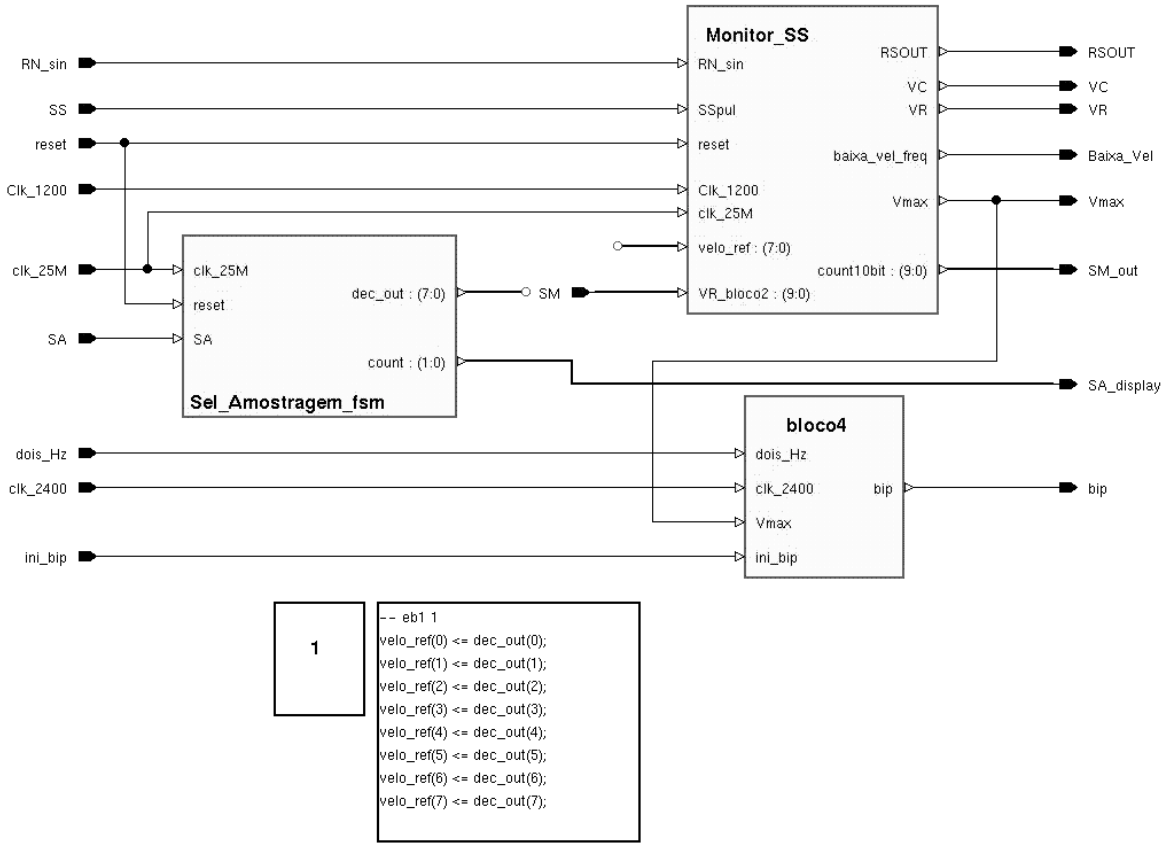


Figura 41: Diagrama esquemático do bloco PV.

2.3.2 Bloco SM

O bloco SM recebe o sinal de saída do bloco PV proveniente do contador de 10 bit, que monitora o sinal VC, Cont_10bt_VC. Na ocorrência de um evento no sinal de Set_vel, este valor é passado para a saída SM_OUT do bloco SM, realimentando o bloco PV com um novo valor de sinal de referência. O Bloco SM é composto por *latches* que passam o valor contido na entrada D para a saída Q, quando ocorre um evento em sua porta de controle. Multiplexadores são usados para estabelecer um valor inicial na saída do bloco SM bem como para realimentar o valor anterior até que um novo pulso de SET_VEL ocorra. A Figura 42 ilustra o diagrama esquemático do bloco SM.

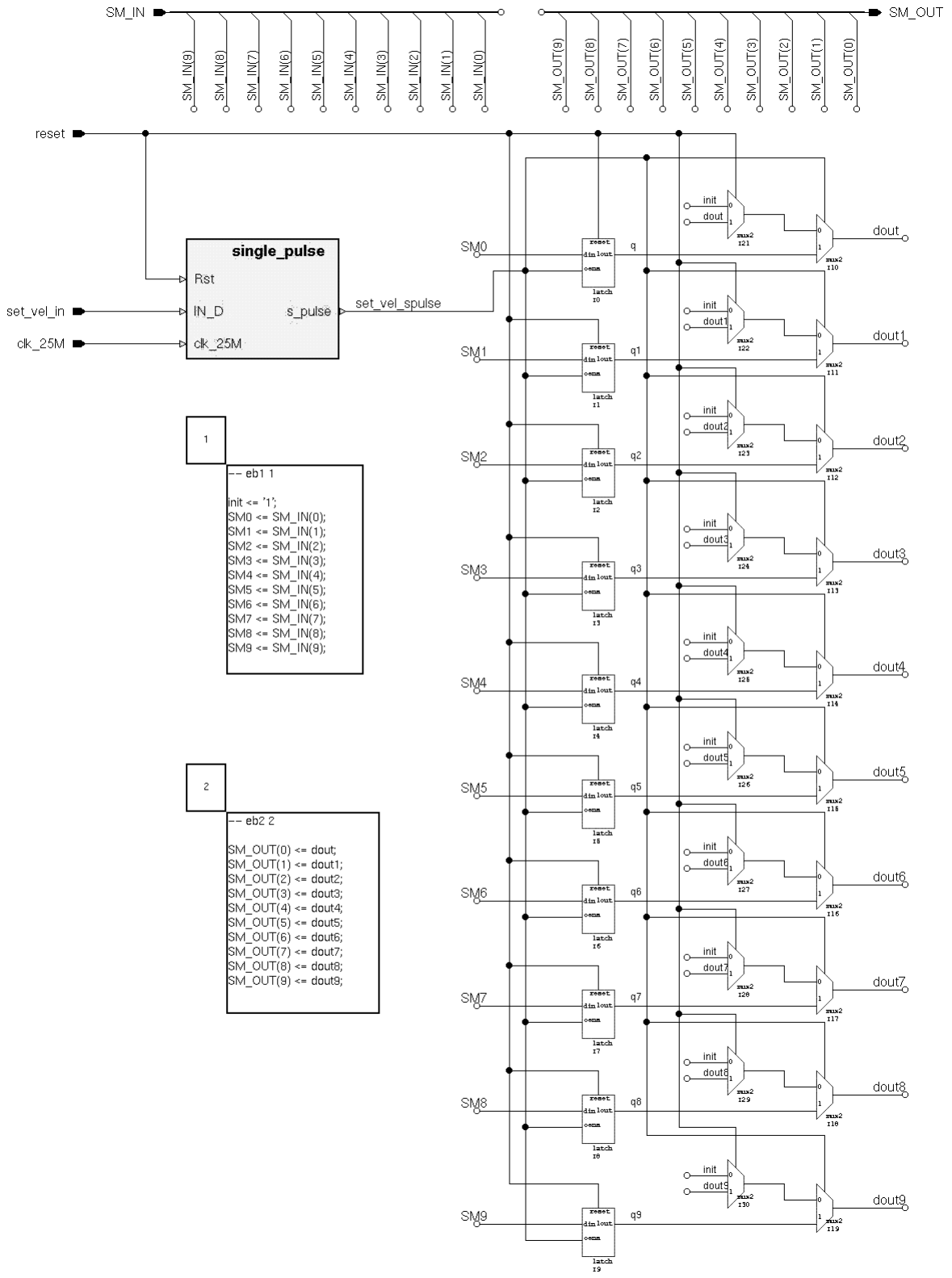


Figura 42: Diagrama esquemático do bloco SM.

A Figura 43 ilustra a simulação do bloco SM, que recebe um valor inicial máximo e após um pulso de set_vel _in, disponibiliza na saída do bloco SM o número de pulsos de clk_1200 contidos no pulso VC.

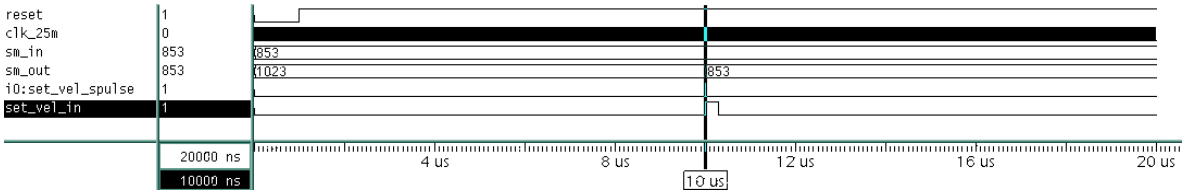


Figura 43: Simulação do Bloco SM.

2.3.3 Bloco Display

O bloco Display faz a interface entre os valores quantificados no bloco PV e o usuário. O Bloco Display recebe os sinais VC e SA_Display. Três contadores de década contam a quantidade de pulsos de *clock* 1200 Hz existente no sinal VC. A saída do sinal é um valor BCD, que será armazenado a cada pulso de VC e enviado para um decodificador BCD-7 segmentos. O Bloco Display é formado por outros 5 blocos: single_pulse, counter_display_esq, display_mem, dec_7seg e dec_7seg_SA.

O Bloco single_pulse tem a função de receber o sinal invertido de VC e transformá-lo num pulso de um período unitário do *clock* de 25.175 MHz. Este sinal aciona as portas dos *latches* do bloco Display_mem para que os valores da contagem sejam transportados para os decodificadores de 7 segmentos. O diagrama esquemático do bloco single_pulse já foi descrito anteriormente.

O Bloco couter_display_esq é um contador de década de 0 à 9, que conta no sistema BCD. É reinicializado sincronamente pelo sinal VC ou quando atinge o valor 9. Este bloco conta o sinal de *clock* 1200Hz; o sinal de *enable* pode ser habilitado ou desabilitado, dependendo do valor dos contadores correspondentes a unidade, dezena e centena. A Figura 44 ilustra o diagrama esquemático do bloco couter_display_esq.

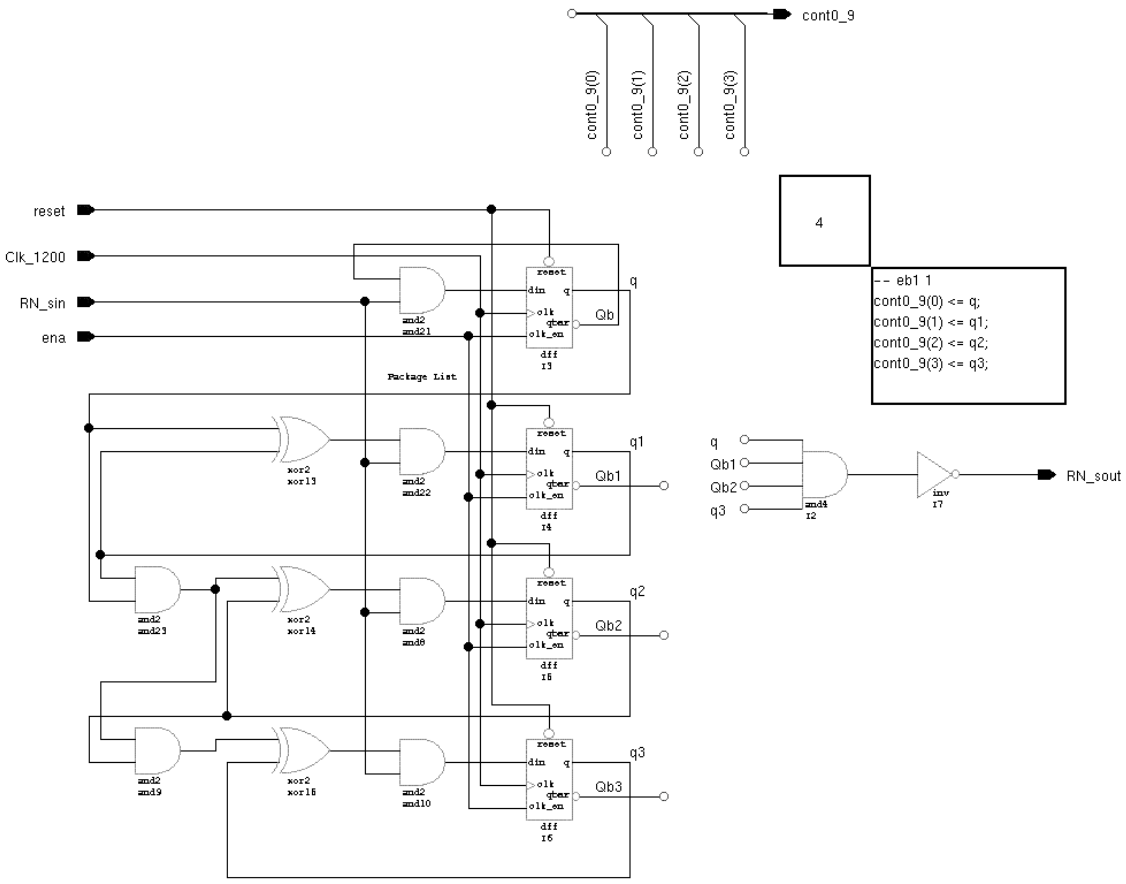


Figura 44: Diagrama esquemático do bloco couter_display_esq.

O bloco display_mem recebe o sinal vindo do bloco single_pulse que, habilita a passagem do sinal vindo do bloco counter_display_esq para a sua saída. A Figura 45 ilustra o diagrama esquemático do bloco display_mem.

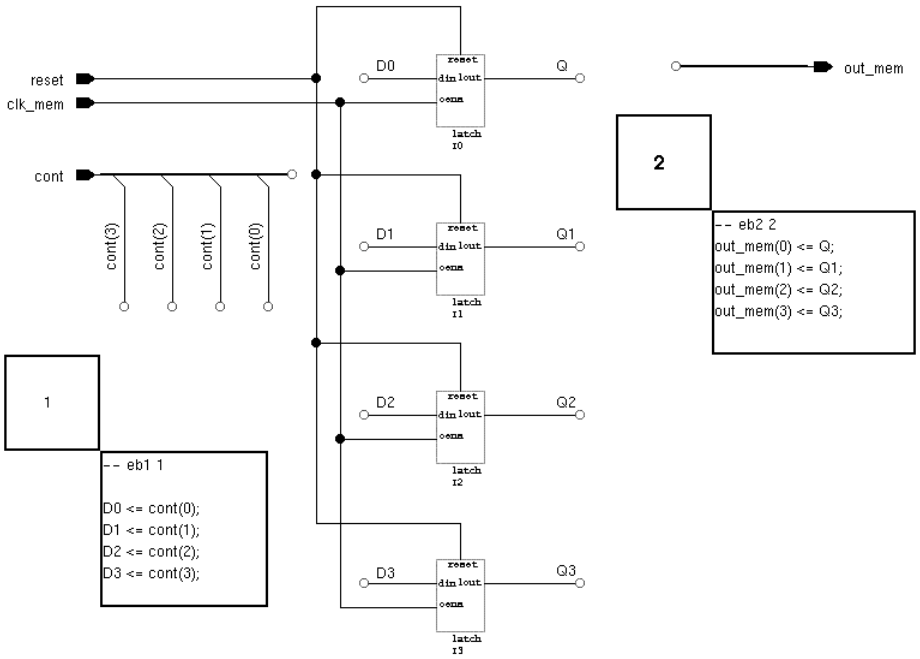


Figura 45: Diagrama esquemático do bloco display_mem.

O bloco dec_7seg que é descrito na forma de tabela verdade, recebe um valor BCD e devolve um valor codificado na forma de Display de 7 segmentos. A lógica utilizada acende os segmentos em valor lógico 0 (uma característica dos displays de 7 segmentos que serão utilizados). A Figura 46 ilustra a tabela verdade utilizada para a codificação do bloco Dec_7seg.

	A	B	C	D	E	F	G	H	I	J	K
1	Q3	Q2	Q1	Q0	a	b	c	d	e	f	g
2	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'1'
3	'0'	'0'	'0'	'1'	'1'	'0'	'0'	'1'	'1'	'1'	'1'
4	'0'	'0'	'1'	'0'	'0'	'0'	'1'	'0'	'0'	'1'	'0'
5	'0'	'0'	'1'	'1'	'0'	'0'	'0'	'0'	'1'	'1'	'0'
6	'0'	'1'	'0'	'0'	'1'	'0'	'0'	'1'	'1'	'0'	'0'
7	'0'	'1'	'0'	'1'	'0'	'1'	'0'	'0'	'1'	'0'	'0'
8	'0'	'1'	'1'	'0'	'1'	'1'	'0'	'0'	'0'	'0'	'0'
9	'0'	'1'	'1'	'1'	'0'	'0'	'0'	'1'	'1'	'1'	'1'
10	'1'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'
11	'1'	'0'	'0'	'1'	'0'	'0'	'0'	'1'	'1'	'0'	'0'

Figura 46: Bloco Dec_7seg descrito na forma de tabela verdade.

O bloco dec_7seg_SA, que também é descrito na forma de tabela verdade, recebe o valor da seleção de amostragem escolhida pelo usuário e devolve um valor codificado na forma de Display de 7 segmentos. A lógica utilizada também acende os segmentos em valor lógico 0. A Figura 47 ilustra a tabela verdade utilizada para a codificação do bloco dec_7seg_SA.

	A	B	C	D	E	F	G	H	I
1	Q1_SA	Q0_SA	a_sa	b_sa	c_sa	d_sa	e_sa	f_sa	g_sa
2	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'1'
3	'0'	'1'	'1'	'0'	'0'	'1'	'1'	'1'	'1'
4	'1'	'0'	'0'	'0'	'1'	'0'	'0'	'1'	'0'

Figura 47: Bloco Dec_7seg_SA descrito na forma de tabela verdade.

A Figura 48 ilustra o diagrama esquemático do bloco Display com seus sub-blocos e interligações.

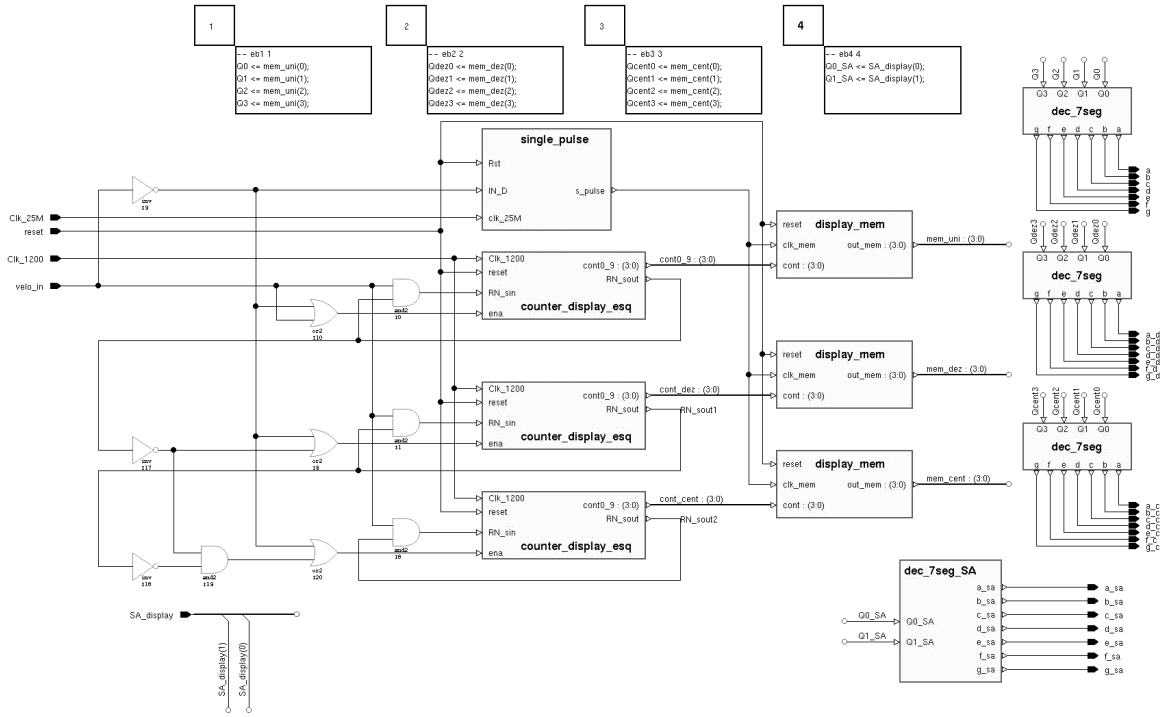


Figura 48: Diagrama esquemático do bloco Display.

2.3.4 Bloco Gera_clk

O bloco Gera_clk recebe um sinal de freqüência de 25,175 MHz. Esse sinal entra no bloco Gera_clk_25M_4800, que é um contador de pulsos de 14 bits reinicializado sincronamente quando a contagem atinge o valor pré-determinado de 5245 em decimal (ou 1010001111101 em binário). Este valor resulta em um sinal com a freqüência de 4800 Hz, que é invertido e entra no *enable* dos flip-flops de um divisor de freqüência por dois de 10 bits, resultando nas freqüências múltiplas de 2400Hz, 1200Hz, 8Hz e 2Hz. A Figura 49 ilustra o bloco Gera_clk e a Figura 50 ilustra o contador de pulsos do Bloco Gera_clk_25M_4800.

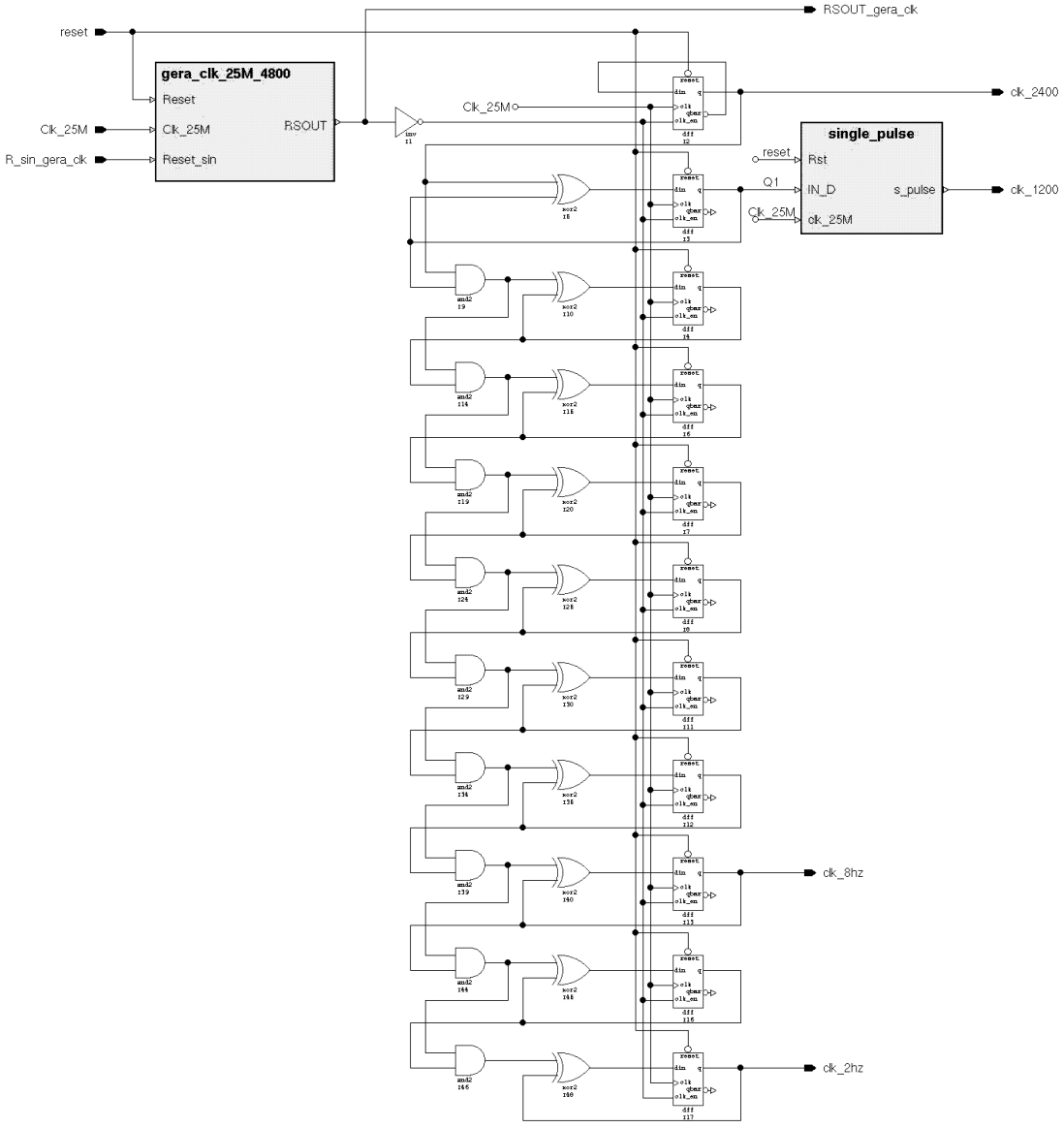


Figura 49: Diagrama Esquemático do bloco Gera_clk.

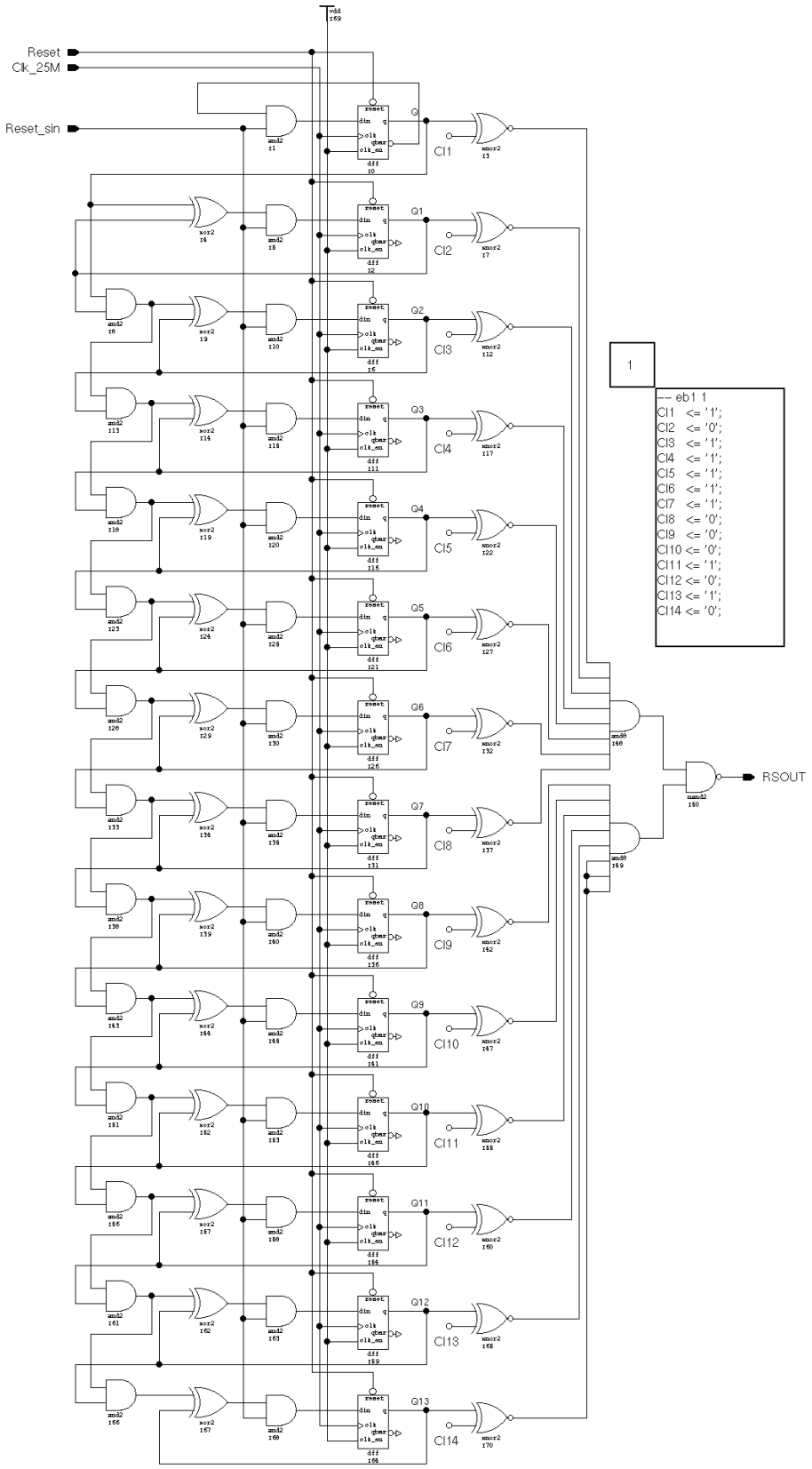


Figura 50: Diagrama Esquemático do bloco Gera_clk_25m_4800.

A Figura 51 ilustra a simulação dos blocos Gera_clk_25M_4800 e Gera_clk. No Bloco Gera_clk a simulação foi dividida em duas partes, devido a grande diferença das freqüências existentes na especificação.

Figura 51: a) Simulação do bloco Gera_clk_25M_4800. b) Simulação do bloco Gera_clk mostrando a Freqüência de 2400 Hz e o pulso de 1200 Hz. c) Simulação do bloco Gera_clk mostrando as Freqüências de 8 e 2 Hz respectivamente.

2.3.5 Bloco Filtro_PB

Este bloco tem a função de eliminar o contato "*bounce*" que ocorre quando botões são pressionados. A Figura 52 ilustra o diagrama esquemático do bloco Filtro_PB.

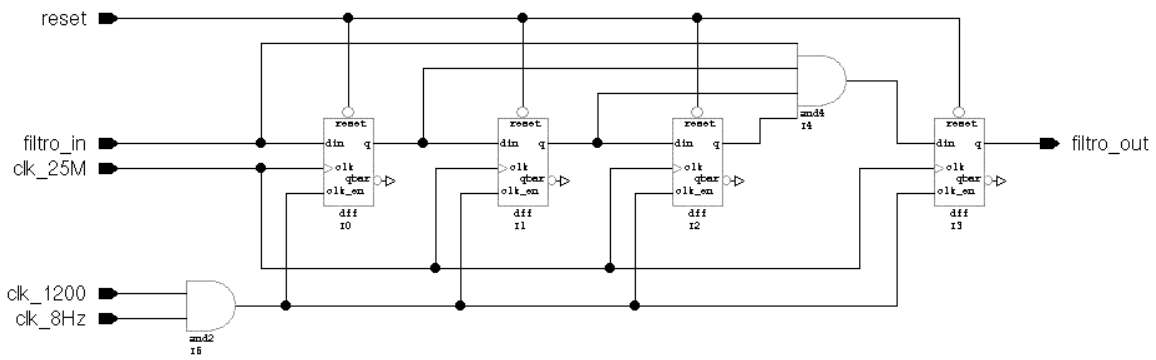


Figura 52: Diagrama esquemático do bloco Filtro_PB.

A Figura 53 ilustra a simulação do bloco Filtro_PB.

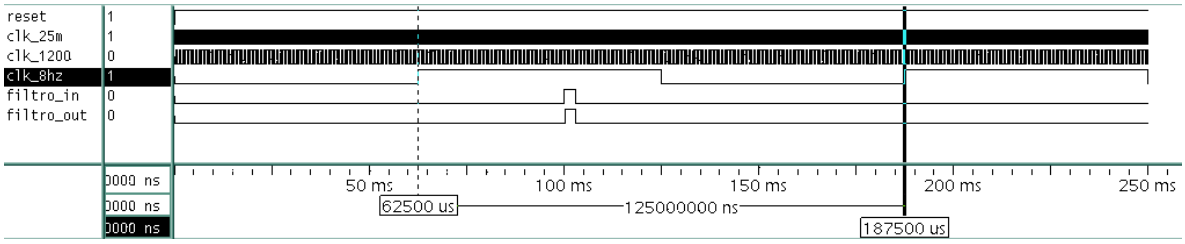


Figura 53: Simulação do bloco Filtro_PB.

2.4 Simulação Global

Após a captura dos esquemáticos, descrição na forma de tabela verdade e descrição na forma de máquinas de estado, os blocos foram simulados individualmente (quando possível) ou em blocos, quando necessário para garantir a fidelidade dos sinais envolvidos. Uma vez verificada a funcionalidade dos blocos eles foram reunidos (Figura 26) a fim de verificar a funcionalidade e especificação global.

Um arquivo de estímulos foi criado para facilitar a simulação. Este arquivo é denominado *test bench*, e nele estão contidos os valores dos sinais de entrada (e como estes variam ao longo do tempo). Os sinais de saída são analisados e tratados de forma a serem ilustrados da melhor forma possível. Como exemplos temos: os circuitos do bloco Display que tem 28 saídas representando os segmentos dos displays utilizados nos testes. O *test bench* recebe o valor destas saídas, analisa-as e devolve-as em forma de unidade, dezena, centena e display_sa, representando os display de sete segmentos que serão utilizados nos testes.

Para a simulação global, foram programados em VHDL sinais que representem as entradas reais que serão utilizadas nos testes. Todos os sinais de entrada recebem estimulo para que estes estejam em estado conhecido durante a simulação. Os sinais que recebem estímulos são: CLK_25M, RESET, SS, INIBIP, SET_VEL e SA. O sinal CLK_25M simula o *clock* do cristal de 25,175 MHz. O sinal RESET simula a reinicialização assíncrona dos circuitos, colocando suas saídas em estado lógico conhecido baixo. O sinal SS simula a velocidade vinda do sensor de velocidade do veículo. Este sinal foi projetado de forma que seu valor varie ao longo da simulação, simulando, desta forma, um aumento e/ou diminuição de velocidade. A velocidade será discutida na dimensão de freqüência (Hertz) para facilitar o entendimento, e devido à determinação da velocidade real depender de outros fatores externos já discutidos anteriormente.

O sinal INIBIP será simulado como uma chave que inibe o sinal sonoro quando o veículo atinge a velocidade máxima especificada, deixando apenas o sinal visual (LED). Os sinais SET_VEL e SA simulam os pulsos vindo de botões que o usuário aperta para determinar a velocidade limite e a seleção de amostragem desejada. Estando os estímulos definidos, uma simulação global foi realizada com um tempo total de 60 segundos, dentre os quais a funcionalidade foi testada, variando parâmetros como velocidade e SET_VEL. A Figura 54 ilustra o CIMV_clk_tester projetado para a gerar os estímulos iniciais.

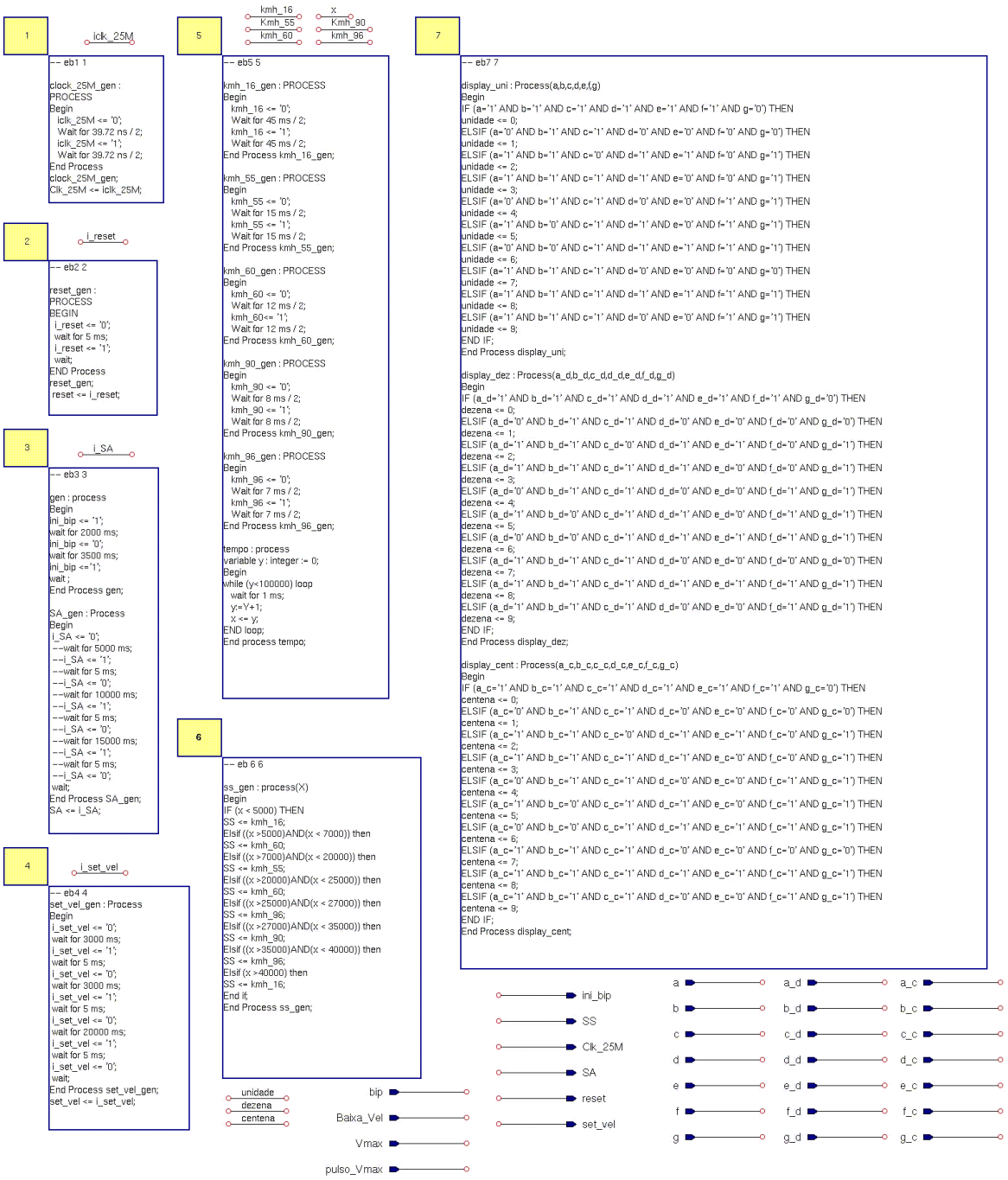


Figura 54: CIMV_clk_tester projetado para gerar estímulos iniciais.

O circuito da Figura 55 foi o circuito *test bench* utilizado para realizar a simulação final do circuito monitor de velocidade. As informações enviadas pelo CIMV_clk_tester simulam a funcionalidade especificada do circuito monitor de velocidade e estão em um intervalo de 60 segundos. A freqüência de entrada que representa a velocidade do veículo

varia. Pulsos (que representam o usuário especificando uma velocidade) ocorrem em tempos determinados, de modo que possa ser analisado se o veículo esta acima, abaixo ou em velocidade de cruzeiro.

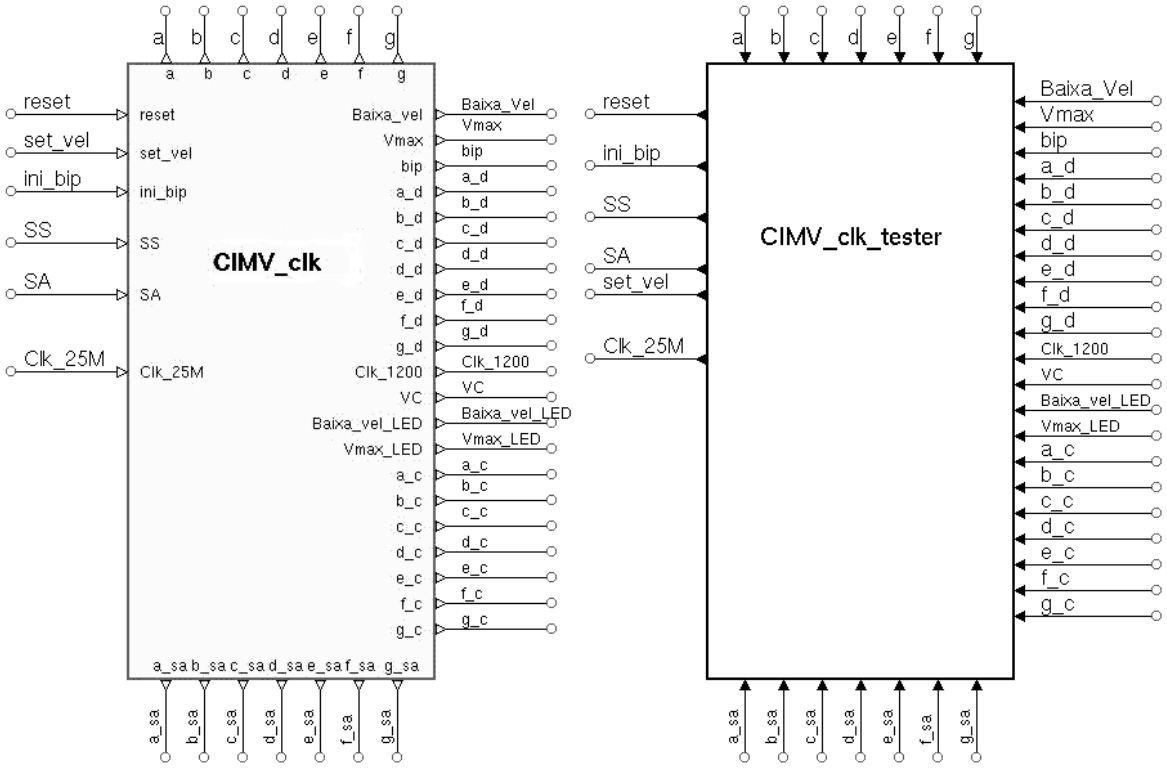


Figura 55: Circuito utilizado para realização da simulação da funcionalidade do circuito monitor de velocidade.

Os 60 segundos da simulação total foram divididos em partes, visando melhor ilustrar e descrever os momentos do circuito em cada situação. Do tempo inicial até 15 segundos. O sinal estimulado na entrada SS é de freqüência baixa, e no circuito CIMV_clk_tester foi nomeado de kmh_16. Depois de 3 s ocorre um pulso de Set_vel, e esta velocidade é armazenada tornando-se a velocidade limite. Em 5 s a velocidade aumenta até uma freqüência nomeada de kmh_60. Em 6 s ocorre um pulso de Set_vel, mudando a velocidade limite para kmh_60. Em 7s a velocidade cai para kmh_55 e se mantém neste valor até 20s. A Figura 56 ilustra a seqüência descrita, onde se pode notar que quando ocorreu um evento Set_vel, o indicador de velocidade máxima foi para nível lógico 1 e aparece um sinal Bip intermitente, que foi inibido entre 2s e 5.5s pelo sinal

INI_BIP. Pode-se verificar também o funcionamento do circuito display, que mostra em valor de unidade, dezena e centena, as variações de freqüência em SS.

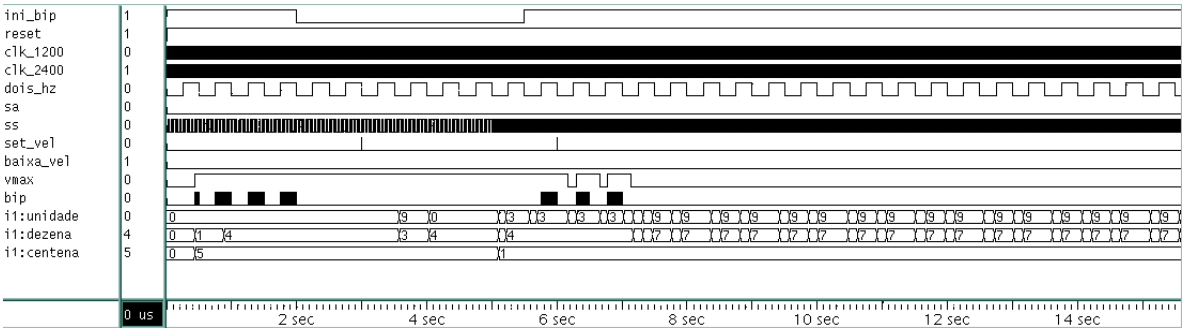


Figura 56: Simulação global do circuito monitor de velocidade no intervalo 0 a 15s.

No parágrafo anterior foi dito que a freqüência de kmh_55 se manteve até 20s, como vemos que ocorre na Figura 57. Depois dos 20s, a freqüência sobe para kmh_60 durante 5s, subindo para kmh_96 no instante 25s e ficando neste valor por 2s. Um pulso Set_vel ocorre no instante 26s e no instante 27s a freqüência diminui para kmh_90 permanecendo neste valor até o instante 35s.

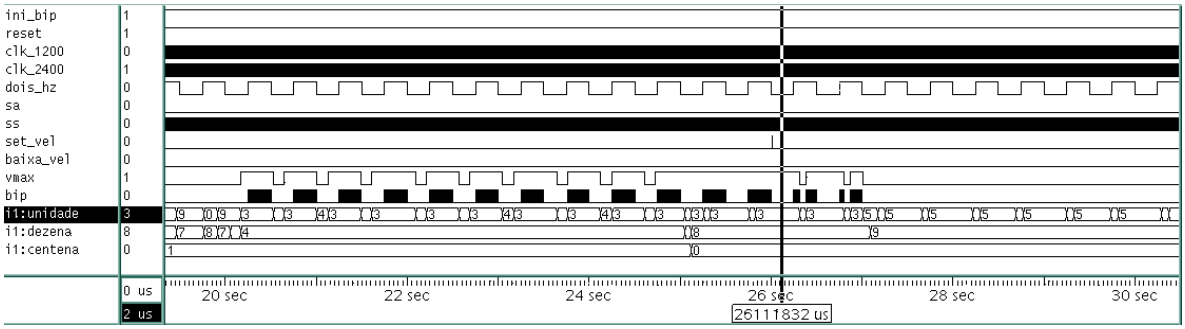


Figura 57: Simulação global do circuito monitor de velocidade no intervalo 20 a 30s.

No instante 35s a freqüência sobe para kmh_96 e permanece por 5 s. Depois cai drasticamente para kmh_16 no instante 40s, permanecendo neste valor até o instante 60s. Pode-se notar que quando kmh_16 ocorre, o sinal BAIXA_VEL recebe nível lógico 1. De fato, kmh_16 é ¼ da velocidade de kmh_96. A Figura 58 ilustra esta situação.

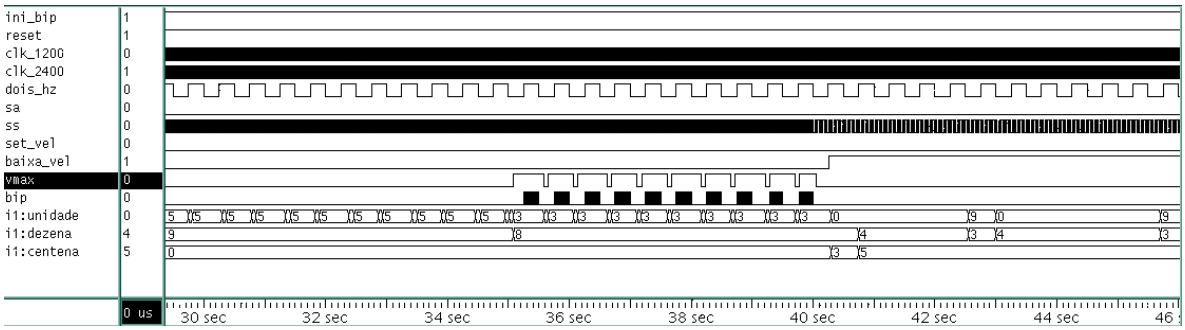


Figura 58: Simulação global do circuito monitor de velocidade no intervalo 30 a 45s.

A Figura 59 ilustra toda a simulação e seus eventos desde o instante 0 até 60s.

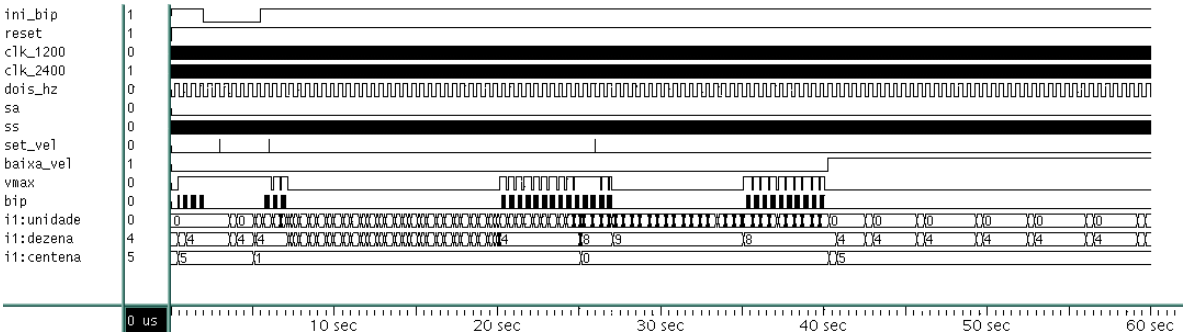


Figura 59: Simulação global do circuito monitor de velocidade no intervalo 0s a 60s.

Analisando a Figura 59 pode-se verificar que o circuito monitor de velocidade atende as especificações quanto a sua funcionalidade. Deste modo, seguindo a seqüência do fluxo de projetos digitais, a síntese lógica é o próximo passo a ser seguido.

Capítulo 3 – Resultados Experimentais

3.1 - Introdução

Visto que o circuito monitor de velocidade atende as especificações, o passo seguinte é a síntese lógica. A ferramenta utilizada para realizar a síntese lógica é o Leonardo Spectrum [27], [28].

A ferramenta utiliza os arquivos HDLs gerados em linguagem VHDL, toda vez que um bloco capturado no HDL Designer Series é compilado. As descrições VHDL de todos os blocos capturados neste projeto são apresentadas em relatório individual no Centro de Pesquisas Renato Archer (CenPRA). O resultado da síntese são arquivos de saída no formato EDIF, que podem ser exportados para outras ferramentas. Neste caso, a ferramenta utilizada é o Max + Plus II 7.21 [30]. O Arquivo EDIF gerado pelo Leonardo Spectrum é exportado para o Max Plus, para ser compilado, e desta forma gerar os arquivos necessários para gravar o projeto em PLD. O projeto será gravado em uma placa educacional UP1 da Altera [30]. Esta placa dispõe de dois dispositivos: EPM7128S dispositivo com 84 pinos da família MAX7000S e EPF10k20 dispositivo de 240 pinos da família FLEX 10K. A família MAX7000S não suportou a gravação do circuito monitor de velocidade. Então o circuito foi gravado no dispositivo da família FLEX 10K.

Para utilizar a totalidade dos recursos disponíveis na placa UP1, foi projetado um circuito simples, apenas com buffers, para que algumas saídas do FLEX 10K utilizassem o recurso de DISPLAY (disponível na placa apenas para o dispositivo da família MAX7000S). Este circuito foi denominado By-Pass e será descrito adiante.

Com o projeto gravado nos dispositivos, o passo seguinte é o teste elétrico funcional. Deve-se ressaltar que o dispositivo da família MAX7000S é um configurável, ou seja, suas informações ficam gravadas até que seja configurada novamente. Já, o dispositivo da família FLEX 10K é programável, suas informações se perdem toda vez que a placa deixa de ser alimentada. Necessitando ser gravado novamente para realizar os testes, se necessário, em ocasiões posteriores.

Os equipamentos utilizados para o teste elétrico funcional foram um osciloscópio Tektronix TDS 3054 500 MHz, um gerador de freqüência HP 3314A – Function Generator, a placa educacional UP1 e um computador pessoal com o software Max + PLUS II versão 7.21 para gravar o projeto nos dispositivos da placa educacional.

3.2 - Síntese Lógica

A síntese lógica é uma etapa relativamente rápida. Isto é de se esperar, porque, como descrito anteriormente, para atender a constante evolução dos circuitos integrados, é necessário flexibilidade e rapidez ao colocar um produto no mercado o mais rápido possível, para que não fique obsoleto antes mesmo de entrar no mercado. Estando a especificação validada na simulação, o protótipo tem que poder ser testado com rapidez. A ferramenta Leonardo Spectrum atende a esta necessidade do mercado e é muito rápida no processo de síntese.

Sendo os HDLs sintetizáveis, dependendo do tamanho do circuito o Leonardo Spectrum realiza a síntese em menos de 15 minutos.

O usuário necessita escolher para que tecnologia o projeto será sintetizado, dentre uma gama de fabricantes e modelos de dispositivos. Em nosso caso foram escolhidos os componentes EPM7128S e EPF10k20 da Altera. Caso alguma *foundry* também disponibilize tecnologia para síntese digital, Leonardo Spectrum também é capaz de sintetizar ASICs. A Figura 60 ilustra o circuito CIMV sintetizado no Leonardo Spectrum.

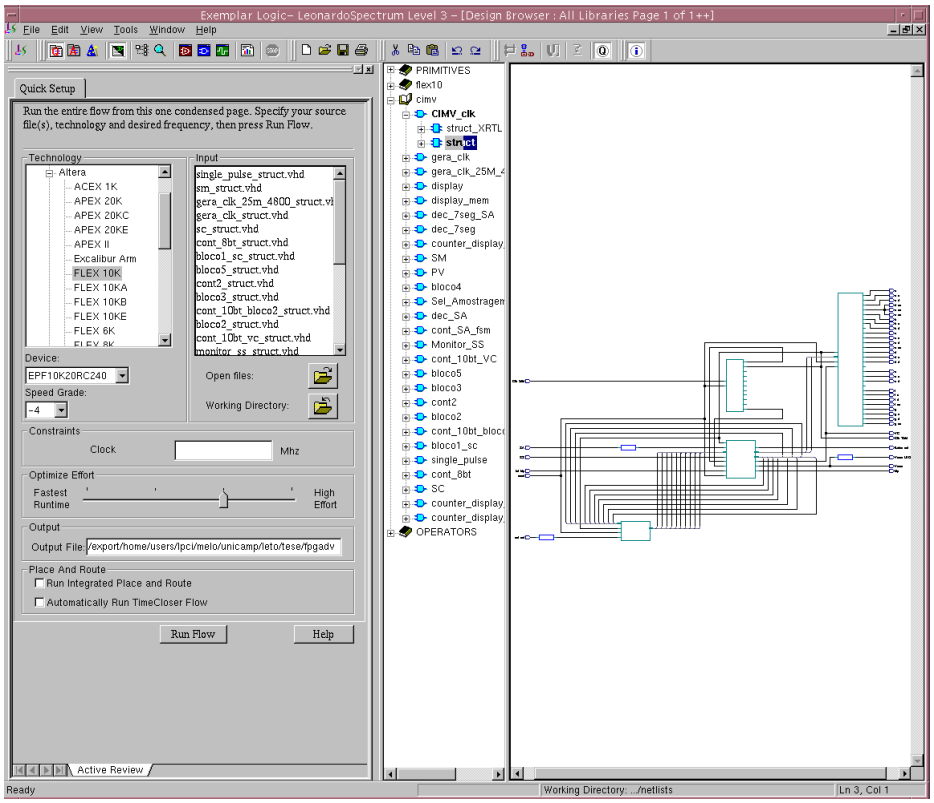


Figura 60: Circuito CIMV sintetizado no Leonardo Spectrum.

Finalizada a síntese, um arquivo de resumo é disponibilizado para o usuário de modo que ele possa saber se o dispositivo comporta o projeto sintetizado. Foi desta forma que o dispositivo da família MAX7000 foi descartado. A Figura 61 ilustra o arquivo de resumo gerado para o dispositivo da família FLEX 10K e como o projeto sintetizado é alocado nas células internas do dispositivo.

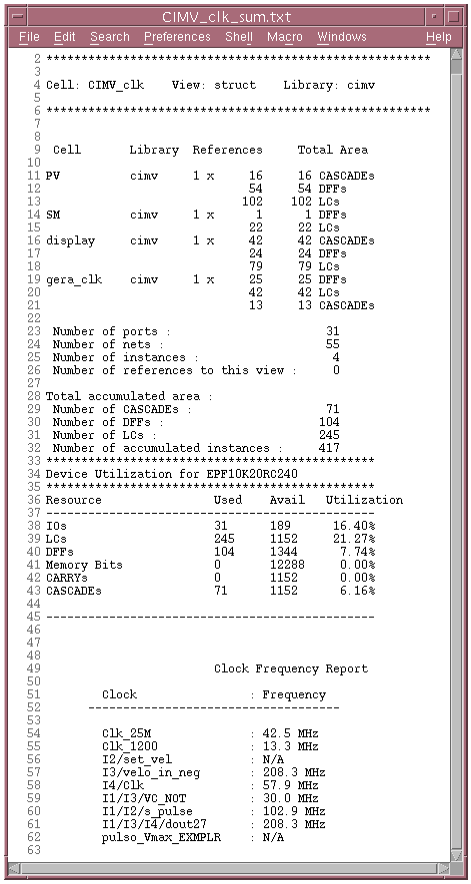


Figura 61: Arquivo de resumo do resultado da síntese.

A Figura 62 ilustra o CIMV sintetizado para o componente FLEX 10K e pode ser comparada com a Figura 26 do capítulo 2, no qual o circuito CIMV é capturado no HDL Designer Series.

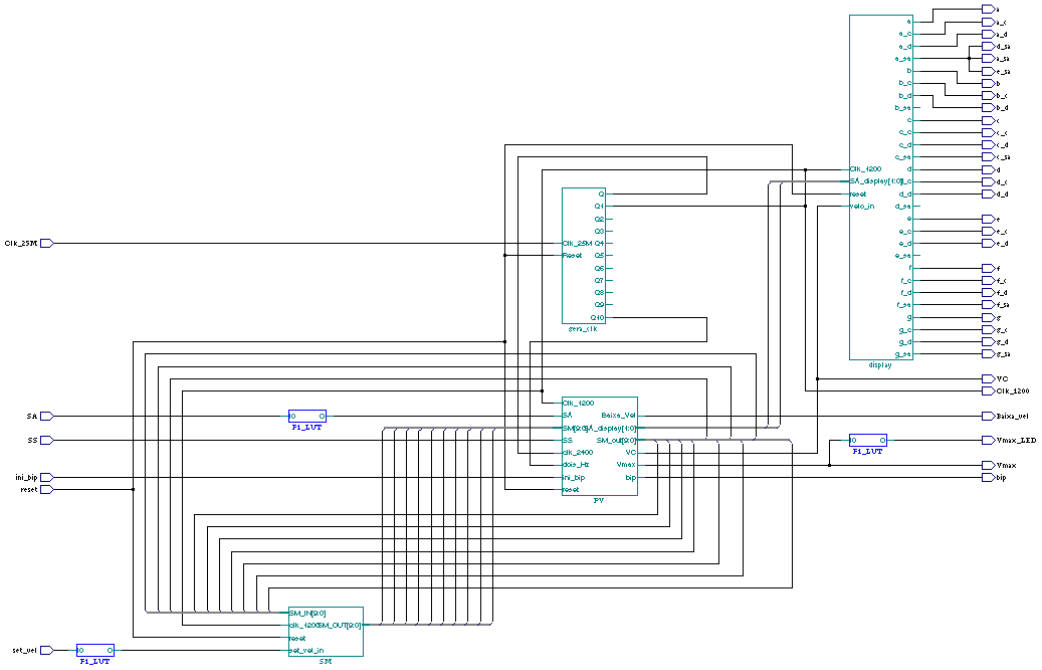


Figura 62: Circuito CIMV sintetizado para o componente da família FLEX 10 K.

3.3 - Programação Lógica

Para utilizar toda a funcionalidade da placa de desenvolvimento UP1, foi sintetizado um outro circuito denominado By_Pass. Este circuito foi gravado no componente configurável da placa de desenvolvimento UP1, da Família Max 7000S. Sua função é receber algumas saídas do componente FLEX e destiná-las aos Displays de 7 segmentos. Assim, os Sinais de display para "centena" e para a seleção de amostragem são direcionados a este componente. Quatorze buffers recebem o sinal que é redirecionado para os Displays de 7 segmentos reservados para o componente da família MAX 7000, o MAX_DIGIT. As Figuras 63 e 64 ilustram o esquemático do circuito By_Pass e sua síntese, respectivamente.

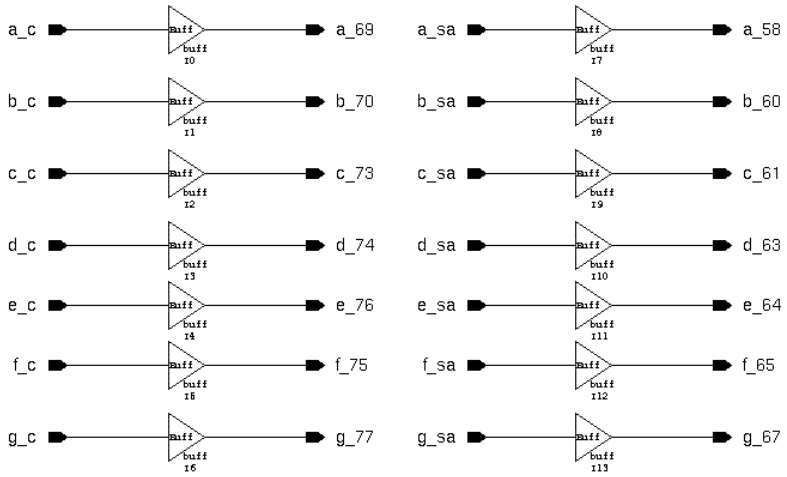


Figura 63: Circuito By_Pass capturado no HDL Designer Series.

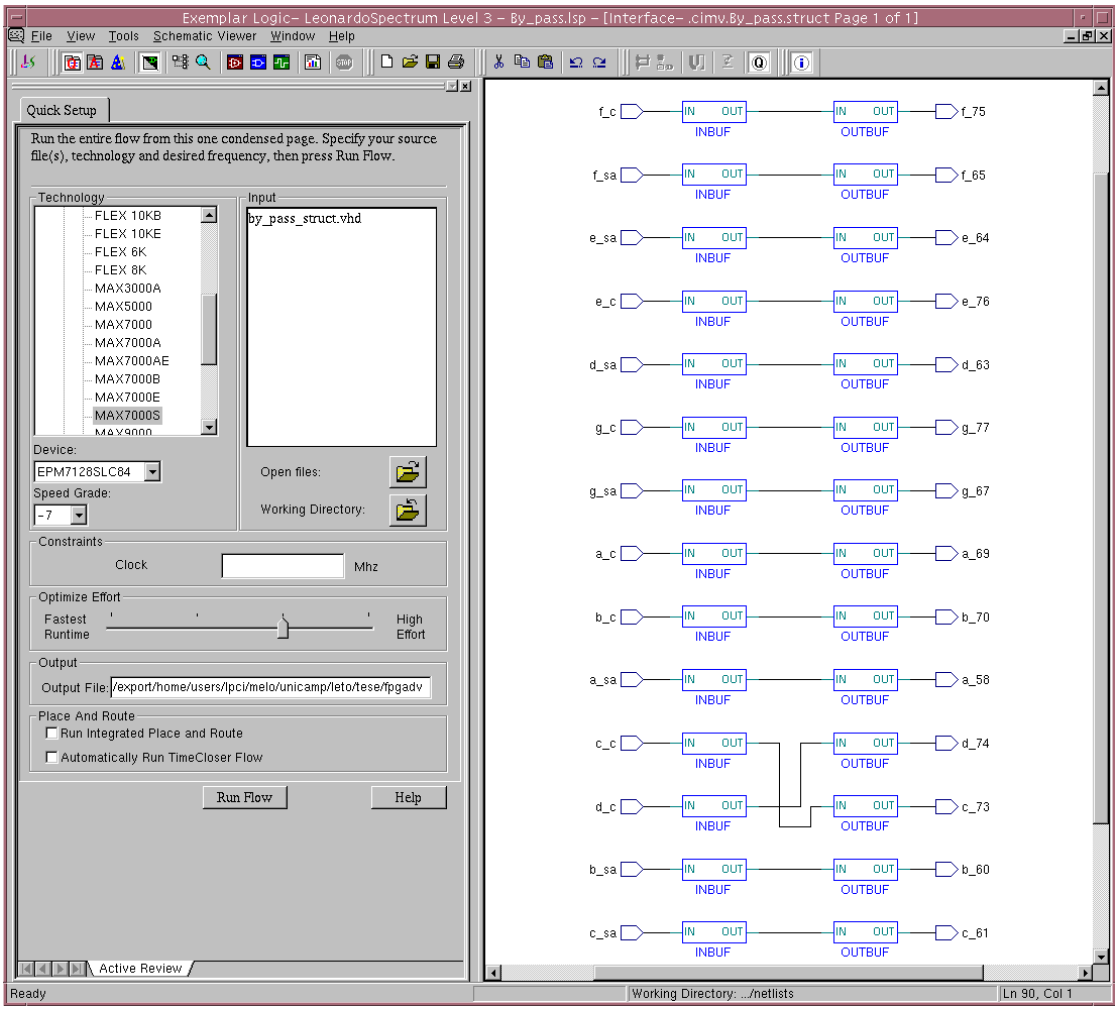


Figura 64: Circuito By_Pass sintetizado no Leonardo Spectrum.

Com a síntese de todos os circuitos realizada, o arquivo edif gerado é exportado para o software Max + Plus da Altera. O arquivo edif então é compilado e gerado um

floorplan, com a distribuição das entradas e saídas do circuito às entradas e saídas do componente da família FLEX 10 K e MAX 7000S. A Figura 65 ilustra o floorplan realizado para o circuito CIMV, o arquivo edif compilado no Max Plus e o software de programação da Altera configurando o componente EPF10K20.

.

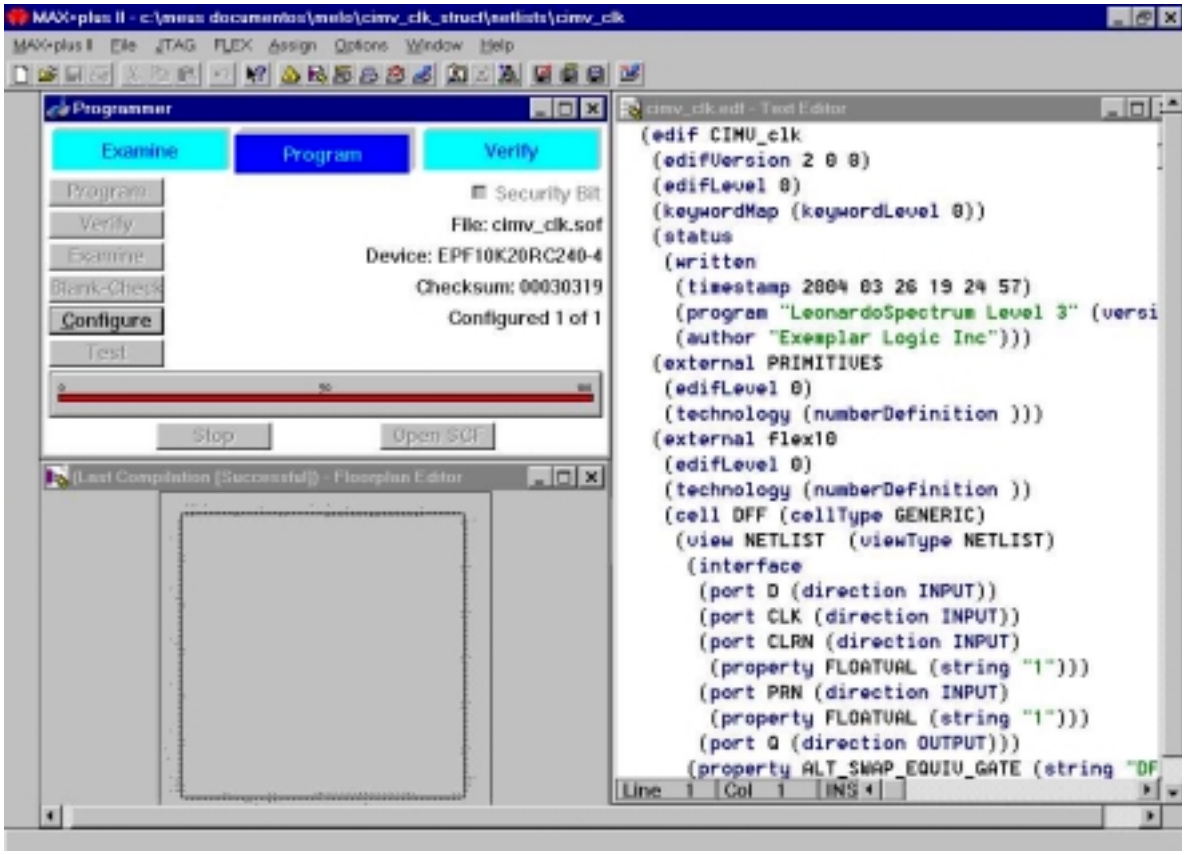


Figura 65: Floorplan realizado no Circuito CIMV, arquivo edif importado do Leonardo Spectrum e software de programação da Altera.

As pinagens dos circuitos foram distribuídas segundo documentação da placa de desenvolvimento educacional UP1, cujo diagrama é ilustrado na Figura 66 [29].

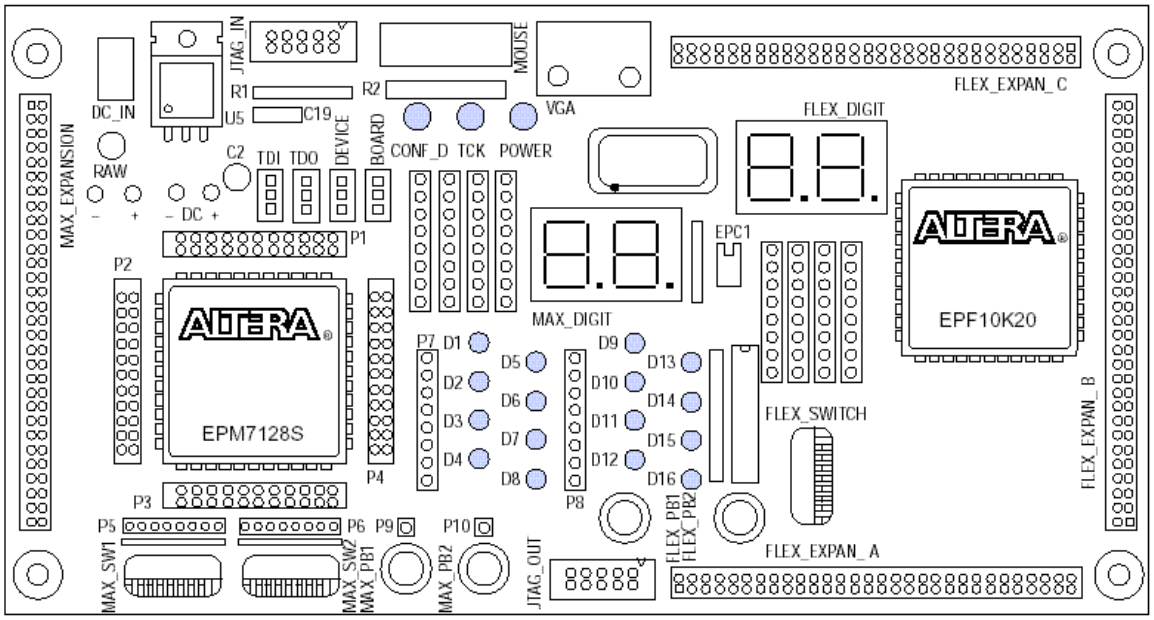


Figura 66: Distribuição dos componentes na placa de desenvolvimento educacional UP1 [29].

Analisando a Figura 66 pode-se observar a distribuição dos componentes na placa de desenvolvimento. Destes componentes serão usados:

Para a família MAX 7000S: O componente configurável EPM7128S com o circuito By_Pass gravado. Este componente programável vem em soquete de 84 pinos e tem 128 macrocelulas com uma capacidade de 2500 gates. Dos recursos disponíveis na placa para este dispositivo serão utilizados: os Prototyping Headers P2 e P3; o MAX_PB2; Os Led´s D4 e D9 e dois displays de 7 segmentos MAX_DIGIT.

Para a família FLEX 10K: O componente programável EPF10K20 com o circuito CIMV programado. Este componente reconfigurável é baseado em elementos SRAM, está soldado na placa em um encapsulamento de 240 pinos e tem 1152 elementos lógicos e 6 blocos de arrays com uma capacidade de 20.000 gates. Dos recursos disponívies na placa para este dispositivo serão utilizados: o barramento de expansão FLEX_EXPAN_A; os Push Butons FLEX_PB1 e FLEX_PB2; uma chave FLEX_SWITCH e os displays de 7 segmentos FLEX_DIGIT.

A Tabela 7 descreve os sinais do CIMV associados a pinagem do FLEX_EXPAN_A e o componente EPF10K10. Na Tabela 8 é descrita a associação dos sinais do CIMV aos Push Butons e a chave Switch. A Tabela 9 demonstra a distribuição dos sinais nos Displays FLEX_DIGIT. A Tabelas 10 descreve a distribuição dos sinais que

saem do FLEX_EXPAND_A e vão para os Prototyping Headers P2 e P3. A Tabela 11 demonstra a associação dos sinais que saem do componente EPM7128S e alimentam os Displays MAX_DIGIT [29].

FLEX_SPAN_A						
Hole number	Signal/pin	Sinal/CIMV		Hole number	Signal/pin	Sinal/CIMV
19	50	Reset		20	51	Baixa_vel_led
21	53	Vmax_Led		22	54	A_sa
23	55	A_c		24	56	B_sa
25	61	B_c		26	62	C_sa
27	63	C_c		28	64	D_sa
29	65	D_c		30	66	E_sa
31	67	E_c		32	68	F_sa
33	70	F_c		34	71	G_sa
35	72	G_c		36	73	Baixa_vel
37	74	Clk_1200		38	75	VC
39	76	Vmáx		40	78	BIP
41	79	SS		42	80	VR

Tabela 7: Sinais do CIMV associados a pinagem do FLEX_EXPAN_A e o componente EPF10K10.

FLEX_PB – Push-Buttons				FLEX_SW1 – Switches		
FLEX_PB1	28	Set_vel		FLEX_Switch-1	41	Ini_Bip
`FLEX_PB2	29	SA				

Tabela 8: Associação dos sinais do CIMV aos Push Butons e a chave Switch.

FLEX_DIGIT					
Display Segment	Pin for Digit 1	Sinal/CIMV		Pin for Digit 2	Sinal/CIMV
A	6	A_d		17	A
B	7	B_d		18	B
C	8	C_d		19	C
D	9	D_d		20	D
E	11	E_d		21	E
F	12	F_d		23	F
G	12	G_d		24	G
Decimal point	14				

Tabela 9: Distribuição dos sinais nos Displays FLEX_DIGIT.

Prototyping Header							
P2				P3			
Outside	Signal/By_Pass	Inside	Signal/By_Pass	Outside	Signal/By_Pass	Inside	Signal/By_Pass
12	A_c	15	A_sa	33	G_c	36	E_sa
16	B_c	23	B_sa			40	F_sa
20	C_c	27	C_sa			44	G_sa
24	D_c	31	D_sa				
28	E_c						
30	F_c						

Tabela 10: Distribuição dos sinais do FLEX_EXPAND_A e vão para os Prototyping Headers P2 e P3.

MAX_DIGIT					
Display Segment	Pin for Digit 1	Sinal/By_Pass		Pin for Digit 2	Sinal/By_Pass
A	58	A_58		69	A_69
B	60	B_60		70	B_70
C	61	C_61		73	C_73
D	63	D_63		74	D_74
E	64	E_64		76	E_76
F	65	F_65		75	F_75
G	67	G_67		77	G_77
Decimal point	68			79	

Tabela 11: Associação dos sinais que saem do componente EPM7128S e alimentam os Displays MAX_DIGIT.

Estando todos os sinais associados aos respectivos componentes de entrada e saída da placa, o passo seguinte é a configuração dos componentes. A Figura 67 ilustra a configuração do circuito CIMV no componente da família FLEX 10K.

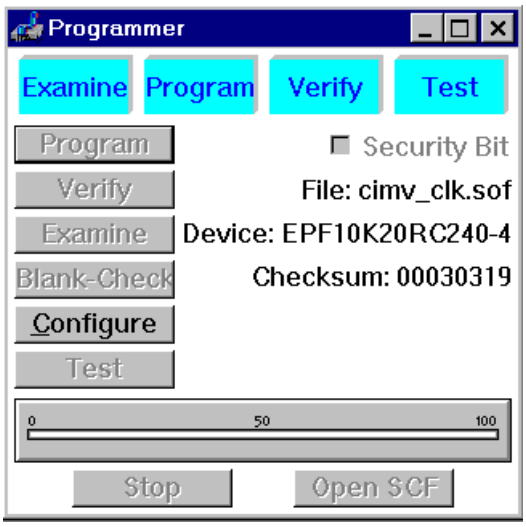


Figura 67: Configuração do circuito CIMV no componente da família FLEX 10K [30].

Estando os circuitos By_Pass e CIMV programados e configurados, dá-se inicio aos testes elétricos funcionais.

3.4 - Teste Elétrico Funcional

Os testes elétricos foram realizados utilizando, além da placa de desenvolvimento educacional UP1 configurada, o gerador de freqüência HP 3314A – Function Generator e o osciloscópio Tektronix TDS 3054 500 MHz. A Figura 68 ilustra o ambiente de testes.

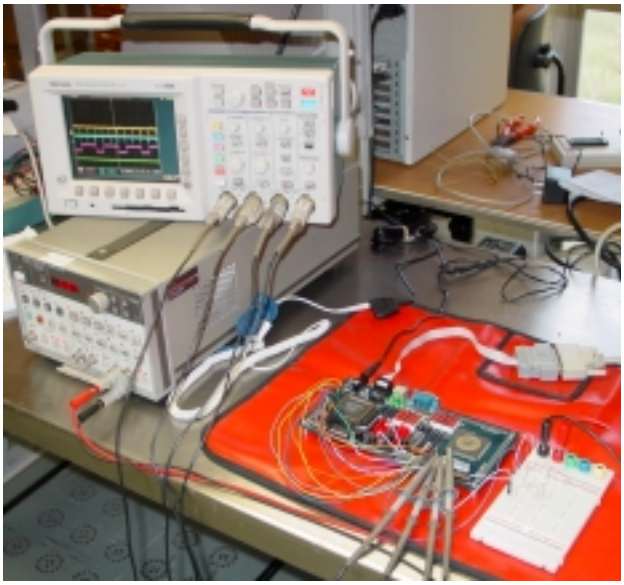


Figura 68: Ambiente de testes

A freqüência escolhida para a realização dos testes funcionais foi a de 60 Hz. Estando a placa alimentada e os componentes configurados, o *push button* MAX_PB2 que aciona o sinal *reset* é apertado e o circuito CIMV fica em estado conhecido.

A freqüência é aumentada até 61 Hz, o FLEX_PB2 é apertado para acionar o sinal SET_VEL. Estando o circuito com a freqüência de 61Hz armazenada é possível realizar todos os testes funcionais desejados, pois, reduzindo a freqüência para aproximadamente 15 Hz o circuito fica com ¼ da velocidade de referência e o sinal Baixa_vel é acionado. Com a freqüência em 60 Hz é possível verificar o sinal amostrado VC com maior facilidade pelo osciloscópio.

Quando a freqüência está entre 16 e 60 Hz, simula-se a velocidade de cruzeiro. E como esperado nada, acontece com os sinais de advertência. Quando a freqüência chega novamente a 61 Hz, ou mais, o sinal de velocidade máxima é acionado e o LED que simula a sirene do sinal BIP começa a piscar na freqüência especificada. As figuras que seguem ilustram a passagem da freqüência pelas varias situações especificadas e comprovam os resultados esperados na simulação global.

A Figura 69 ilustra o sinal SS de entrada em 60 Hz no canal 1, do osciloscópio, e o sinal amostrado VC no canal 2 com uma freqüência de 5.5 Hz aproximadamente 6 Hz resultado dos 10 pulsos de SS amostrados.

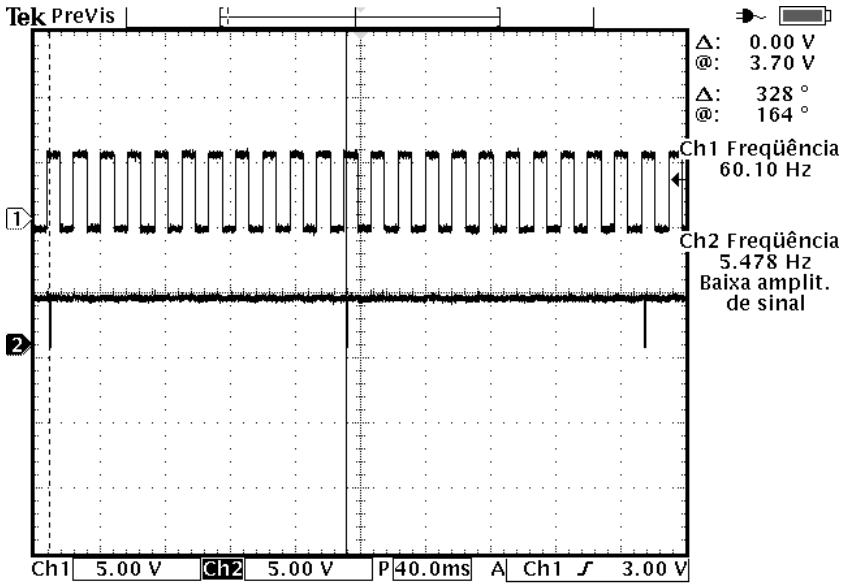


Figura 69: Sinal SS no canal 1 e sinal VC amostrado em 10 pulsos de SS no canal 2.

A Figura 70 ilustra o funcionamento do circuito quando a velocidade SS é igual ou menor que ¼ da velocidade limite especificada. No canal 1 do osciloscópio está o sinal

amostrado VC, com um período quatro vezes maior que VR; no canal 2 o sinal de referencia VR; no canal 3 está o sinal de velocidade máxima e no canal 4 está o sinal Baixa_Vel que recebe nível lógico alto segundo a especificação.

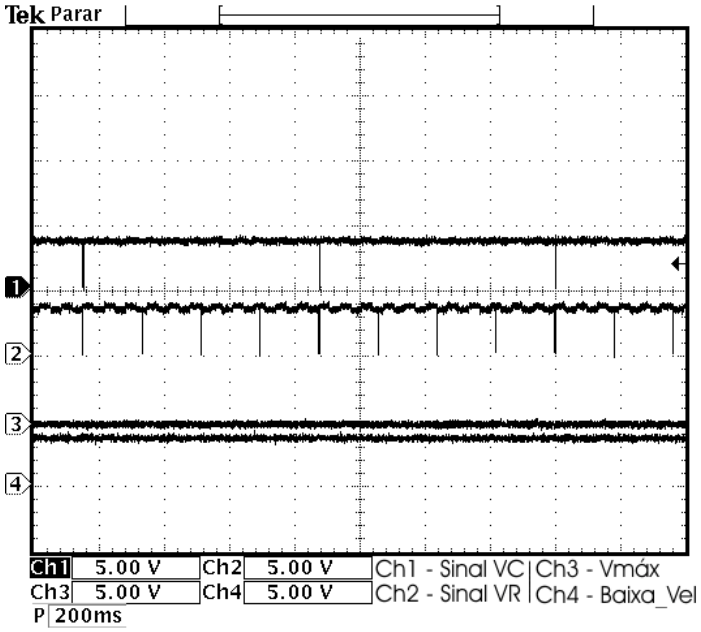


Figura 70: Teste do sinal Baixa_Vel, o canal 1 é o sinal VC, o canal 2 o sinal VR, o canal 3 o sinal Vmáx e o canal 4 o sinal Baixa_Vel.

A Figura 71 ilustra o funcionamento do circuito quando a freqüência está entre os valores de referencia máximos e mínimos, aproximadamente 30 Hz. A Figura 72 ilustra o funcionamento do circuito quando a freqüência está no limiar da velocidade máxima. A seqüência dos canais do osciloscópio utilizada é a mesma que a utilizada para a Figura 70.

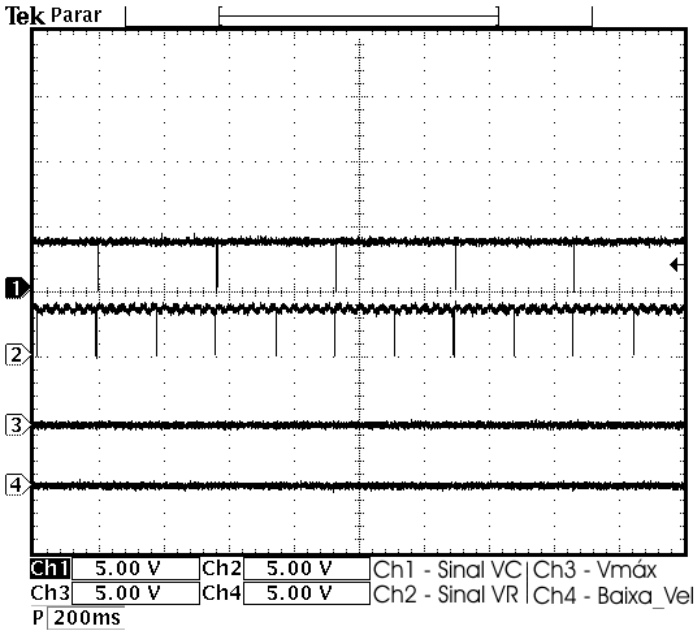


Figura 71: Teste de velocidade de cruzeiro com SS aproximadamente em 30Hz, o canal 1 é o sinal VC, o canal 2 o sinal VR, o canal 3 o sinal Vmáx e o canal 4 o sinal Baixa_Vel.

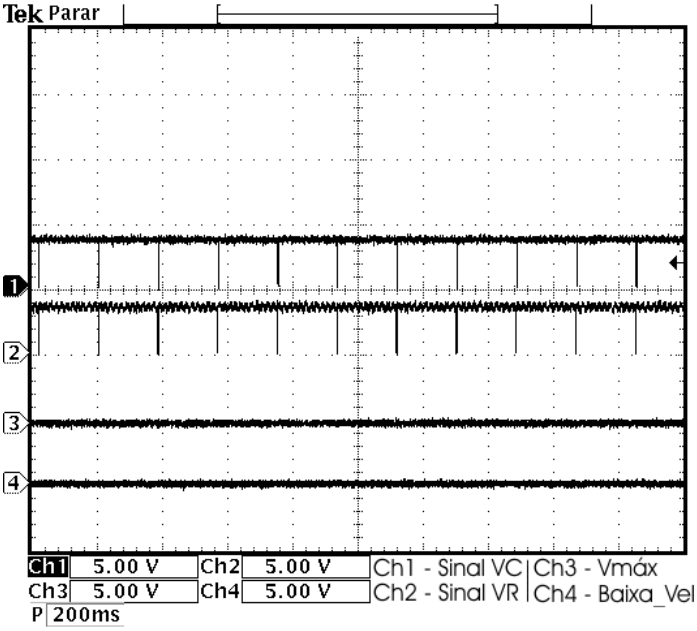


Figura 72: Teste de velocidade de cruzeiro com VC no limiar da velocidade de referencia VR, o canal 1 é o sinal VC, o canal 2 o sinal VR, o canal 3 o sinal Vmáx e o canal 4 o sinal Baixa_Vel.

A Figura 73 ilustra o funcionamento do circuito quando a freqüência é superior a velocidade limite especificada VR. Percebe-se que o contador do bloco2 é reinicializado sincronamente pelo sinal de VC. Deste modo, o sinal VR é continuo e o sinal Vmáx fica em nível lógico alto. Resultado esperado decorrente das simulações.

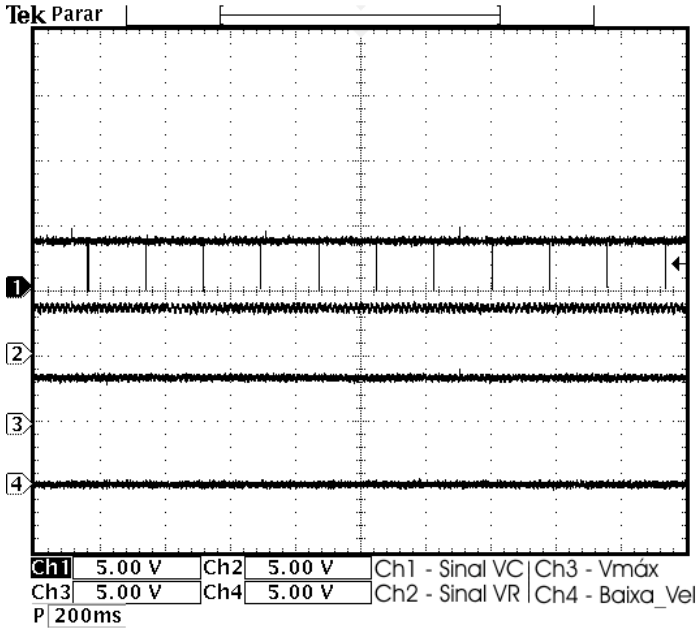


Figura 73: Teste da velocidade máxima de referencia: o canal 1 é o sinal VC, o canal 2 o sinal VR, o canal 3 o sinal Vmáx e o canal 4 o sinal Baixa_Vel.

Com a análise destas curvas é finalizado o teste funcional do CIMV, sendo a sua funcionalidade validada segundo a especificação.

A seguir serão analisados sinais internos ao circuito como: clk_1200, Clk_25M, o sinal BIP, Clk_2400, Clk_2hz, INI_BIP e os sinais VC e VR, para que o CIMV possa ser considerado como tendo atendido a todas as especificações iniciais.

A Figura 74 ilustra o sinal CLK_1200, na forma de pulsos simples, de freqüência de 25,175 MHz, justamente como o especificado. Este resultado de teste valida o circuito GERA_CLK.

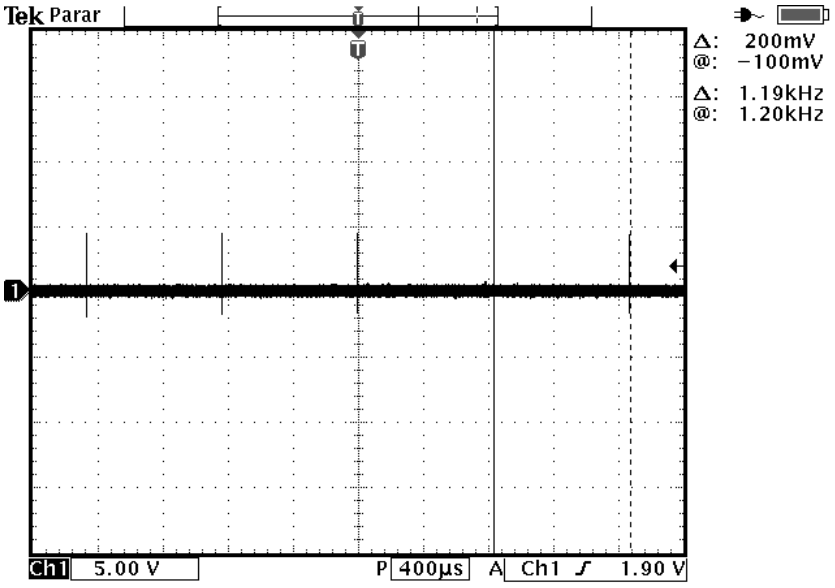


Figura 74: Sinal de relógio de 1200 Hz validando o sinal clk_1200.

A Figura 75 ilustra o sinal de pulso simples de 25,175 MHz que está contido no sinal clk_1200. A freqüência de 25,175 MHz é gerada pelo cristal oscilador da placa de desenvolvimento estudantil UP1. A saída do oscilador entra no *clock* global do dispositivo EPM7128S no pino 83 e no dispositivo EPF10K20 no pino 93, e é utilizado como *clock* de sincronismo de Flip-Flops dos quatro blocos principais do CIMV.

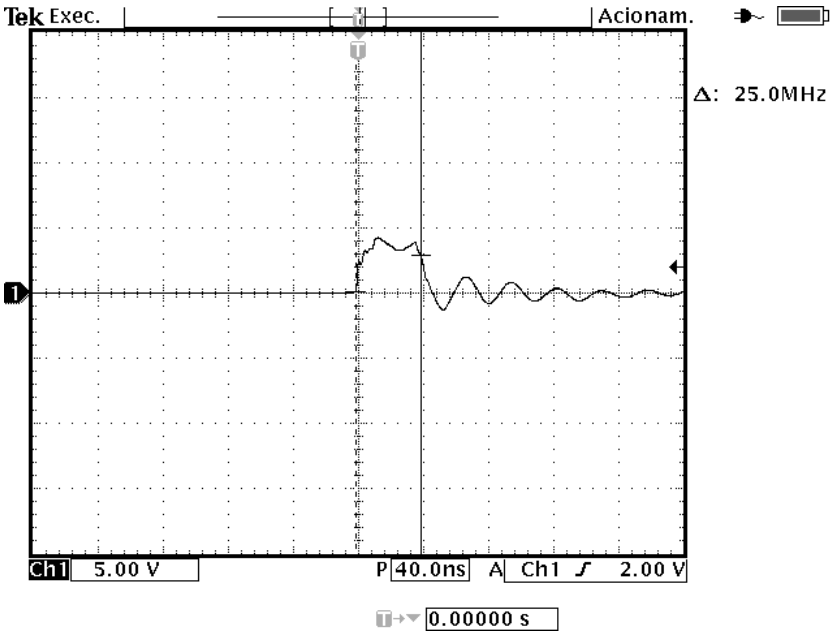


Figura 75: Sinal de Pulso simples de freqüência 25,175 MHz.

Seguindo com a analise dos sinais internos do CIMV, o sinal BIP é analisado e com isto pode-se verificar se os sinais Clk_2400, Clk_2Hz e INI_BIP estão compatíveis com a especificação. A Figura 76 ilustra o sinal BIP quando a velocidade limite é atingida. Com o sinal Vmáx em nível lógico 1 (canal 2 do osciloscópio) verifica-se um sinal intermitente composto de uma freqüência de 2400Hz modulada por uma de 2Hz.

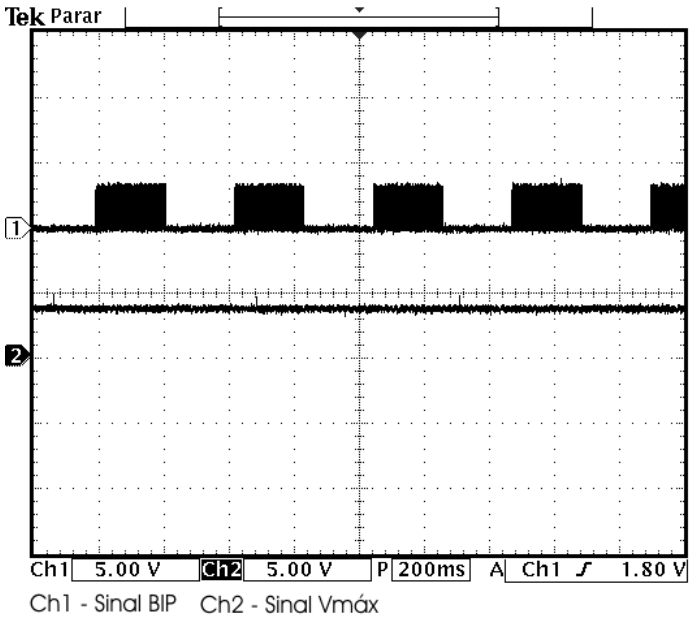


Figura 76: Sinal composto de uma freqüência de 2400Hz modulada por uma de 2Hz no canal 1 do osciloscópio e o sinal Vmáx em nível lógico 1 no canal 2.

A Figura 77 ilustra a freqüência de 2400Hz e a Figura 78 ilustra a freqüência moduladora de 2Hz. Estas freqüências vem do bloco GERA_CLK, o que mais uma vez confirma sua especificação, juntamente com o bloco 4 responsável pelo sinal BIP.

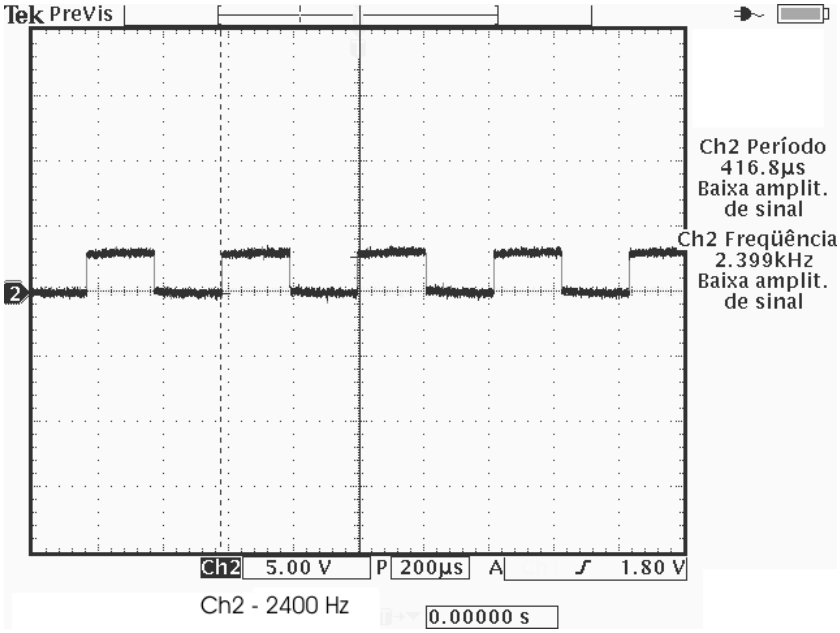


Figura 77: Freqüência de 2400Hz do sinal BIP.

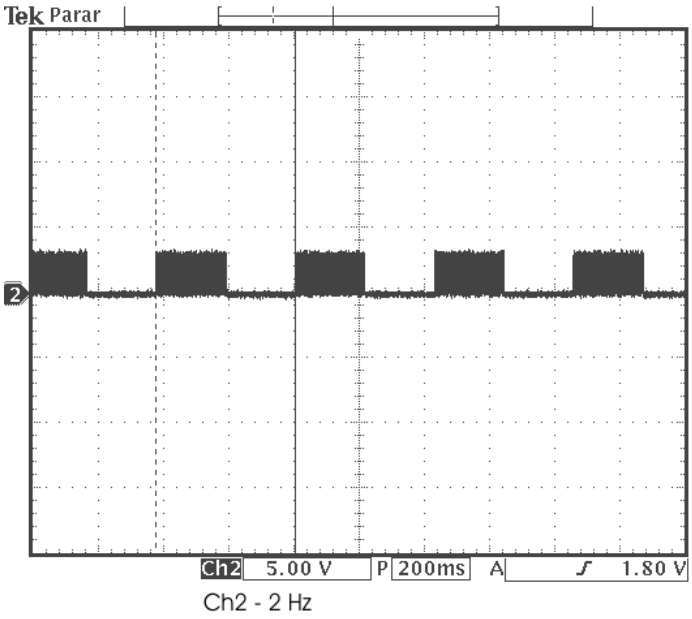


Figura 78: Sinal modulador de 2Hz do sinal BIP.

A Figura 79 ilustra o sinal BIP quando a chave seletora INI_BIP está acionada. Verifica-se que mesmo com o sinal Vmáx em nível lógico 1, o sinal BIP permanece em nível lógico 0.

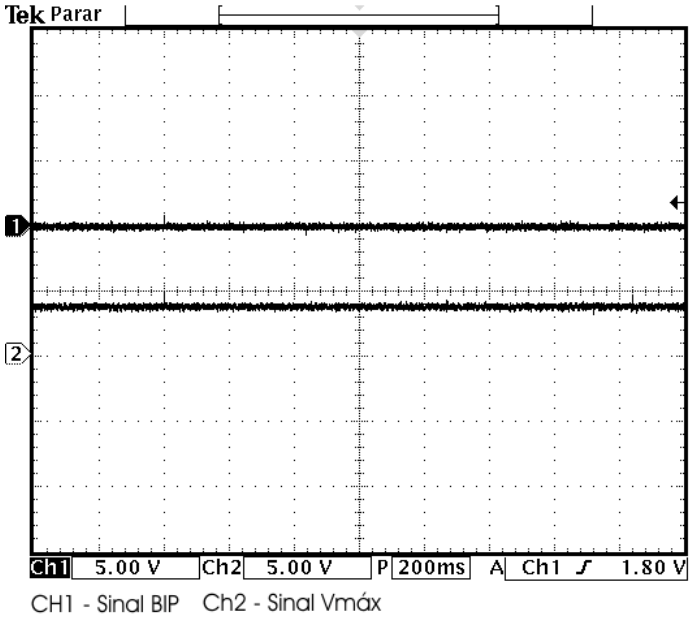


Figura 79: Sinal Bip em nível lógico 0, quando a chave seletora INI_BIP está acionada, no canal 1 do osciloscópio, mesmo quando Vmáx está em nível lógico 1, no Canal 2.

Finalizando, os sinais VC e VR serão analisados para confirmar suas especificações e as simulações realizadas. A Figura 80 ilustra os sinais SS, VC, VR e Vmáx, respectivamente nos canais 1, 2, 3 e 4 do osciloscópio. A freqüência SS está em 60 Hz e a freqüência de referência é 61 Hz. Verifica-se que o período dos pulsos da freqüência amostrada VC é maior que VR. O Contador do bloco2 é reinicializado sincronamente pelo sinal VR quando este atinge o valor de referência armazenado pelo usuário, e não pelo valor de VC. Isto significa que a velocidade ainda não atingiu a velocidade limite e o veiculo esta em velocidade de cruzeiro. Afirmação esta verdadeira, se o sinal Vmáx está em nível lógico 0.

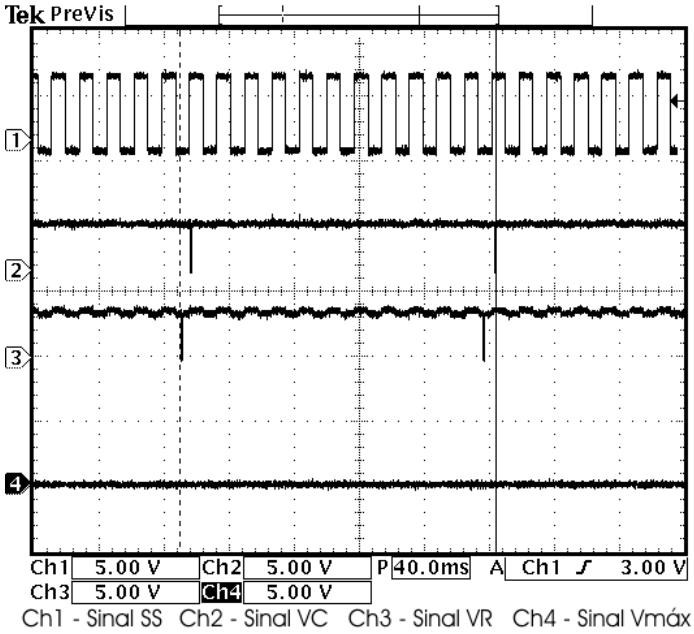


Figura 80: Sinal SS com freqüência menor que a de referencia, Vmáx em nível lógico 0.

A Figura 81 ilustra a situação quando o sinal SS é igual ao sinal de referência escolhido pelo usuário. O sinal VC é igual ao sinal VR. Nesta situação, o sinal Vmáx fica intermitente, pois ora o circuito está na velocidade máxima, ora ele está pouco abaixo. Mesmo intermitente, o sinal Vmáx executa sua função, que é a de sinalizar a velocidade máxima atingida.

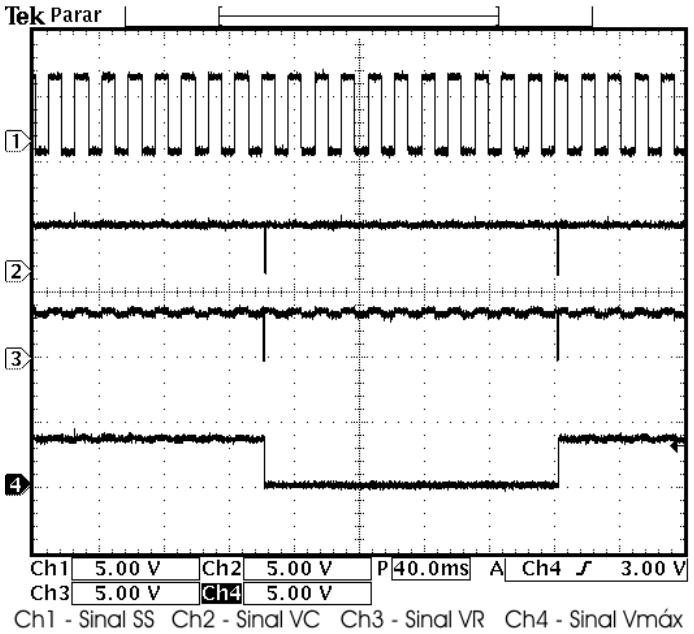


Figura 81: Sinal VC igual ao sinal VR, sinal Vmáx intermitente.

A Figura 82 mostra o sinal SS quando a velocidade é superior a velocidade de referência. Nesta situação o contador do Bloco2 é reinicializado sincronamente pelo sinal VC, e o sinal VR fica permanentemente em nível lógico 1. Sendo o contador reinicializado pelo sinal VC, significa que a velocidade real é maior que a de referência. Assim, a lógica utilizada no bloco3 é validada, pois o sinal Vmáx recebe nível lógico 1.

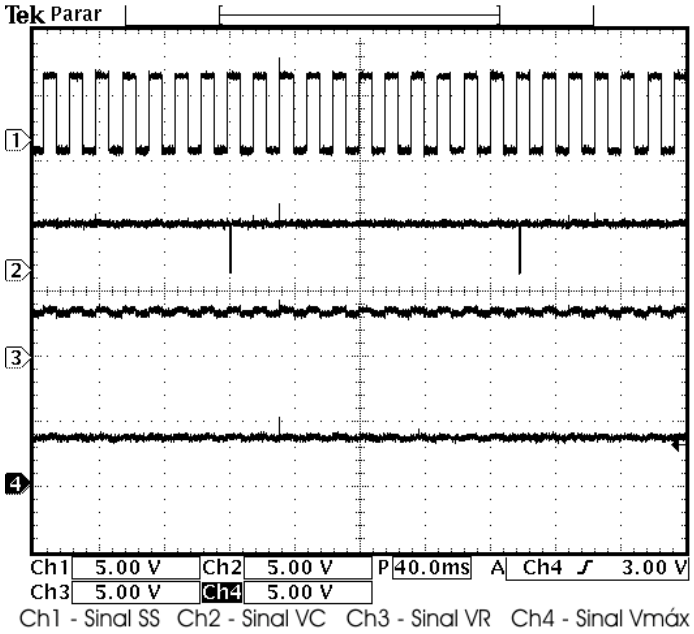


Figura 82: Sinal SS maior que a velocidade de referência, VC reinicializa o contador do bloco2, veículo com velocidade acima da especificada.

Com este ultimo teste pode-se considerar que o protótipo do CIMV atende as especificações do projeto. A Figura 83 ilustra o funcionamento da placa de desenvolvimento quando o sinal SS é maior que a velocidade de referência. O LED de sinalização de velocidade máxima esta aceso e o LED no *protoboard*, que simula uma sirene para o sinal BIP, pisca na freqüência de 2Hz. A Figura 84 ilustra o ambiente de teste com todos os equipamentos utilizados: um computador pessoal, a placa de desenvolvimento, o gerador de freqüência e o osciloscópio.

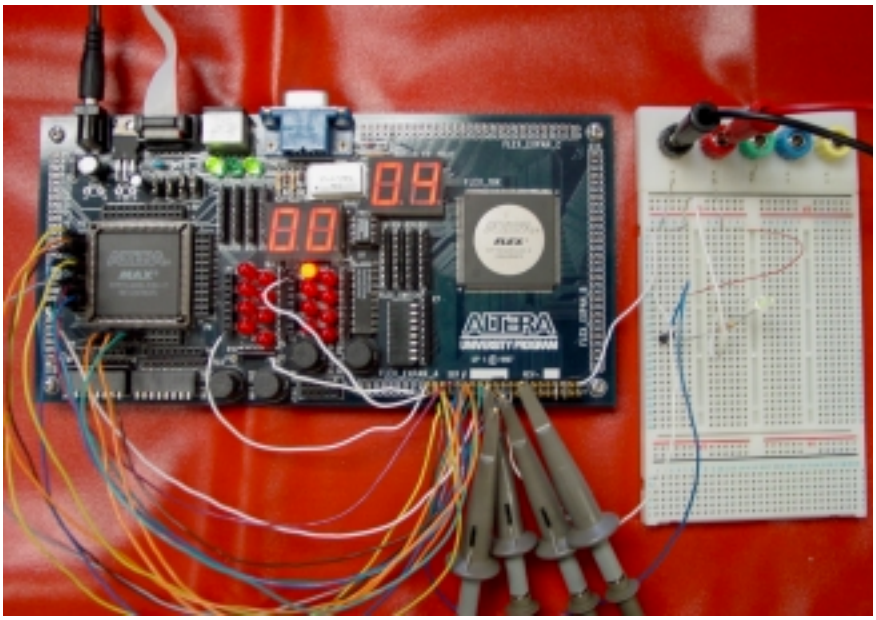


Figura 83: Placa de desenvolvimento com a freqüência SS superior a de referência. Velocidade máxima atingida.

Figura 84: Ambiente de teste com todos os equipamentos utilizados

Seguindo o fluxo de projetos digitais, após a validação do protótipo o produto estaria pronto para ser fabricado e comercializado. Supondo que o volume comercializado fosse suficientemente alto, que justificasse sua produção em ASIC para redução de custos, deveria ser realizado o layout do circuito integrado, sua verificação física e envio para a fabricação. O Capítulo 4 descreve estas etapas.

Capítulo 4 – Layout e Verificação Física

4.1 – Introdução

Após realizados os testes elétricos funcionais a última etapa antes da fabricação de um circuito integrado é o layout. No nosso caso, o envio para a fabricação não se faz necessário, devido ao alto custo envolvido e pela finalidade deste projeto de tese que é percorrer o fluxo completo de projetos digitais.

Para a realização do layout foi escolhida a tecnologia CMOS 0.6 µm da Austriam Micro Systems – AMS e as ferramentas utilizadas serão: Design Architect, Qhsim e IC Station V. 8.6.2.1, Mentor Graphics.

Os esquemáticos sintetizados pelo Leonardo Spectrum foram copiados e capturados no Design Architect utilizando as células básicas correspondentes a tecnologia escolhida. Capturados os esquemáticos, uma simulação global para confirmar a funcionalidade do CIMV deve ser realizada. Confirmada esta simulação, o layout pode ser efetuado. Como descrito em Capítulos anteriores o layout pode ser manual ou automático. Neste caso pode-se dizer que será feito um layout semi-automático, pois o *core* será gerado automaticamente, mas as ligações no anel de *pads* serão feitas manualmente.

Terminado o layout, as verificações físicas típicas DRC e LVS serão realizadas. Eliminados os erros de verificação física, completa-se o fluxo de projetos digitais. As ferramentas EDA escolhidas foram as disponíveis para utilizar na Divisão de Concepção de Sistemas de Hardware (DCSH) do Centro de Pesquisas Renato Archer (CenPRA).

4.2 – Captura de Esquemáticos e Simulação

A captura dos esquemáticos foi feita copiando os circuitos sintetizados pelo Leonardo Spectrum. Todos os circuitos foram capturados na forma de esquemáticos e a Figura 85 ilustra o CIMV em diagrama esquemático, semelhante ao que foi capturado no HDL Design Series e sintetisado no Leonardo Spectrum.

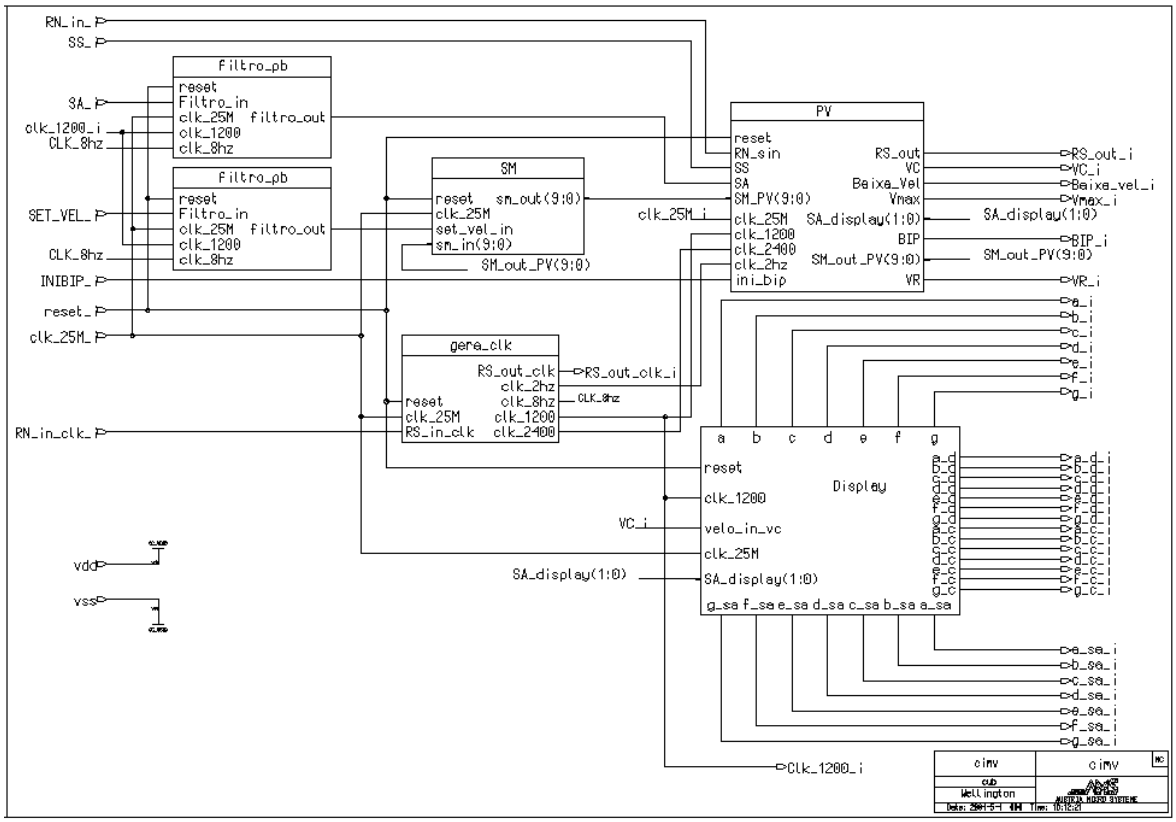


Figura 85: CIMV em diagrama esquemático.

A Figura 86 ilustra o CIMV com o Anel de *Pads.* Este circuito é muito importante para a realização da verificação física e para a simulação do circuito.

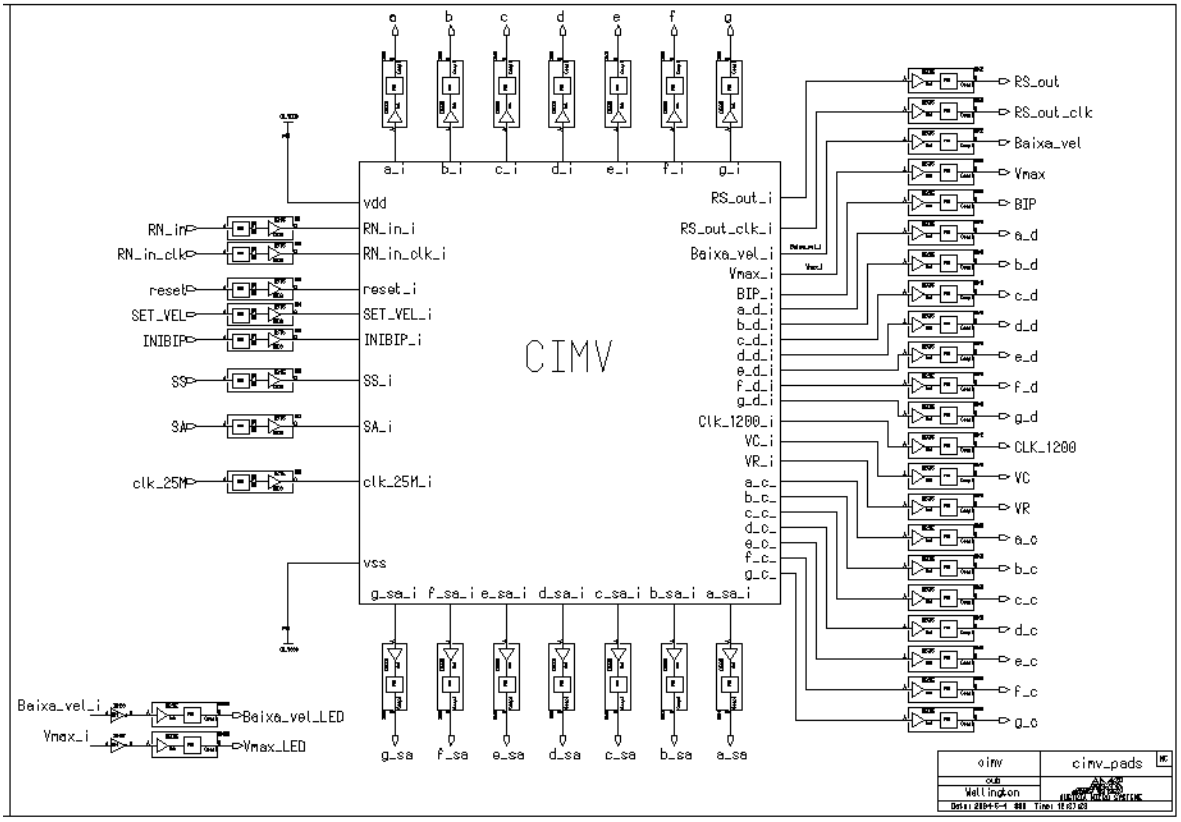


Figura 86: CIMV com o Anel de Pads.

A Figura 87 ilustra uma simulação do CIMV com o Anel de *Pads*. Deste modo é possível verificar se há influência dos circuitos de proteção dos *pads* no funcionamento do CIMV.

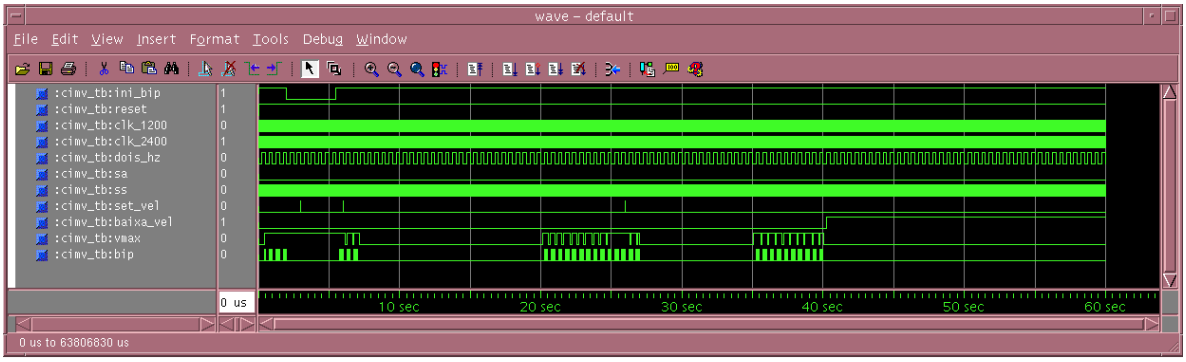


Figura 87: Simulação do CIMV com o Anel de *Pads*.

4.3 – Layout

O layout foi realizado de maneira semi-automática, gerando o *core* das células digitais automaticamente e fazendo sua ligação no anel de *pads* de maneira manual. A

Figura 88 ilustra o *core* do CIMV gerado automaticamente e a Figura 89 ilustra o CIMV com suas ligações no chip total: anel de *pads* e *core* interligados.

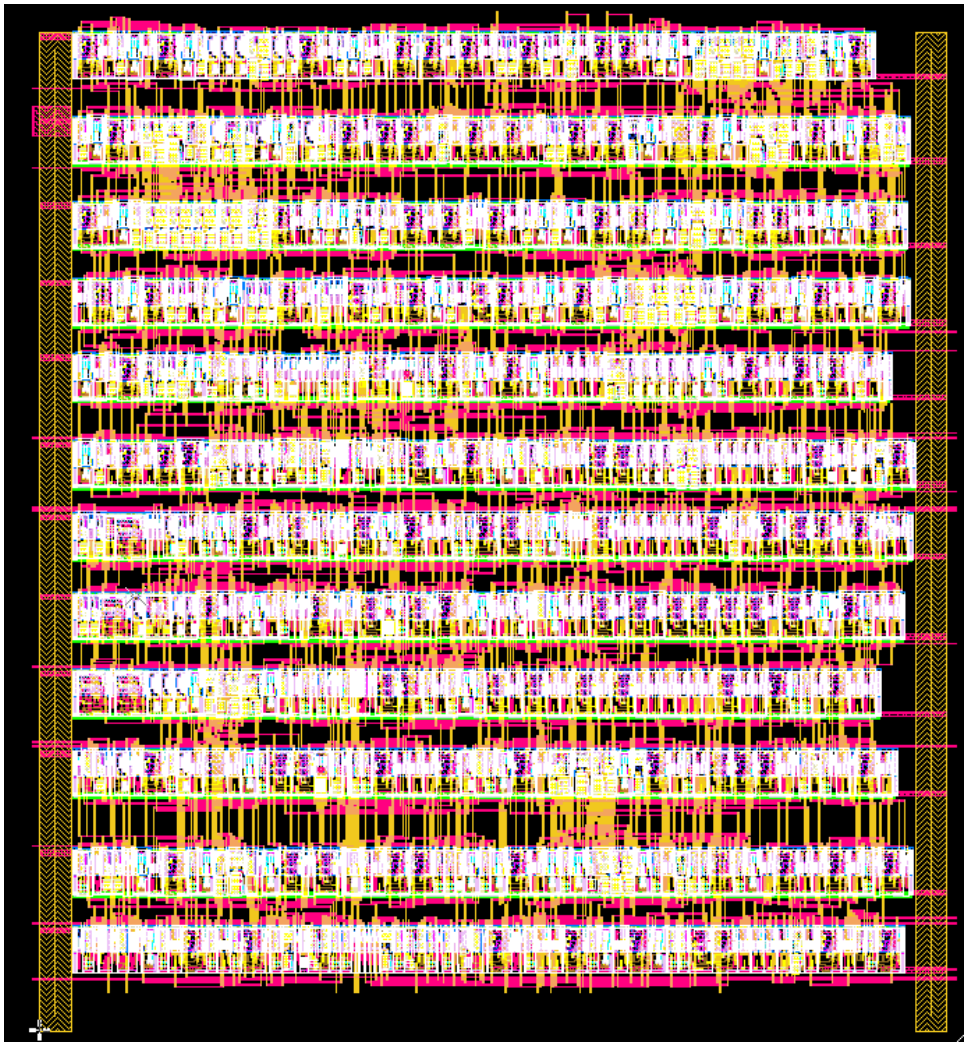


Figura 88: Core do CIMV gerado automaticamente.

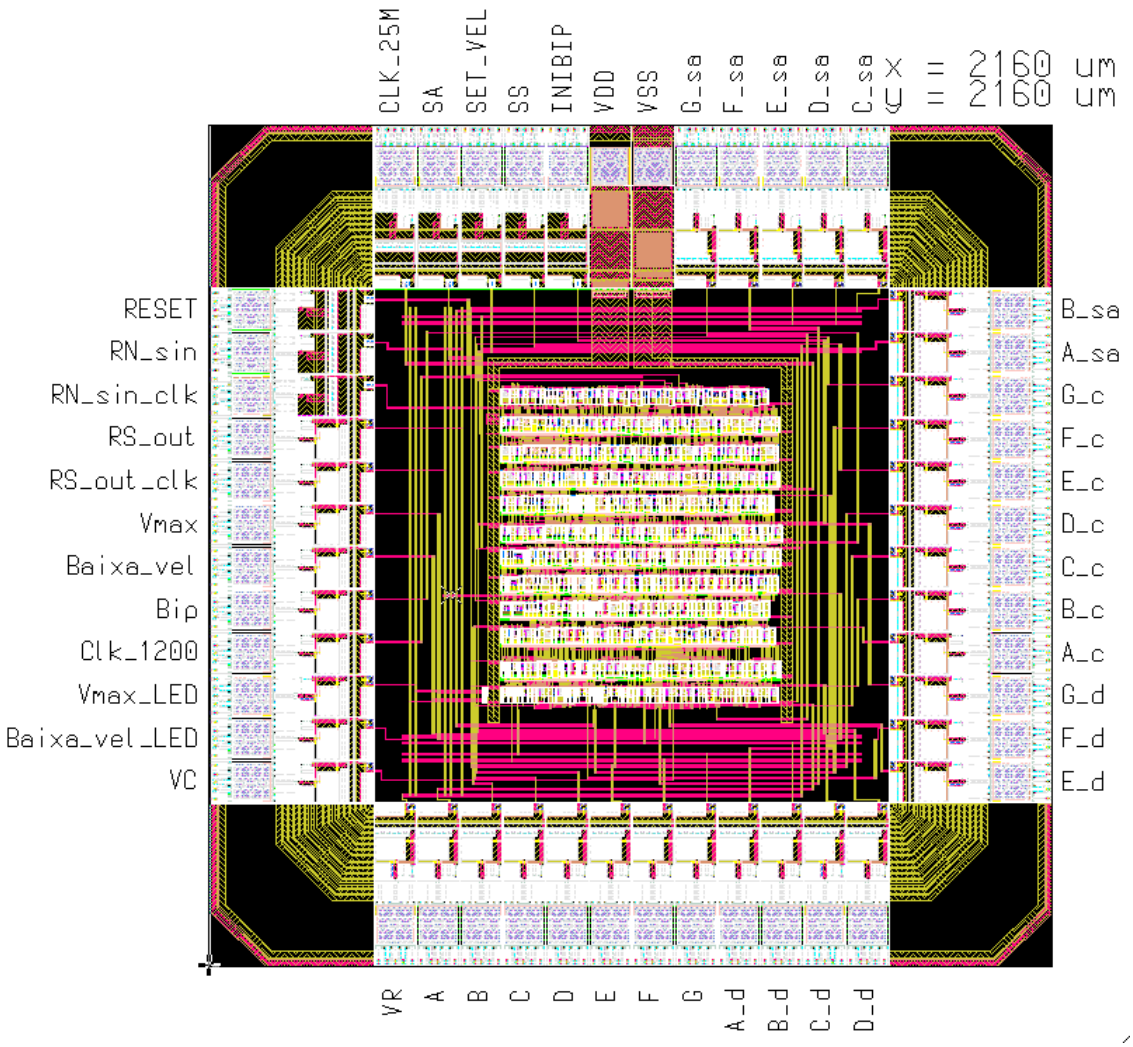


Figura 89: CIMV com anel de *pads* e core interligados.

4.4 – Verificação Física

Terminado o layout, a verificação do DRC e LVS foram realizadas. Normalmente quando se está fazendo um layout manual, estas verificações são realizadas em blocos individuais à medida que vão sendo finalizados para evitar a propagação de erros em hierarquias superiores. Como o núcleo digital (*core*) foi feito automaticamente à incidência de erros de DRC e LVS é reduzida. Isto facilitou a realização das verificações sem perder muito tempo corrigindo erros. A parte que ficou destinada ao layout manual foi a ligação do núcleo digital aos *pads*. A incidência de erros de verificação também não foi muito alta devido à pequena quantidade de ligações para os *pads*. A Figura 90 ilustra o resultado da verificação física DRC e LVS com o circuito pronto para ser gerado o arquivo de intercâmbio GDSII.

```
*****************************************************************************************************************************
                                    OVERALL COMPARISON RESULTS
*****************************************************************************************************************************

                                  ####################
                                  #                  #       _   _
                                  #                  #       *   *
                                  #     CORRECT      #         |
                                  #                  #       \___/
                                  ####################

-----------------------------------------------------------------------------------------------------------------------------
```

Figura 90: Resultado da verificação física DRC e LVS.

A Figura 91 ilustra o resultado do LVS com suas informações e estatísticas a nível de transistores. A Figura 92 ilustra o resultado do LVS com suas informações e estatísticas a nível de portas lógicas.

```
*****************************************************************************************************************************
                                    INFORMATION AND WARNINGS
*****************************************************************************************************************************

                  Matched     Matched     Unmatched     Unmatched     Component
                   Layout      Source       Layout        Source     Type
                  -------     -------     ---------     ---------    ---------
   Ports:           48          48           0             0
   Nets:          4788        4788           0             0
   Instances:     4273        4273           0             0        mn (nmos4)
                    38          38           0             0        lddn (nmosh)
                  4405        4405           0             0        mp (pmos4)
                    13          13           0             0        r (rdiffp3)
                    59          59           0             0        d (nd)
                    48          48           0             0        d (nwd)
                    56          56           0             0        d (pd)
                  -------     -------     ---------     ---------
   Total Inst:    8892        8892           0             0

o Statistics:

   4 isolated layout nets were deleted.

   1000 layout mos transistors were reduced to 272.
     728 mos transistors were deleted by parallel reduction.
   8 source mos transistors were reduced to 4.
     4 mos transistors were deleted by parallel reduction.

   1962 parallel layout diodes were reduced to 142.
   184 parallel source diodes were reduced to 4.


o Isolated Layout Nets:

      (Layout nets which are not connected to any instances or ports).

   861(1745.500,2102.000) 862(1855.500,2102.000) 863(1965.500,2102.000) 876(2185.500,2102.000)


o Initial Correspondence Points:

   Ports:        VDD VSS clk_25M SA SET_VEL INIBIP g_sa f_sa e_sa d_sa c_sa b_sa a_sa g_c f_c
                 e_c d_c c_c b_c a_c f_d e_d d_d c_d b_d a_d g f e d c b a VR VC Baixa_vel_LED
                 Vmax_LED CLK_1200 BIP Baixa_vel Vmax RS_out_clk RS_out RN_in_clk RN_in reset
```

Figura 91: Resultado do LVS com informações e estatísticas a nível de transistores.

```
********************************************************************************
                          INFORMATION AND WARNINGS
********************************************************************************

                 Matched    Matched    Unmatched   Unmatched   Component
                  Layout     Source      Layout      Source     Type
                 -------    -------    ---------   ---------   ---------
   Ports:          48         48          0           0
   Nets:         2285       2285          0           0
   Instances:      38         38          0           0        lddn (nmosh)
                   42         42          0           0        mp (pmos4)
                   13         13          0           0        r (rdiffp3)
                   59         59          0           0        d (nd)
                   48         48          0           0        d (nwd)
                   56         56          0           0        d (pd)
                 1252       1252          0           0        INV
                  272        272          0           0        NAND2
                   27         27          0           0        NAND3
                   49         49          0           0        NAND4
                    5          5          0           0        NAND5
                  143        143          0           0        NOR2
                   13         13          0           0        NOR3
                    3          3          0           0        NOR4
                   62         62          0           0        AOI_2_1
                   38         38          0           0        OAI_2_1
                  622        622          0           0        SDW2
                   98         98          0           0        SDW3
                  814        814          0           0        SUP2
                 -------    -------    ---------   ---------
   Total Inst:   3654       3654          0           0

o Statistics:

   4 isolated layout nets were deleted.

   1000 layout mos transistors were reduced to 272.
     728 mos transistors were deleted by parallel reduction.
   8 source mos transistors were reduced to 4.
     4 mos transistors were deleted by parallel reduction.

   1962 parallel layout diodes were reduced to 142.
   184 parallel source diodes were reduced to 4.
```

Figura 92: Resultado do LVS com informações e estatísticas a nível de portas lógicas.

Com o layout terminado e sem erros de DRC e LVS, o circuito CIMV está pronto para ser enviado para a fabricação. Neste ponto encerramos o fluxo de projetos proposto.

Conclusões

A proposta deste trabalho de dissertação foi percorrer as etapas do fluxo de projeto digitais, utilizando como veículo a especificação de um circuito integrado de aplicação especifica monitor de velocidade.

Neste contexto, foi realizada uma pesquisa do que há disponível em termos de técnicas de desenvolvimento e ferramentas de suporte que auxiliem e facilitem chegar ao final do fluxo com resultado positivo, diminuindo o esforço, aumentando o rendimento, a confiabilidade e reduzindo o NRE.

A especificação deste projeto pode ser considerada simples, uma vez que não era esse o interesse, mas sim percorrer o fluxo de projetos digitais.

As ferramentas utilizadas foram as disponíveis na Divisão de Concepção de Sistemas de Hardware (DCSH) do Centro de Pesquisa Renato Archer (CenPRA). Ferramentas da Mentor Graphics: FPGA Advantage for HDL Design, Design Archteti, QSim e IC Station; e Ferramentas da Altera: Max + PLUS II e a placa desenvolvimento educacional UP1.

O Fluxo foi percorrido, especificando-se o projeto e descrevendo os circuitos utilizando as diferentes facilidades que as ferramentas proporcionam: captura de esquemáticos, tabela verdade, máquina de estados e finalmente descrição em linguagem HDL (no caso deste projeto à linguagem escolhida foi VHDL). Devido à ferramenta FPGA Advantage ser baseada em HDL´s, independente da descrição utilizada, após a compilação o arquivo base gerado é uma descrição em VHDL.

O projeto foi descrito em vários blocos e todos os blocos foram simulados e atenderam a especificação inicial. Uma vez que a especificação inicial foi validada pela simulação, o passo seguinte no fluxo adotado foi à síntese lógica. A síntese foi realizada para os componentes FPGA: EPM7128SLC84-7 e EPF10K20RC240-4, ambos integrantes da placa de desenvolvimento educacional UP1. A síntese gerou arquivos de transporte *edif* que foram exportados para o software Max + PLUS II. Estes arquivos foram compilados e configurados e/ou programados na placa UP1.

Com o circuito programado na placa UP1, os testes elétricos foram realizados e o circuito atendeu às especificações.

Com um protótipo que atende às especificações, o passo seguinte é sua produção em série. Neste sentido foi realizado o layout e a verificação física de um circuito dedicado *full custom*. Eliminando os erros da verificação física o projeto está pronto para ser enviado para a fabricação. Neste ponto encerramos o fluxo de projeto digital proposto devido aos custos envolvidos e o tempo estimado para o retorno do circuito processado.

Confrontar-se com as dificuldades inerentes a realização de um projeto, superá-las, e apresentá-las de forma didática criando um documento que possa ser útil para todos que se iniciam no projeto de CI´s digitais foi o principal objetivo deste trabalho.

Trabalhos futuros

Como trabalhos futuros podemos citar: a realização de um bloco conversor da quantidade de pulsos amostrados VC, que permita mostrar a velocidade real do veículo na saída do bloco Display.

Havendo possibilidade de financiamento de alguma agencia de fomento, a base de dados gerada pelo circuito monitor de velocidade pode ser enviada para a difusão.

Referências

1. BASS M. J., CHRISTENSEN C. M. **The Future of the Microprocessor Business.** IEEE SPECTRUM – abril 2002 – pg. 34-39;
2. LAYER D. H. **Digital Radio Takes to the Road**. – IEEE SPECTRUM – Julho 2001 – pg. 40-46;
3. MARTIN B. **Automation Comes to Analog.** IEEE SPECTRUM – Junho 2001 – pg. 70-75;
4. WAKERLY, J. F. **Digital Design Principles & Practices**. Third Edition, New Jersey – Prentice Hall – 2000 – Chapter 1, p. 1-8.
5. MAMMANA, C.I.Z.; MAMMANA, A. P. **Introdução ao Projeto de Circuitos Integrados.** Edição Preliminar, Campinas, 1987, p. 11-19.
6. WORKSHOP/HANDSON/CenPRA-HITECH. **Fluxo de Projetos Digitais com Componentes Programáveis (ASICs/FPGAs/EPLDs).** Março de 2002, não paginado.
7. MENTOR GRAPHICS. **Designing with FPGA Advantage A Mentor Graphics Training Workbook.** Education Services. Version 4.0. December 2000, não paginado.
8. REDDY, S. V. **Digital Design Flow Options**. The Ohio State University, 2001, Master Degree These, Chapter 1, p. 1-11.
9. WAKERLY, J. F. **Digital Design Principles & Practices**. Third Edition, New Jersey – Prentice Hall – 2000 – Chapter 1, p. 1-23, Chapter 4, p. 193-298.
10. BROWN, S., VRANESIC, Z. **Fundamentals of Digital logic with VHDL Design**. Boston, McGraw-Hill – 2000 – Chapter 8, p.427-507.
11. PALNITKAR, S. **Verilog HDL – A Guide to Digital Design and Synthesis**. SunSoft Press, Chapter 1 e 2, p. 3-25.
12. MENTOR GRAPHICS TRAINING. **Comprehensive VHDL Training Workbook**. November 2000 – não paginado.
13. MAZOR, S., LANGSTRAAT P. **A Guide to VHDL**. Second Edition, Kluwer Academic Publishers, Chapter 1, p. 1-15.

14. MENTOR GRAPHICS. **Leonardo Spectrum Synthesis and technology Manual**. V2001.1, não paginado
15. HUBER, J. P., ROSNECK, M. W. **Successful ASIC Design the First Time Through**. Van Nostrand Reinold, New York, Chapter 1, p. 1-15, Chapter 5, p. 77-107.
16. REDDY, S. V. **Digital Design Flow Options**. The Ohio State University, 2001, Master Degree These, Chapter 5, p. 65-116.
17. MENTOR GRAPHICS. **Icverify Manual**. V8.7-3 Chapter1 – Overview, não paginado.
18. MENTOR GRAPHICS. **Standard Verification Rule Format (SVRF) Manual**. V8.7, April 1999 – Design Rule Checking System, p. 1-20 – 1-25.
19. MENTOR GRAPHICS. **Standard Verification Rule Format (SVRF) Manual**. V8.7, April 1999 – Conectivity Extraction, p. 1-51 – 1-59.
20. AMS HIT-KIT. **Mentor User's Guide**. Version 2.40. December 1995.
21. MENTOR GRAPHICS. **Getting Started wuith FPGA Advantage Tutorial** – Software Version 5.3 – 20 June 2002.
22. MENTOR GRAPHICS. **Start Here Guide for the HDL Designer Series** – Version 2002.1 – 14 March 2002.
23. MENTOR GRAPHICS. **ModuleWare Reference Guide For The HDL Designer Series** – Version 2002.1 – 6 March 2002.
24. MENTOR GRAPHICS. **Start Here for ModelSim SE** – Version 5.6a – 23 April 2002.
25. MENTOR GRAPHICS. **ModelSim SE User's Manual** – 5.6a – 23 April 2002.
26. MENTOR GRAPHICS. **ModelSim SE Tutorial** – 5.6a – 23 April 2002.
27. MENTOR GRAPHICS. **Leonardo Spectrum User's Manual** – Software Release 2002b – April 2002.
28. MENTOR GRAPHICS. **Leonardo Spectrum Synthesis and Technology Manual** – Software 2002b – April 2002.
29. ALTERA. **Altera – University Program Design Laboratory Package – User Guide** August – 1997.
30. ALTERA. **Max + Plus II Ver 7.0 Help Contents**

Fontes de Pesquisa na Internet

31. FINIT STATE MACHINE DESIGN. – http://www.redbrick.dcu.ie/academic/CLD/chapter8/chapter08.doc.html
32. http://www.injetronic.com.br/dica02.htm
33. http://www.injetronic.com.br/dica15.htm
34. http://www.injetronic.com.br/dica03.htm
35. http://www.injetronic.com.br/dica01.htm
36. http://www.geocities.com/SiliconValley/Program/3430/velocimetro.htm
37. http://www.multazero.com.br/multa-zero/home/instalacao.htm
38. http://www.neotech.com/brazil/install.htm
39. http://www.oficinabrasil.com.br/PagMateria.asp?ID_Materia=515
40. http://www.elexp.com/t_bounc.htm
41. http://www.allaboutcircuits.com/vol_4/chpt_4/4.html