UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica

Departamento de Semicondutores, Instrumentos e Fotônica

Um Sistema Especialista para Verificação de Regras de Projeto Elétrico em Circuitos Integrados de Tecnologia CMOS VLSI.

Tulio Ibañez Nunes

Orientador: Prof. Dr. Furio Damiani

Banca examinadora:

Prof. Dr. Furio Damiani

Prof. Dr. Eduardo Moreira da Costa

Prof. Dr. Fernando Antonio Campos Gomide

Dissertação submetida como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Campinas, dezembro 1990

Este exemplar corresponde à redação final da tese defendida por Tulio Ibañez Nunes e aprovada pela Comissão Julgadora em 15/02/91.

Orientador

Conteúdo

SUMÁRIO

Este trabalho apresenta os aspectos teóricos e práticos envolvidos na construção de um sistema especialista para verificação de regras elétricas em circuitos integrados de tecnologia CMOS.

Inicialmente é feito um estudo sobre os paradigmas e técnicas de construção de sistemas especialistas. É apresentada uma arquitetura de um sistema especialista para verificação de regras. Na apresentação da arquitetura são mostrados os aspectos teóricos das regras elétricas que um circuito deve respeitar.

Por fim é apresentado um protótipo do sistema verificador de regras elétricas, implementado em PROLOG, apresentando-se os resultados alcançados.

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas que me auxiliaram, direta ou indiretamente na execução deste trabalho. Algumas destas pessoas foram imprescindíveis a este trabalho. Entre elas considero duas as mais importantes. O meu orientador, que sugeriu este assunto, conseguindo conciliar o meu trabalho de pesquisa no CPqD TELEBRÁS com esta tese e também pelas discussões que tivemos sobre o assunto. A outra pessoa importante, sem dúvida, foi o meu Pai quem sempre me estimulou e valorizou o meu trabalho e a ele dedico esta tese.

Agradeço também aos meus colegas de trabalho: Alexandre, Serginho, Pisani, Narcizo, Bernadete, Marcelo e outros pelas discussões sobre este trabalho.

Por fim agradeço ao CPqD TELEBRÁS em particular à área de circuitos integrados que forneceu os meios materiais para execução desta dissertação de mestrado.

CAPÍTULO UM

1. INTRODUÇÃO

Esta dissertação mostra os aspectos teóricos e práticos envolvidos na construção de um sistema especialista Conferidor de REgras de Projeto Elétrico: o sistema CREPE.

Este trabalho surgiu de uma necessidade encontrada durante o trabalho do projetista de circuitos integrados: a de verificar regras elétricas em projetos de células VLSI. Dadas as características do problema decidiu-se adotar uma abordagem com sistemas especialistas. A tarefa foi dividida em duas fases: a primeira, de estudos de técnicas de desenvolvimento de sistemas especialistas, e a segunda de uma aplicação destas técnicas a um sistema de verificação de regras elétricas em circuitos integrados de tecnologia CMOS VLSI. A interdependência entre estas fases existe na medida em que a primeira fase fornece subsídios metodológicos para a segunda fase onde se mostra a utilidade do uso deste tipo de programação.

1.1. Objetivos e Motivação

O projeto de circuitos integrados em VLSI segue usualmente a metodologia mostrada no diagrama de fluxo de atividades da figura 1.1.

Este trabalho procura preencher um espaço deixado pelas ferramentas de software convencionais, que não fazem a verificação de regras elétricas antes da simulação, atualmente feita pelo projetista sem o auxílio de ferramentas adequadas.

A motivação principal para a construção do CREPE foi a idéia de se automatizar a tarefa de depuração, de forma a melhorar a confiabilidade dos circuitos integrados e diminuir o tempo de projeto. Além disto, a verificação de regras elétricas é ardua quando deixada para a simulação elétrica, pois a interpretação dos resultados da simulação exige experiência e tempo do projetista.

O uso da abordagem de sistemas especialistas também foi uma motivação, na medida em que se aprenderam técnicas novas de implementação de sistemas de auxílio ao projeto de circuitos integrados, pudemos ter visto mais aprofundada a validade do uso de tais sistemas.

1.2. Os Sistemas Especialistas para Verificação de Projeto VLSI

As técnicas de inteligência artificial foram inicialmente desenvolvidas para solução de problemas genéricos e complexos, mostrando-se em geral ineficientes. Dentro deste quadro aparecem os sistemas especialistas, que procuram resolver uma classe restrita de problemas tornando-se sistemas eficientes em termos de desempenho e confiabiliadade dos resultados.

Os sistemas especialistas possuem duas partes principais, a representação da classe de problemas e a estrutura de controle.

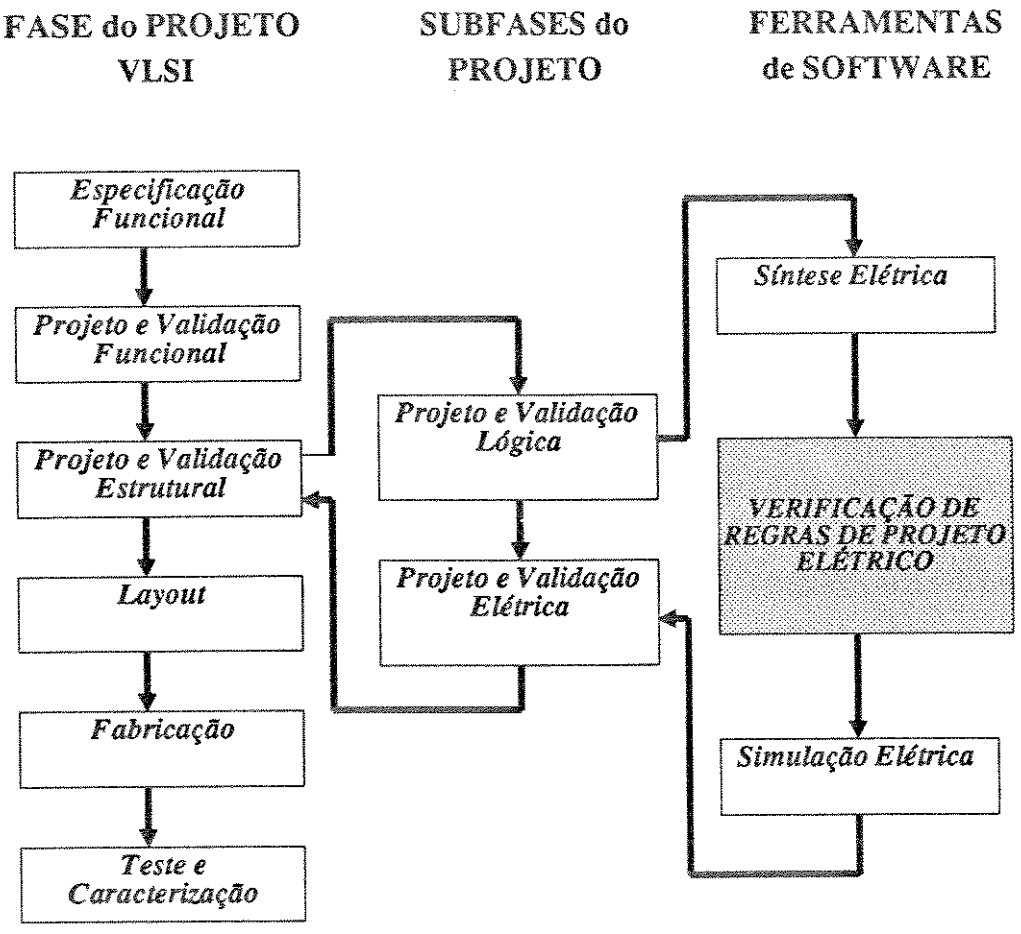


Figura 1.1 Atividades e ferramentas de software para o projeto de circuitos integrados.

A representação da classe de problemas trata da escolha apropriada de uma maneira de representar o conhecimento e o espaço do problema. As representações mais comuns são as baseadas em regras, as redes semâmticas e os " Frames ".

A estrutura de controle trata das decisões de quando e onde uma certa parte do conhecimento deve ser usada para identificar o estado do problema e definir o próximo estado em direção à solução. O tipo de estrutura de controle está relacionado com o tipo de problema que se pretende resolver. As técnicas de raciocínio para a frente (" Forward Chainning ") são normalmente usadas para tarefas de síntese e construção enquanto que tarefas de análise usam preferencialmente técnicas de raciocínio para trás (" Backward Chainning ").

Existem hoje alguns sistemas especialistas que abordam o problema da verificação do projeto de circuitos. Cada sistema tem abordagens e soluções diferentes para este tipo de problema. Abaixo estão citados alguns destes sistemas.

NCR Design Advisor [37]: É um sistema de verificação de regras de amplo espectro. Verifica regras de testabilidade, " timing ", interface e barramentos. Além disto faz análises de desempenho do circuito (Mentor Graphics).

RUBIC [22]: Sistema para verificação de regras elétricas em circuitos integrados NMOS. Analisa regras simples de conectividade e regras mais complexas de relógio, " timing ", " clock skew ", " races " e outras (Universidade de Berkeley).

CRITIC [36]: Sistema que procura erros em projetos construidos sob uma metodologia inadequada. Verifica erros do tipo conexão, distribuição de cargas, " timing " e testabilidade (Universidade de Berkeley).

DIALOG [3]: Sistema para verificação de regras de ligações lógicas e " timing " em circuitos CMOS VLSI.(Universidade Católica de Leuven).

VERIFY [13] : Sistema para verificar regras de ligações em projetos digitais com circuitos grandes (Laboratótio da FAIRCHILD).

DESIGN AUDIT [7]: Sistema para verificação genérica, nas fases de projeto funcional, projeto lógico, projeto elétrico e " layout ". Faz parte de um sistema integrado de ferramentas da ES2 para circuitos integrados VLSI (ES2).

O CREPE implementa regras para a tecnologia CMOS, assim como o RUBIC o faz para a tecnologia NMOS. A principal diferença entre o CREPE e o RUBIC é a capacidade de explanação que o CREPE possui e o RUBIC não. Os sistemas CRITIC, NCR, DIALOG, VERIFY e o DESIGN AUDIT fazem auditorias em circuitos a nível lógico, enquanto que o CREPE as faz a nível elétrico .

Com estas características o CREPE contribui com a formalização de regras de verificação elétrica para circuitos CMOS e com a implementação de explanação sobre as regras implementadas.

1.3. Visão Geral do Sistema CREPE

A aplicação básica do sistema está no projeto de células e blocos "full custom" para circuitos integrados, onde a experiência do projetista é muito importante.

O conhecimento está organizado no que se chama base de conhecimento do sistema especialista. Esta base foi construida através da aquisição de conhecimento existente na literatura sobre projeto de circuitos integrados e na experiência dos projetistas peritos no assunto.

O processo de verificação do circuito é feito a partir da aplicação de regras da base de conhecimento sobre a descrição do circuito. Como resultado são gerados diagnósticos sobre o funcionamento do circuito a nível elétrico. Além disto, o CREPE apresenta uma explanação, o processo de raciocínio que o levou

a um dado resultado. Esta explanação auxilia o projetista na tarefa de encontrar os defeitos de projeto.

Em suma o sistema faz automaticamente a análise e verificação que um projetista faria manualmente antes de submeter o circuito a uma simulação elétrica.

Na construção do sistema CREPE optou-se pela abordagem de sistema especialista pelos motivos:

- Necessidade constante de aumento do número de regras a serem verificadas, isto exige uma base de conhecimento facilmente ampliável,
- Necessidade de se dar explanações sobre os avisos gerados para facilitar o processo de identificação do erro no circuito,
- A tarefa de análise de um circuito exige experiência e heurísticas, as quais podem ser representadas adequadamente com os sistemas especialistas,
- Necessidade de classificação e reconhecimento das estruturas de circuito o que sugere um sistema baseado em regras.

O sistema está representado em linhas gerais na figura 1.2.

O usuário fornece uma descrição do circuito em formato SPICE III. O sistema converte os dados para fatos em PROLOG colocando uma certa "inteligência" nos dados. Estes fatos são analisados e a partir destes dados podem ser identificadas estruturas pré definidas (inversores, nor, nand, etc). Com os fatos PROLOG e as estruturas encontradas, o sistema aplica as regras que fazem parte da base de conhecimento. As regras são aplicadas em uma seqüência pré definida sendo os resultados armazendos em uma saída padrão, por exemplo, via uma impressora. A saída de dados são os avisos de problemas detetados pela aplicação das regras ao circuito em questão. O usuário pode fazer perguntas

para esclarecimento do raciocínio usado pelo programa para uma determinada regra.

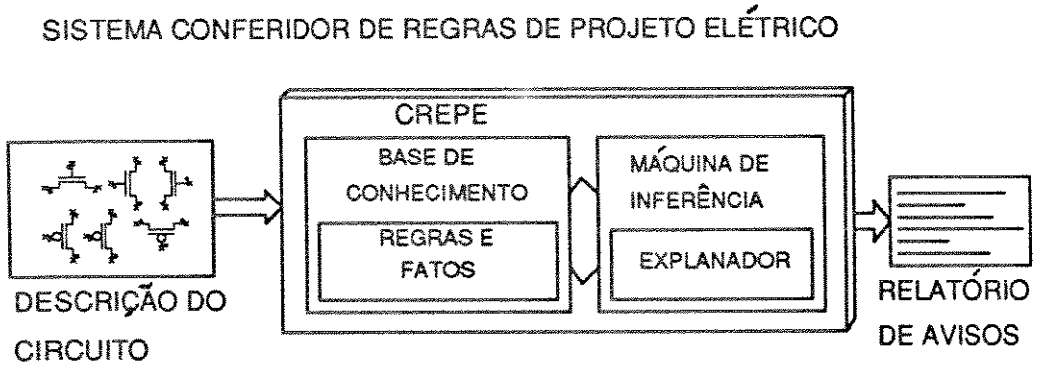


Figura 1.2 Módulos componentes do sistema conferidor de regras elétricas CREPE.

1.4. Organização desta Dissertação

A dissertação está dividida em cinco capítulos, bibliografia e dois anexos:

O capítulo um faz uma introdução ao tipo de trabalho que foi realizado, dando um enfoque genérico aos sistemas especialistas.

O capítulo dois apresenta um estudo mais detalhado das técnicas de sistemas especialistas voltado para esta aplicação em particular.

O capítulo três descreve o sistema CREPE (Conferidor de REgras de Projeto Elétrico) em detalhes. Durante a descrição do sistema são feitas as considerações teóricas envolvidas na conceituação das regras.

O capítulo quatro mostra a implementação do sistema como um todo, e de seus módulos componentes, bem como os resultados obtidos com um protótipo experimental.

O capítulo cinco faz considerações finais e sugestões de continuação do trabalho.

Os anexos contém:

- Anexo A: Exemplos de aplicação do sistema em sete circuitos de teste, mostrando os avisos gerados, estruturas identificadas,
- Anexo B: Um exemplo de uma execução completa do CREPE. Mostra também os arquivos usados e gerados pelo sistema.

CAPÍTULO DOIS

2. ESTUDO DOS SISTEMAS ESPECIALISTAS

2.1. Introdução

Os sistemas especialistas vêm tendo inúmeras aplicações devido ao sucesso obtido na solução de problemas onde os sistemas convencionais são ineficientes. Os sistemas especialistas incorporam conhecimento de uma área específica e, usando mecanismos de inferência, resolvem problemas. Neste capítulo fazemos um estudo com enfase nos tópicos:

- Visão geral dos sistemas especialistas e comparação com sistemas convencionais,
- Classes de problemas que os sistemas especialistas se propoem a resolver,
- Conceituação e características de sistemas especialistas,
- Construção de sistemas especialistas,
- Ferramentas para auxiliar na construção de sistemas especialistas,
- Avaliação de sistemas especialistas.

2.2. Visão Geral

Os sistemas especialistas se aplicam a problemas que exigem conhecimento e experiência em um domínio restrito. As características principais que os diferenciam dos convencionais são:

- Perícia, que consiste no conhecimento sobre um domínio específico de problemas e na habilidade em resolve-los,
- Heurística, que permite determinar soluções baseadas em dados incompletos ou com erros e que também permite elucidar e representar o conhecimento adquirido no processo de solução,
- Resolução de problemas que não são resolvidos com métodos algorítmicos,
- A aquisição do conhecimento humano, representando-o no computador de modo a facilitar sua reprodução e exploração.

O objetivo do sistema especialista é solucionar problemas de forma inteligente. A idéia central é que o sistema obtenha uma solução de forma seletiva e eficiente a partir de um espaço de alternativas. Esta solução depende do conhecimento sobre o assunto e será satisfatória na medida em que encontrar um resposta suficientemente boa para os recursos existentes. Portanto, um sistema especialista poderá não encontrar a melhor resposta mas encontrará uma resposta satisfatória para um problema de sua alçada.

2.2.1. Conhecimento e Base de Conhecimento

Os sistemas especialistas baseiam-se na manipulação do conhecimento, diferentemente dos convencionais que apenas manipulam dados. Portanto, as definições de conhecimento e de base de conhecimento são fundamentais para entender o funcionamento de sistemas especialistas.

Uma definição dada por Hayes [28] é apresentada a seguir:

" O conhecimento consiste das descrições, relações e procedimentos num dado domínio".

A base de conhecimento é formada por:

- Descrições das relações, que identificam e diferenciam objetos e classes, que são sentenças em uma linguagem de programação,
- Relações, que expressam as dependências e associações entre os ítens da base de conhecimento,
- Procedimentos, que especificam as operações a executar quando acionados para resolver um problema.

Resumindo, o conhecimento consiste de:

- Uma descrição simbólica que caracteriza as relações empíricas definidas num dado domínio,
- O procedimento para manipular estas descrições.

2.2.2. Aplicações dos Sistemas Especialistas.

Os sistemas especialistas tem aplicações em áreas do conhecimento humano que necessitam da experiência, da perícia e do conhecimento em um domínio muito restrito. As aplicações dos sistemas especialistas normalmente se enquadram em alguns dos tipos de tarefas a seguir: interpretação, predição, diagnóstico, projeto, planejamento, monitoração, depuração, instrução e controle. Estas tarefas, para serem executadas por programas de computador, necessitam de sistemas que adquiram o conhecimento e os métodos de aplicação deste conhecimento. Só assim estas tarefas serão executadas com sucesso e eficiência.

2.2.3. Componentes de um Sistema Especialista: Exemplo

Existem na literatura várias maneiras de representar os componentes de um sistema especialista, dependendo do tipo de problema que se pretende resolver. Uma maneira genérica de se representar os componentes de um sistema especialista está mostrada na figura 2.1.

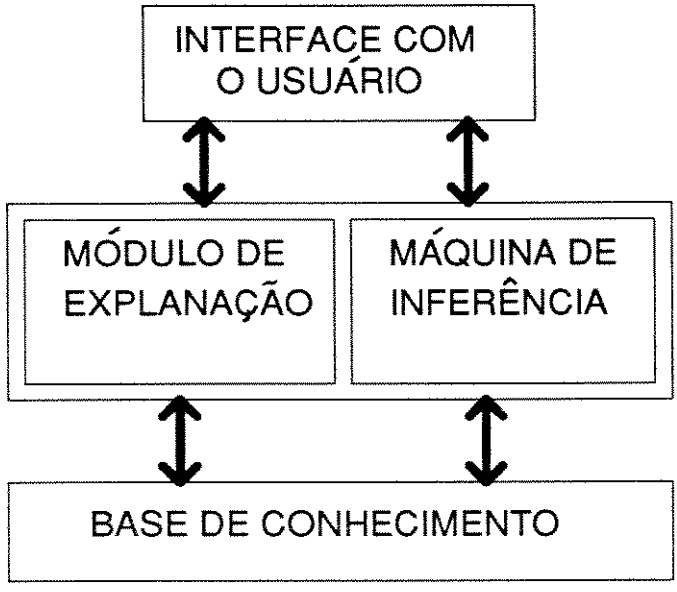


Figura 2.1 Componentes de um sistema especialista.

Os componentes básicos de um sistema especialista são descritos abaixo:

- Interface com o usuário: esta interface permite a comunicação com o usuário para apresentação da solução, perguntas, explanações etc.,
- Máquina de inferência: contém o conhecimento geral para a solução de problemas,

- Base de conhecimento: contém o conhecimento específico do domínio em questão.
- Módulo de explanação: permite ao usuário entender o processo de raciocínio usado pelo sistema especialista para chegar a uma solução.

2.3. Classes de Problemas e Arquiteturas dos Sistemas Especialistas.

Neste item são apresentadas as classes de problemas que os sistemas especialistas se propõem a resolver. A complexidade, a aplicação e o tipo de conhecimento disponível determinam a organização geral mais apropriada para o sistema, isto é, sua arquitetura.

A figura 2.2 mostra essas classes de problemas e os requisitos que a arquitetura do sistema especialista deve considerar[28].

O caso 1 considera uma arquitetura onde os dados são constantes, o conhecimento é confiável e o espaço de busca é restrito. Neste caso pode ser utilizada uma arquitetura simples , com uma única linha de raciocínio, busca exaustiva e raciocínio monotônico[34].

No caso 2 considera-se que os dados ou o conhecimento podem conter erros. Aqui podem-se usar modelos probabilísticos ou lógica nebulosa. Estes métodos são baseados na idéia de que combinando-se evidências obtem-se um aumento na confiabilidade dos dados . Estes métodos exigem um conhecimento sobre o conhecimento que é necessário para combinar estas evidências.

No caso 3 são considerados dados variando no tempo onde se usa o cálculo de situações e regras de transições[28] introduzidos por McCarthy e Hayes. A idéia básica é incluir "situações" que indicam o estado para o qual o objeto está indo, ou onde estava quando os dados mudaram.

Os casos restantes tratam das maneiras de reduzir um espaço de busca amplo através de um esquema hierárquico de geração e teste de soluções[28].

No quarto caso 4 considera-se um espaço de busca amplo que pode ser fatorado em sub-espaços, de forma a eliminar rapidamente as soluções que não satisfizerem às condições do problema. O método de geração e teste só se aplica quando as soluções que não satisfazem às condições do problema puderem ser eliminadas rapidamente; por isso é considerado um método pobre. Além disto, nem em todas as aplicações é necessário considerar todas as possíveis soluções para escolher a melhor, basta obter-se uma que seja satisfatória.

No caso 5, a geração e teste de soluções não pode fazer uma avaliação parcial das soluções. Utiliza-se então uma forma de raciocínio baseada em abstrações, adequada quando o espaço de busca for amplo e não se podem eliminar as soluções parciais. Este método requer que todas as informações para o teste das soluções parciais estejam disponíveis antes que se prossiga para um novo subproblema.

No caso 6 considera-se o refinamento de soluções onde é feita uma análise das soluções em cada nível da hierarquia, iniciando-se pelo nível mais alto e prosseguindo-se em direção ao nível mais baixo. Desta forma pode-se reduzir, de maneira mais eficiente, o espaço de busca . Neste caso as abstrações são compostas por um conjunto de conceitos num espaço de busca hierarquizado.

O caso 7 envolve o princípio do último confiável, que diz se as decisões devem ser postergadas até que se tenha um conjunto suficiente de informações. Isto requer a habilidade de suspender atividades de solução de subproblemas, movimentação de dados entre subproblemas e de reiniciar o processo de solução quando as informações estiverem disponíveis.

O caso 8 aborda a situação em que o gerador e testador de soluções não encontra uma solução imediatamente; então é necessário fazer uma suposição para que o processo de solução continue. A quantidade de suposições necessárias é proporcional á falta de conhecimento. Tem-se constatado que, normalmente as

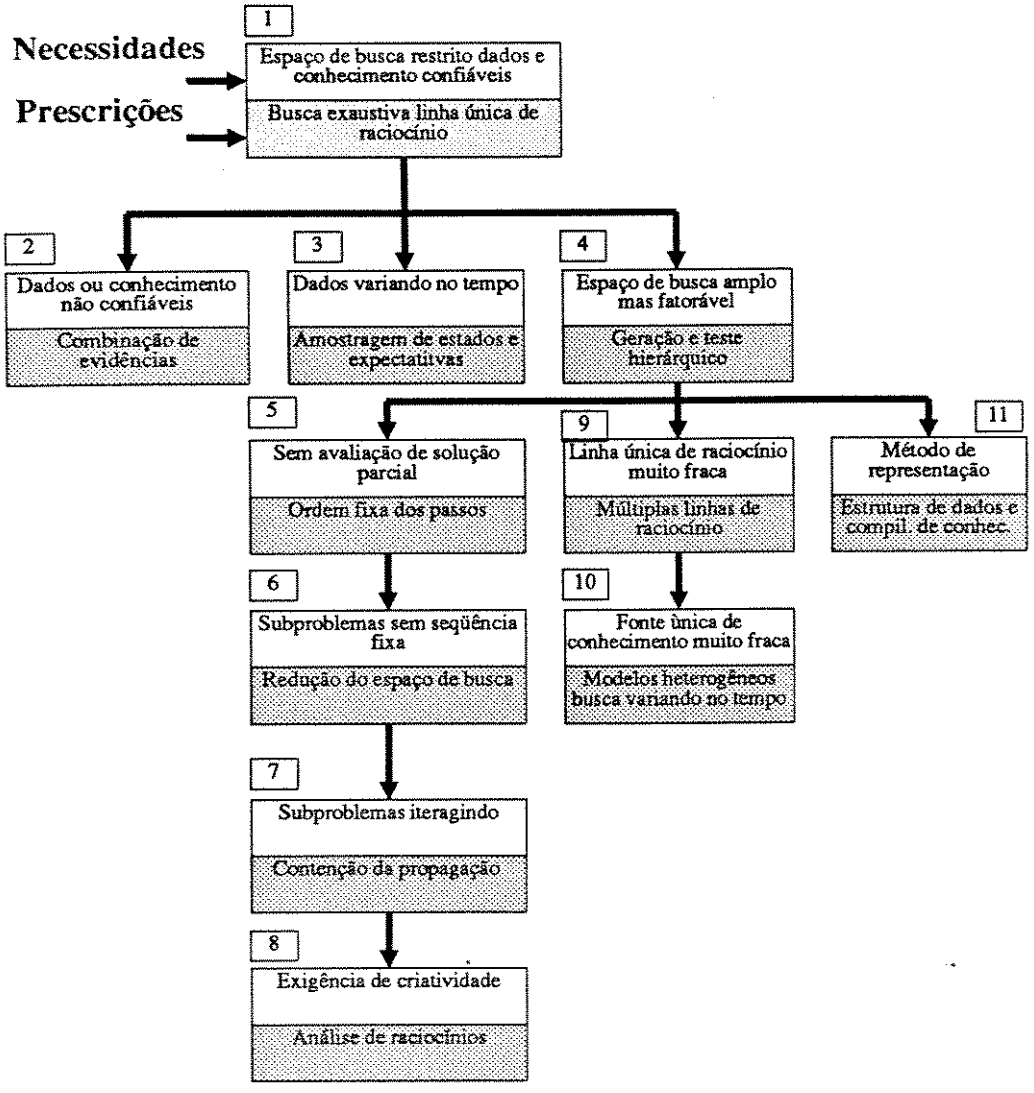


Figura 2.2. Árvore com as classes de problemas. Cada retângulo corresponde a um caso. As linhas ordenam os casos numa árvore, onde a seqüência dos ramos indica aumento na complexidade do conhecimento e na complexidade da estratégia de busca.

bases de conhecimento são incompletas, isto requer muitas suposições o que pode gerar soluções não confiáveis.

Os casos 9 e 10 referem-se ao uso de múltiplas linhas de raciocínio com o objetivo de aumentar a capacidade de solucionar problemas. E também referem-se ao uso de modelos de abtração variados para captura de variados tipos de conhecimento.

O caso 11 mostra a utilização de métodos para acelerar o processamento e a recuperação da informação, sendo conhecidos por estrutura de dados especializada e compiladores de conhecimento[28].

2.4. Características e Conceitos Relacionados aos Sistemas Especialistas

2.4.1. Definição de Sistema Especialista.

Para uma definição de sistema especialista [28] devem ser levados em conta os conceitos de: perícia, raciocínio, inteligência e complexidade, associados a: reformulação, raciocínio sobre si mesmo e tarefas. Estes conceitos são comentados abaixo:

- Perícia: é a capacidade de desempenho de alto nível, análoga a que tem o ser humano e a sua capacidade de produzir respostas de alta qualidade em um curto espaço de tempo,
- Raciocínio: é o processo de encontrar uma solução eficiente, de evitar buscas inúteis fazendo uma rápida eliminação de hipóteses em cada processo de inferência,
- Inteligência: é a habilidade de resolver problemas genéricos num dado domínio,
- Complexidade: Os problemas devem ser suficientemente complexos para necessitarem de um especialista para sua solução,
- Reformulação: é a capacidade de reformular hipóteses e conclusões durante o processo de solução,
- Raciocínio sobre si mesmo: é a capacidade do sistema de olhar para si mesmo, para sua linha de raciocínio, a fim de entender o processo como se fosse o especialista,
- Tarefas: os problemas devem ser do tipo dos resolvidos por especialistas, e possuir um domínio reduzido.

Pode-se definir um sistema especialista como aquele que foi criado para resolver problemas usando as características citadas acima. Desta forma o sistema deve conter as seguintes habilidades:

- Adquirir conhecimento para aumentar sua experiência,
- Reconceituar,
- Adquirir conhecimento mais genérico,
- Ter senso comum,
- Raciocinar por analogia.

Os sistemas especialistas serão incapazes de reconhecer ou resolver problemas para os quais o seu conhecimento for insuficiente ou inaplicável; não tem capacidade para conferir se suas conclusões são razoáveis. Além disto explicações de seu raciocínio algumas vezes são incompletas. Hoje em dia os sistemas especialistas se aplicam a uma pequena faixa do desenvolvimento da inteligência artificial.

O estado da arte dos sistemas especialistas[28] pode ser resumido abaixo:

- Domínio restrito de perícia,
- Linguagens limitadas para expressar fatos e relações,
- Suposições limitadas sobre problemas e métodos de solução,
- Linguagem de entrada e saída estilizada,
- Linha de raciocínio estilizada,
- Pequeno conhecimento do seu escopo e limitações,
- Base de conhecimento extensível.

2.4.2. Fundamentos dos Sistemas Especialistas

O coração do sistema especialista está na base de conhecimento, onde o conhecimento é acumulado durante o processo de construção [28]. A acumulação de experiências e sua codificação é um dos mais importantes aspectos de um sistema especialista.

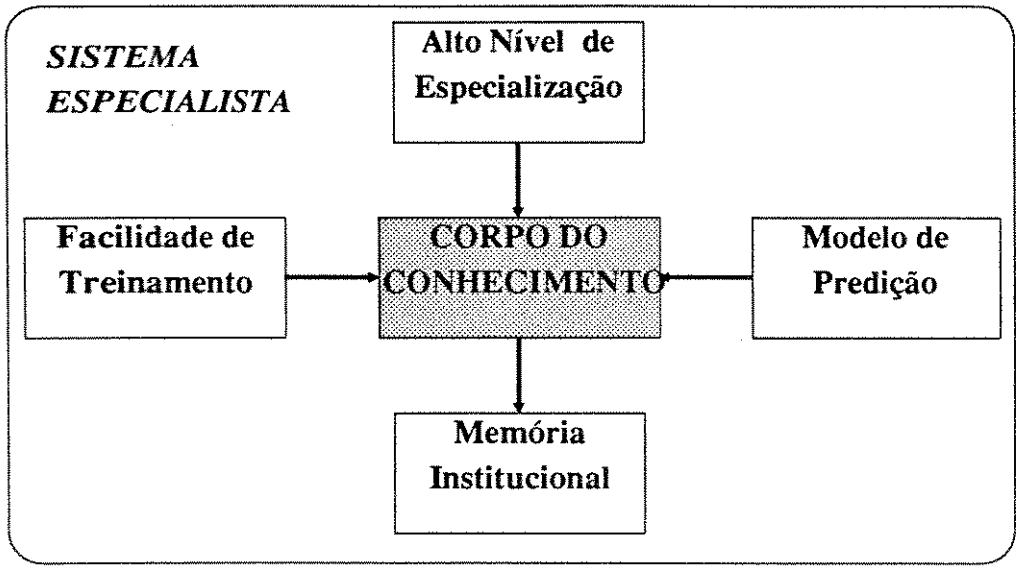


Figura 2.3 Diagrama mostrando as relações que o sistema especialista tem com a base de conhecimento.

As capacidades de um sistema especialista podem ser vistas quando mostramos as relações que o sistema tem com a base de conhecimento, mostradas na figura 2.3.

Estas relações são comentadas abaixo:

- Alto nível de especialização: fornece condições para a solução de problemas específicos, na medida em que esta especialização pode

representar o melhor dos especialistas numa dada área, gerando soluções criativas, eficientes e precisas.

- O modelo de predição: atua como um processador de informação ou como um método de resolução de problemas numa dada área, fornecendo a solução desejada. O sistema especialista pode explicar com detalhes como chegou a esta solução. Também pode mostrar como outros dados a afetariam, permitindo ao usuário entender e avaliar os efeitos dos dados nessa solução.
- A memória institucional: armazena o conhecimento de um especialista ou chefe de equipe que, desta forma, pode sair da mesma sem que se perca o seu conhecimento. Este conhecimento foi adquirido a partir das opiniões de alto nível de especialistas. E a este conhecimento pode-se incluir e armazenar as melhores estratégias de soluções de problemas.
- A capacidade de treinamento: alguns sistemas possuem a capacidade de treinar seus usuários, visto que já contém o conhecimento necessário e a habilidade de explicar os processos de raciocínio.

Participantes na Construção de Sistemas Especialistas.

A figura 2.4 mostra a interrelação entre os participantes, as ações e os resultados no processo de construção de um sistema especialista. Nessa figura podemos ver que o participante central desse processo é o engenheiro de conhecimento, que utiliza a ferramenta de software para a construção do sistema especialista, faz as entrevistas com o especialista, constroi, refina e testa a base de conhecimento. Os consultores completam o processo verificando a qualidade do conhecimento adquirido.

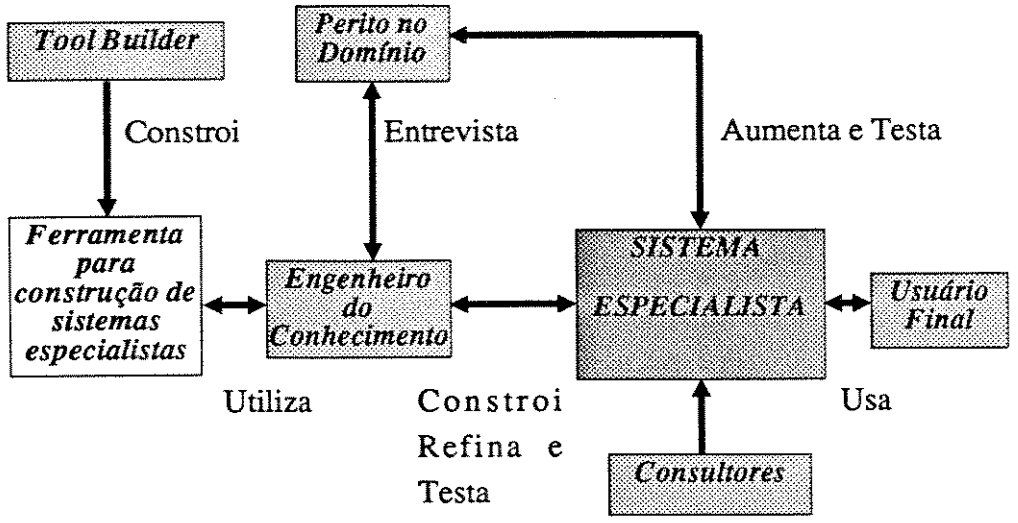


Figura 2.4 Diagrama das relações e participantes na construção de sistemas especialistas.

2.4.3. Definições e Conceitos Relacionados com os Sistemas Especialistas.

Terminologia básica dos sistemas especialistas:

- Inteligência artificial: parte da ciência da computação que trata do desenvolvimento de programas de computador inteligentes,
- Engenheiro de conhecimento: pessoa que projeta e constroi o sistema especialista,
- Especialista na área: pessoa que por sua grande experiência tornou-se um eficiente solucionador de problemas da área,

- Usuário final: pessoa que vai usar o sistema, pessoa para a qual o sistema foi concebido,
- Representação: processo de formulação ou visualização do problema buscando facilitar sua solução,
- Busca: processo eficiente de procura de uma solução aceitável dentro do conjunto de soluções possíveis para o problema,
- Busca cega: processo onde se procura o objetivo de uma maneira não seletiva, isto é, todos os fatos e regras são percorridos de forma indiscriminada,
- Busca heurística: processo de procura seletiva segundo critérios heurísticos,
- Ambiente de Suporte: é o conjunto das características das ferramentas de software que auxiliam na interação do usuário com o sistema especialista,
- "Tool Kit": consiste da linguagem de programação e do conjunto de ferramentas de suporte usadas para construir o sistema especialista,
- "Tool Builder": pessoa que projeta e constroi as ferramentas de software para auxiliar na construção de sistemas especialistas,
- Base de Dados: conjunto de fatos, assertivas e conclusões com base nos quais se aplicam as regras de um sistema especialista,
- Ciclo de Inferência: seqüência de passos ou aplicações de regras que são usadas por um sistema especialista para chegar a uma solução,
- Método de Inferência: técnica usada pela máquina de inferência para aplicar e acessar o conhecimento,
- Encadeamento para Trás (Backward Chaining): método de inferência onde o sistema inicia com um objetivo por provar e tenta estabelecer os fatos que provam esse objetivo,

- Encadeamento para Frente (Forward Chaining): método de inferência onde as regras são aplicadas a fatos para estabelecer novos fatos,
- Símbolo: é uma cadeia de caracteres utilizada para representar algum objeto real.

2.4.4. Caracterização das Tarefas do Especialista.

Os especialistas são pessoas que executam várias tarefas genéricas:

Interpretação: é a determinação do significado dos dados através de sua análise: busca-se encontrar uma interpretação correta e consistente dos mesmos. É importante considerar que os dados podem ser falhos ou incompletos e portanto:

- Pode-se eventualmente contar apenas com informações parciais,
- Os dados podem ser contraditórios,
- O interpretador deve ser capaz de formular hipóteses,
- Mesmo que o caminho de raciocínio seja longo o programa interpretador deve ser capaz de dizer como chegou ao seu resultado.

Diagnóstico: é o processo para encontrar falhas em sistemas, baseando-se na interpretação de dados válidos. Para o diagnóstico deve-se conhecer a organização dos sistemas em análise e suas interações com os sub-sistemas.

Alguns problemas chave inerentes desta tarefa:

- Algumas vezes as falhas podem ser mascaradas por outras falhas,
- As falhas podem ser intermitentes,
- O diagnóstico as vezes tem que ser exaustivo,

- O próprio processo de diagnóstico pode ter falhas,
- Algumas informações sobre o sistema em diagnóstico podem ser inacessíveis.

Monitoração: é o processo contínuo de interpretação de sinais e emissão de alarmes quando for necessária uma intervenção.

Previsão: é a previsão do curso do futuro a partir de um modelo do passado e do presente. A previsão requer relações de tempo entre presente e passado.

Planejamento: é a preparação de um programa de ação para alcançar os objetivos.

Projeto: é o processo de construção de elementos a partir de uma definição de requisitos, que satisfaçam às necessidades específicas.

2.4.5. Metodologia para construção de sitemas especialistas.

Dentro desta metodologia o processo de extração do conhecimento do especialista e sua codificação em programa é chamado aquisição de conhecimento. Esta transferência e transformação da fonte do conhecimento para um programa, é o coração do desenvolvimento de um sistema especialista.

A metodologia para aquisição do conhecimento inclui as seguintes etapas:

- IDENTIFICAÇÃO: Determinação das características do problema,
- CONCEPÇÃO: Estabelecimento dos conceitos para representar o conhecimento,
- FORMALIZAÇÃO: Projeto das estruturas e organização do conhecimento,
- IMPLEMENTAÇÃO: Formulação das regras que incorporam o conhecimento ao sistema especialista,

- TESTE: Validação das regras que incorporam o conhecimento.

2.4.6. Meta Conhecimento.

O desempenho de um sistema especialista pode ser melhorado providenciando-se um conhecimento sobre o conhecimento. A este chamamos de meta conhecimento. Por exemplo: uma regra que estabelece que regras para o salvamento de vidas tem precedência sobre regras que dão notícias, está utilizando-se de meta conhecimento.

Quando o sistema especialista crescer e ficar complexo poderá ser difícil entender o que se passa; uma explanação ou justificativa para seus resultados será fundamental e muito útil. Devido a característica de conhecimento sobre o conhecimento o meta conhecimento pode auxiliar no processo de explanação.

Quando um sistema especialista se tornar muito grande deverá ser feita uma seleção na aplicação de suas regras, caso contrário haverá queda no seu desempenho; o meta conhecimento poderá facilitar essa seleção.

Um exemplo de meta conhecimento útil para melhorar o desempenho de uma busca:

- Se < o espaço de busca for relativamente pequeno >
- Então < será possível uma busca exaustiva >

O meta conhecimento poderá auxiliar o sistema a se adaptar ao ambiente de execução, como no exemplo:

- Se < um pedaço do código for muito usado >
- Então < será digno de uma otimização >

Resumindo o meta conhecimento ou as meta regras são utilizadas com os seguintes objetivos:

- Guiar a alocação e a seleção das regras,
- Fornecer informações sobre o conhecimento do domínio e sobre as regras,
- Justificar regras, aumentando a capacidade de explanação.
- Ajudar na detecção de defeitos em regras novas, inseridas para a expansão da base de conhecimento.

2.4.7. Métodos de Representação do Conhecimento

As representações de dados e conhecimento mais utilizadas são normalmente feitas com regras, com "frames" ou com redes semânticas.

Regras: são o método de representação mais conhecido na área de sistemas especialistas. As regras fornecem uma maneira formal de representação de diretivas, recomendações ou estratégias. A técnica de regras deverá ser usada quando o conhecimento for resultado de uma longa experiência resolvendo problemas na área, com o conseqüente estabelecimento de correlações empíricas. As regras são expressões do tipo se ... então ... como mostra o exemplo:

- Se < o circuito tiver curtos >
- Então < o circuito poderá não funcionar >
- Se < a capacitância do nó for muito alta >
- Então < o nó poderá ter atrasos indesejados >

Redes semânticas: são uma representação do conhecimento baseada numa estrutura de redes. As redes consistem de pontos chamados nós conectados por setas chamadas arcos. Os nós representam objetos, conceitos ou eventos. Os arcos defininem as relações entre os nós. Normalmente os arcos representam hierarquias do tipo "isto é um" ou "isto é uma parte de". A figura 2.6 B mostra um exemplo de rede semântica:

Frames: são uma maneira de representar conceitos e situações comuns. A estrutura dos "frames" é muito semelhante à das redes semânticas. Os "frames" são uma rede de nós e relações organizados segundo uma hierarquia. Os nós do topo da hierarquia representam os conceitos mais genéricos; a medida que se desce na hierarquia os conceitos vão ficando mais específicos. No sistema de "frames" o conceito em cada nó é definido por um conjunto de atributos e de valores destes atributos. Os atributos são chamados de "slots". Cada "slot" tem associado um procedimento que controla o seu preenchimento, podendo haver outros procedimentos para efetuar outras funções. A figura 2.6 C mostra um exemplo de estrutura baseada em "frames".

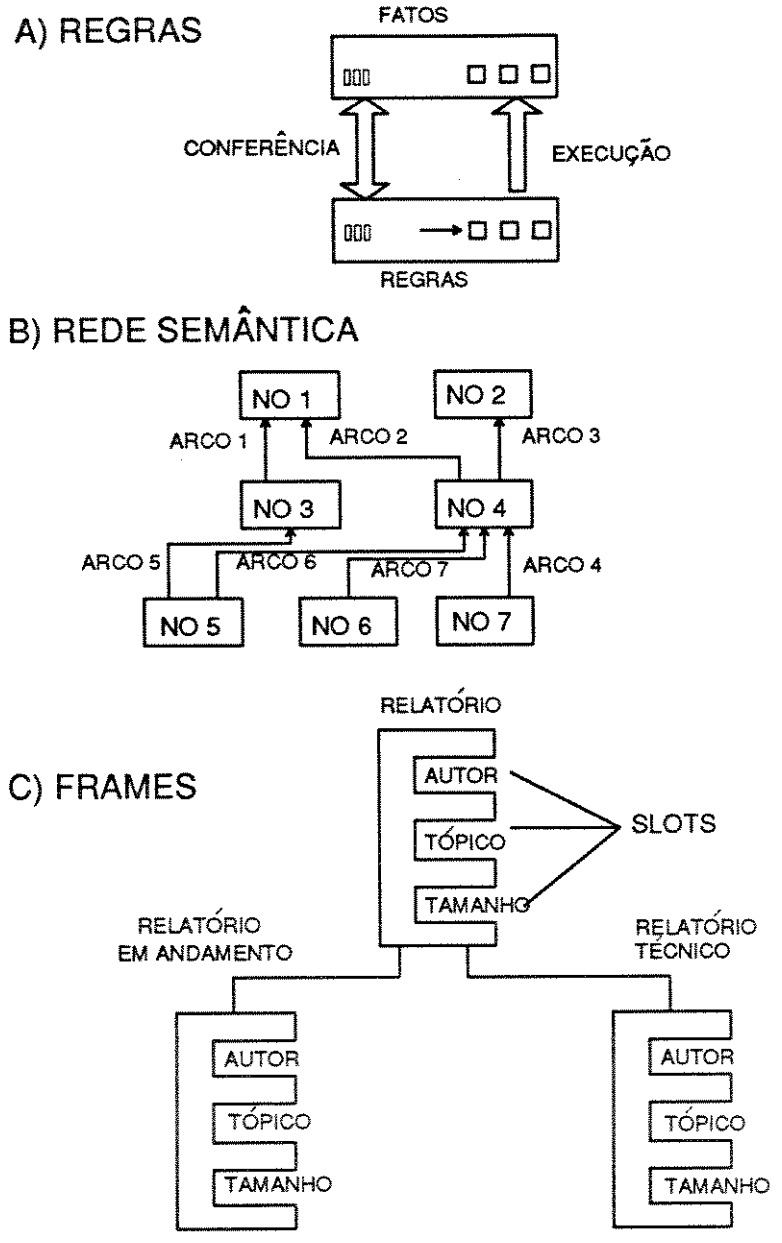


Figura 2.6 Diagrama de representações do conhecimento: A) Regras B) Rede semântica C) Frames.

2.5. Construindo um Sistema Especialista

Quando o conhecimento sobre um domínio for fixo e formalizado, programas algorítmicos podem resolver os problemas relacionados a este domínio. No entanto, quando o conhecimento for subjetivo, os sistemas especialistas com a incorporação de heurísticas são mais apropriados. A dificuldade reside na extração do conhecimento e de sua transferência para um programa. Este processo é chamado de aquisição de conhecimento. Os sistemas especialistas normalmente nascem pequenos e restritos com a intenção de crescer em profundidade e abrangência depois de sua implementação inicial.

2.5.1. Aquisição de Conhecimento

É o processo de transferência e transformação de um modo de resolver problemas para um programa de computador. Fontes potenciais de conhecimento são os livros, os bancos de dados, os especialistas ou as experiências próprias. A aquisição de conhecimento pode ser automática, parcialmente automática ou manual.

Modos de Aquisição de Conhecimento

A aquisição de conhecimento é o gargalo na construção de um sistema especialista. Ao engenheiro de conhecimento cumpre a tarefa de ajudar um especialista a estruturar seu conhecimento para identificar e formalizar os conceitos do domínio. O conhecimento pode ser adquirido de vários modos, mas todos envolvem a transferência da capacidade necessária para a solução de problemas da fonte de conhecimento (o especialista) para o programa (sistema especialista). O processo de aquisição de conhecimento é feito pelo engenheiro de conhecimento ou por um programa inteligente, como mostra a figura 2.7. O conhecimento é separado da máquina de inferência o que torna as estruturas transparentes e flexíveis. Uma vez que os especialistas constroem

sua base de conhecimento de experiências passadas e livros, há razões para crer que um programa de indução possa fazer esta tarefa, sendo esta outra maneira de se construir uma base de conhecimento.

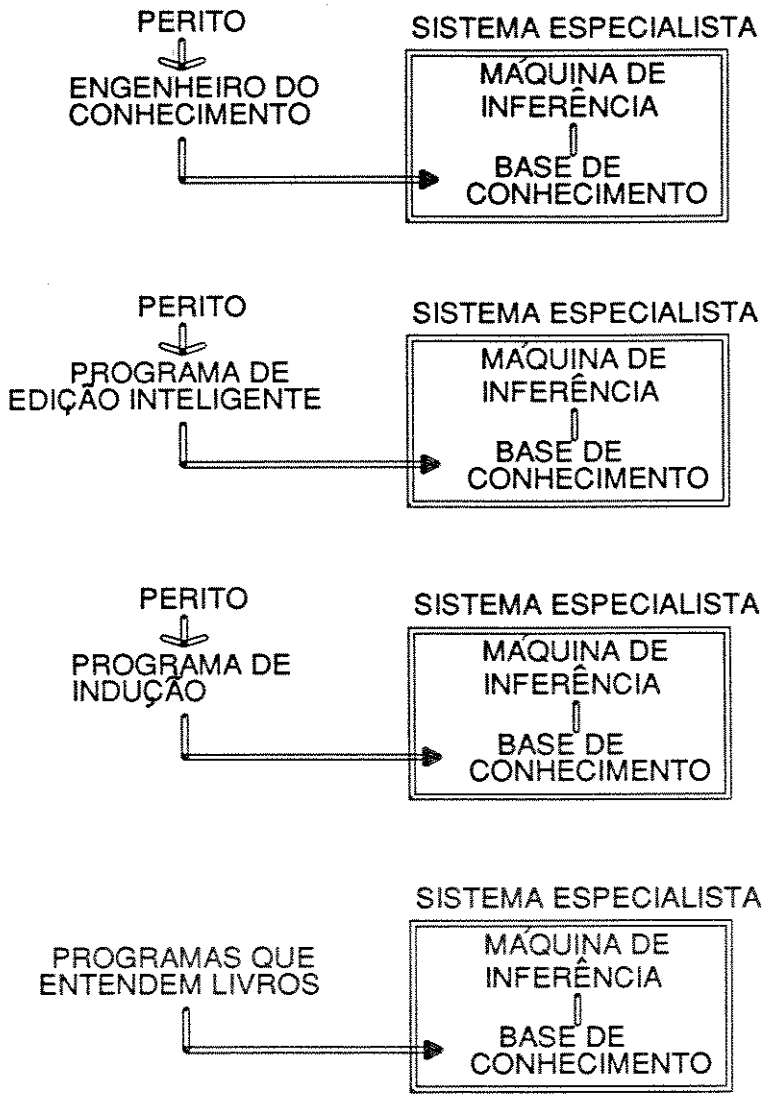


Figura 2.7 Modos de aquisição de conhecimento.

Compondo o Cenário para a Aquisição de Conhecimento

Apresentamos aqui alguns passos importantes que o engenheiro de conhecimento executa durante a evolução e desenvolvimento de um sistema especialista:

- Passo 1: Familiarizar-se com o assunto. Isto inclui a localização das fontes de especialistas e de conhecimento em geral,
- Passo 2: Depois de algum estudo informal o engenheiro de conhecimento dominará o problema (o suficiente para conversar inteligentemente com o especialista) e iniciará o processo de identificação e caracterização do problema.

Quando o problema estiver adequadamente descrito, o engenheiro de conhecimento iniciará a determinação dos conceitos maiores do domínio que serão necessários para a execução da tarefa. Durante os encontros com o especialista o engenheiro de conhecimento deverá captar e entender quais são os conceitos importantes e relevantes do problema, pedindo ao especialista que explique e justifique o seu raciocínio sobre o problema. O segundo tipo de conhecimento que o engenheiro de conhecimento deverá captar é a estratégia que o especialista usa quando resolve os problemas. Quando se tem estes conhecimentos reunidos pode-se formar uma estrutura de inferência.

Durante a conceituação o engenheiro de conhecimento também pensará em como formalizará o conhecimento, isto é, a arquitetura do sistema especialista que melhor representará a organização do conhecimento. Durante a formalização do conhecimento haverá uma realimentação do engenheiro de conhecimento para o especialista onde serão validadas as regras.

2.5.2. Estágios da Aquisição de Conhecimento

A aquisição do conhecimento passa por uma seqüência de várias fases e, a medida que se encontram problemas, pode-se voltar para a fase anterior, de maneira a alcançar uma aquisição de conhecimento satisfatória. Na figura 2.8 são mostrados os estágios de aquisição de conhecimento.

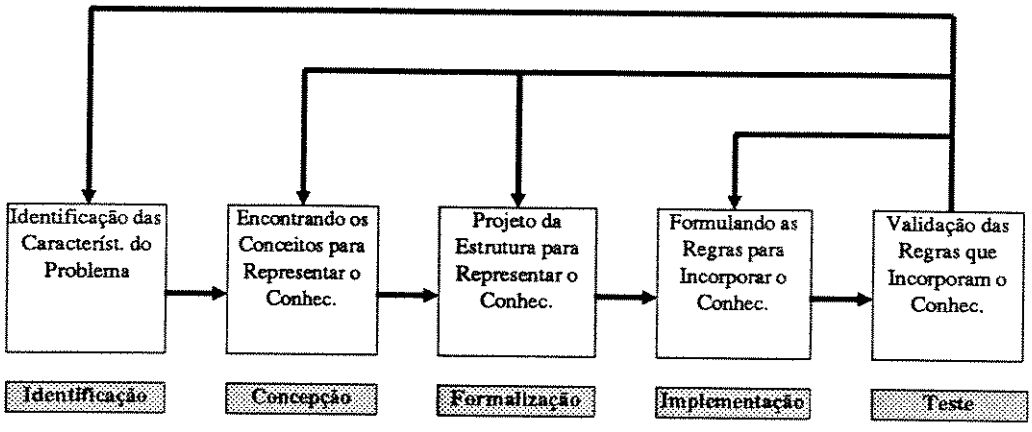


Figura 2.8 Estágios da aquisição do conhecimento.

Estágio de identificação.

O primeiro passo é caracterizar os aspectos importantes do problema. Isto envolve:

- Identificação dos participantes,
- Caracterização do problema,
- Identificação dos recursos,
- Identificação dos objetivos.

Identificação dos participantes: antes de iniciar o processo, os participantes devem ser selecionados e seus papéis definidos. Normalmente os participantes são o especialista e o engenheiro de conhecimento, mas podem-se usar diversos especialistas e engenheiros do conhecimento de diversas áreas de conhecimento.

Caracterização do problema: uma vez escolhidos os participantes, os procedimentos do engenheiro de conhecimento e do especialista vão na direção de caracterizar o problema e o conhecimento que o suporta. Para caracterização do problema é importante responder as seguintes perguntas:

- Que classe de problemas o sistema especialista pretende resolver?
- Como estes problemas podem ser caracterizados ou definidos?
- Quais são os subproblemas importantes e o particionamento das tarefas?
- Quais são os dados?
- Quais são os termos importantes e suas interrelações?
- Qual é a forma da solução e que conceitos são usados nela?
- Quais os aspectos da perícia humana que são essenciais para resolver este problema?
- Qual é a natureza e extensão do conhecimento que são usados nas soluções dadas pelos humanos?

Respondendo a estas perguntas os participantes isolam e verbalizam o conhecimento relevante para resolver o problema e identificam os elementos chave da descrição do problema.

Identificação dos Recursos: as fontes de recursos do especialista incluem a resolução de problemas passados, livros e exemplos de problemas e soluções. E para o engenheiro de conhecimento as fontes são a experiência em problemas

análogos, conhecimento dos métodos de representação e ferramentas para a construção do sistema especialista.

Identificação dos objetivos: Normalmente uma boa prática é relacionar os objetivos para que estes constituam restrições adicionais podendo até indicar se o sistema for factível ou não.

Exemplo de possíveis objetivos:

- Gerar um conjunto informal de práticas,
- Disseminação da perícia,
- Ajudar os especialistas na resolução de problemas,
- Automatização de rotinas do trabalho do especialista.

Estágio de Conceituação

Durante a fase de conceituação as relações e conceitos chave devem ser explicitados. Para facilitar o entendimento do processo de conceituação são sugeridas as questões abaixo. Estas devem ser respondidas antes do procedimento de conceituação.

- Quais são os tipos de dados disponíveis?
- O que é dado e o que é inferido?
- Quais os nomes que as subtarefas devem ter?
- Quais os nomes que os estágios devem ter?
- Existem hipóteses parciais identificáveis comumente usadas? Quais são elas?
- Como são os objetos do domínio?
- Pode-se diagramar a hierarquia e nomear as relações causais, inclusão de conjuntos, relações parte -- todo etc... Como aparecem?

- Quais são os processos envolvidos na solução do problema?
- Quais são as restrições nestes processos?
- Qual é o fluxo de informação?
- Pode-se identificar e separar o conhecimento necessário para resolver o problema do conhecimento para justificá-lo?

O engenheiro de conhecimento poderá sair desta fase sem uma definição muito rígida, pois há várias realimentações durante processo de aquisição.

Estágio de Formalização.

A formalização envolve o mapeamento dos conceitos chave, subproblemas e características do fluxo de informação, em uma representação formal, isolados durante a conceituação. Para auxiliar na tarefa de mapeamento do conceitos existem várias ferramentas de engenharia do conhecimento ou "frameworks". O resultado deste estágio é um conjunto parcial de especificações descrevendo como o problema pode ser representado com a ferramenta ou "framework" escolhido. Três são fatores importantes na formalização:

- Espaço de hipóteses; para se entender a estrutura do espaço de hipóteses, é necessário formalizar os conceitos e determinar como se ligam para formar uma hipótese. A estrutura dos conceitos também deve ser definida nesta fase. Isto é útil para descrever conceitos como objetos estruturados.
- Modelo básico do processo; um passo importante na formalização do conhecimento é descobrir um modelo de processo e usá-lo para gerar as soluções para o problema.
- Características dos dados; o entendimento da natureza dos dados no domínio do problema é outro fator importante na formalização.

As perguntas abaixo objetivam um aprofundamento no entendimento da natureza dos dados sobre o domínio.

- Os dados são insuficientes ou incompletos?
- Os dados são esparsos ou redundantes?
- Existe incerteza associada aos dados?
- A interpretação lógica dos dados depende de sua ordem de ocorrência no tempo?
- Qual é o custo da aquisição de dados?
- Como são os dados, adquiridos ou deduzidos?
- Quais são as classes de questões necessárias para se obter os dados?
- Como certas características dos dados são reconhecidas quando amostrados ou extraidas de um fluxo de dados, como podem as características serem extraidas de formas de onda, desenhos ou tradução de entrada de linguagem natural?
- Os dados são recuperáveis e precisos?
- Os dados são consistentes e completos para resolver o problema?

O resultado do estágio de formalização é uma especificação parcial de uma de base de conhecimento.

Estágio de Implementação.

A implementação envolve o mapeamento do conhecimento formalizado do estágio anterior em uma representação "framework" associada à ferramenta escolhida para o problema. Como o conhecimento no "framework" é consistente, compatível e organizado para definir um fluxo e controle de informação, ele torna-se um programa executável. Na formalização alguns erros ou implementações não realizáveis podem ocorrer e, deverão ser eliminados nesta fase.

Estágio de Teste.

Neste ponto faz-se a avaliação do protótipo e da forma de representação nele utilizada. Os elementos do sistema especialista que usualmente causam problemas são:

- Características de entrada e saída,
- Regras de inferência,
- Estratégia de controle,
- Exemplo de teste.

Caso a avaliação conclua que há problemas ou erros deve-se retornar aos estágios anteriores e recomeçar a tarefa de aquisição.

2.5.3. Construção do Sistema Protótipo

A construção do sistema protótipo exige que o engenheiro do conhecimento se preocupe com os seguintes tópicos:

- Familiarizar-se com o problema antes de começar exaustivas iterações com o especialista.
- Identificar e caracterizar claramente os aspectos importantes do problema.
- Identificar o protocolo detalhado usado pelo especialista na solução do problema.
- Escolher uma ferramenta que minimiza os desencontros de representação dos sub-problemas.
- Iniciar a construção do protótipo assim que o primeiro exemplo for entendido.
- Trabalhar intensamente com o núcleo do conjunto de representações dos problemas.

- Identificar e separar as partes do problema que causaram problemas de I.A. no passado.
- Separar o conhecimento da área específica do conhecimento global de solução de problemas. Buscar simplicidade na máquina de inferência.
- Inicialmente não se preocupar com o tempo de execução e espaço de memória.
- Encontrar ou fazer as ferramentas que o auxiliem no processo de escrita de regras.
- Tomar cuidado com a documentação.
- Não esperar até que as regras formais estejam prontas.
- Durante o teste, procurar erros nas entradas e saídas, regras de inferência, estratégia de controle, exemplos de teste.

2.5.4. Estendendo a Versão Protótipo do Sistema

Quando o engenheiro de conhecimento for estender o protótipo do sistema especialista, ele deverá se preocupar em construir uma interface amigável. Deverá dar-lhe capacidade para examinar o conhecimento do sistema. Prover capacidades de explanar a linha de raciocínio.Deverá também manter uma biblioteca de casos apresentados ao sistema especialista, juntamente com os resultados alcançados.

Encontrando e Escrevendo Regras.

Para escrever as regras, além de falar com o especialista, devem-se seguir os exemplos dados pelo mesmo. Os termos e os métodos que o especialista usa devem ser utilizados no sistema especialista.

Mantendo o Especialista Interessado.

Engajar o especialista no desafio de projetar uma ferramenta útil é muito importante, pois mantém o especialista interessado. Isolar o especialista, bem como o usuário, dos problemas técnicos deixa o especialista mais a vontade para discutir assuntos específicos do domínio.

2.5.5. Construindo o Sistema final.

Com a implementação do protótipo concluida pode-se iniciar a construção do sistema final a partir da estensão protótipo. Na tarefa de construir o sistema final o engenheiro do conhecimento deve levar em consideração os seguintes pontos:

- Generalidade: tornar o sistema mais abrangente que o protótipo.
- Identificação com os usuários interessados no sistema final.
- Sistema de entrada e saída familiar ao usuário.

A partir deste ponto o sistema vai crescer a medida em que se incluir mais conhecimento.

2.5.6. Avaliação do Sistema.

Na avaliação do sistema deve-se analisar o sucesso do seu esforço tão logo quanto possível. Deve-se entender como o especialista avaliaria a performance do sistema. A interface com o usuário é crucial para a aceitação do sistema pelos futuros usuários.

2.6. Ferramentas para a Construção de Sistemas Especialistas

Mostra-se a seguir as características de oito ferramentas para a construção de sistemas especialistas; EMICYN, KAS, EXPERT, OPS5, ROSIE, RLL, HEARSAY III e AGE. Nesta análise serão consideradas técnicas de projeto, representação do conhecimento e características de desempenho.

O EMYCIN possui um ambiente de trabalho bom e uma boa interface com o usuário, principalmente para sistemas de diagnóstico especializado. Possui mecanismos de aquisição automática de conhecimento, permite rápida construção da base de conhecimento, possui módulos de consulta e explanações. O EMYCIN só possui mecanismo de inferência do tipo "backward chaining".

O KAS possui modelos de representação em forma de redes semânticas. Modelo de inferência em "forward chaining" e "backward chaining". Este sistema possui procedimentos de entrada e saída que facilitam o seu uso. A ferramenta não possui mecanismos de manipulação de dados variando no tempo ou de múltiplos objetivos. Tem um sistema de entrada de conhecimento sofisticado que facilita muito o processo de aquisição do conhecimento. Além disto o KAS tem mecanismos de aumentar a base de conhecimento bem como, reconhecer sinônimos e resumir o conhecimento.

O EXPERT possui uma interface com o usuário bastante flexível. A maior vantagem do EXPERT é permitir uma rápida construção de protótipos. O EXPERT é muito usado para resolver problemas de classificação. A estrutura do EXPERT não permite soluções simples, e é necessário se implementar um "backtrack" em outra linguagem. Permite implementar objetivos múltiplos mas com um dispendio muito grande de esforço.

O OPS5 não tem possibilidades de expandir ou fazer explanações sobre a base de conhecimento, mas pode processar objetivos múltiplos e possui uma grande generalidade na linguagem. Não tem uma estratégia pronta para a solução de problemas, mas possui uma boa capacidade de encontrar padrões. Contudo, a falta de um ambiente de programação é uma das maiores falhas desta ferramenta.

O ROSIE não possui mecanismos de estruturação e construção da base de conhecimento, mas tem rotinas de entrada e saída e de explanação. A maior vantagem é a sintaxe "English like". Possui um bom sistema de reconhecimento de padrões. Tem capacidade de acessar rotinas do sistema operacional local. No ROSIE há falta de acesso às próprias regras e ao mecanismo de controle e há falta de uma estratégia de controle. Esta ferramenta não possui a habilidade de manipular operações assíncronas e concorrentes.

O RLL possui um esquema de entrada e saída muito rudimentar, poucos recursos de explanação e construção da base de conhecimento. Por outro lado possui um modelo competente de programação e generalidade da estrutura de dados e algoritmos. O RLL pode raciocinar sobre a sua própria estrutura. O RLL possui mecanismos de modificação e flexibilização dos modelos; no entanto esta flexibilidade causa dificuldades de escolha e definição de estruturas.

O HEARSAY III possui mecanismos de entrada e saída fracos, esquemas de construção da base de conhecimento e de explanação rudimentares. No entanto possui a capacidade de manipular dados que estão chegando e de tratar objetivos paralelos. A sua maior vantagem é a uma estrutura genérica de representação do conhecimento que permite representar conhecimento de várias formas. Por outro lado, a falta de uma linguagem de representação de alto nível é uma falha.

O AGE apresenta poucos recursos de entrada e saída, de construção da base de conhecimento e de explanação. Possui mecanismos de aplicação de "frameworks" gerais e tem dois sistemas de controle: o "blackboard-framework" e o "back-chaining framework"; caso o sistema que se pretende construir se encaixe em algum destes "frameworks", sua implementação ficará muito facilitada. O AGE possui uma flexibilidade de representação da dados. Sua maior falha é não poder tratar de problemas que não tenham estrutura "framework".

2.6.1. Seleção da Ferramenta Apropriada.

Na seleção da ferramenta a ser usada para implementar um sistema especialista devem ser considerados os seguintes pontos:

- Generalidade: Quanto mais genérico for o sistema especialista, maiores serão as dificuldades de representar o conhecimento. Muita generalidade causa dispersão, portanto a ferramenta deve ser especializada o suficiente para o problema.
- Teste: A construção de um sistema dispende muito esforço. Portanto é necessário testar sua validade, durante todo o processo de desenvolvimento, evitando caminhos que não levem a um bom desempenho do sistema.
- Acessabilidade: Possibilidade de manutenção da ferramenta dada pelo fornecedor desse software.
- Velocidade de desenvolvimento: Se o projeto do sistema especialista tem urgência a velocidade é critica. Portanto a ferramenta deve conter internamente recursos de interação e capacidade de explanação. Isto não se reflete só na velocidade, mas também num sistema mais inteligível.

Um dos maiores problemas na escolha de ferramentas é o de adequar as características do problema com as da ferramenta. Os requisitos necessários são dependentes de três características:

- Características do domínio do problema,
- Características da proposta de solução do problema,
- Características desejáveis do sistema especialista a construir.

2.7. Avaliação do Sistema Especialista

O desenvolvimento e implementação de um sistema especialista envolve constante avaliação dos progressos conseguidos, considerando questões como:

- O esquema de representação de conhecimento é adequado ou deve ser modificado ou estendido?
- O sistema está crescendo, gerando respostas certas usando raciocínios coerentes?
- O conhecimento contido é consistente com o dos especialistas?
- Os usuários tem facilidade de interação com o sistema?
- Quais as recursos e capacidades que os usuários precisam?

Durante o processo de desenvolvimento e implementação, o conhecimento é alterado, somado, retirado, refinado etc... Isto faz com que a avaliação seja um fator importante no desenvolvimento. Os sistemas especialistas devem ser avaliados para testar a utilidade e precisão dos resultados, para verificar se o sistema é fácil de se interagir, verificar a inteligibilidade e credibilidade do sistema, em suma, a sua eficiência.

A avaliação auxilia o especialista no entendimento do assunto de sua experiência, promovendo a comunicação entre os membros da equipe de desenvolvimento. Os usuários podem testar a competência do sistema e determinar se produz resultados inteligíveis. A avaliação auxilia o usuário na decisão de quais capacidades o sistema deve ter. A avaliação resulta algumas vezes na iniciação de novas pesquisas ou esforços de desenvolvimento.

Princípios de Avaliação

As avaliações do sistema especialista deverão ser constantes, sendo instrutivas para se verificar como estão seu desempenho e seus resultados.

Para avaliar um sistema especialista são seguidos quatro princípios básicos:

- Princípio 1: Objetos complexos ou processos não podem ser avaliados por critérios simples ou números,
- Princípio 2: Um grande número de critérios distintos, avaliados ou medidos, geram mais informação facilitando a avaliação como um todo,
- Princípio 3: Algumas pessoas, dependendo de seu interesse específico, podem discordar do significado dos critérios usados na avaliação,
- Princípio 4: Qualquer coisa pode ser medida experimentalmente na medida em que se saiba como as medidas foram definidas.

Para se obter sucesso no processo de avaliação devem ser evitados os problemas:

- Os avaliadores podem falhar na avaliação do que está sendo avaliado,
- Os avaliadores podem falhar na indicação dos objetivos da avaliação,
- Casos pré-selecionados podem tornar tendenciosos os resultados das interpretações dos avaliadores, pelo estreitamento da faixa de problemas,
- Os avaliadores podem falhar na seleção de um contra padrão apropriado que compara o desempenho do sistema especialista,
- Os avaliadores tendem a generalizar os resultados obtidos num ambiente reduzido,
- Pode existir confusão sobre os objetos de estudo,
- A avaliação pode ser inapropriada para o estágio de desenvolvimento do sistema,

- Existem algumas dificuldades inerentes ao projeto de testes elegantes,

2.7.1. Quando Avaliar Sistemas Especialistas.

A avaliação deve ser contínua durante os nove estágios abaixo, que resumem a construção de um sistema especialista:

- Durante o projeto de alto nível,
- Durante a implementação do sistema protótipo,
- Durante o refinamento do sistema,
- Durante a fase de teste informais,
- Quando se for tornar mais amigável a interface do protótipo do sistema especialista com o usuário,
- Na revisão do sistema especialista com realimentação do usuário,
- Na avaliação estruturada do desempenho,
- Na avaliação estruturada da aceitabilidade pelos usuários,
- Durante o funcionamento em serviço, por um longo período de tempo em ambiente de protótipo,
- Durante a condução dos estudos para demonstrar a larga utilização do sistema,
- Quando forem necessárias mudanças no programa para permitir larga distribuição,
- Na liberação final e "marketing" com planos de manutenção e expansão.

2.7.2. Considerações Relacionadas a Avaliação de Sistemas Especialistas.

Cada sistema especialista possui características próprias e portanto uma avaliação própria.

A tecnologia de sistemas especialistas ainda está em processo de disseminação, portanto muitos critérios como o quanto corretas são suas respostas, qual a sua eficiência, qual sua capacidade de comunicação com o usuário, usados para avaliações de seres humanos, também podem ser usados para a avaliação desses sistemas. Contudo, se ninguém sabe ao certo como avaliar especialistas humanos, que dirá sistemas especialistas!

CAPÍTULO TRÊS

3. SISTEMA ESPECIALISTA PARA VERIFICAÇÃO DE REGRAS ELÉTRICAS

3.1. Introdução.

O sistema especialista CREPE pode ser utilizado em duas das fases componentes da metodologia de projeto de circuitos integrados:

- A primeira utlilização é durante a fase de verificação de regras elétricas do projeto elétrico de células, antes da simulação. Nesta fase o projetista pode verificar os possíveis erros de topologia, isto é, erros decorrentes somente da estrutura de ligações do circuito. Estes erros de projeto são do tipo: curto circuito, circuito aberto, capacitâncias, dimensionamento de transistores e outros.
- A segunda utilização é após o layout pronto e extraido. O sistema pode verificar se o layout está coerente, sem erros de topologia, acusando erros antes da simulação de caracterização elétrica da célula.

O processo de verificação, antes da simulação, poupa o projetista da árdua tarefa de encontrar problemas de topologia de circuitos após a simulação. O sistema pode também detectar erros que a propria simulação mascara, isto ocorre porque os modelos de simulação nem sempre são completos.

O programa, além de gerar os avisos de possíveis problemas, gera uma lista de elementos que foram identificados como, por exemplo: portas lógicas, inversores, circuitos de memória ou elementos pré-definidos. Isto facilita o processo de análise e entendimento do circuito, principalmente em tarefas de engenharia reversa de circuitos, onde é necessária uma análise mais elaborada.

A entrada de dados que descreve o circuito é uma lista de transistores e capacitores, os quais representam os principais elementos dos circuito digitais em tecnologia CMOS.

Sobre a descrição estrutural do circuito são aplicadas regras, deduzidas a partir do conhecimento de especialistas e da literatura existente. Estas regras são aplicadas em em seqüência, analisando três grandes categorias de erros de projeto:

- categoria de regras de conectividade e dimensionamento,
- categoria de erros de "timing" e cargas,
- categoria de regras heurísticas adquiridas a partir do conhecimento dos especialistas.

Durante o processo de classificação e de aplicação de regras o programa vai acumulando resultados intermediários como, por exemplo: fatos, inferências e informações. Estes elementos serão utilizadas na análise das regras e principalmente na tarefa de explanação dos resultados encontrados, numa espécie de explicação da linha de raciocínio que o programa usou para chegar a um dado resultado.

O processo de auditoria informa ao projetista os possíveis erros. Isto não significa que o projeto está incorreto ou não vai funcionar, somente indica que foram detetadas situações anômalas que, em circuitos padrões, não são

permitidas. Quando for detetado um possível problema, o sistema informa ao projetista o transistor ou conjunto de transistores em questão.

O capítulo três está dividido em três partes. A primeira é uma introdução à linguagem PROLOG usada em quatro dos cinco módulos que implementam o sistema. A segunda parte é uma descrição geral da arquitetura e do fluxo de controle e dados do sistema. A terceira parte é o detalhamento dos módulos componentes em termos de: funções, entradas, saídas e implementação.

3.2. Linguagem PROLOG

Neste item serão abordados alguns aspectos da linguagem PROLOG, dando noções básicas dos componentes da linguagem que são importantes para o entendimento do sistema CREPE.

3.2.1. Visão geral

O PROLOG é uma linguagem voltada para símbolos, para a computação não numérica, que é especialmente aplicável à resolução de problemas que envolvam objetos e relações entre objetos. O PROLOG possui uma característica peculiar, a de ser possível se entender os resultados do programa sem exatamente saber como o sistema chegou a um dado resultado. A habilidade do PROLOG de tratar de muitos detalhes procedurais autonomamente, facilita a construção do programa do ponto de vista procedural. Isto faz com que o programador se preocupe mais com o ponto de vista declarativo da linguagem, que é mais simples de entender que o lado procedural.

Como a lógica, o PROLOG é usado para expressar fatos e relações entre fatos e para inferir soluções para os problemas. Veja um exemplo:

Sócrates é mortal ?

- 1. Sócrates é um homem.
- 2. Todo homem é mortal.
- 3. Sócrates é mortal.

O exemplo mostra a sentença 1 como um fato, a sentença 2 como uma regra e a sentença 3 como uma inferência.

Os fatos representam algo que contém verdades. Regras expressam relações entre fatos, uma relação é verdadeira se somente se as outras relações da regra são verdadeiras.

Em PROLOG os fatos são relacionados como cláusulas. Os componentes de uma cláusula são o predicado e os argumentos. O predicado descreve a relação entre os argumentos. Exemplo:

- homem(Socrates).
- gosta(João,Maria).

Um programa PROLOG possui três interpretações semânticas:

- Interpretação declarativa.
- Interpretação procedural.
- Interpretação operacional.

Interpretação declarativa.

São as cláusulas que definem o programa e descrevem a teoria de primeira ordem [25]. Isto permite ao programador modelar problemas através de assertivas acerca dos objetos de domínio do discurso, simplificando a tarefa de programar.

Interpretação procedural.

Nesta interpretação as cláusulas são vistas como comandos para um método de refutação [25]. Esta permite ao programador identificar e descrever o problema pela redução do mesmo em subproblemas, através de uma definição de uma série de chamadas de procedimentos.

Interpretação operacional.

As cláusulas são vistas como comandos para um particular procedimento de refutação. Esta interpretação reintroduz a idéia de controle de execução através da ordem das cláusulas.

3.2.2. Fluxo do programa

Os três conceitos fundamentais do programa PROLOG são: o sucesso, a falha e unificação.

Falha e sucesso.

Os conceitos de falha e sucesso estão relacionados com o "backtrack" [24]. Quando o programa é questionado para satisfazer um objetivo ou uma consulta, este inicia procurando a resposta ou o objetivo através dos fatos e das regras. Os objetivos podem ter sucesso ou não. Se um objetivo falha o PROLOG faz um "backtrack" com o intuito de satisfazer o primeiro objetivo que vem imediatamente antes do subobjetivo que falhou. Se este não puder ser satisfeito são feitos "backtracks" até que ocorra uma das situações abaixo:

1. Todas as tentativas de satsifazer o objetivo principal e dos subobjetivos falharam. Neste caso o objetivo falhou.

2. Um objetivo ou subobjetivo é satisfeito, neste caso o processo continua através dos subobjetivos. Se a solução satisfaz todos os subobjetivos então esta solução satisfaz o objetivo principal. Caso contrário faz "backtrack".

Unificação.

Em linguagens convencionais, uma característica básica é o assinalamento de valores para as variávies. No PROLOG é usado um tipo genérico de assinalamento que é a unificação. A unificação é um processo de assinalamento

que pode falhar ou ter sucesso. Se tiver sucesso, os termos unificam e então são equivalentes.

3.2.3. Estrutura de dados

O PROLOG contém a habilidade de construir e manipular estruturas complexas compostas de estruturas simples. Os tipos de dados incluem variáveis, átomos e estruturas. A figura a seguir mostra os tipos de estruturas de dados em PROLOG.

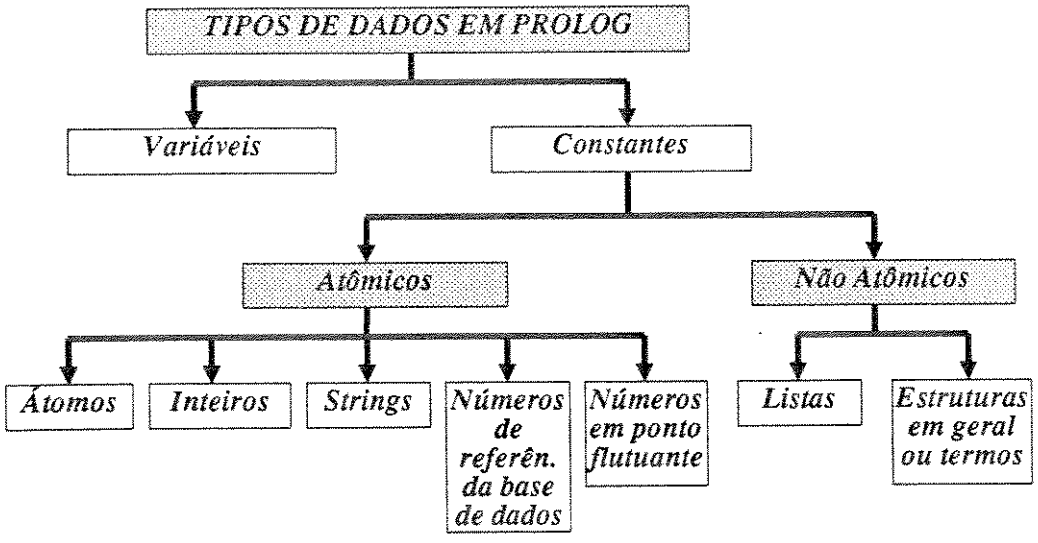


Figura 3.1 Estruturas de dados PROLOG

3.2.4. Controle do programa

Em PROLOG, o programador controla o caminho pelo qual o programa vai seguir ordenando as cláusulas de acordo com o seu planejamento e os objetivos de cada cláusula. O PROLOG procura satisfazer todos os subobjetivos antes de ir para outro objetivo, em outros termos, isto é um processo de busca em profundidade ("depth-first search"[34]).

3.3. Módulos e Fluxo de Dados.

Cada sistema especialista possui, para fins de implementação, uma arquitetura própria. A arquitetura do sistema proposto [12] está definida em termos de módulos (funções) e fluxo de dados entre os módulos e, com a interface com o usuário. O diagrama de blocos desta arquitetura está representado na figura 3.2.

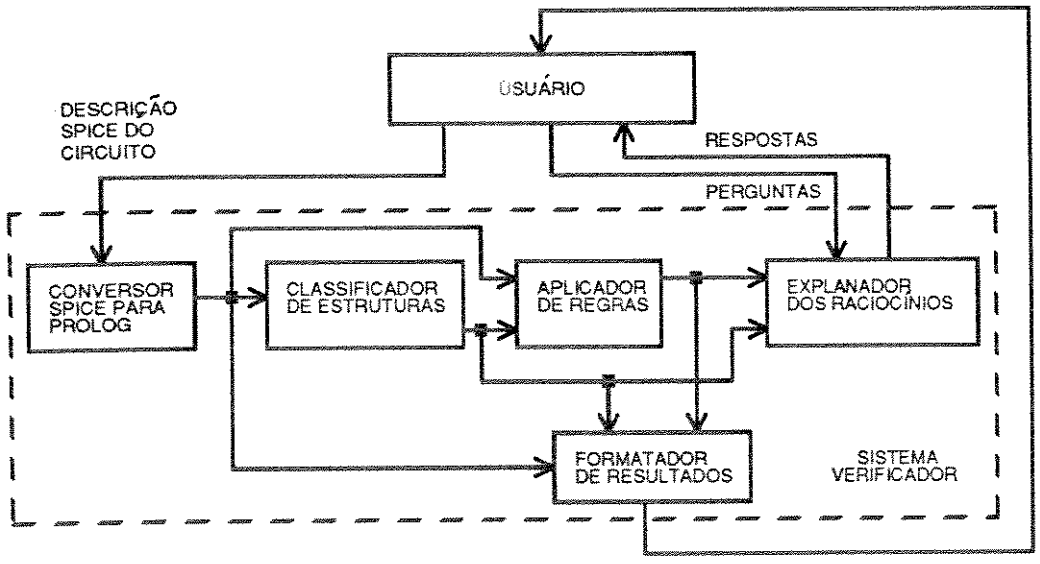


Figura 3.2 Módulos e fluxo de dados do CREPE.

Como mostra o diagrama de blocos da figura 3.2, o usuário participa do processo, fornecendo a descrição SPICE do circuito. O usuário também participa, no final da execução do programa, quando podem ser feitas perguntas ao sistema para explanações dos resultados alcançados.

O módulo conversor SPICE-PROLOG lê a descrição SPICE e faz algumas perguntas ao usuário como:

- Qual é o número do nó de VDD ?
- Qual é o número do nó de VSS ?
- Qual é o número do nó de CK ?
- Quais são os números dos nós de entrada ?
- Quais são os números dos nós de saída ?

A partir destas informações o módulo gera cláusulas para o classificador de estruturas, para o aplicador de regras e para o formatador de resultados.

O módulo classificador de estruturas interpreta as cláusulas geradas pelo conversor SPICE-PROLOG e identifica os elementos que foram pré-definidos como: inversores, portas lógicas etc... São geradas cláusulas, uma para cada elemento identificado. Estas cláusulas geradas são passadas para o aplicador de regras e para o formatador de resultados.

O módulo aplicador de regras é o cerne do sistema. Nele está a base de conhecimento adquirida dos projetistas. Nesta base de conhecimento estão as regras de projeto elétrico a serem verificadas pelo sistema. Aqui são aplicadas as regras sobre o circuito definido pelas cláusulas de transistores e capacitores e também pelas cláusulas geradas pelo classificador de estruturas. São gerados avisos na medida em que forem sendo encontrados possíveis problemas de violações de regras elétricas. O avisos gerados são cláusulas que serão posteriormente usadas pelos módulos explanador e formatador resultados.

O módulo explanador de raciocínios é um módulo interativo com o usuário onde podem ser feitas perguntas e são dadas respostas. Este módulo mostra o processo de raciocínio que o programa seguiu para chegar a um dado resultado.

O último módulo é o formatador de resultados onde os resultados dos outros módulos, com exceção do explanador, são organizados de maneira a serem facilmente entendidos pelos usuários.

Os ítens a seguir definem os módulos e entram em detalhes de implementação e de teoria envolvidos em cada um destes módulos.

3.4. Módulo Conversor de Formatos.

Existem muitas formas de se descrever circuitos a nível de transistores, e também muitas linguagens de descrição. A descrição SPICE tem sido uma das mais utilizadas devido ao fato de ser o SPICE o simulador elétrico mais conhecido atualmente. SPICE passou a ser sinônimo de simulação elétrica. Neste contexto foi escolhida uma descrição tipo SPICE como entrada de dados para o programa verificador de regras elétricas.

O sistema aceita qualquer descrição SPICE que não contenha chamadas de subcircuitos.

O módulo conversor de formatos só interpreta os transistores e capacitores do circuito, que em circuitos CMOS são os componentes relevantes a nível elétrico e lógico.

Este módulo deve cumprir duas funções principais:

- Gerar as cláusulas PROLOG, que descrevem a topologia e os elementos do circuito,
- Interpretar os dados de entrada e representá-los de modo a facilitar as tarefas de classificação de estruturas e de aplicação de regras e, como resultado desta interpretação, gerar cláusulas de interligações e cláusulas de estatísticas.

3.4.1. Cláusulas PROLOG.

O conversor gera dez cláusulas PROLOG sobre a descrição do circuito descritas abaixo:

- vdd(no).

- vss(no).
- ck(no).
- ent([H | T] [1]).
- sai([H | T]).
- mos(n, d, g, f, b, tipo, l, w).
- cap(no, valor).
- no([H | T]).
- numnos(n).
- numtrans(n).

Cláusulas vdd, vss, ck, ent e sai.

Estas cláusulas são geradas a partir da interação com o usuário onde o programa pergunta o número do nó de VDD, o número do nó de VSS, o número do nó de CLOCK, os nós de entrada e os nós de saída.

O formato das cláusulas está mostrado abaixo:

- vdd (x). onde x é o nó associado,
- vss (x). onde x é o nó associado,
- ck (x). onde x é o nó associado,
- ent ([H | T]). onde H | T é uma lista de nós de entrada,
- sai ([H | T]). onde H | T é uma lista de nós de saída.

1 [H | T] *representação de listas em PROLOG.*

Cláusula transistor (mos).

Esta cláusula representa um transistor com as suas ligações e atributos retirados da descrição SPICE do transistor. O transistor é representado pelo SPICE com o seguinte formato:

Mn D G F B TIPO L W AD AF PD PF

onde:

- M indica a primitiva transistor mos,
- n é o número do transistor,
- D é o número do nó de dreno do transistor,
- G é o número do nó de gate do transistor,
- F é o número do nó de fonte do transistor,
- B é o número do nó de substrato do transistor,
- TIPO é o tipo do transistor, PMOS ou NMOS,
- L é o comprimento do canal do transistor,
- W é a largura do canal do transistor,
- AD é a área do terminal de dreno,
- AF é a área do terminal de fonte,
- PD é o perímetro do terminal de dreno,
- PF é o perímetro do terminal de fonte.

A cláusula gerada é do seguinte tipo:

mos (n, d, g, f, b, tipo, l, w).

onde n, d, g, f, b, tipo, l, w são os parâmetros descritos anteriormente.

Cláusula capacitância (cap).

Esta cláusula é o resultado do cálculo da capacitância de cada nó do circuito. Esta capacitância é calculada levando-se em conta os seguintes dados:

- L, W dimensões do retângulo do canal,
- AD, AF áreas de dreno e fonte,
- PD, PF perímetro de dreno e fonte.

Os parâmetros abaixo são os dependentes de processo e se encontram no arquivo de tecnologia:

- CJAp e CJAn capacitância de junção por unidade de área.
- CJPp e CJPn capacitância de junção por unidade de perímetro.
- COX capacitância por unidade de área do canal.

A capacitância do nó de gate é calculada pela fórmula abaixo:

$C_{gate}=C_{ox} \times L \times W$ (1).

As capacitâncias de nó de dreno e fonte, dos transistores PMOS, são calculadas pelas fórmulas abaixo:

Para transistores PMOS;

$C_{dreno}=AD \times CJA_p + PD \times CJP_p$ (2).

$C_{fonte}=AF \times CJA_p + PF \times CJP_p$ (3).

OBS: para transistores NMOS os parâmetros usados são CJA_n e CJP_n.

A capacitância de trilha é um valor da descrição SPICE dado pela primitiva cap.

Finalmente as capacitâncias são somadas para cada nó. Por exemplo um nó que possui um terminal de dreno, um de fonte e um de gate ligados a ele. A capacitância individual por terminal é calculada por (1), (2) e (3) e são somadas. A esta soma parcial é adicionada o valor da capacitância das trilhas associadas, dadas pelas primitivas "cap" do SPICE.

A cláusula gerada é a "cap" que tem o seguinte forma:

cap(no, valor).

onde:

valor é dado em Faradays.

no é o número do nó.

Cláusulas numnos e numtrans.

Estas cláusulas são a somatória dos nós e a somatória dos transistores do circuito e forma implementadas para fins de estatística.

Cláusula de estrutura de nós.

As cláusulas de estrutura de nós relacionam o nó ao transistor e ao terminal do transistor, isto é, uma lista que tem como cabeça o número do nó, e como cauda listas de dois elementos: o número do transistor e o tipo de terminal (dreno, fonte ou gate).

O objetivo destas listas é facilitar a tarefa de busca de informação pelas cláusulas PROLOG. Esta estrutura organiza as ligações de maneira que fica fácil identificar o tipo de ligação, a quantidade de ligações e quais transistores estão ligados a um dado nó.

A estrutura destas listas é mostrada a seguir

no([n1, [t1, g], [t2, d],... [tn, f]]).

no([n2, [t1, d], [t2, f],... [tn, g]]).

no([n3, [t1, g], [t2, f],... [tn, f]]).

.....

no([nn, [t1, g], [t2, g],... [tn, g]]).

Existe um exemplo que deixa bem clara a vantagem desta estrutura, que é o de se identificar dois transistores em série. Para que dois transistores estejam em série é preciso que:

- Os dois transistores tenham nó de dreno ou fonte em comum.
- O nó em comum seja único no circuito.

Em uma busca sequencial seriam percorridos todos os dados da descrição para se identificar um nó que seja único. Com esta descrição de listas basta verificar se o nó possui somente duas ligações e se estas são com dreno ou fonte dos transistores.

Exemplo:

no ([1, [2, d], [5, f]]).

Os transistores T2 e T5 estão em série, como nó 1 em comum.

Este módulo foi implementado em linguagem C, que foi escolhida pelo fato da facilidade de se manipular strings, e também por ser conhecida pelo autor da dissertação.

3.5. Módulo Classificador de Estruturas.

O processo de identificação e classificação dos elementos de ciruitos digitais é a base para uma posterior verificação de regras elétricas. Este processo se desenvolve da seguinte maneira; inicia-se de uma composição simples de componentes, como dois pares de transistores em série ou paralelo. Com os pares procura-se as relações de topologia destes pares para se identificar estruturas maiores, como portas lógicas. A partir das portas lógicas pode-se identificar os elementos de memória ou outros elementos mais complexos.

Este processo é necessário na medida em que se reduz gradualmente o espaço de busca para cada elemento pré-definido a ser identificado. Para a identificação são usadas as cláusulas "mos" e "nos" geradas pelo conversor SPICE-PROLOG.

O número de estruturas a serem identificadas depende do tipo de regra a ser aplicada. Por exemplo, para se aplicar uma regra a estruturas dinâmicas tem-se que identificar estas estruturas. Portanto para cada regra que necessita uma identificação existe um identificador de elemento. Na medida em que forem aparecendo mais regras de projeto elétrico os identificadores podem ser incluídos na base de conhecimento.

O classificador gera tantas cláusulas quantos forem os elementos pré-definidos. Durante o processo de identificação dos elementos são calculados algums parâmetros como, tempos de subida e de descida das portas lógicas, dos inversores, dos buffers etc.

Regras de identificação

As regras que identificam os elementos de circuito são descrições da topologia mais usadas em circuitos integrados CMOS digitais que são basicamente:

- Inversores
- Gates de transmissão
- Buffers
- Portas Nor
- Portas Nand
- Latch

A regras que identificam os buffers, inversores e gates de transmissão são baseadas em uma topologia simples e definida por uma única regra para cada elemento. Por exemplo a regra que identifica o inversor:

```
inversor:-
        vdd(Vdd),
        vss(Vss),
        mos(M1, D1, G1, F1,_ ,p ,_ ,_ ),
        mos(M2, D2, G2, F2,_ ,n ,_ ,_ ),
        G1 == G2,
        same_set( [D1,F1], [O,Vdd] ),
        same_set( [D2,F2], [O,Vss] ),
        assertz ( inv (N, M1, M2, G1, O) ),
        fail.
inversor.
```

Esta regra mostra a topologia de um inversor, onde o transistor PMOS dado pela primeira cláusula mos e transistor NMOS dado pela segunda cláusula mos, estão ligados a VDD e VSS conforme a configuração dada pela cláusula

same_set. A cláusula same_set indica a intercambiabilidade dos nós de dreno e fonte com as ligações do nó de saída (nó O) e o VDD ou VSS. Além disto os nós de gate G1 e G2 devem estar conectados. Satisfeitas estas condições podemos incluir na base de conhecimento o inversor encontrado, isto é feito pela cláusula assertz, por último é colocado um "fail" para que se tenha "backtracks" e se encontrem todos os inversores do circuito.

As regras que identificam portas lógicas exigem uma pré-identificação de elementos. A pré-identificação consiste em encontrar pares de transistores do mesmo tipo em configurações série ou paralelo. Após a identificação de todos os pares série ou paralelo são aplicadas regras semelhantes as do inversor só que são considerados, ao invés de transistores, os pares paralelo ou série em uma topologia tal que se identifique as portas lógicas.

A regra que identifica um latch é uma composição de inversores e transistores. Este módulo auxilia na identificação de estruturas em um circuito extraído de um layout pronto. Isto é muito útil em processos de engenharia reversa de circuitos.

3.6. Módulo Aplicador de Regras.

Este módulo é o componente principal e o mais complexo do sistema. Sendo assim, a sua descrição foi dividida em três partes:

- Entradas.
- Regras verificadas e os conceitos teóricos envolvidos.
- Saídas.

3.6.1. Entradas

As entradas do módulo são provenientes de dois módulos; o conversor SPICE-PROLOG e o classificador de estruturas. Do conversor SPICE-PROLOG são usadas as cláusulas " mos ", " vdd ", " vss ", " ck ", " ent", " sai " e " cap ". Do classificador de estruturas são usadas as cláusulas dos elementos pré-definidos identificados, e também os cálculos feitos para cada elemento identificado.

3.6.2. Regras verificadas

Uma divisão em categorias de regras foi implementada para facilitar o processo de análise das regras. Outro fator que levou a divisão em categorias é a possibilidade de se selecionar uma ou duas ou todas as categorias para uma verificação seletiva.

A divisão foi a seguinte:

- Regras tipo I : regras relacionadas a conectividade.
- Regras tipo II : regras relacionadas ao tamanho dos transistores.

- Regras tipo III : regras relacionadas com as cargas, regras de "timing" e regras heurísticas.

Regras tipo I.

Estas regras estão encarregadas de verificar a coerência das ligações do ponto de vista topológico. E verificam parâmetros mínimos e máximos dos transistores e capacitores do circuito.

As regras do tipo I para efeitos de implementação foram subdivididas em dois tipos:

- Regras de verificação de parâmetros.
- Regras de verificação de conectividade.

Regras de Parâmetros.

Estas regras verificam se o comprimento (l) e a largura (w) dos transistores estão dentro de uma faixa admissível.

Outro parâmetro verificado é a capacitância máxima admissível para um nó do circuito digital.

Os parâmetros, tanto da capacitância como dos tamanhos, estão definidos no arquivo de tecnologia. O valores definidos são dependentes do rigor com que o sistema vai verificar as regras.

Regras de Conectividade.

Estas regras verificam se as ligações entre os transistores estão coerentes. Os possíveis problemas que o circuito pode conter são inúmeros, e neste sistema são verificados os principais. Estes possíveis erros estão listados abaixo:

- Transistor com curto entre dreno e fonte,
- Transistor com curto entre dreno e gate,
- Transistor com curto entre gate e fonte,
- Nó que contenha uma única ligação, que não seja nem entrada nem saída do circuito,
- Transistor com ligação do substrato incorreta; por exemplo um transistor PMOS com o nó de substrato ligado a VSS, onde o correto seria estar ligado a VDD,
- Transistor com dreno e fonte ligados a VDD e VSS respectivamente ou vice-versa,
- Nó com ligações somente em gates de transistores que não são nem entrada nem saída do circuito,
- Nó de clock em curto com VDD ou VSS,
- Regra do caminho CMOS, isto é, cada caminho entre VDD e VSS deve necessariamente passar por um transistor PMOS e um transistor NMOS.

Os avisos gerados por estas regras do tipo I são cláusulas nas duas formas abaixo:

1. regra_xxxx(N,'comentário',argumentos).

onde:

N é o número do transistor com problemas.

'comentário' é uma breve explanação do erro encontrado.

argumentos são por exemplo: o comprimento do canal, largura do canal, nós em curto, etc...

3.5. Módulo Classificador de Estruturas.

O processo de identificação e classificação dos elementos de ciruitos digitais é a base para uma posterior verificação de regras elétricas. Este processo se desenvolve da seguinte maneira; inicia-se de uma composição simples de componentes, como dois pares de transistores em série ou paralelo. Com os pares procura-se as relações de topologia destes pares para se identificar estruturas maiores, como portas lógicas. A partir das portas lógicas pode-se identificar os elementos de memória ou outros elementos mais complexos.

Este processo é necessário na medida em que se reduz gradualmente o espaço de busca para cada elemento pré-definido a ser identificado. Para a identificação são usadas as cláusulas "mos" e "nos" geradas pelo conversor SPICE-PROLOG.

O número de estruturas a serem identificadas depende do tipo de regra a ser aplicada. Por exemplo, para se aplicar uma regra a estruturas dinâmicas tem-se que identificar estas estruturas. Portanto para cada regra que necessita uma identificação existe um identificador de elemento. Na medida em que forem aparecendo mais regras de projeto elétrico os identificadores podem ser incluídos na base de conhecimento.

O classificador gera tantas cláusulas quantos forem os elementos pré-definidos. Durante o processo de identificação dos elementos são calculados algums parâmetros como, tempos de subida e de descida das portas lógicas, dos inversores, dos buffers etc.

Regras de identificação

As regras que identificam os elementos de circuito são descrições da topologia mais usadas em circuitos integrados CMOS digitais que são basicamente:

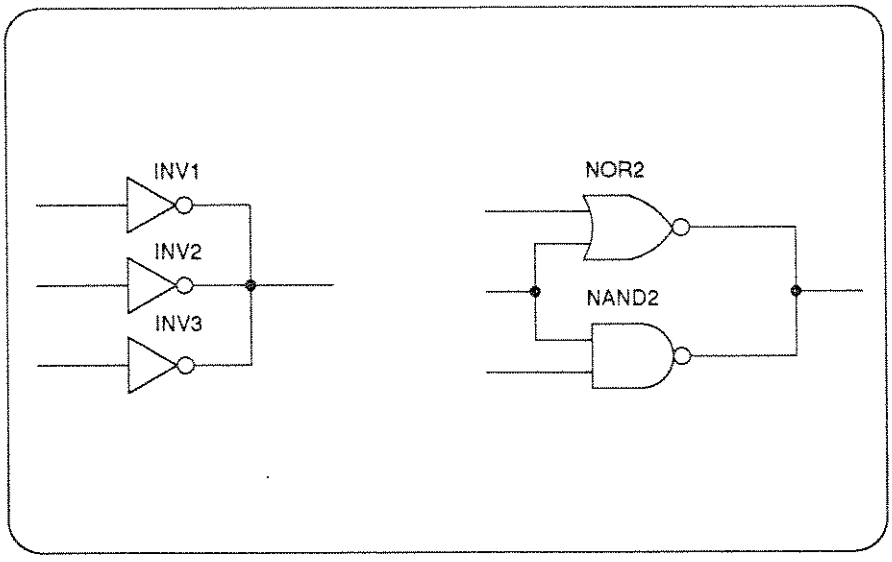


Figura 3.3 Inversores com as saídas em curto e um Nand e um Nor com saídas em curto.

Problemas com tempos de subida e descida:

Durante o projeto de células deve-se manter como premissa a seguinte afirmação:

Tempo de subida = Tempo de descida (1).

Para calcular o tempo de subida (t_r) e o tempo de descida (t_f) das portas lógicas é necessário um estudo das características de chaveamento dos transistores em configuração CMOS.

A velocidade de chaveamento está relacionada com o tempo de carga e descarga da capacitância do nó de saída da porta.

Considerando-se o inversor:

Quando o inversor chaveia existem dois tempos envolvidos na descarga da capacitância CL do nó, por exemplo chaveando a saída de 1 para 0 (tempo de descida):

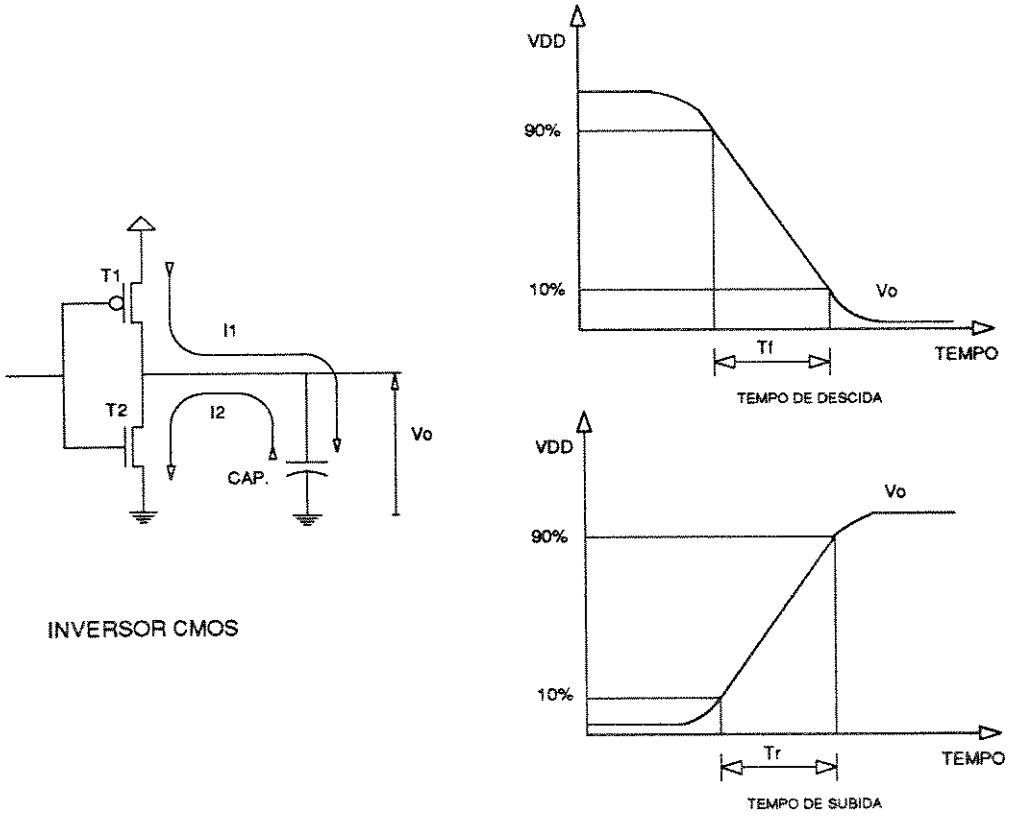


Figura 3.4 Circuito do inversor e como se caracteriza o tempo de subida e de descida do inversor.

- 1. t_{f1} quando o transistor T2 está na região de saturação até que a tensão de saída caia para VDD - Vt, descarregando o capacitor CL até este ponto.

- 2. t_f2 Quando o transistor T2 está na região linear, descarregando o capacitor CL até 10% da tensão de alimentação.

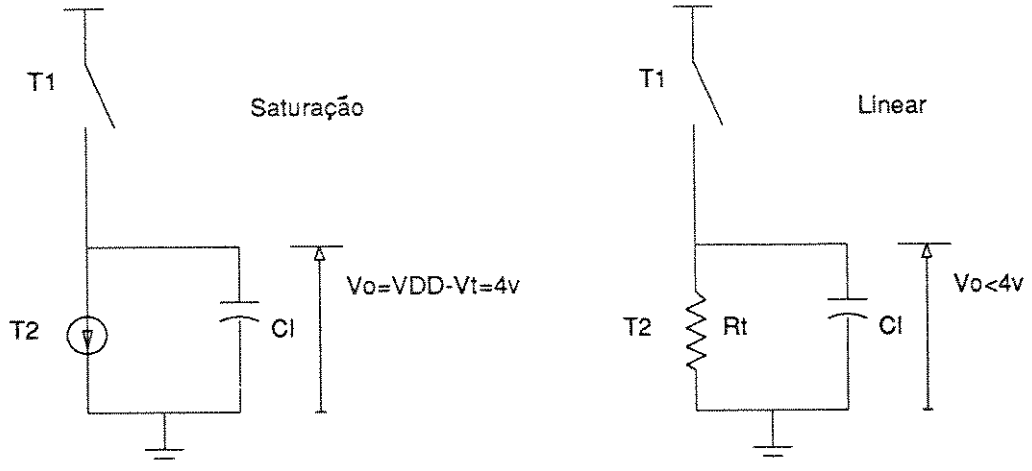


Figura 3.5 mostra os dois tempos envolvidos no cálculo do tempo de descida de um inversor, assumindo-se VDD = 5V e Vt = 1V.

Para o cálculo do tempo de descida, é utlizada a lei das malhas e integrando-se de t = t1 correspondente a 90% do VDD e t = t2 correspondente a VDD - Vt, e com a aproximação de Vt igual a 20% de VDD, tem-se a equação abaixo. Este cálculo pode ser verificado com detalhes em [27].

$$t_f = \frac{4 \times Cl}{\beta_n \times VDD}$$

onde:

$$\beta_n = K_n \left(\frac{W_n}{L_n} \right)$$

K_n constante relacionada ao processo e W_n e L_n respectivamente o comprimento e largura do canal do transistor NMOS.

Um procedimento análogo pode ser aplicado para o tempo de subida do inversor (t_r) e se obtém o seguinte:

$$t_r = \frac{4 \times Cl}{\beta \times VDD}$$

onde:

$$\beta_p = K_p \left(\frac{W_p}{W_n} \right)$$

K_p constante relacionada ao processo e W_p e L_p respectivamente o comprimento e largura do canal do transistor PMOS.

Considerando-se a premissa (1) temos:

$$\frac{4 \times Cl}{\beta_p \times VDD} = \frac{4 \times Cl}{\beta_n \times VDD}$$

donde se conclui que:

$$\beta_n = \beta_p \qquad (2)$$

Que são dados por:

$$\beta_n = \frac{\mu_n \times \varepsilon}{T_{ox} \times \frac{W_n}{L_n}} \qquad (3)$$

e

$$\beta_p = \frac{\mu_p \times \varepsilon}{T_{ox} \times \frac{W_p}{L_p}}$$

μ_n = mobilidade dos elétrons.

μ_p = mobilidade das lacunas.

t_{ox} = espessura do óxido de gate.

ε = permissividade elétrica.

Normalmente nos processos típicos a mobilidade das lacunas (μ_p) é a metade da mobilidade dos elétrons (μ_n).

$$\mu_n = 2 \times \mu_p$$

substituindo-se em (3) e assumindo-se (1) temos:

$$\frac{2 \times \mu_p \times \varepsilon}{T_{ox} \times \frac{W_n}{L_n}} = \frac{\mu_p \times \varepsilon}{T_{ox} \times \frac{W_p}{L_p}}$$

$$2\left(\frac{W_n}{L_n}\right) = \frac{W_p}{L_p} \qquad (4)$$

Para se ter um tempo de subida igual ao tempo de descida devemos manter a relação de tamanhos dada por (4).

A regra a ser aplicada verifica se os inversores seguem a relação de tamanho dada em (4). Como o processo pode variar e também os parâmetros de projeto

podem não ser muito rígidos, foi implentada uma faixa de variação permitida, por exemplo de 0.8 a 1.2 no valor β do inversor.

O estudo anterior foi feito para os inversores. A seguir é feita uma análise para um Nor de m entradas e um Nand de m entradas.

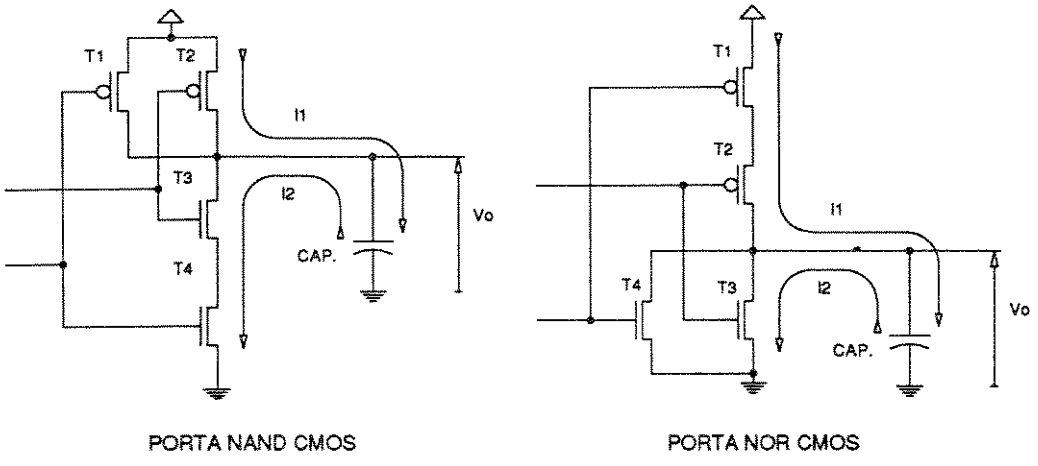


Figura 3.6 Fluxo de corrente em uma porta lógica Nor de duas entradas.

Para o caso do Nor de m entradas os transistores PMOS são os que alteram o tempo de subida, pois tem-se um aumento da resistência em série quando estes transistores estão ligados. Uma comparação pode ser feita em relação ao cálculo dos tempos de subida e descida do inversor.

$$\beta_{inv} = \frac{\beta_n}{\beta_p}$$

$$R_c = K\,(\frac{W}{L})$$

$$\beta_n = \frac{W_n}{L_n}$$

$$\beta_p = \frac{W_p}{L_p}$$

Relacionando-se o β do inversor com o β do Nor temos o seguinte:

β_n igual ao β de um dos transistores NMOS está ligado pois a resistência de descarga é a maior.

β_p igual a um β que considere os m transistores PMOS ligados, caso com a maior resistência.

Relacionando-se o tempo de subida com a constante RC dada pela resistência dos transistores em série e pela capacitância de carga do nó [27], temos:

$$t_r = m \times R_p \times (m \times Cd + Cl)$$

$$t_f = R_n \times (m \times Cd + Cl)$$

onde:

m é o número de entradas do nor.

R_n resistência relativa ao transistor NMOS.

R_p resistência relativa ao transistor PMOS.

($m \times Cd+CL$) capacitância total do nó de saída da porta nor.

Assumindo a premissa (1):

$$m \times R_p \times (m \times Cd + Cl) = R_n \times (m \times Cd + Cl)$$

$$m \times R_p = R_n$$

substituindo-se em (2) temos:

$$m \left(\frac{4}{\beta_p \times VDD}\right) = \frac{4}{\beta_n \times VDD}$$

$$\frac{\beta_p}{\beta_n} = m \qquad \text{(5) para o pior caso de tempos de subida e descida.}$$

Para o caso do nand de m entradas os transistores NMOS são os que alteram o tempo de descida, pois aumenta a resistência em série.

Relacionando-se o β do inversor com o β do nand temos o seguinte:

β_p igual ao β quando somente um dos transistores PMOS está ligado pois a resistência de descarga é a maior.

β_n igual a um β que considere os m transistores NMOS ligados em série, caso com a maior resistência.

Relacionando-se o tempo de subida com a constante RC dada pela resistência dos transistores em série e pela capacitância de carga do nó [27], temos:

$$t_r = R_p \, (m \times Cd + Cl)$$

$$t_f = m \times R_n \, (m \times Cd + Cl)$$

e assumindo a premissa (1):

$$R_p\,(m \times Cd + Cl) = m \times R_n\,(m \times Cd + Cl)$$

$$R_p = m \times R_n \qquad (5)$$

substituindo-se em (2) temos:

$$\frac{4}{(\beta_p \times VDD)} = m\left(\frac{4}{\beta_n \times VDD}\right)$$

$$\frac{\beta_p}{\beta_n} = \frac{1}{m} \qquad \text{(6) para o pior caso de tempos de subida e descida.}$$

Satisfazendo as relações (5) e (6) e o valor m, que é o número de entradas da porta lógica, pode-se verificar que os tempos de subida e de descida são iguais.

Problemas com a capacidade de carga e descarga de nós ("Fan-out")

O "fan-out" da saída de uma porta lógica é a capacidade que uma saída tem de acionar mais de uma entrada. Cada entrada acionada possui uma capacitância associada bem como a capacitância associada da saída da porta em questão. A figura 3.7 mostra uma situação característica de problema com "fan-out".

Como foi visto nos ítens anteriores a capacitância de carga, associada a saídas de portas CMOS, é um fator determinante no tempo de subida e de descida dos sinais neste nó. O problema do "fan-out" está diretamente relacionado com o tempo gasto para a carga e descarga deste capacitor, da seguinte maneira:

Se o tempo de carga ou descarga está muito alto temos um consumo excessivo de potência pois os transistores PMOS e NMOS estão na região linear. Por outro lado não podemos aumentar indefinidamente a capacidade de "drive" do buffer que aciona o nó pois tem-se perda de área de layout e problemas de ruído de chaveamento devido a grande quantidade de carga chaveada. Portanto deve

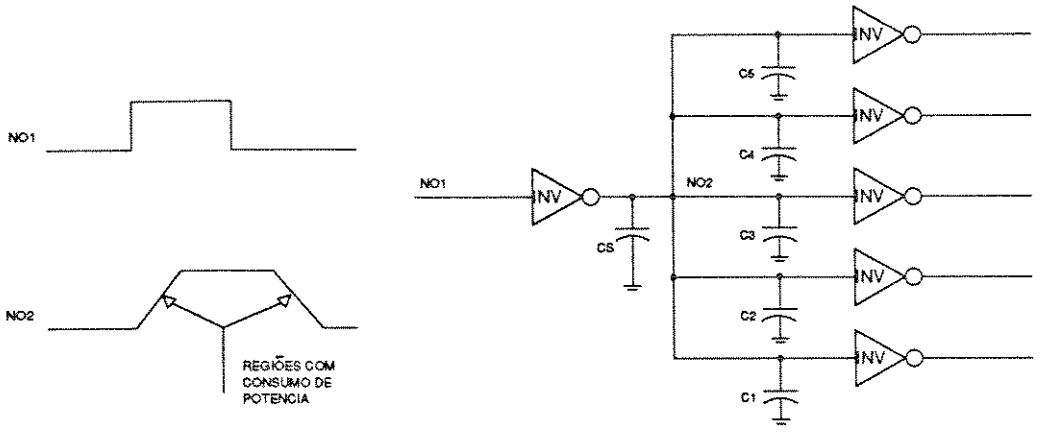


Figura 3.7 Análise de "fan-out" de portas lógicas e buffers, onde são mostradas formas de onda de um sinal de entrada no NO1 e respectiva saída no NO2.

existir uma relação de compromisso entre o tempo máximo de subida ou descida de um sinal para um dado nó e o tamanho máximo do drive de saída que aciona este nó.

A implementação desta regra é feita comparando-se os tempos de subida e de descida do buffer ou da porta que aciona o nó, com os parâmetros do arquivo de tecnologia de máximo tempo de subida e máximo tempo de descida. No cálculo dos tempos de subida e descida estão considerados os tamanhos dos transistores do buffer ou porta.

Consultando-se alguns programas comercias de análise de "fan-out" constatou-se que, para se considerar um nó com pouco consumo de potência, os tempos máximos são da ordem de 10ns.

O aviso gerado com relação a este tipo de problema é uma sugestão de se diminuir o número de ligações a um nó. Pode-se resolver este problema

colocando-se mais um buffer ou ainda, caso a porta que esteja acionando não seja buffer, trocá-la por um buffer.

Problemas com a margem de ruído ("spike").

A análise de ruído em portas lógicas CMOS é uma tarefa que possui um enorme número de variáveis envolvidas, como: processos, condições de operação, freqüências de chaveamento e outros efeitos. Para o caso deste trabalho a análise relativa a ruídos se limita a identificar a largura máxima de pulso permitida para que uma porta lógica não mude o seu estado, isto é, a suscetibilidade a pulsos espúrios. Contudo a análise mais acurada do problema deve ser feita pelo projetista.

A largura mínima que um pulso de entrada em uma porta lógica pode interferir no resultado da saída é o tempo de propagação deste pulso pela porta. Portanto por exemplo uma porta que tenha tempo de propagação igual a 20ns os pulsos espúrios que acontecerem com largura menor que 20ns não alterarão o valor da saída.

Regras do tipo III

Estas regras são as que incorporam a experiência do projetista especialista no assunto. A incluem vivência do projetista ao analisar um projeto de circuito.

Estas regras são resultados de um intensa interação do enegenheiro do conhecimento com o projetista para o entendimento do processo que o projetista usa para identificar problemas em circuitos.

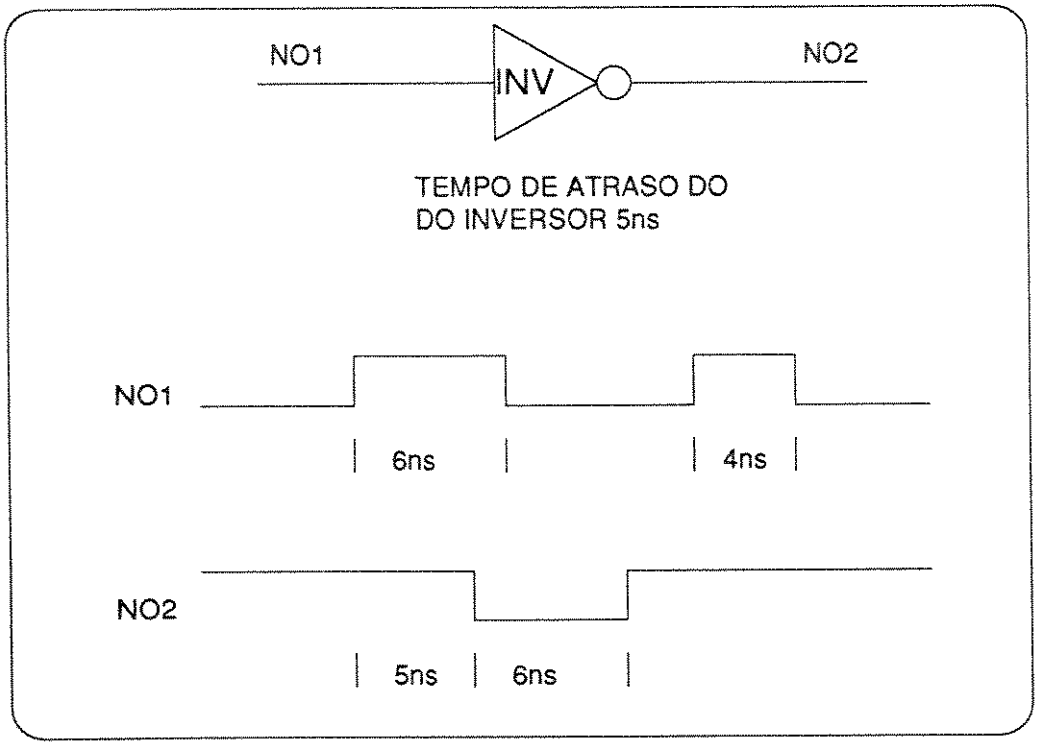


Figura 3.8 Filtragem de spikes de uma porta lógica; onde são mostrados a forma de onda de entrada pelo NO1 e forma de onda de saída no NO2, verifica-se que o pulso de 4 ns não causou mudança de estado na saída do inversor.

3.7. Módulo Explanador de Raciocínios.

O processo de explanação dos raciocínios feitos pelo sistema especialista envolve explicar o processo usado, com auxílio da base de conhecimento, que levou a uma dada resposta. Esta explicação está relacionada com o modo como o sistema entende seus próprios problemas e inclui também uma estrutura de diálogo entre o sistema e o usuário.

A explanação envolve dois aspectos: o primeiro explicar o ambiente onde ocorreu um problema, o segundo explicar uma decisão tomada segundo certas regras.

A explanação possui quatro objetivos principais:

- 1. Ajudar o usuário a entender as conclusões encontradas pelo sistema,
- 2. Ajudar o engenheiro do conhecimento a depurar o sistema e o ambiente de solução de problemas,
- 3. Convencer o usuário que as conclusões que o sistema especialista chegou são razoáveis,
- 4. Ensinar ao usuário inexperiente a fazer um bom projeto, isto é, melhorar a qualidade dos projetos do usuário.

Como primeira conseqüência deste processo de explanação, tem-se que o usuário aprende com o sistema. Na medida em que são explicados os processos de análise feitos pelo sistema especialista, o usuário pode compreender com mais facilidade os erros que cometeu durante o projeto.

Para dar respostas ao usuário é essencial saber como o conhecimento foi usado durante o processo de solução do problema. Este conhecimento de como foi o processo de solução do problema pode ser adquirido armazenado-se a seqüência de aplicação das regras.

O conteúdo da explicação pode ser montado e apresentado de duas formas:

- Por introspecção: examinando-se os componentes da atividade de solução do problema, que foram armazenados no retrospecto da solução, ou repassando a parte da base do conhecimento que foi usada nesta solução em particular.
- Por preparação: produzindo-se uma explicação que não esta relacionada com o caminho percorrido para a solução, mas que torna a resposta plauzível. Este processo é usado quando não se tem acesso aos passos intermediários feitos pelo programa.

3.7.1. Interface Homem-Máquina.

O conteúdo da explanação relacionado acima produz, conceitualmente e lógicamente, toda a informação necessária para as explanações que o usuário necessita. Contudo, isto não é suficiente, é necessária uma tradução para uma linguagem natural.

O sistema de explanação para o verificador de regras de projeto está baseado em duas premissas:

O engenheiro de conhecimento tem acesso a passos intermediários, que mostram a seqüência de regras que foram aplicadas para se chegar a um dado aviso.

O sistema não faz perguntas intermediárias, isto é, não é considerado um sistema de consulta. Portanto as questões do tipo "Porque ?" não se aplicam. O tipo de pergunta implementado para explanar os raciocínios é o "Como ?" que faz uma retrospectiva das regras utilizadas.

Uma caracterísitica do sistema é a de analisar todas as regras em uma execução, devido a isto, as explanações só podem ser feitas com relação ao último aviso

gerado. A princípio isto representaria uma deficiência do sistema, mas considerando-se que as regras são as mesmas e o processo de raciocínio é o mesmo para um aviso ou para vários esta deficiência é irrelevante.

3.8. Módulo Formatador de Resultados.

Este módulo transforma as cláusulas de avisos, estruturas identificadas e estatísticas para linguagem natural. Esta transformação facilita ao usuário entender os avisos e identificar melhor os problemas com o circuito.

A apresentação dos resultados é mostrada na forma de uma lista composta de três partes:

- Estruturas identificadas, com os números dos transistores que compoem as estruturas,
- Avisos de problemas, com as estruturas ou transistores com problemas e os parâmetros que foram violados,
- Dados relativos ao circuito, número de nós, número de transistores e consumo de potência estimado.

CAPÍTULO QUATRO

4. IMPLEMENTAÇÃO E RESULTADOS

4.1. Introdução

Este capítulo descreve um sistema protótipo que implementa as regras principais descritas no capítulo três, bem como apresenta os resultados alcançados com este protótipo. Na implementação foram usadas duas linguagens de programação: C e PROLOG.

A parte do sistema correspondente à conversão de dados SPICE em fatos PROLOG foi implementada em linguagem C devido a facilidade de manipulação de cadeias de caracteres que esta linguagem oferece. A base de conhecimento, a máquina de inferência e o explanador foram implementados em PROLOG.

4.2. Descrição do Protótipo

O protótipo implementa a arquitetura descrita no capítulo tres e se insere no contexto dos sistemas especialistas como está mostrado na figura 4.1.

Como pode ser visto na figura 4.1 o sistema possue uma base de conhecimento, uma máquina de inferência, um módulo de explanação e uma interface com o usuário.

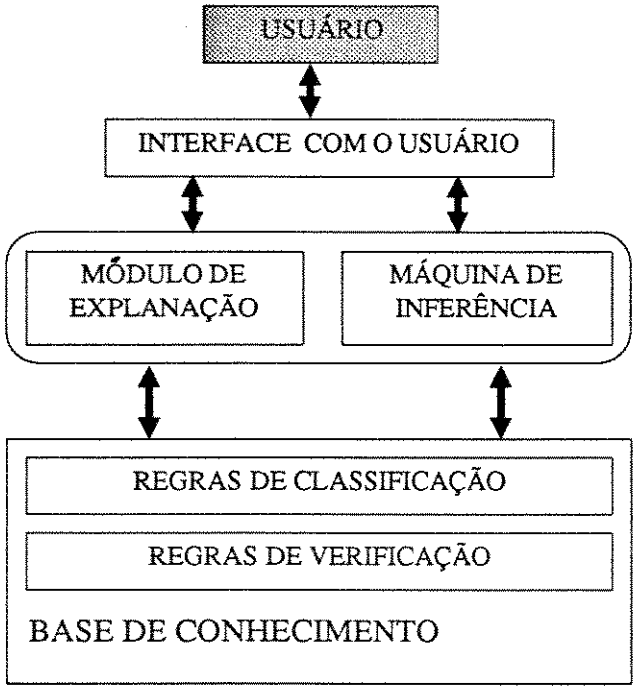


Figura 4.1 Diagrama esquemático do sistema especialista CREPE.

4.2.1. Interface com o usuário.

Esta interface permite ao usuário uma interação com o sistema especialista para:

- 1. Entrar com os dados do circuito,
- 2. Analisar os resultados através do relatório gerado,
- 3. Fazer consultas e obter respostas do módulo explanador.

A entrada de dados é feita através do arquivo ***circuito.spi***, que contém a descrição SPICE do circuito. Além deste arquivo é necessário que o usuário informe os nós de entrada, nós de saida, nó de VDD, nó de VSS e o nó de relógio.

A saída de dados é feita pelo módulo formatador de resultados que gera um arquivo ***circuito.rel*** que contém as estruturas identificadas, os avisos gerados, o número de nós do circuito, o número de transistores e uma estimativa de potência consumida.

O módulo explanador, que é acionado logo após a análise do circuito, quando solicitado pelo usuário, justifica o diagnóstico feito pelo sistema especialista.

As restrições de tecnologia estão descritas no arquivo ***tec.ari*** que contém os seguintes dados:

- vdd : valor nominal da alimentação terminal positivo ex: 5V,
- vss : valor nominal da alimentação terminal negativo ex: 0V,
- subs : valores da polarização do nó de sustrato,
- max_tr : valor máximo admissível de tempo de subida de uma saída CMOS,
- max_tf : valor máximo admissível de tempo de descida de uma saída CMOS,
- lmin : comprimento mínimo que o canal do transistor admite ex: 1.0 U,
- lmax : comprimento máximo que o canal do transistor admite ex: 100 U,
- wmin : largura mínima que o canal do transistor admite ex: 1.0 U,
- wmax : largura máxima que o canal do transistor admite ex: 100 U,
- capmax : Indica o máximo valor de capacitância admissível por um nó,
- tox: Espessura da camada de óxido do gate,

- up : Mobilidade das lacunas,
- un : Mobilidade dos elétrons,
- e: Permissividade do silicio,
- variacao_beta_max: Parâmetro que define a máxima diferença percentual entre o tempo de subida e o tempo de descida,
- variacao_beta_min: Parâmetro que define a mínima diferença percentual entre o tempo de subida e o tempo de descida,
- freq: define a freqüência média que o circuito vai operar para fins de cálculo de consumo de potência,
- l_buffer: Comprimento mínimo do canal de um transistor para este fazer parte de um buffer,
- w_buffer: Largura mínima do canal de um transistor para este fazer parte de um buffer,
- spike: largura máxima para que um pulso seja considerado "spike".

Os valores das constantes acima são estipulados pelo usuário, conforme a tecnologia específica de fabricação do circuito integrado que está sendo usada e conforme o tipo de circuito que deve ser diagnosticado. Por exemplo, se o circuito deve ser imune a spikes de 2ns o usuário definirá a constante "spike" com esse valor e o sistema verificará se as portas lógicas do circuito respeitam este limite; se for desejado relaxar as condições de contorno do projeto, o usuário poderá definir o valor dessa variável com um valor grande, por exemplo, 100ns..

Base de Conhecimento.

e de conhecimento implementa as regras discutidas no capítulo três. Para nplementação, as regras que representam o conhecimento estão divididas ›is tipos :

- Regras de classificação.
- Regras de verificação.

. Regras de classificação.

gras de classificação são as responsáveis pela identificação dos elementos efinidos de circuito. O protótipo identifica as seguintes estruturas:

- nand2 : porta lógica Nand de duas entradas.
- nor2 : porta lógica Nor de duas entradas.
- inv : inversor.
- buffer : inversor com maior capacidade de carga.
- latch : elemento de memória.
- gates de transmissão : par de transistores conectados em paralelo, um transistor PMOS e um transistor NMOS.

nfigurações de transistores relativas a estas estruturas são mostradas na a 4.2.

gras de classificação são baseadas nas interligações dos elementos, por plo:

Para identificar um inversor a regra procura dois transistores, um PMOS e um NMOS, com os terminais de fonte ligados a VDD e VSS respectivamente, com os terminais de gate em curto e com os terminais de dreno em curto.

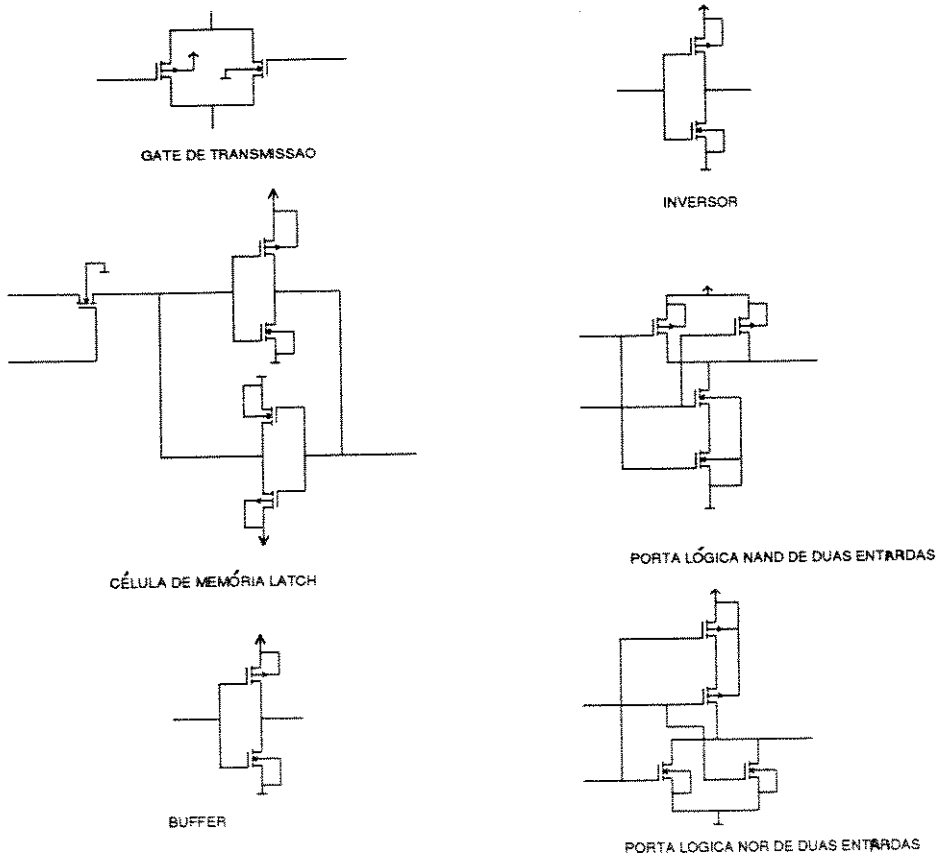


Figura 4.2 configurações de transistores das estruturas identificadas.

Para identificar uma porta lógica Nor de duas entradas são tomados como base dois pares de transistores, desta forma, um par de transistores PMOS em série e um par de transistores NMOS em paralelo, sendo os gates dos NMOS ligados aos gates dos PMOS. Um terminal do par série ligado a VDD, um terminal do par paralelo ligado a VSS e os terminais restantes ligados em curto.

4.3.2. Regras de verificação.

As regras de verificação implementadas são as tipo I e II descritas no capítulo três. Estas regras estão descritas na base de conhecimento na forma abaixo:

SE: premissas
ENTÃO: conclusão

Onde as premissas são as condições para que o circuito apresente problemas. Estas premissas também podem ser regras e assim por diante. As conclusões são os avisos de problemas encontrados. Estes avisos são armazenados para serem usados pelo módulo de explanação e para formatação dos resultados.

4.3.3. Máquina de Inferência

A máquina de inferência usada foi a máquina embutida no PROLOG, que possui como características o "Backward Chaining" [24,34] o qual tem sido muito utilizado nos processos de análise e diagnóstico.

4.3.4. Módulo de explanação

Este módulo faz a retrospectiva da utilização das regras para um dado aviso. Para execução desta retrospectiva foi implementada uma pilha onde, durante o processo de aplicação das regras são armazenadas as seqüências de aplicação das regras que geraram avisos. Quando o usuário consulta o sistema para pedir explicações sobre um aviso, o explanador desempilha as seqüências de regras usadas e vai explicando o "porquê" de cada regra. O explanador é composto das cláusulas de explanação e de uma cláusula que desempilha as seqüências e as coloca em linguagem natural para o usuário.

4.4. Resultados

Foram implementadas 30 regras das quais 14 são regras de simples verificação, onde uma premissa gera uma conclusão. Por exemplo:

> **SE:** L < Lmin do transistor Tx
> **ENTÃO:** : Avisa que o transistor Tx está com o comprimento do canal menor que a permitida pelo processo.

Foram também implementadas regras de maior complexidade que exigem mais de um nível de análise, isto é, uma regra que procura validar outra regra para chegar a um aviso de problema. Por exemplo:

> **SE:** os transistores Tx e Ty formam um inversor
> e o tempo de subida for TS
> e tempo de descida for TD
> e TS e TD não forem iguais
> **ENTÃO:** O inversor estará mal dimensionado.

Implementadas as regras tipo I e tipo II foi feita uma análise de desempenho do sistema especialista. Com os circuitos dos testes de 1 a 6, mostrados no anexo A, foram obtidos os resultados abaixo:

Obs: o programa foi executado por um interpretador Arity PROLOG num microcomputador IBM-PC XT.

CIRCUITO	TRANSISTORES	NÓS	TEMPO	ESTRUTURAS	AVISOS
TESTE1	19	15	48 s	6	2
TESTE2	15	15	39 s	0	13
TESTE3	12	11	35 s	4	1
TESTE4	6	8	27 s	3	4
TESTE5	8	7	28 s	4	2
TESTE6	10	8	31 s	3	8

Tabela 1. Resultados dos testes 1 a 6, cujos circuitos estão descritos no anexo A.

O que se pode concluir desta tabela é que o sistema com a atual implementação é considerado útil para células de circuito com um número reduzido de transistores. Para circuitos maiores serão necessários ajustes e uma versão compilada do programa poderá tornar menor o tempo de execução.

Os resultados mais significativos foram obtidos com a aplicação do sistema CREPE a casos reais. O sistema foi usado para validar circuitos de uma biblioteca de células do CPqD da TELEBRÁS. Em uma célula desta biblioteca o sistema CREPE identificou uma falha potencial. Foi encontrado um caminho

de VDD para VSS passando somente por transistores NMOS. No anexo A são apresentados os resultados obtidos pelo CREPE neste e em outros testes.

4.4.1. Avaliação do Sistema CREPE

A avaliação de um sistema especialista, como foi discutida no capítulo dois, é uma tarefa contínua que deve se iniciar na fase de protótipo. A avaliação feita no protótipo implementado, foi executada com um conjunto de testes que procurou cobrir todos as regras discutidas no capítulo três.

Uma avaliação mais completa só poderá ser obtida com o uso exaustivo do sistema em casos reais de circuitos. Esta avaliação começou com a aplicação do sistema em células de uma biblioteca que está sendo desenvolvida no CPqD da TELEBRÁS.

A obtenção de resultados que permitam uma avaliação segura do sistema, dependerá de seu uso por diferentes projetistas. Como estes poderão ter opiniões diferentes do que um sistema de verificação de regras elétricas deva oferecer, suas considerações enriquecerão a avaliação.

CAPÍTULO CINCO

5. CONCLUSÕES

5.1. Considerações Finais.

O conhecimento necessário para o projeto de circuitos integrados não difere muito de projetista para projetista. Isto permite que a aquisição deste conhecimento seja feita com maior facilidade. Contudo este conhecimento vai se aprimorando com o passar do tempo, na medida em que os projetistas vão se tornando mais capazes.

Como os projetistas, o sistema CREPE pode, quando se incluem mais regras na base de conhecimento, tornar-se mais capaz de analisar e criticar os circuitos projetados. Esta característica de ampliação da base de conhecimento dos sistemas especialistas norteou a construção deste sistema.

A arquitetura aqui proposta cumpriu com razoável sucesso o objetivo citado acima, visto que a modularidade das regras permite que o sistema se aprimore e se torne um analista cada vez melhor.

O sistema CREPE contém uma grande quantidade de informação codificada em suas regras. Como estas informações podem ser acessadas de modo organizado, se constituem em um poderoso instrumento de aprendizado e de elucidação dos diagnósticos para os usuários. Com a facilidade de explanação

implemetada pelo sistema CREPE mostramos que se pode auxiliar os projetistas menos experientes a aprender e também facilitar a tarefa de depuração do sistema pelo engenheiro do conhecimento.

É muito comum que grupos de projeto de circuitos integrados sejam constituídos por um especialista e alguns projetistas menos experientes e, além disto, que este grupo possua uma biblioteca de células muito específica. Neste caso o sistema CREPE pode se adaptar a esta biblioteca específica e incorporar o conhecimento do especialista, otimizando o acesso a este conhecimento. Desta forma o especialista pode se dedicar a tarefas mais complexas e ao mesmo tempo os projetista menos experientes vão aprendendo com o uso do sistema especialista.

No capítulo dois foi feita uma revisão bibliográfica das técnicas de construção de sistemas especialistas. Estas informações foram organizadas de tal forma que podem ser utilizadas como um guia para quem deseja se iniciar nesta área da computação.

5.2. Trabalhos Futuros

Durante o desenvolvimento deste trabalho foram detetadas algumas melhorias que tornariam o sistema mais amigável e eficiente. A seguir listamos alguns destes ítens.

5.2.1. Aumento da base de conhecimento.

A cada circuito que se projeta podem-se criar novas regras. À medida que estas regras forem aparecendo, deverão ser incorporadas a base de conhecimento inicial. No momento, esta inclusão só pode ser feita pelo engenheiro de

conhecimento que projetou o sistema. Como continuidade deste trabalho sugerimos a implementação de um módulo de aquisição automática de regras que facilitaria em muito o processo de ampliação da base de conhecimento.

5.2.2. Uso de " Workstation " e compilação.

O uso de uma máquina mais poderosa aumenta o desempenho a nível de tempo de execução, bem como permite que o sistema analise circuitos maiores.

5.2.3. Interface com o usuário.

Como a preocupação principal deste trabalho foi a validação de uma arquitetura de sistema especialista, a interface com o usuário ficou relegada a um segundo plano, portanto é nesta interface que podemos sugerir melhorias mais significativas como:

- Integração com uma ferramenta de edição de esquemáticos em primeira instância e a seguir, integração com um sistema completo de projeto de circuitos integrados.
- Incluir uma parte gráfica de visualização de erros. Por exemplo: piscar na tela um nó que esteja com o " Fan-out " excedido.
- Criar um sistema de janelas de verificação no esquemático para que o sistema só verifique as regras elétricas na parte do circuito que estiver dentro da janela.

BIBLIOGRAFIA

[1] Chung P.H.W. and Kingston J.K.C. " State of the Art Knowledge-Based Toolkits ", Artificial Inteligence Applications Institute, University of Edinburg.

[2] Richman B.A., Hansen J.E. and Cameron K. " A Deterministic Algorithm for Automatic CMOS Transistor Sizing ", IEEE Journal of Solid State Circuits.

[3] De Man H., Bolsens I. and Meersh E.V. " An Expert System for Logical and Eletrical Debbuging of MOS VLSI Networks ", ICCAD, 1984, pg 203-205.

[4] Bolsens I. " Experiments with PROLOG ", Internal Report ESAT, June 1984.

[5] Clocksin W.F. " Logic Programming and Digital Analysis ", The Journal of Logic Programming, 1987, pg. 60-82.

[6] Kerr E.S. " Pad Audit - A Rule Based System in PROLOG ", Internal Report, ES2/TELEBRÁS, 1989.

[7] Kerr E.S. " Design Audit - Draft Specification ", Internal Report, ES2, 1989.

[8] Meersh E.V. " ERC 1.0: Introdution and User's Manual ", Internal Report CPqD TELEBRÁS, 1988.

[9] Wick M.R. and Slage J.R. " An Explanation Facility for Today's Expert Systems ", IEEE Expert, Spring 1989, pg 26-36.

[10] Chandrasekaran B., Tanner M.C. and Josephson J.R. " Explaining Control Strategies in Problem Solving ", IEEE Expert Spring 1989, pg 9-24.

[11] Meersh E.V. " Construction of an Accurate Timming Model for Digital MOS Circuits ", Porto Alegre RS, SBMICRO 1989.

[12] Nunes T.I. e Damiani F. " Arquitetura de um sistema especilista para verificação de regras elétricas em circuitos integrados CMOS ", Campinas SP, SBMICRO 1990.

[13] Barrow H.G. " VERIFY : A Program for Proving Connectness of Digital Hardware Designs ", Artificial Inteligence 24, 1984, pg. 437-491.

[14] Michaelson R.H., Miche D. and Boulanger A. " The Technology of Expert Systems ", BYTE, April 1985, pg 303-312.

[15] Kowalski T.J. " The VLSI Design Automation Assistant: A Synthesis ", Expert AT&T Technical Journal, Jan-Feb 1987, pg 81-92.

[16] Srinivas N.C.E. and Agrawal D.V. " Formal Verification of Digital Circuits Using Hibrid Simulation ", IEEE Circuit and Devices Magazine, Jan 1988, pg 19-27.

[17] Thompson B. " Topics in Knowledge-Based Languages ", Dr Dobbs Journal, April 1988, pg 40-49.

[18] Subrahmanyam P.A. " Synopse: An Expert System for VLSI Design ", IEEE Computer, July 1986, pg 78-89.

[19] Bulter C.W., Hodil E.D. and Richardson G.L. " Building Knowledge-Based Systems with Procedural Languages ", IEEE Expert, Summer 1988, pg 47-59.

[20] Rosenstiel W. Bergstrasser T. " Artificial Inteligence for Logic Deisgn ", Logic Design and Simulation, 1986, pg 229-257.

[21] PCAD Corporation " PC-ERC's User Guide ", 1990, pg 13-1 to 13-36.

[22] Lob C. " RUBICC: A rule based expert system for integrated circuit critique ", UCB/ERL M84/86, 28 Sept 1984.

[23] Microsoft Corporation " Microsoft Quick C Compiler, Programmer's guide ", 1987.

[24] Clocksin W.F. and Mellish C.S. " Programming in PROLOG ", Springer-Verlag, New York, 1987.

[25] Marcus C. " PROLOG Programming ", Arity corporation, Addison-Wesley, Nov 1986.

[26] Weiss S.M. e Kulikowski. " Guia Prático para Projeto de Sistemas Especialistas ", 1988.

[27] Weste N.H.E. and Eshraghian K. " Principles os CMOS VLSI Design a Systems Perstective ", Addison-Wesley, 1985.

[28] Hayes F., Waterman D.A. and Lenat D.B. " Building Expert Systems ", Vol I, Addison - Wesley, 1983.

[29] Arity Corporation. " The Arity/Prolog programming language ", CSA press, Concord Massachusetts, 1986.

[30] Swartout W.R. " XPLAIN: a system for creating and explaining expert consulting programs ", USC/Information Sciences Institute Marina del Rey, Artificial Intelligence, Vol. 21 Number 3, Sept 1983, pg 285-325.

[31] Thompson B.A. and Thompson W.A. " Inside an expert system ", BYTE, April 1985, pg 315-330.

[32] Rubinoff R. " Explaining concepts in expert systems : the CLEAR system ", Proc. Second Conf. Artificial Intelligence Applications, IEEE Computer Society Press, California, 1985, pg 416-421.

[33] Clancey W.J. " The epistemology of a rule-based expert system - A framework for explanition ", Stanford University, Artificial Intelligence Vol. 20 Number 3, May 1983, pg 215-251.

[34] Nilsson N.J. " Principles of Artificial Intelligence ", Tioga Press, California, 1980.

[35] Glasser L.A. and Dobberpuhl D.W. " The design and analysis of VLSI circuits ", Addison-Wesley, 1985.

[36] Spickelmier R.L. and Newton R.A. " CRITIC: A knowledge-based program for critiquing circuits designs ", IEEE, California, 1988.

[37] Richardson S., Steele R. and Ellsworth D. " AI for asic's pinpoints potencial problems ", ESD: The eletronic System Design Magazine, June 1988.

ANEXO A

Exemplos de Execução do Programa CREPE

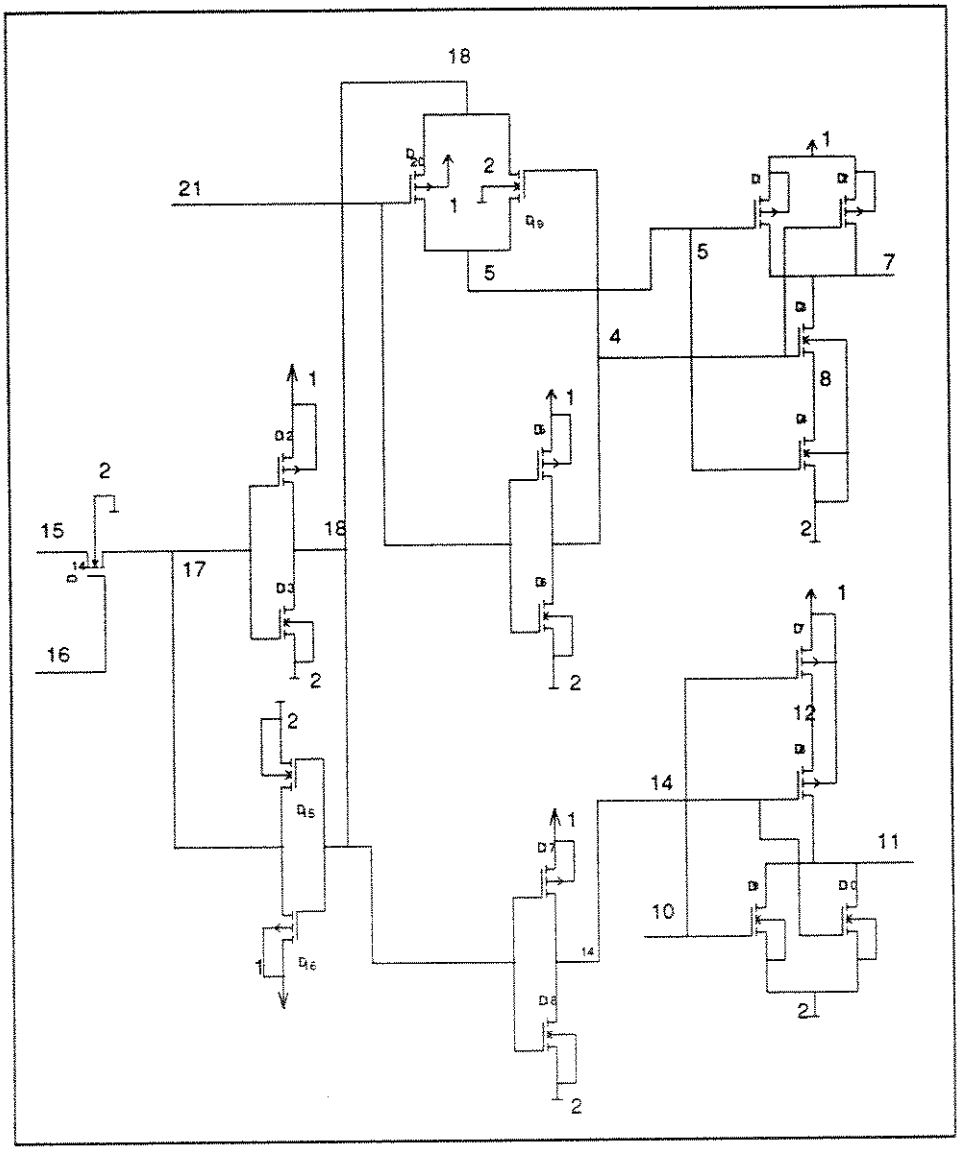
A.1. Circuit : TESTE1

```
M6 4 21 2 2 NMOS L=1.25U W=4.5U 4 4 8 8
M7 1 10 12 1 PMOS L=1.25U W=15.0U 4 4 8 8
M5 1 21 4 1 PMOS L=1.25U W=7.5U 4 4 8 8
M8 12 14 11 1 PMOS L=1.25U W=15.0U 4 4 8 8
M1 1 5 7 1 PMOS L=1.25U W=4.5U 4 4 8 8
M2 1 4 7 1 PMOS L=1.25U W=4.5U 4 4 8 8
M3 7 4 8 2 NMOS L=1.25U W=6.0U 4 4 8 8
M4 8 5 2 2 NMOS L=1.25U W=6.0U 4 4 8 8
M9 11 10 2 2 NMOS L=1.25U W=3.0U 4 4 8 8
M10 11 14 2 2 NMOS L=1.25U W=3.0U 4 4 8 8
M17 1 18 14 1 PMOS L=10U W=45U 40 40 80 80
M18 14 18 2 2 NMOS L=10U W=20U 40 40 80 80
M12 1 17 18 1 PMOS L=1.25U W=4.5U 4 4 8 8
M13 18 17 2 2 NMOS L=1.25U W=4.5U 4 4 8 8
M16 1 18 17 1 PMOS L=4.5U W=2.0U 4 4 8 8
M15 17 18 2 2 NMOS L=1.25U W=2.0U 4 4 8 8
M14 15 16 17 2 NMOS L=1.25U W=4.5U 4 4 8 8
M20 18 21 5 1 PMOS L=1.25U W=2.0U 4 4 8 8
M19 5 4 18 2 NMOS L=1.25U W=2.0U 4 4 8 8
C3  4 2 1000
C4 11 2 1000
C5  7 2 1000
```

```
                CREPE
   SISTEMA PARA VERIFICACAO DE REGRAS ELETRICAS
Circuito analisado: teste1
 ESTRUTURAS IDENTIFICADAS

Inversores encontrados
  inv 1 transistores  5  6   no de entrada= 21 no de saida= 4
Buffers  encontrados
  buf 1 transistores  17  18   no de entrada= 18 no de saida= 14
Portas nand encontradas
  nand2 1 transistores  1  2  3  4   nos de entrada= 5  4   no de saida= 7
Portas nor encontradas
  nor2 1 transistores  7  8  9  10   nos de entrada= 10  14   no de saida= 11
Gates de transmissao encontrados
  tra 1 transistores  20  19   no de entrada= 18   no de saida= 5
Latchs encontrados
  latch 1 inversores 2  3   transistor 14   D= 15   En= 16   Q= 18

     AVISOS DE PROBLEMAS COM REGRAS
TIPO   COMENTARIO        ARGUMENTOS
i6    No com uma unica conexao   15   14   d
i6    No com uma unica conexao   16   14   g
Numero de transistores encontrados no circuito : 19
Numero de nos encontrados no circuito : 15
Consumo de potencia estimado para a frequencia media  : 0.00111838
```



TESTE1 IDENTIFICAÇÃO DE ESTRUTURAS

A.2. Circuit: TESTE2

M1 1 1 3 2 NMOS L=1.25U W=2.0U 6 6 9 9
M2 3 4 4 2 NMOS L=1.25U W=2.0U 6 6 9 9
M3 4 5 4 2 NMOS L=1.25U W=2.0U 6 6 9 9
M4 1 6 2 1 PMOS L=1.25U W=2.0U 6 6 9 9
M6 1 7 8 1 PMOS L=1.25U W=2.0U 6 6 9 9
M7 8 7 9 1 PMOS L=1.25U W=2.0U 6 6 9 9
M8 9 10 11 1 PMOS L=1.25U W=2.0U 6 6 9 9
M9 11 10 2 1 PMOS L=1.25U W=2.0U 6 6 9 9
M10 1 13 12 2 NMOS L=1.25U W=2.0U 6 6 9 9
M11 12 13 14 2 NMOS L=1.25U W=2.0U 6 6 9 9
M12 14 16 15 2 NMOS L=1.25U W=2.0U 6 6 9 9
M13 15 16 2 2 NMOS L=1.25U W=2.0U 6 6 9 9
M14 1 18 17 1 PMOS L=1250U W=2.0U 6 6 9 9
M15 17 18 19 1 NMOS L=1.25U W=2.0 6 6 9 9
M16 19 18 2 2 NMOS L=1.0U W=1.0U 6 6 9 9

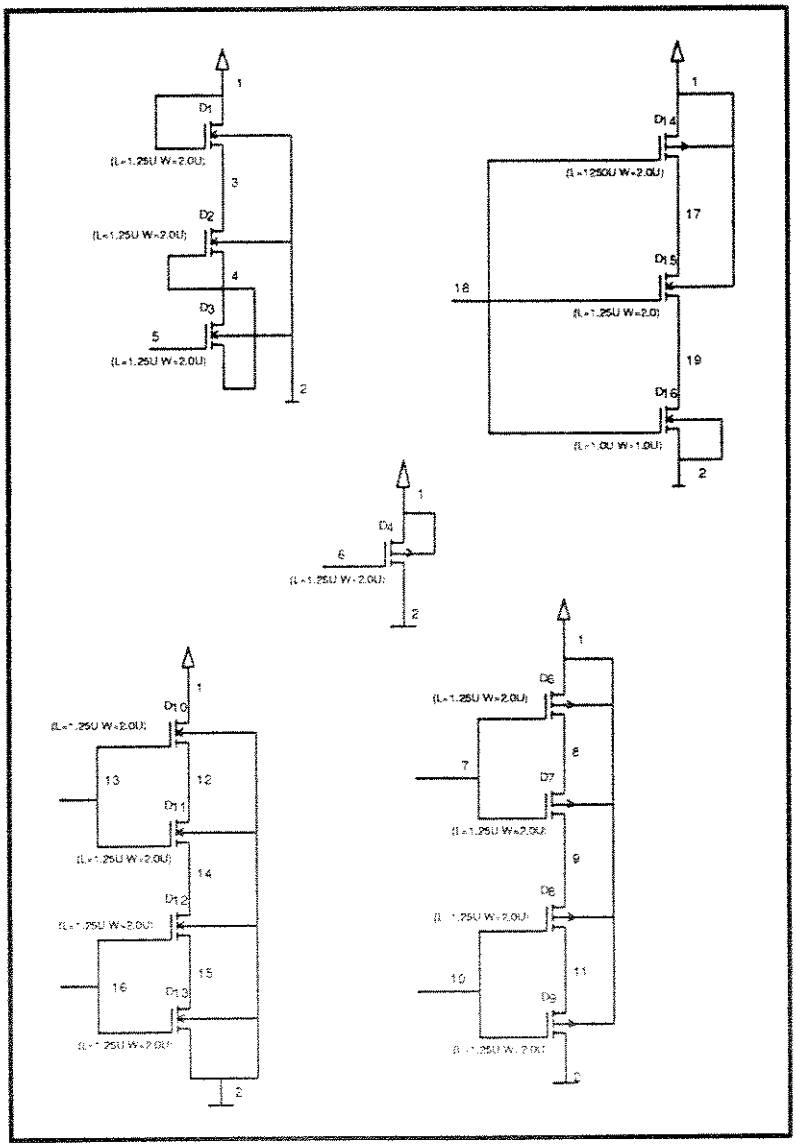
CREPE
SISTEMA PARA VERIFICACAO DE REGRAS ELETRICAS
Circuito analisado: teste2
ESTRUTURAS IDENTIFICADAS

Inversores encontrados
Buffers encontrados
Portas nand encontradas
Portas nor encontradas
Gates de transmissao encontrados
Latchs encontrados

AVISOS DE PROBLEMAS COM REGRAS

TIPO	COMENTARIO	ARGUMENTOS
i1	W maior que o maximo	15 2000000.0 100
i2	W menor que o minimo	16 1 1.25
i3	L maior que o maximo	14 1250 100
i4	L menor que o minimo	16 1 1.25
i6	No com uma unica conexao	5 3 g
i6	No com uma unica conexao	6 4 g
i7	Transistor com dreno e fonte em curto	3 4
i8	Transistor com dreno e gate em curto	1 1
i9	Transistor com gate e fonte em curto	2 4
i10	Transistor com dreno e fonte em vdd e vss	4 1 2
i11	Tem pelo menos um caminho nmos entre vdd e vss	_27AD
i12	Tem pelo menos um caminho pmos entre vdd e vss	_27AD
i14	Transistor com o substrato mal polarizado	15

Numero de transistores encontrados no circuito : 15
Numero de nos encontrados no circuito : 19
Consumo de potencia estimado para a frequencia media : 0.9572575



TESTE2 CIRCUITO COM ERROS BÁSICOS

A.3. Circuit : TESTE3

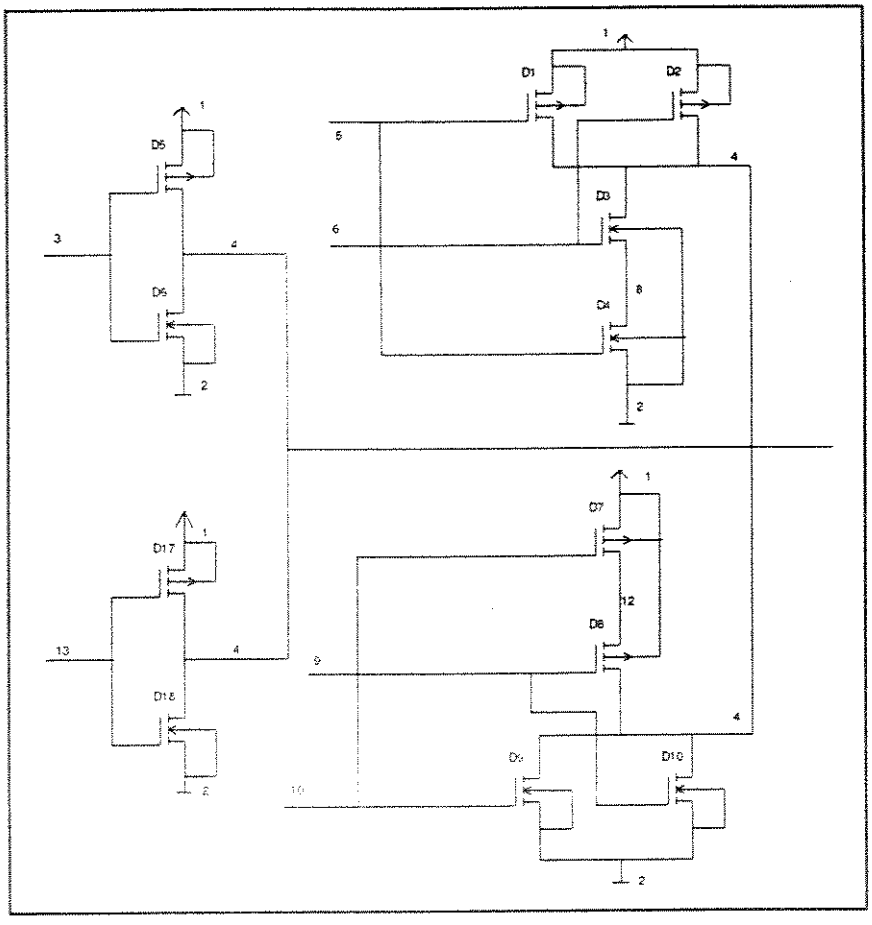
M6 4 3 2 2 NMOS L=1.25U W=4.5U 4 4 8 8
M7 1 10 12 1 PMOS L=1.25U W=15.0U 4 4 8 8
M5 1 3 4 1 PMOS L=1.25U W=7.5U 4 4 8 8
M8 12 9 4 1 PMOS L=1.25U W=15.0U 4 4 8 8
M1 1 5 4 1 PMOS L=1.25U W=4.5U 4 4 8 8
M2 1 6 4 1 PMOS L=1.25U W=4.5U 4 4 8 8
M3 4 6 8 2 NMOS L=1.25U W=6.0U 4 4 8 8
M4 8 5 2 2 NMOS L=1.25U W=6.0U 4 4 8 8
M9 4 10 2 2 NMOS L=1.25U W=3.0U 4 4 8 8
M10 4 9 2 2 NMOS L=1.25U W=3.0U 4 4 8 8
M17 1 13 4 1 PMOS L=10U W=45U 40 40 80 80
M18 4 13 2 2 NMOS L=10U W=20U 40 40 80 80

```
          CREPE

   SISTEMA PARA VERIFICACAO DE REGRAS ELETRICAS
Circuito analisado: teste3
 ESTRUTURAS IDENTIFICADAS

Inversores encontrados
  inv 1 transistores  5  6   no de entrada= 3 no de saida= 4
Buffers  encontrados
  buf 1 transistores  17  18   no de entrada= 13 no de saida= 4
Portas nand encontradas
  nand2 1 transistores  1  2  3  4   nos de entrada= 5  6   no de saida= 4
Portas nor encontradas
  nor2 1 transistores  7  8  9  10   nos de entrada= 10  9   no de saida= 4
Gates de transmissao encontrados
Latchs encontrados

     AVISOS DE PROBLEMAS COM REGRAS
TIPO   COMENTARIO          ARGUMENTOS
ii1    Lista de portas em curto   4   [(buf , 1),(inv , 1),(nand2 , 1),(nor2 , 1)]
Numero de transistores encontrados no circuito : 12
Numero de nos encontrados no circuito : 11
Consumo de potencia estimado para a frequencia media  : 0.00034162
```



TESTE3 PORTAS EM CURTO

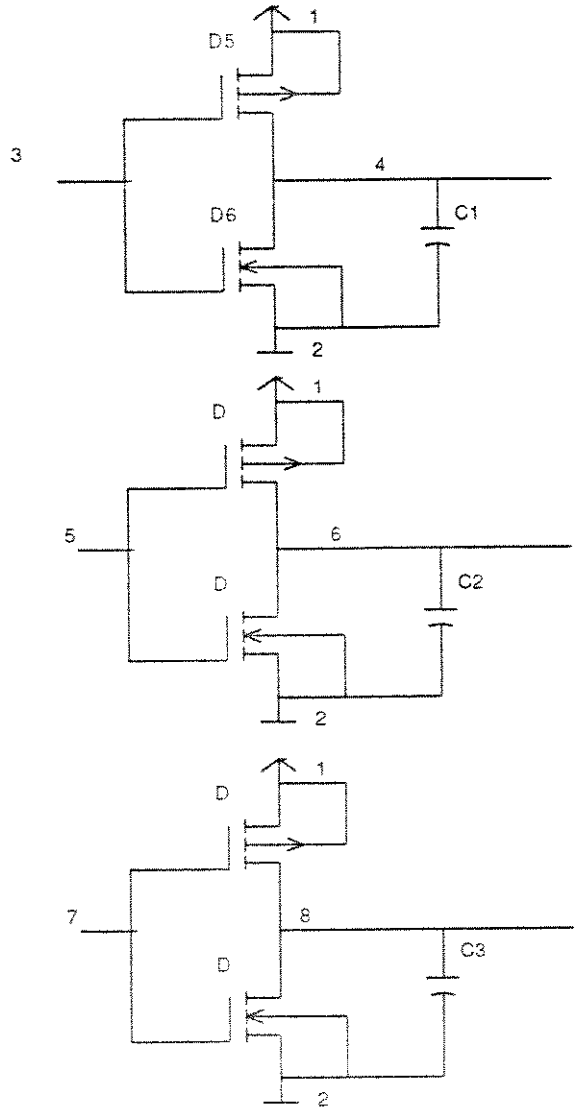
A.4. Circuit : TESTE4

M6 4 3 2 2 NMOS L=1.25U W=4.5U 4 4 8 8
M2 8 7 2 2 NMOS L=1.25U W=4.5U 4 4 8 8
M5 1 3 4 1 PMOS L=1.25U W=7.5U 4 4 8 8
M3 1 7 8 1 PMOS L=1.25U W=4.5U 4 4 8 8
M4 1 5 6 1 PMOS L=7.5U W=1.25U 4 4 8 8
M7 6 5 2 2 NMOS L=1.25U W=4.5U 4 4 8 8
C1 4 2 1000
C2 6 2 1000
C3 8 2 1000

```
                CREPE
   SISTEMA PARA VERIFICACAO DE REGRAS ELETRICAS
Circuito analisado: teste4
 ESTRUTURAS IDENTIFICADAS

Inversores encontrados
   inv 1 transistores  5  6   no de entrada= 3 no de saida= 4
   inv 2 transistores  3  2   no de entrada= 7 no de saida= 8
   inv 3 transistores  4  7   no de entrada= 5 no de saida= 6
Buffers  encontrados
Portas nand encontradas
Portas nor encontradas
Gates de transmissao encontrados
Latchs encontrados

     AVISOS DE PROBLEMAS COM REGRAS
TIPO   COMENTARIO          ARGUMENTOS
ii2     Beta do inversor mal dimensionado   2  0.6  1.4
ii2     Beta do inversor mal dimensionado   3  0.6  1.4
ii5     Fan-out do inversor excedido   3  0.00000014  3.32015810E-09
ii6     Inversor sensivel a spikes   3  0.00000004  0.00000001
Numero de transistores encontrados no circuito : 6
Numero de nos encontrados no circuito : 8
Consumo de potencia estimado para a frequencia media  : 0.00077778
```



TESTE4 DIMENSIONAMENTO DOS INVERSORES

A.5. Circuit : TESTE5

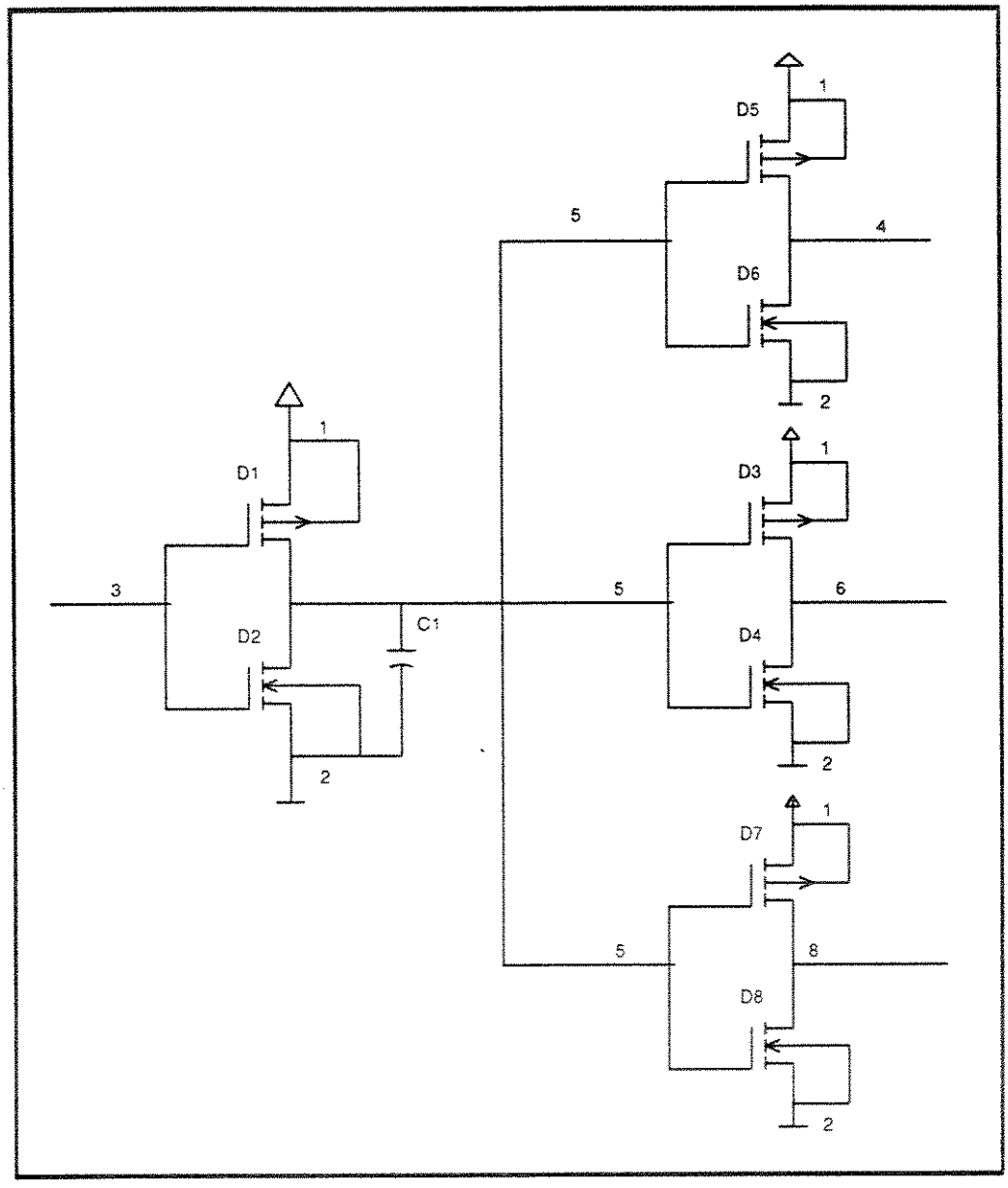
```
M6 4 5 2 2 NMOS L=2.5U W=4.5U 4 4 8 8
M8 8 5 2 2 NMOS L=2.5U W=4.5U 4 4 8 8
M5 1 5 4 1 PMOS L=2.5U W=7.5U 4 4 8 8
M7 1 5 8 1 PMOS L=2.5U W=7.5U 4 4 8 8
C1 5 2 1000F
M1 1 3 5 1 PMOS L=12.5U W=30U 4 4 8 8
M2 5 3 2 2 NMOS L=12.5U W=15U 4 4 8 8
M3 1 5 6 1 PMOS L=2.5U W=7.5U 4 4 8 8
M4 6 5 2 2 NMOS L=2.5U W=4.5U 4 4 8 8
```

```
                CREPE

   SISTEMA PARA VERIFICACAO DE REGRAS ELETRICAS
Circuito analisado: teste5
 ESTRUTURAS IDENTIFICADAS

Inversores encontrados
  inv 1 transistores  5  6   no de entrada= 5 no de saida= 4
  inv 2 transistores  7  8   no de entrada= 5 no de saida= 8
  inv 3 transistores  3  4   no de entrada= 5 no de saida= 6
Buffers  encontrados
  buf 1 transistores  1  2   no de entrada= 3 no de saida= 5
Portas nand encontradas
Portas nor encontradas
Gates de transmissao encontrados
Latchs encontrados

     AVISOS DE PROBLEMAS COM REGRAS
TIPO   COMENTARIO          ARGUMENTOS
ii5    Fan-out do buffer excedido   1  0.00000001  0.00000001
ii6    Buffer sensivel a spikes     1  0.00000001  0.00000001
Numero de transistores encontrados no circuito : 8
Numero de nos encontrados no circuito : 7
Consumo de potencia estimado para a frequencia media  : 0.00051558
```



TESTE5 FAN-OUT DO BUFFER P/CARGA DE 3 INVERSORES

A.6. Circuit : TESTE6

M6 6 3 2 2 NMOS L=5.25U W=4.5U 4 4 8 8
M5 1 3 6 1 PMOS L=5.25U W=4.5U 4 4 8 8
M1 1 5 7 1 PMOS L=2.5U W=4.5U 4 4 8 8
M2 1 4 7 1 PMOS L=2.5U W=4.5U 4 4 8 8
M3 7 4 8 2 NMOS L=5.0U W=6.0U 4 4 8 8
M4 8 5 2 2 NMOS L=5.0U W=6.0U 4 4 8 8
M7 1 10 12 1 PMOS L=10.25U W=9.0U 4 4 8 8
M8 12 14 11 1 PMOS L=10.25U W=9.0U 4 4 8 8
M9 11 10 2 2 NMOS L=10.25U W=3.0U 4 4 8 8
M10 11 14 2 2 NMOS L=10.25U W=3.0U 4 4 8 8
C1 7 2 1000
C2 11 2 1000
C3 6 2 1000

CREPE

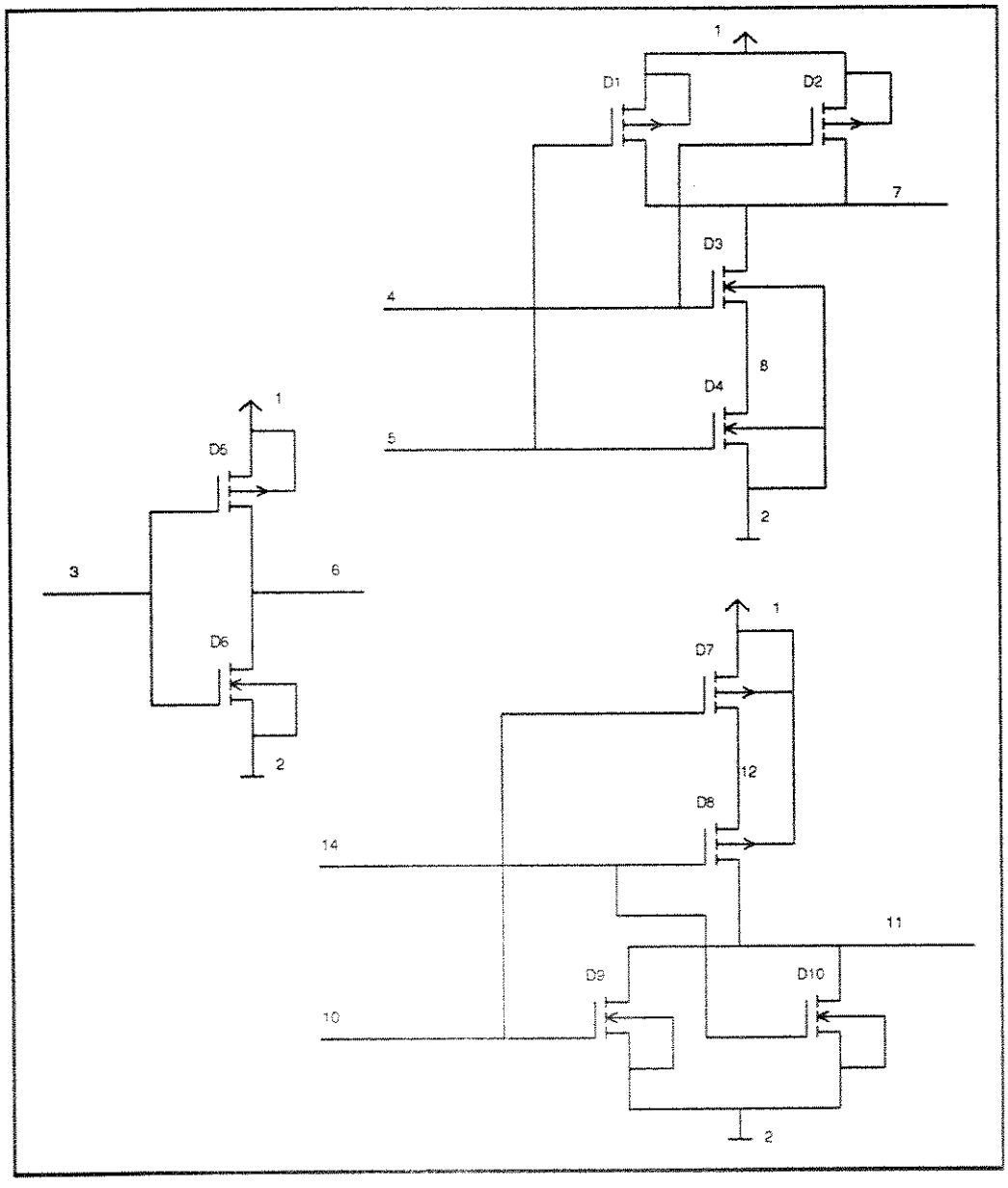
SISTEMA PARA VERIFICACAO DE REGRAS ELETRICAS
Circuito analisado: teste6
ESTRUTURAS IDENTIFICADAS

Inversores encontrados
inv 1 transistores 5 6 no de entrada= 3 no de saida= 6
Buffers encontrados
Portas nand encontradas
nand2 1 transistores 1 2 3 4 nos de entrada= 5 4 no de saida= 7
Portas nor encontradas
nor2 1 transistores 7 8 9 10 nos de entrada= 10 14 no de saida= 11
Gates de transmissao encontrados
Latchs encontrados

AVISOS DE PROBLEMAS COM REGRAS

TIPO	COMENTARIO	ARGUMENTOS
ii2	Beta do inversor mal dimensionado	1 0.6 1.4
ii2	Beta do nor mal dimensionado	1 1.4 0.6
ii5	Fan-out do inversor excedido	1 0.00000003 0.00000001
ii5	Fan-out do nand excedido	1 0.00000001 0.00000002
ii5	Fan-out do nor excedido	1 0.00000005 0.00000001
ii6	Inversor sensivel a spikes	1 0.00000001 0.00000001
ii6	Nand sensivel a spikes	1 0.00000001 0.00000001
ii6	Nor sensivel a spikes	1 0.00000002 0.00000001

Numero de transistores encontrados no circuito : 10
Numero de nos encontrados no circuito : 12
Consumo de potencia estimado para a frequencia media : 0.00091373



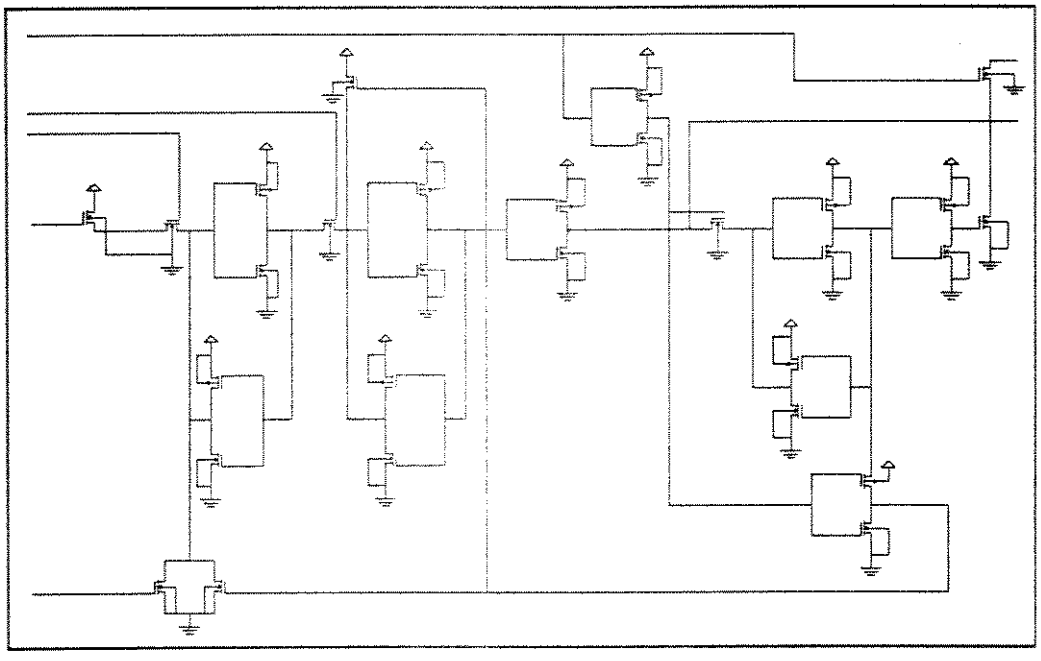
TESTE6 VERIFICAÇÃO DE SPIKES E DIMENSIONAMENTO

A.7. Circuit: TESTE7

M1 6 5 4 4 PMOS L=1.25U W=4.50U
M2 6 7 4 4 PMOS L=1.25U W=4.50U
M3 9 8 4 4 PMOS L=1.25U W=7.50U
M4 11 10 4 4 PMOS L=1.25U W=7.50U
M5 4 5 12 4 PMOS L=1.25U W=2.00U
M6 5 12 4 4 PMOS L=1.25U W=4.50U
M7 4 13 7 4 PMOS L=1.25U W=4.50U
M8 13 7 4 4 PMOS L=1.25U W=2.00U
M9 4 15 14 4 PMOS L=1.25U W=4.50U
M10 15 14 4 4 PMOS L=1.25U W=2.00U
M11 12 17 16 18 NMOS L=1.25U W=4.50U
M12 18 5 12 18 NMOS L=4.50U W=2.00U
M13 5 12 18 18 NMOS L=1.25U W=4.50U
M14 18 19 12 18 NMOS L=1.25U W=4.50U
M15 18 13 7 18 NMOS L=1.25U W=4.50U
M16 20 7 6 18 NMOS L=1.25U W=6.00U
M17 18 5 20 18 NMOS L=1.25U W=6.00U
M18 14 21 13 18 NMOS L=1.25U W=4.50U
M19 18 15 14 18 NMOS L=1.25U W=4.50U
M20 15 14 18 18 NMOS L=4.50U W=2.00U
M21 18 7 13 18 NMOS L=4.50U W=2.00U
M22 23 22 15 18 NMOS L=1.25U W=4.50U
M23 4 24 23 18 NMOS L=1.25U W=4.50U
M24 15 19 18 18 NMOS L=1.25U W=4.50U
M25 18 25 15 18 NMOS L=1.25U W=4.50U
M26 4 10 26 4 PMOS L=1.25U W=2.00U
M27 10 26 4 4 PMOS L=1.25U W=4.50U
M28 25 27 10 4 PMOS L=1.25U W=4.50U
M29 26 9 6 18 NMOS L=1.25U W=4.50U
M30 18 10 26 18 NMOS L=4.50U W=2.00U
M31 10 26 18 18 NMOS L=1.25U W=4.50U
M32 18 8 9 18 NMOS L=1.25U W=4.50U
M33 13 25 4 18 NMOS L=1.25U W=4.50U
M34 18 27 25 18 NMOS L=1.25U W=4.50U

M35 18 10 11 18 NMOS L=1.25U W=4.50U
M36 28 11 18 18 NMOS L=1.25U W=15.75U
M37 29 8 28 18 NMOS L=1.25U W=15.75U

```
                CREPE
    SISTEMA PARA VERIFICACAO DE REGRAS ELETRICAS
Circuito analisado: TESTE7
  ESTRUTURAS IDENTIFICADAS
 Inversores encontrados
    inv 1 transistores  3  32   no de entrada= 8 no de saida= 9
    inv 2 transistores  4  35   no de entrada= 10 no de saida= 11
    inv 5 transistores  7  15   no de entrada= 13 no de saida= 7
    inv 6 transistores  8  21   no de entrada= 7 no de saida= 13
    inv 7 transistores  9  19   no de entrada= 15 no de saida= 14
    inv 8 transistores  10  20   no de entrada= 14 no de saida= 15
    inv 9 transistores  26  30   no de entrada= 10 no de saida= 26
    inv 10 transistores  27  31   no de entrada= 26 no de saida= 10
 Buffers  encontrados
 Portas nand encontradas
    nand2 1 transistores  1  2  16  17   nos de entrada= 5  7   no de saida= 6
 Portas nor encontradas
 Gates de transmissao encontrados
    tra 1 transistores  8  33   no de entrada= 13   no de saida= 4
 Latchs encontrados
    latch 1 inversores 4  3   transistor 11   D= 16   En= 17   Q= 5
      AVISOS DE PROBLEMAS COM REGRAS
TIPO   COMENTARIO          ARGUMENTOS
i11    Tem pelo menos um caminho nmos entre vdd e vss   _266D
ii2    Beta do inversor mal dimensionado   5  0.6  1.4
ii2    Beta do inversor mal dimensionado   6  0.6  1.4
ii2    Beta do inversor mal dimensionado   7  0.6  1.4
ii2    Beta do inversor mal dimensionado   8  0.6  1.4
ii2    Beta do inversor mal dimensionado   9  0.6  1.4
ii2    Beta do inversor mal dimensionado   10  0.6  1.4
ii2    Beta do nand mal dimensionado   1  1.4  0.6
Numero de transistores encontrados no circuito : 37
Numero de nos encontrados no circuito : 26
Consumo de potencia estimado para a frequencia media  : 0.00009505
```



TESTE 7 CÉLULA DA BIBLIOTECA DO CPqD TELEBRÁS COM ALGUNS PROBLEMAS DE PROJETO

ANEXO B

EXEMPLO DE UMA EXECUÇÃO COMPLETA DO SISTEMA CREPE

```
**************************************************************************
**                                                                      **
**          Conferidor de REgras de PRojeto Eletrico                    **
**                                                                      **
**                       CREPE                                          **
**                                                                      **
**      versao  0.0                                                     **
**                                                                      **
**      por : TULIO IBANEZ NUNES        jun 1990                        **
**                                                                      **
**************************************************************************
```

```
 Entre com o nome do circuito ..... teste4.

INDIQUE QUE TIPOS DE REGRAS QUER VERIFICAR

Regras do tipo I (s/n)?

s.


Regras do tipo II (s/n)?

s.

Regras do tipo III (s/n)?

s.


    SISTEMA PARA CONVERSAO DE FATOS EM FORMATO SPICE

                PARA CLAUSULAS PROLOG


DE O NOME DO ARQUIVO DE ENTRADA -- teste4.spi

DE O NOME DO ARQUIVO DE SAIDA  --  teste4.ari

ENTRE COM O NUMERO DO NO DE VDD  ----  1

ENTRE COM O NUMERO DO NO DE VSS  ----  2

ENTRE COM O NUMERO DO NO DE CLOCK

CASO NAO TENHA CLOCK TECLE -1 :  ---- -1

ENTRE COM OS NOS DE ENTRADA DO CIRCUITO
```

```
PARA TERMINAR TECLE -1

NO DE ENTRADA --- 3

NO DE ENTRADA --- 5

NO DE ENTRADA --- 7

NO DE ENTRADA --- -1

ENTRE COM OS NOS DE SAIDA DO CIRCUITO

PARA TERMINAR TECLE -1

NO DE SAIDA ---  4

NO DE SAIDA ---  6

NO DE SAIDA ---  8

NO DE SAIDA ---  -1
```

```
-----------       ENCONTRANDO AS ESTRUTURAS        --------

-----------          APLICANDO AS REGRAS           --------

REGRAS DO GRUPO I

REGRAS DO GRUPO II

REGRAS DO GRUPO III

----------- GERANDO O ARQUIVO DE RELATORIO (.rel) --------
```

```
**************************************************
**                                              **
**     MODULO DE EXPLANACAO DOS AVISOS          **
**                                              **
**************************************************
Quer fazer alguma pergunta ao sistema?
Perguntar COMO chegou a um aviso ?
sim ou nao ---  sim.

Qual o aviso que esta interessado? (Por exemplo: aviso ii1)
(OBS: para terminar tecle fim)
Diga o tipo de aviso ---  ii1.

De onde se concluiu que:
Eu nao gerei este tipo de aviso!
Voce deve estar enganado.
Tente outra vez !!!

Diga o tipo de aviso ---  ii2.
O objetivo era provar a regra_ii2

ESTA REGRA VERIFICA SE A RELACAO BETA
```

DOS COMPONENTES ESTA PROXIMA DE UM PARA TERMOS OS TEMPOS

DE SUBIDA E DE DESCIDA DOS COMPONENTES PROXIMOS

O objetivo era provar a regra_ii2a

ESTA REGRA COMPARA OS BETAS ENCONTRADOS

COM OS BETA MAXIMOS E MINIMOS ADMISSIVEIS

De onde se concluiu que:

Beta do inversor mal dimensionado : Componente = 2 0.6 1.4

Diga o tipo de aviso --- ii5.

O objetivo era provar a regra_ii5

ESTA REGRA VERIFICA SE O FAN-OUT

DAS SAIDAS DOS COMPONENTES ESTA SENDO RESPEITADO

O objetivo era provar a regra_ii5a

ESTA REGRA COMPARA O TEMPO DE SUBIDA

E TEMPO DE DESCIDA COM OS VALORES MAXIMOS PARA QUE SEJA

EVITADO CONSUMO EXCESSIVO DE POTENCIA

De onde se concluiu que:

Fan-out do inversor excedido : Componente = 3 0.00000014 3.32015810E-09

Diga o tipo de aviso --- ii6.

```
O objetivo era provar a regra_ii6

ESTA REGRA INDICA A SENSIBILIDADE

DOS COMPONENTES A SPIKES NAS ENTRADAS

O objetivo era provar a regra_ii6a

ESTA REGRA COMPARA O TEMPO DE ATRASO

COM O VALOR MAXIMO PARA QUE SEJAM EVITADOS SPIKES

De onde se concluiu que:

  Inversor sensivel a spikes : Componente = 3 0.00000004 0.00000001


Diga o tipo de aviso --- ii8.


De onde se concluiu que:

Eu nao gerei este tipo de aviso!

Voce deve estar enganado.

Tente outra vez !!!


Diga o tipo de aviso --- fim.


ENTAO TUDO BEM SEM MAIS EXPLICACOES PARA O MOMENTO,


ATE MAIS !!!
```

B.1. ARQUIVO DE AVISOS GERADOS: teste4.rel

```
                    CREPE

     SISTEMA PARA VERIFICACAO DE REGRAS ELETRICAS
Circuito analisado: teste4
   ESTRUTURAS IDENTIFICADAS
  Inversores encontrados
     inv 1 transistores  5  6   no de entrada= 3 no de saida= 4
     inv 2 transistores  3  2   no de entrada= 7 no de saida= 8
     inv 3 transistores  4  7   no de entrada= 5 no de saida= 6
  Buffers  encontrados
  Portas nand encontradas
  Portas nor encontradas
  Gates de transmissao encontrados
  Latchs encontrados
```

AVISOS DE PROBLEMAS COM REGRAS

TIPO	COMENTARIO	ARGUMENTOS		
ii2	Beta do inversor mal dimensionado	2	0.6	1.4
ii2	Beta do inversor mal dimensionado	3	0.6	1.4
ii5	Fan-out do inversor excedido	3	0.00000014	3.32015810E-09
ii6	Inversor sensivel a spikes	3	0.00000004	0.00000001

Numero de transistores encontrados no circuito : 6

Numero de nos encontrados no circuito : 8

Consumo de potencia estimado para a frequencia media : 0.00077778

B.2. ARQUIVO DE ENTRADA EM FORMATO SPICE : teste4.spi

```
M6  4  3  2  2 NMOS L=1.25U W=4.5U 4 4 8 8

M2  8  7  2  2 NMOS L=1.25U W=4.5U 4 4 8 8

M5  1  3  4  1 PMOS L=1.25U W=7.5U 4 4 8 8

M3  1  7  8  1 PMOS L=1.25U W=4.5U 4 4 8 8

M4  1  5  6  1 PMOS L=7.5U W=1.25U 4 4 8 8

M7  6  5  2  2 NMOS L=1.25U W=4.5U 4 4 8 8

C1  4  2  1000

C2  6  2  1000

C3  8  2  1000
```

B.3. ARQUIVO GERADO PELO CONVERSOR SPICE-PROLOG: teste4.ari

```
vdd(1).

vss(2).

ck(-1).

ent([3,5,7]).

sai([4,6,8]).

mos(6,4,3,2,2,n,1.25,4.5).

mos(2,8,7,2,2,n,1.25,4.5).

mos(5,1,3,4,1,p,1.25,7.5).

mos(3,1,7,8,1,p,1.25,4.5).

mos(4,1,5,6,1,p,7.5,1.25).

mos(7,6,5,2,2,n,1.25,4.5).

numnos(8).

numtrans(6).

cap(4,1.008E-012).

cap(2,1.2E-014).

cap(3,2.295E-014).
```

```
cap(8,1.008E-012).

cap(7,1.72125E-014).

cap(1,1.2E-01#).

cap(6,1.008E-012).

cap(5,2.295E-014).

cap_total(3.11111E-012).

no([1 ,[3, d],[4, d],[5, d]]).

no([2 ,[2, f],[6, f],[7, f]]).

no([3 ,[5, g],[6, g]]).

no([4 ,'5, f],[6, d]]).

no([5 ,[4, g],[7, g]]).

no([6 [4, f],[7, d]]).

no([7 ,[2, g],[3, g]]).

no([8 ,[2, d],[3, f]]).
```