Boletim Técnico da Escola Politécnica da USP

Departamento de Engenharia Eletrônica

ISSN 1413-2206
BT/PEE/9708

Requisitos para o Mapeamento Tecnológico em Projetos de Microeletrônica

Luciano de Oliveira Corrêa de Brito
José Roberto de Almeida Amazonas

São Paulo - 1997

O presente trabalho é um resumo da dissertação de mestrado apresentada por Luciano de Oliveira Corrêa de Brito, sob orientação do Prof. Dr. José Roberto de Almeida Amazonas: "Requisitos para o Mapeamento Tecnológico em Projetos de Microeletrônica", defendida em 20/12/96, na Escola Politécnica.

A íntegra da dissertação encontra-se à disposição com o autor e na Biblioteca de Engenharia de Eletricidade da Escola Politécnica/USP.

Brito, Luciano de Oliveira Corrêa de
Requisitos para o mapeamento tecnológico em projetos de microeletrônica / L.O.C. de Brito, J.R.A. Amazonas. -- São Paulo : EPUSP, 1997.
p. -- (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia Eletrônica, BT/PEE/9708)

1. Microeletrônica 2. Circuitos integrados I. Amazonas, José Roberto de Almeida II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Eletrônica III. Título IV. Série

ISSN 1413-2206 CDD 621.381
621.3815

REQUISITOS PARA O MAPEAMENTO TECNOLÓGICO EM PROJETOS DE MICROELETRÔNICA

Luciano de Oliveira Corrêa de Brito[i] e José Roberto de Almeida Amazonas[ii]

RESUMO

O problema abordado consiste em um processo automático de tradução, a partir de duas sintaxes distintas que definem as descrições válidas de um circuito integrado. Mais do que simplesmente traduzir de uma descrição para outra, deseja-se que o processo avalie a semântica da tradução, isto é, procure efetuar a melhor das possíveis traduções segundo algum critério.

Inicialmente, desejava-se resolver o problema através da implementação de uma ferramenta computacional utilizada em microeletrônica, denominada mapeador tecnológico. Contudo, as dificuldades encontradas fizeram com que o trabalho se redirecionasse no sentido de analisá-las e criar uma infra-estrutura adequada que permita solucionar o problema inicial de tradução, através de um mapeador tecnológico.

O presente trabalho irá detalhar os requisitos necessários à implementação de um mapeador tecnológico. Será apresentado um panorama das soluções descritas em literatura, o detalhamento das condições necessárias para o suporte adequado a algoritmos de mapeamento e a proposta de uma implementação de infra-estrutura para esse tipo de ferramenta. Finalmente, serão apresentadas partes iniciais de implementações da estrutura proposta.

ABSTRACT

The problem in focus consists of an automatic translation process based upon two distinct syntaxes, which define valid descriptions of an integrated circuit. More than a simple translation, from one description to another, the process should deal with the semantics of the translation, that is, it tries to achieve the best translation in accordance to certain criteria.

Formerly, the original goal was to solve the problem with a microelectronics software tool, named technology mapping tool. However, the difficulties that have been met, lead this paper to describe them ad propose an adequate infra-structure to allow the solution of the initial translation problem, using a technology mapping tool.

The present work will detail requirements to implement a technology mapper. It will be presented a broad discussion of solutions reported in the literature, the refinement of the requirements to build mapping algorithms and the proposal of an infra-structure to support this type of tool. Finally, it will be presented portions of the code that implements the proposed infra-structure.

---

i FORMTEK, Inc. - A Lockheed Martin Co. - luciano_brito@formtek.com

ii Escola Politécnica da Universidade de São Paulo - Departamento de Engenharia de Eletricidade - jra@lcs.poli.usp.br

1. INTRODUÇÃO

1.1. Objetivo

Qualquer que seja o projeto de um CI, é necessário que, a partir de um certo momento, este circuito esteja descrito em termos de células vinculadas a uma tecnologia de fabricação. Tais células são portas lógicas que implementam funções booleanas básicas, como AND, NAND, OR, NOR, XOR, contadores, registradores e demais estruturas utilizadas em projetos de CI. Os fabricantes costumam descrever as propriedades das células, em livros e arquivos eletrônicos, chamados de bibliotecas. Tais bibliotecas serão referenciadas neste trabalho como bibliotecas específicas por estarem associadas a uma determinada "foundry".

Uma biblioteca genérica, por outro lado, pode ter a mesma estrutura e sintaxe de uma biblioteca específica, porém suas células não são, necessariamente, reais. Desta forma, ela é abstrata, fictícia, podendo representar diferentes condições de projeto. Uma aplicação para uma biblioteca genérica é sua utilização como representativa de uma tecnologia, através da incorporação de parâmetros e elementos próximos aos reais, descritos por diversos fabricantes. Outra aplicação é seu uso no estudo e testes de ferramentas computacionais, quando a utilização de valores extremos (e irreais) dos parâmetros pode contribuir para a análise de sensibilidade dos sistemas e sugerir estratégias de abordagem do problema.

Denomina-se mapeador tecnológico, um algoritmo que permita o mapeamento da descrição de um circuito integrado em uma determinada tecnologia de fabricação para outra, buscando otimizar a área ou os tempos de propagação do circuito final. Para tanto é necessária uma infra-estrutura específica para suportar o algoritmo. Como qualquer programa computacional, são necessários dados de entrada e definições de variáveis e estruturas capazes de suportar o subsequente processamento e geração de resultados.

Este trabalho procura prover o leitor com informações suficientes a respeito da infra-estrutura necessária de modo a permitir a implementação de mapeadores tecnológicos mais robustos e abrangentes que os reportados em literatura.

1.2. Justificativa

Um mapeador tecnológico é necessário em um ciclo de projetos de circuitos integrados, quer se utilize uma biblioteca tecnológica específica ou uma genérica. Quando se sabe ao certo a tecnologia a ser utilizada, o mapeamento tecnológico se justifica como um otimizador do circuito [BRG88] e permite o trabalho com um "second source". Quando a tecnologia de fabricação é desconhecida, o processo permite que se projete em uma biblioteca genérica e, posteriormente, quando o projeto já estiver validado, se defina a biblioteca alvo.

Para a elaboração do mapeador tecnológico é necessária:

a) a concepção e o provimento das bibliotecas em formato eletrônico, tornando-as disponíveis ao algoritmo de processamento;

b) a definição de uma estrutura de dados capaz de suportar a manipulação das informações de modo eficiente e

c) uma maneira de abordar o problema e ou graus de intervenção que o operador irá dispor para guiar o processo de mapeamento e orientá-lo em direção ao ótimo.

As bibliotecas são fundamentais ao processo, pois constituem a base de conhecimento sobre a qual irá operar qualquer algoritmo. Como será detalhado, normalmente não é possível utilizar diretamente as bibliotecas fornecidas pelos fabricantes, porque nem todas as informações necessárias ao mapeamento estão explicitamente nelas descritas. Além disso, há que se definir um padrão para bibliotecas que se destinam a ferramentas de mapeamento tecnológico, pela necessidade de conexão e coerência com outras ferramentas do ambiente de projeto e de especificação de um formato para as bibliotecas genéricas.

A estrutura de dados que irá armazenar as informações coletadas nas bibliotecas e permitir sua manipulação adequada para obter um bom mapeamento é a parte mais complexa do sistema. Tal estrutura deve permitir um preenchimento rápido, mesmo em se lidando com arquivos hierárquicos, suportar as diversas células existentes, permitir o rápido acesso a informações sobre elas e possibilitar a rápida obtenção da estrutura topológica do circuito.

A base deve ser flexível de modo a permitir observar o circuito de diferentes pontos de vista - topológico, lógico, testabilidade, hierárquico, nivelado - em qualquer fase do processamento. Com isso, permite-se a aplicação de algoritmos que implementam diferentes heurísticas, sobre uma mesma estrutura de armazenamento interna, podendo-se, então, obter comparações de desempenho muito mais acuradas.

Finalmente, a capacidade de intervenção do usuário no processo de mapeamento é fundamental do ponto de vista prático. Em certos casos, não se deseja submeter parte de um circuito ao processo, querendo o projetista traduzi-lo manualmente. Em outros casos, deseja-se que o mapeamento seja realizado somente dentro de cada módulo do sistema, preservando-se a interface entre os diversos sub-circuitos. Mesmo quando se permite à ferramenta a intervenção total no circuito proposto, ainda assim são necessários parâmetros e condições de contorno que guiem a execução do algoritmo. Se, durante esta execução, for permitida a intervenção do projetista, então pode-se aumentar as chances de se obter um circuito otimizado em casos onde a simples execução do programa não esteja oferecendo resultados satisfatórios.

1.3. Organização do Trabalho

O presente trabalho é composto dos seguintes capítulos:

1. Introdução
2. Cenário Atual e Proposta de Trabalho
3. Requisitos para o Mapeamento Tecnológico em Microeletrônica
4. Proposta de Mapeador Tecnológico
5. Conclusões

2. Cenário Atual e Proposta de Trabalho

2.1. Classificação das Heurísticas de Mapeamento

Os mapeadores tecnológicos descritos em literatura podem ser classificados segundo a estratégia utilizada, a saber:

a) mapeadores baseados em operações booleanas: McMAP [BRG88] e DECAF [LIS87]
b) mapeadores baseados em sistemas especialistas: SOCRATES [GEU85], LORES/EX[ISH88],
c) mapeadores baseados na topologia do circuito: ASYL [CRA91], DAGON [KEU87]
d) mapeadores voltados a FPGAs: CHORTLE-CRF [FRA91], Xmap [KEV91]
e) mapeadores para casos específicos: MIS [BRA87], Lilly [MAS91], Fusion [YOS91]
f) mapeadores híbridos: CERES [MAI93], DIRMAP[LEG88], SKOL[BER91].

2.2. Sumário dos Trabalhos Reportados em Literatura.

Nome	Heurística	Opera em circuitos do tipo	Descrição do circuito	Otimização
DECAF [LIS87]	Operações Booleanas	Combinacionais	Expandida	Global
SOCRATES [GEU85]	Sistema de Regras/Topologia	Combinacionais	Expandida, Equações e saídas ESPRESSO	Local
ASYL [CRA91]/	Topologia	Combinacionais	Expandida	Local

DAGON[KEU87]				
Chortle-crf [FRA91]	Específico para FPGAs	FPGAs	Expandida	Local
Techmap(MIS) [BRA87]	Topologia e testabilidade	Combinacionais	Expandida, Equações	Local
Lilly(MIS) [MAS91]	Topologia, área de "lay-out" e interconexão	Combinacionais	Expandida, Equações	Local
FUSION [YOS91]	Topologia	Combinacionais e seqüenciais	Hierárquico	Local
CERES [MAI93]	Operações Booleanas Topologia	Combinacionais	Hierárquico	Global por partes

Dentre as deficiências dos mapeadores acima, pode-se mencionar a deficiência de uma base de dados capaz de suportar os diversos pontos de vista através dos quais se pode manipular uma descrição de um circuito. A ausência de menção sobre como prover as descrições reais de células, existentes nos manuais dos fabricantes, e como lidar com aspectos práticos do ponto de vista do projetista também constituem aspectos não abordados previamente. De maneira geral, os algoritmos apresentados lidam somente com células combinacionais e limitadamente com outros tipos de células. A seguir, irá se definir requisitos reais para o mapeador tecnológico e, a partir daí, desenvolver uma infra-estrutura adequada à ferramenta.

3. REQUISITOS PARA O MAPEAMENTO TECNOLÓGICO

3.1. Requisitos comuns

O mapeador tecnológico apresenta requisitos comuns em ambos os casos de otimização:

3.1.1. Requisitos dependentes do algoritmo:

a) a minimização deverá assumir um caráter global, para garantir o melhor mapeamento possível;

b) o mapeamento deve ser feito a partir de qualquer biblioteca para qualquer biblioteca, sem impor restrições tecnológicas, não se limitando a elementos combinacionais; o programa também deve suportar o uso de uma biblioteca genérica;

c) validação do algoritmo por circuitos de validação ("benchmarks") genéricos, reais e teóricos;

d) suportar a definição de bibliotecas adequadas e sua expansão, inclusive para biblioteca genérica.

3.1.2. Requisitos funcionais e de implementação:

a) habilidade para permitir como entrada um arquivo hierárquico de descrição do circuito, o arquivo expandido ou equações booleanas (minimizadas ou não, em dois ou mais níveis), de modo a se inserir mais facilmente em diversos ambientes e metodologias de projeto, podendo ser acessado de diversos pontos em um fluxo de projeto. O uso de arquivos "flat" é necessário quando se deseja minimização de tempo; isso porque, durante o mapeamento, o caminho crítico pode estar passando por um módulo composto de células de biblioteca, que precisa, então, ser remapeado;

b) ser codificado em e de acordo com uma linguagem padronizada e amplamente disponível, como C, para garantir máxima portabilidade do programa;

c) estabelecimento de índices que determinem a capacidade total de memória a ser utilizada e a ordem de grandeza do tempo de processamento em função do número de portas a serem processadas;
d) suportar modularização, refinamento e escopo.

3.1.3. Requisitos de projeto:

a) manutenção do número de entradas primárias e de saídas primárias do circuito original pois elas estão vinculadas a pinos do CI;
b) capacidade de gerar múltiplos mapeamentos, pois caso o resultado de uma simulação seja insatisfatório, o projetista tem a possibilidade de tentar outro resultado que atenda suas restrições;
c) capacidade de gerar múltiplos formatos de saída (descrição hierárquica, descrição expandida ou equações). É fundamental prover mecanismos que permitam fazer a correspondência entre as descrições na biblioteca origem e as correspondentes descrições mapeadas na biblioteca alvo, para permitir eventuais alterações que o projetista deseje fazer;
d) suportar alteração da função custo;
e) admitir diversos parâmetros de intervenção do usuário, assim como pontos de verificação do processamento;
f) integrar-se com outras ferramentas no mesmo nível de abstração.

A implementação de diversos itens acima, como o suporte a bibliotecas genéricas, arquivos de entrada e saída com suporte à hierarquia e a integração com ferramentas de mesmo nível de abstração, impõe a necessidade global de se criar uma base de dados adequada ao suporte de tais requisitos.

A seguir detalham-se alguns dos aspectos mencionados.

3.2. Circuitos de teste

Uma vez criado um mapeador, o problema que se põe é como validar a ferramenta, isto é, responder às seguintes perguntas:

a) a ferramenta é capaz de levar um circuito descrito em uma biblioteca qualquer para a biblioteca destino, ou seja, chega-se a pelo menos um resultado, sempre ?
b) em se conseguindo uma descrição na biblioteca destino, é possível obter outra melhor, em termos de área ou de tempos de propagação ? Ou seja, o mapeamento para a mesma biblioteca realmente leva a resultado melhor ? Quão melhor ?
c) qual o poder de interferência que a ferramenta oferece para guiar o processo em direção a uma descrição ?
d) qual a sensibilidade do algoritmo ? As restrições para se chegar a uma solução são fortes ou fracas ?

A primeira pergunta é básica e procura mostrar que, conceitualmente, o mapeador executa sua função primária: transformar uma descrição não implementável em uma factível, se possível atendendo às restrições fornecidas. Em alguns casos, isso poderá não ser possível.

Por exemplo, caso se saiba, previamente, que um circuito já se encontra em uma descrição mínima de área e tempo, o processo de otimização não poderá levar a solução, exceto se os mecanismos de intervenção fornecidos permitirem ao algoritmo não abortar uma solução que não atenda os critérios. A solução em um caso desses é primeiro traduzir (mapear) e, depois, tentar otimizar.

A segunda pergunta procura avaliar a capacidade de otimização da ferramenta e, nesse caso, há uma sobreposição conceitual com ferramentas de síntese. Não é de se esperar resultados excepcionais nesses casos. Porém, quando for possível guiar o processo de modo a se mapear e otimizar simultaneamente, então, espera-se chegar a diversas soluções otimizadas. Alguns dos mapeadores já apresentados exploram essa característica.

A terceira pergunta responde quão controlável será o sistema e a quarta, quão sensível a parâmetros e condições será a solução.

O problema prático que se põe é determinar quais circuitos ("benchmarks") deverão ser exercitados para se responder às perguntas acima. Como exemplo, o uso dos circuitos de teste do Microelectronics Center of North Carolina (MCNC), também conhecidos como circuitos ISCAS [ISC89]. Este último conjunto foi um dos primeiros conjuntos de "benchmarks" desenvolvidos e muito comum em literatura e em trabalhos de testabilidade. O conjunto ISCAS é questionável para esse trabalho por diversas razões:

a) são circuitos meramente combinatórios;

b) são circuitos caracterizados pela dificuldade de teste, com redundâncias e de minimização difícil;

c) constituem exemplos de circuitos reais ou circuitos sintetizados para avaliar capacidades de síntese e testabilidade, não tendo sido, portanto, elaborados para teste de mapeamento.

Assim, nada os torna melhor do que outros circuitos quaisquer, para propósitos desse trabalho. Mesmo o seu uso para comparação com as ferramentas de literatura não levarão a resultados conclusivos: caso se obtenha resultados melhores, usando os ISCAS, não se pode dizer que o algoritmo ou a ferramenta é melhor que qualquer uma já reportada. O mesmo vale se os resultados forem piores.

Pode-se, por analogia, adotar circuitos altamente seqüenciais para testes. É necessário, portanto, elaborar uma estratégia que apresente circuitos que, ao serem exercitados, permitam extrapolar seus resultados para os demais. Nesse sentido, sugere-se a criação de circuitos especiais, como descrito a seguir.

Assumindo-se por premissas que:

a) dado um circuito a ser projetado, quanto maior a área utilizada para sua implementação, menor o tempo de propagação entre entradas e saídas (maior a velocidade);

b) as diferentes tecnologias de fabricação oferecem bibliotecas com parâmetros próximos;

pode-se construir um gráfico do tipo da Figura 1 - Gráfico área x atraso para diversas implementações.:

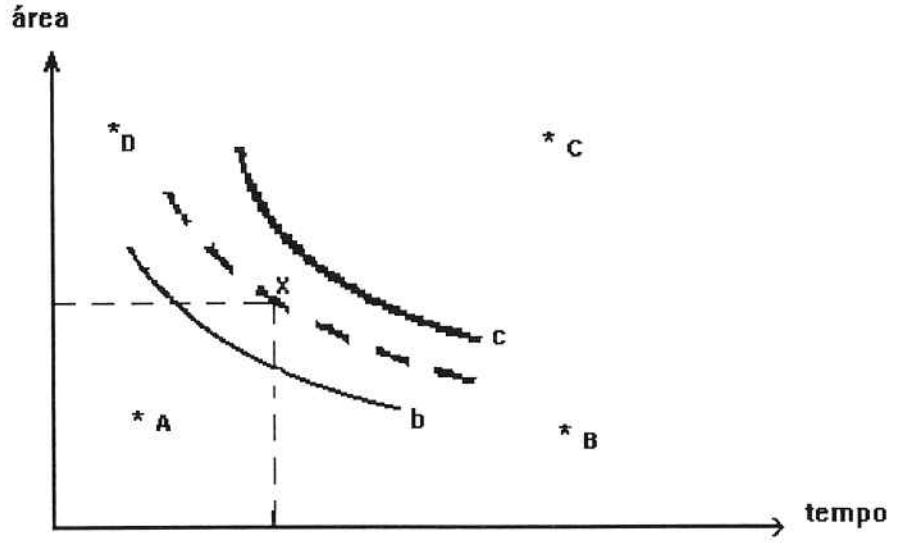

Figura 1 - Gráfico área x atraso para diversas implementações.

onde as curvas b e c representam duas famílias de curvas relativas a uma tecnologia de fabricação típica. O ponto X representa um circuito real qualquer e implementável e os pontos A, B,

C e D, circuitos fictícios, não implementáveis por representarem pontos de uma curva de tecnologia absurda.

Os pontos absurdos do gráfico são aqueles que irão permitir responder à primeira das indagações, porque estão suficientemente "longe" das curvas reais e, portanto, não consistem em circuitos que poderiam ser mapeados apenas por estarem próximos a implementações reais.

Assim, por exemplo, circuitos tipo A são aqueles de área e tempos de propagação mínimos, podendo ser circuito implementados em lógica de dois níveis, onde as áreas das portas sempre é unitária e o tempo de propagação unitário. Logo, a área A de um circuito tipo A é A(A)= 2*número de portas e tempo de propagação, T, de um circuito tipo A é dado por T(A)= 2.

Circuitos D são aqueles em dois níveis onde a área de cada porta é, por exemplo, igual a número de entradas da porta (ou outro valor, como o máximo da biblioteca) e os tempos de propagação sempre iguais a 1. Então, área A(D)= Σ (entradas de porta i) e tempo de propagação T(D)= 2.

Circuitos B são aqueles em dois níveis, cujas áreas das portas são unitárias, mas os tempos de propagação iguais ao número de entradas da porta (ou outro valor, como o máximo da biblioteca). Então, A(B)=2 e T(B)= Σ (entradas de porta i). Finalmente, circuitos C, são aqueles cujas áreas e tempos de propagação das portas são proporcionais ao número de entradas de cada porta. A(C)= Σ (entradas de porta i) e T(C)= Σ (entradas de porta i).

O uso de circuitos dos tipos apresentados podem, então gerar as respostas às perguntas acima. Uma vez se atingindo uma curva real de implementação, o mapeamento de biblioteca para biblioteca permite avaliar como a ferramenta desloca a implementação dentro da curva, como os parâmetros de interferência afetam o processo e com que sensibilidade.

3.3. *Bibliotecas de mapeamento e sua geração*

Os arquivos de consulta de um mapeador tecnológico devem levar em conta os seguintes aspectos:

a) o mapeador encontra-se aproximadamente no mesmo nível hierárquico das ferramentas de testabilidade e auditoria, no eixo estrutural de um diagrama em Y [GAJ88];
b) os arquivos de consulta do mapeador devem poder atender às necessidades das demais ferramentas de mesmo nível; nesse sentido, o arquivo de consulta passa a ser definido como uma biblioteca, no sentido das bibliotecas de projeto utilizadas para simulação;
c) bibliotecas como as utilizadas para simulação encontram-se em um nível estrutural mais baixo, uma vez que suas primitivas vão até transistores e que possuem algumas informações elétricas e de "lay-out";
d) as bibliotecas para programas de mapeamento devem se referir a um nível comportamental-estrutural, necessitando de outros tipos de informações que as disponíveis em bibliotecas normalmente disponíveis;
e) a complexidade e quantidade de informações existentes em um biblioteca justifica a necessidade de criação de um mecanismo de entrada de dados, que assegure sua integridade.

Considerando-se tais aspectos, normalmente se depara com a necessidade de uma biblioteca em um nível de abstração superior ao das usualmente disponíveis. Essa biblioteca deve atender ao conjunto de ferramentas que estejam em nível comportamental-estrutural, como programas de testabilidade, auditoria e mapeamento.

Da mesma forma, um gerador de bibliotecas para nível comportamental-estrutural é necessário para facilitar a criação de arquivos de biblioteca, preferencialmente em formato ASCII. O formato ASCII contribui para manter a portabilidade da biblioteca gerada, embora permita que ela seja acessada externamente por um editor de texto. Um eventual gerador deve verificar e garantir a sintaxe do arquivo gerado não se responsabilizando, contudo, em relação a informações incoerentes que lhe sejam fornecidas. Devem ser suportadas funções de:

a) inclusão de células;
b) remoção de células;
c) consultas;
d) replicação de subconjuntos da biblioteca.

O usuário deve ter a sua disposição uma interface amigável para a execução das operações acima. O programa deve facilitar ao máximo a entrada de dados, no sentido de certificar-se de sua correção sintática e evitar ambigüidades. Nesse sentido, opções devem ser dinamicamente ajustadas dentro do utilitário, de modo a implementar regras sintáticas e semânticas.

É necessário que a biblioteca suporte todas as células descritas em um manual de uma dada tecnologia, garantindo seu processamento no menor tempo possível. Deve-se evitar que o arquivo gerado não se torne muito grande, assumindo-se formato ASCII.

3.4. Arquivos de consulta

O programa irá precisar, para seu processamento, de informações acerca das bibliotecas de origem e de destino. Sugere-se que, para cada biblioteca de uma dada tecnologia, sejam feitos dois arquivos: um contendo uma descrição das células em termos de área, atrasos e outras informações lógico-estruturais e outro com um "netlist" implementável para cada célula, seguindo sintaxe similar à utilizada para descrição de circuitos hierárquicos. Ambos os arquivos criados serão também considerados bibliotecas daquela tecnologia.

3.5. Arquivos de entrada e saída

O programa de mapeamento deve suportar diferentes formatos de entrada para assegurar sua inserção em diversos fluxos de projeto: entrada por meio de "netlist" hierárquico, Entrada por meio de "netlist flat" (nivelados) e por meio de equações como as geradas pelo ESPRESSO [BRA84].

Tanto nos arquivos hierárquicos como nos expandidos, deve ser definida uma extensão de linguagem para assinalar os blocos que não devem sofrer mapeamento algum. Outras eventuais extensões podem ser definidas para evitar sobrecarregar o pré-processamento do programa.

3.6. Arquivos de correspondência

É necessário, também, providenciar alguma forma que permita ao usuário identificar, no circuito gerado, os "gates" do circuito original. Para tanto, no arquivo de saída, deverão ser dadas informações das portas eliminadas e daquelas que foram substituídas pelas suas equivalentes De Morgan. A localização de tais informações estará na mesma região do arquivo onde foram processadas as mudanças, sendo que a descrição original pode ser substituída ou comentada, conforme opção do usuário. Assim, por exemplo, em um arquivo hierárquico, dentro de cada módulo, serão fornecidas tais informações. Quando as mudanças ocorrerem entre módulos, serão criadas novas fronteiras que definirão o novo módulo gerado e dentro das quais estarão as descrições das mudanças.

Células sintetizadas podem ser colocadas em um arquivo a parte, que o usuário poderá incluir, se desejar as descrições propostas. Quaisquer ocorrências deverão ser descritas em um arquivo de informações .LOG, a ser gerado ao longo do processamento. Informações adicionais, como número total de portas obtidas ou outras estatísticas, poderão ser colocadas nesse arquivo.

No caso de se gerar arquivo expandido, deve-se procurar, também, gerar as descrições das alterações em posições equivalentes do arquivo de entrada. Quando a entrada for de um tipo e a saída de outro, então as descrições poderão estar desordenadas. Devem ser mantidas as instanciações definidas nos arquivos originais e identificar os nós criados, no processo.

3.7. Modularização, refinamento e escopo

A modularização do programa pretende estruturar as diversas rotinas em blocos fisicamente isolados, logicamente coerentes e compiláveis, de sorte a se poder alterar partes do programa sem se preocupar com o todo. Esse enfoque permite, também, que sejam alocados blocos para procedimentos ainda não existentes, formando "ganchos", que permitam a fácil inclusão de novas rotinas. O

programa deve buscar o maior número de rotinas possível, em detrimento do tamanho do arquivo executável, a menos que este seja um fator limitante. A modularização permite uma visão clara de dependência de rotinas, possibilitando um enfoque por objetos e facilitando o uso de "overlays", se isso se fizer necessário.

O refinamento na execução do processo reporta a uma visão hierárquica, na qual se define o tipo de procedimento desejado. Neste sentido, definem-se os seguintes níveis:

a) NÍVEL 0 - Tradução - Quando se solicitar um mapeamento em nível 0, está-se definindo apenas um processo de tradução do circuito original. Assim, o mapeador apenas buscará na biblioteca alvo células que sejam iguais às do circuito original, trocando seus nomes. Caso se entre com equações, busca-se implementá-las na biblioteca alvo. Quando um elemento não for encontrado no destino, apela-se para sua descrição e elabora-se sua síntese com as células disponíveis.

b) NÍVEL 1 - Tradução e otimização - Neste nível de mapeamento, a lógica combinatória constituída de primitivas ("hardcells") será mapeada de forma otimizada. As "softcells" não serão expandidas; desta forma, multiplexadores, somadores e outras funções combinatórias serão preservadas, sendo feita sua simples tradução. Com esses dois níveis, já se pode analisar o que é mais vantajoso no mapeamento da biblioteca A para B: utilizar direto o nível 1, ou primeiro utilizar o nível 0 de A para B e, posteriormente, utilizar o nível 1, neste caso mapeando de B para B.

c) NÍVEL 2 - Geral - Neste enfoque as células todas serão expandidas e o procedimento de otimização aplicado. É o caso mais geral.

Em qualquer um desses níveis, entretanto, o usuário poderá definir o âmbito do mapeamento (escopo), isto é, se ele será feito dentro de cada módulo somente ou também entre módulos. As células sequenciais serão traduzidas em quaisquer dos níveis.

Uma ferramenta deverá suportar o caráter global. Todavia, algumas operações podem ser realizadas a nível local, a saber:

a) uma operação de expansão feita internamente a um módulo, em um enfoque local, seria feita todas as vezes que fosse possível, caso se encarasse o circuito expandido;

b) uma operação de fusão interna a um módulo segue o mesmo raciocínio anterior;

c) a operação de determinação de equivalentes De Morgan também se enquadra nessa visão.

Os três procedimentos mencionados permitem ao usuário fazer um mapeamento intra-módulos e, portanto, local. Do ponto de vista algorítmico, pode-se, sempre, aplicar os procedimentos internamente aos módulos e, posteriormente, se for o caso, nas fronteiras entre módulos. Dessa forma, ao operar primeiro localmente, respeitando a hierarquia, está-se reduzindo o espaço de busca em relação ao mesmo procedimento aplicado ao arquivo expandido, uma vez que já se está definindo uma ordem inicial: primeiro serão processadas as portas internas aos módulos.

Essa abordagem acelera o tempo de execução do programa porque uma alteração local em um módulo já se reflete em todas as suas instâncias.

3.8. Parâmetros e pontos de intervenção

Durante o mapeamento, diversos pontos de intervenção e parâmetros podem ser estipulados. Parâmetros normalmente estão ligados ao algoritmo a ser implementado. Número de iterações, constantes associadas a função custo, número de tentativas, número máximo de "fan-outs" e soluções a serem geradas constituem alguns exemplos de parâmetros.

Pontos de intervenção permitem ao usuário parar o processamento e alterar parâmetros ou o próprio circuito em processo. Por exemplo, o usuário pode intervir para definir quais portas devem ser processadas e quais não, expandindo as possibilidades de mapeamento por níveis.

3.9. Base de dados para suporte ao mapeamento

3.9.1. Definições

É necessária uma base de dados que contenha informações do circuito segundo diferentes enfoques. Em um circuito, podemos identificar as classes portas, nós e linhas.

Portas: São aqueles elementos funcionais descritos na biblioteca de origem do circuito.

Nós: Define-se que cada nó (não entrada primária) tem associado a si uma única porta, que é aquela da qual o nó é saída

Linhas: são as conexões de um circuito que partem de uma porta ou de um nó de ramificação múltipla ("fan out stem") e chegam a uma porta ou nó de ramificação múltipla.

Define-se, também, uma classe ramo, análoga às linhas.

Define-se RAMO como aquele que une dois nós, passando por uma porta.

A Figura 2 - Ilustração dos conceitos: nó, porta, linha e ramo., ilustra esses conceitos.

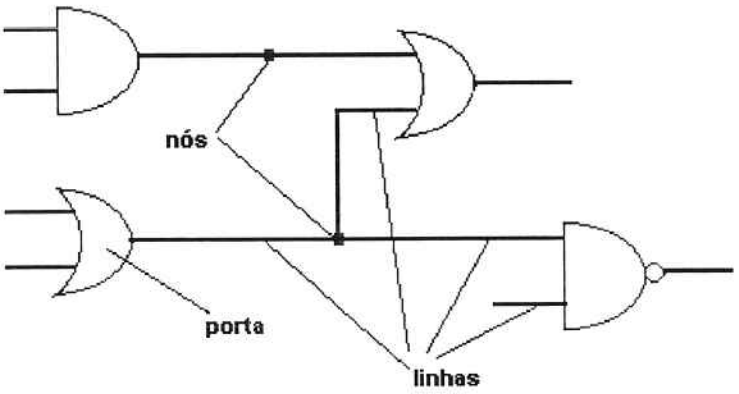

Figura 2 - Ilustração dos conceitos: nó, porta, linha e ramo.

Note que o conceito de ramo é diferente do de linha. Considere uma grafo associado a um circuito, onde os nós do circuito são os nós do grafo. Então, o ramo corresponde às linhas de conexão nesse grafo. Com essa definição, tem-se que cada nó X tem nós que o originam e nós dos quais X é origem. Dados os nós origem e destino de um determinado nó X, estão definidos, implicitamente, os ramos que interligam esses nós: um ramo saindo de cada nó de origem e indo para o nó X e um ramo saindo de X e indo para cada nó destino.

A Figura 3 - Ilustração do conceito de RAMO., ilustra o conceito, para o mesmo circuito mostrado na figura anterior.

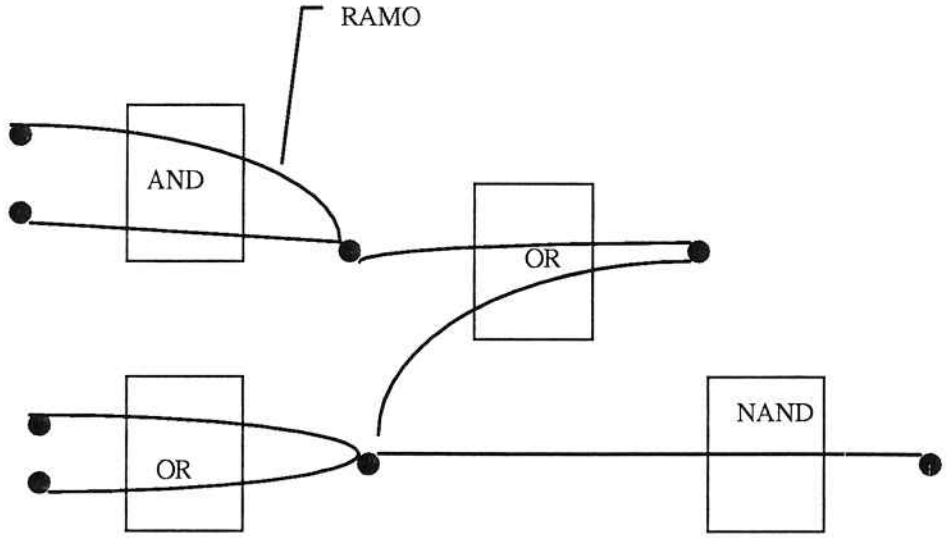

Figura 3 - Ilustração do conceito de RAMO.

A classe RAMO define a conectividade do circuito, podendo-se obter uma descrição para reconhecimento de padrão a partir dela.

3.9.2. Classes

Considere-se os objetos definidos, agrupados em classes:

3.9.2.a. Classe Nó

Identificador: nome do nó; este será o identificador (chave) da base, juntamente com a tabela de módulos, definida mais adiante.

Equação: cada nó admite uma equação (a da célula combinatória a ele associada). Podem ser utilizadas diferentes representações. Para "softcells", quando não expandidas, a informação não é necessária.

Nível: o nível está vinculado ao nó, sendo definido como:

nível de entrada primária =0;
nível do nó X= (máx (nível dos nós de entrada do "gate" cuja saída é X)+T);

onde T pode ser 1, caso se esteja considerando apenas os níveis topológicos do circuito ou o atraso intrínseco da porta associada ao nó, quando se está interessado em avaliar caminhos críticos (nesse caso define-se o objeto **ATRASO**). A informação é obtida a partir da conectividade do circuito.

3.9.2.b. Classe "Gate"

Nome: identifica a porta pelo nome ou instância.

Tipo de porta: é um número codificado que identifica a porta em questão. A codificação visa não somente à compactação da informação, mas permitir, também, acrescentar informações que se julguem convenientes no futuro. Por exemplo, a porta AND pode ser codificada por 0000, e a porta OR por 0001, etc... A codificação é feita durante a montagem da base, a partir da leitura de um arquivo de biblioteca. Quando uma porta for necessária, ela é buscada no arquivo de consulta e somente então armazenada na memória do computador. A codificação acima permite identificar se uma porta é de lógica negada ou não por consulta a um "bit".

Atraso: engloba os parâmetros de atraso intrínseco da porta e seu fator de aumento por unidade de carga. Com essas informações, modela-se o atraso (máximo ou mínimo) por:

T = Ti + K.F, onde F é a carga total que a porta alimenta.

Área: dada em unidades convenientes ("gates equivalentes", mils^2,etc..), de forma a ser computada como inteiro.

3.9.2.c. Classe Ramo

Dado um nó do circuito e o(s) ramo(s) que nele chegam, tem-se:

Origem: referencia **identificador**(es) (ver classe NÓ), isto é, o(s) nó(s) que originam os ramos que chega no nó enfocado.

Destino: referencia **identificador**(es), isto é, o(s) nó(s) para os quais se dirige(m) o(s) ramo(s) que partem do nó enfocado. Note que caso haja mais de um, temos um nó de saída múltipla ("fan out stem").

Porta: referencia o "gate", cuja saída é o nó enfocado; é a porta através da qual os ramos definidos pelos nós de origem e o nó enfocado passam.

Observabilidade[iii]: valor numérico de observabilidade da linha entre o nó enfocado e o "gate" a ele associado.

Controlabilidade[iv]: valor numérico da controlabilidade da linha entre o nó e o "gate" a ele associado.

Estes dois últimos objetos relacionam-se à testabilidade de circuitos. São medidas associadas às **linhas** e não aos ramos, como definidos. Porém, como não se pretende trabalhar com linhas e sim com ramos, tais valores foram agrupados nessa classe. Essa opção se deve ao fato de que a classe ramo suporta as informações necessárias à testabilidade. Na verdade, as definições de observabilidade e controlabilidade acima cobrem parte das linhas do circuito. Não são englobadas aquelas linhas provenientes de um nó de saída múltipla e que alimentam uma porta. Todavia, essas medidas podem ser obtidas rapidamente com as demais, permitindo suportar testabilidade sem a necessidade de criarmos portas especiais ("fan out stem gates"), que constituem objetos inexistentes em um circuito e que costumam ser necessários quando se utiliza uma base orientada por "gates".

4. IMPLEMENTAÇÃO DE ELEMENTOS BÁSICOS DE UM MAPEADOR

4.1. Definição da sintaxe das bibliotecas

4.1.1. Introdução

Conforme mencionado anteriormente, as descrições de bibliotecas assumem o conjunto de células básicas: AND, NAND, OR, NOR, NOT, XOR, XNOR, ff ("flip-flop" tipo d, podendo ter "set" e "reset"), ffsc ("flip-flop" de "scan" duplo "clock") e ffsm ("flip-flop" de "scan" com "shift mode").

Nas descrições que se seguem os símbolos destacados em negrito são palavras-chave, que fazem parte da sintaxe, e devem ter sua grafia respeitada, inclusive em termos de maiúsculas e minúsculas.

Tanto nos arquivos hierárquicos como nos expandidos, é definida a extensão de linguagem **%X**, para os blocos que não devem sofrer mapeamento algum. Assim, tudo que estiver entre dois símbolos **%X** consecutivos será ignorado para efeito de mapeamento. O sinal **%** será indicativo de comentário. Outras eventuais extensões poderão ser definidas para evitar sobrecarregar o pré-processamento do programa.

4.1.2. Arquivos de Descrição de "Netlist"

[iii] Observabilidade é uma medida associada à capacidade de se detectar o valor lógico presente em uma linha, através da imposição de valores na entrada do circuito e observação das saídas.

[iv] Controlabilidade é uma medida associada à capacidade de se impor um valor lógico a uma linha, através da imposição de valores na entrada do circuito.

Conforme mencionado anteriormente, o programa utilizará um arquivo de biblioteca , contendo uma descrição das células em termos de área, atrasos e outras informações lógico-estruturais e outro, com um "netlist" implementável para cada célula, seguindo sintaxe similar à utilizada para descrição de circuitos hierárquicos.

Este último pode ser descrito da seguinte forma (Figura 4 - Definição da sintaxe do arquivo de "netlist".):

```
<nome da célula> [nó de saída ...] [nó de entrada ...]
ATTRB
<equações lógicas>
BEGIN
CASE TIPO =
'MAPEADOR': BEGIN
                    <descrição ... em termos de primitivas>;
                    END;
'AUDITOR':  BEGIN
                    <descrição ... em termos de elementais>;
                    END;
ELSE:       BEGIN
                    <descrição ... em termos de elementais>;
                    END;
END;
END;
```

Figura 4 - Definição da sintaxe do arquivo de "netlist".

A equação lógica é dada para cada saída e os nomes das entradas e saídas seguem a nomenclatura definida para os pinos. A descrição hierárquica conterá, via estruturas do tipo "case usage", descrições convenientes para o mapeador, ferramenta auditora de regras de projeto e programas de testabilidade. Isto porque podem ser necessárias diferentes descrições conforme o caso, ou seja, a descrição de um circuito para o mapeador, por exemplo, pode não ser conveniente para os propósitos de uma ferramenta de auditoria de regras de projeto. Um exemplo de arquivo .NET é apresentado nos anexos que descrevem uma implementação de mapeador realizada.

4.1.3. Arquivos de Descrição das Células

Propõe-se, a seguir, o formato e sintaxe para esse arquivo. São caracteres de controle e restrições da linguagem:

% utilizado para comentários e interpretado como final de linha e retorno de cursor (LF+CR); caso o comentário prossiga na linha seguinte, é necessário colocar este símbolo no começo da linha;

- utilizado quando uma informação não for disponível; pode ser utilizado no meio de nome de células pois sua interpretação somente ocorre quando o símbolo estiver isolado;

os campos não devem ser quebrados quando se muda de linha, cuja tamanho é de até 80 caracteres (incluindo CR e LF);

é importante observar que só há um espaço entre os campos presentes em cada linha;

os arquivos utilizam apenas caracteres ASCII não estendido.

O arquivo de informações lógico-estruturais (.BIB) terá o seguinte formato (Figura 5 - Definição da sintaxe dos arquivos .BIB.):

```
<nome da célula> ITSEBXX NI NO NIO <extra>
<área> <atraso máx./K> <atraso mín./K>
[<YYnn/nível/carga/configuração elétrica/extra>...]
```

Figura 5 - Definição da sintaxe dos arquivos .BIB.

Nesta descrição tem-se:

I - codifica o tipo de célula. Por exemplo, **A**, para comparadores, **B**, para "buffers" e

"pads", etc.

TSEB - são indicadores booleanos (V ou F) da presença ou não de certas características como "tri-state", recurso de "scan" para DFT, habilitação e extensibilidade.

XX - indicadores de características especiais da célula - são dois "flags" utilizados conforme a célula a que se refiram, caso haja necessidade, para descrição de um subtipo ou alguma característica particular; se não utilizados, valem X.

NI é o número de "pinos" de entrada da célula;

NO é o número de "pinos" de saída da célula;

NIO é o número de "pinos" de entrada/saída da célula.

"extra" - pode ser utilizado para a descrição de algumas características especiais que porventura se faça necessária. Todo o espaço até o final da linha é disponível para esse campo:

Área, atraso máximo e mínimo e respectivos fatores de degradação (K):

devem ser definidos em unidades tais que sua representação seja como números inteiros entre 0 e 32765.

A descrição de pinos é feita obedecendo-se a ordem em que são definidos na respectiva biblioteca, ou seja, na ordem em que são evocados no "netlist".

O campo YY assume valores codificados para descrever características dos pinos. Por exemplo, **CI**, para um pino de "carry-in" em um somador, **CK**, para um pino de "clock" de dados, OO, para um pino de saída de dados, etc.

nn - indica a ordenação dos "bits" naquela categoria; assim, OO00 indica a saída menos significativa de uma célula. Isto ocorre para que se possa identificar univocamente as entradas e saídas das células. Caso contrário, se em uma biblioteca os pinos de entrada de um multiplexador, por exemplo, fossem do mais para o menos significativo e, em uma outra biblioteca, o contrário, não se conseguiria identificar a ordem correta e, portanto o mapeamento resultaria errado.

nível - indica o tipo de sinal que ativa o pino.

carga - representa a capacitância de entrada do pino (multiplicada por fator conveniente de modo a ser um número inteiro)

configuração elétrica é codificado de modo a identificar características elétricas. Por exemplo, **u**, para um pino "pull-up", **t**, para um pino "tri-state", **<número inteiro>**, para identificar uma corrente em uA, etc...

Extra pode ser utilizado para descrições adicionais, como "clock" sensível a borda.

Ao final do arquivo, temos as linhas:

<nº de entradas "e"/nº de entradas "ou" da maior AOI da biblioteca>

<número de células no arquivo>

4.1.4. Arquivos de Correspondência

Conforme mencionado anteriormente, é necessário providenciar mecanismos que permitam ao usuário identificar, no circuito gerado, as alterações feitas. Assim, nós criados serão identificados por:

MMN<nº>,

onde nº corresponde a uma numeração seqüencial dada pelo programa à medida em que forem criados os nós.

Os "gates" criados por expansão serão da forma:

[<instância>...]/MME<nº>,

onde nº corresponde a uma numeração seqüencial dada pelo programa à medida em que forem criados os "gates"

E os por fusão:

[<instância>...]/MMF<nº>,

onde a instância poderá assumir um novo nome, quando se fundem portas de módulos distintos. As informações pertinentes a esse processo, constantes do arquivo de saída, devem permitir a identificação das portas que foram fundidas. Outras eventuais extensões poderão ser definidas para evitar sobrecarregar o pré-processamento do programa.

4.1.5. Implementação de um Gerador de Bibliotecas

A entrada dos dados de biblioteca exigem uma rotina específica que auxilie essa tarefa. Pretende-se que o usuário forneça informações acerca de uma célula uma única vez, isto é, terminado o processamento daquela primitiva, não mais seja necessário retornar a ela. Da mesma forma, deseja-se que as informações sejam introduzidas em nível crescente de dificuldade. Desta forma, propõe-se que a primeira tela de informações do gerenciador do banco de dados, colete os seguintes dados:

a) nome da célula;
b) número de "pinos" de entrada, saída e entrada e saída;
c) área;
d) tempo máximo de propagação e respectivo fator de degradação;
e) tempo mínimo de propagação e respectivo fator de degradação;
f) indicador se célula possui "tri-state";
g) indicador se célula possui habilitação;
h) indicador se célula possui recurso de "scan".

A segunda tela pedirá o tipo da célula (somador, registrador, etc...) e informações adicionais sobre ela, conforme o seu tipo. No caso mais geral, serão solicitados:

a) indicador se célula é "bit-slice";
b) os campos X que dão características específicas da célula;
c) campo extra; [1]

A terceira tela irá solicitar a entrada dos pinos, após a qual será atualizado o arquivo de biblioteca. A entrada dos pinos deverá ser a mais amigável possível, dado o grande número de codificações existentes. Mecanismos de verificação de correção sintática deverão ser incluídos. Essas telas poderão ter associadas a si telas de socorro, a serem ativadas quando o usuário pressionar uma tecla específica (p. ex. F1).

A seguir o gerenciador do banco de dados pode abrir um "shell" para o sistema operacional e ativar um programa para a quarta tela. Esta irá orientar o usuário a entrar as descrições do circuito. É possível, também, importar uma descrição de algum arquivo preexistente. A extração da equação lógica do circuito (para o caso combinatório) pode ser feita concomitantemente a essa etapa ou posteriormente, dependendo do algoritmo a ser utilizado.

À geração dos arquivos de biblioteca está associada a geração dos respectivos arquivos de índice que permitirão o acesso imediato às informações.

4.2. Definição da estrutura de dados

4.2.1. Base de Dados

Baseando-se nas definições anteriores, pode-se estabelecer a seguinte base de dados (Figura 6 - Estrutura de Dados sugerida):

Nome da Estrutura	**Campos**	**Observações**
TABELA DE MÓDULOS	nome	nome do módulo
	Ponteiro p/ NÓ	endereça início das estruturas NÓ daquele módulo
	Ponteiro p/ RAMO	endereça início das estruturas RAMO daquele

		módulo
TABELA DE PORTAS	Código	número codificado que identifica a primitiva ("bitwise")
	Atraso Máximo	atraso intrínseco e fator carga
	Atraso Mínimo	atraso intrínseco e fator carga
	Área	ver definições.
NÓ	Identificador	nome do nó (código <> nº)
	Nível	fornece o nível do nó e, por tanto, da porta ou módulo a ele associado
	Atraso	fornece o nível do nó, em termos do atraso máximo de uma entrada ao nó
	Equação	caso a célula de origem seja uma primitiva, é fornecida a equação, caso seja uma "soft- cell" ou módulo, nada é dado.
	Flag ("bitwise")	Indica se o nó já sofreu algum tipo de processamento ou não.
	Ponteiro "gate"	aponta para a TABELA DE PORTAS ou TABELA DE MÓDULOS
	Ponteiro de RAMO	aponta para estrutura RAMO associada ao nó; é uma repetição para agilizar o processo
RAMO	Origem	vetor com a lista dos nós que originam o nó enfocado (pont.)
	Destino	lista ligada dos nós dos quais o nó enfocado é origem (pont.)
	Porta	nome ou instância da porta associada ao nó que acessou a estrutura (código interno).
	Controlabilidade	ver definição acima
	Observabilidade	ver definição acima

Figura 6 - Estrutura de Dados sugerida

Uma vez montada, a base de dados apresenta um aspecto como o da Figura 7 - Representação da Base de Dados

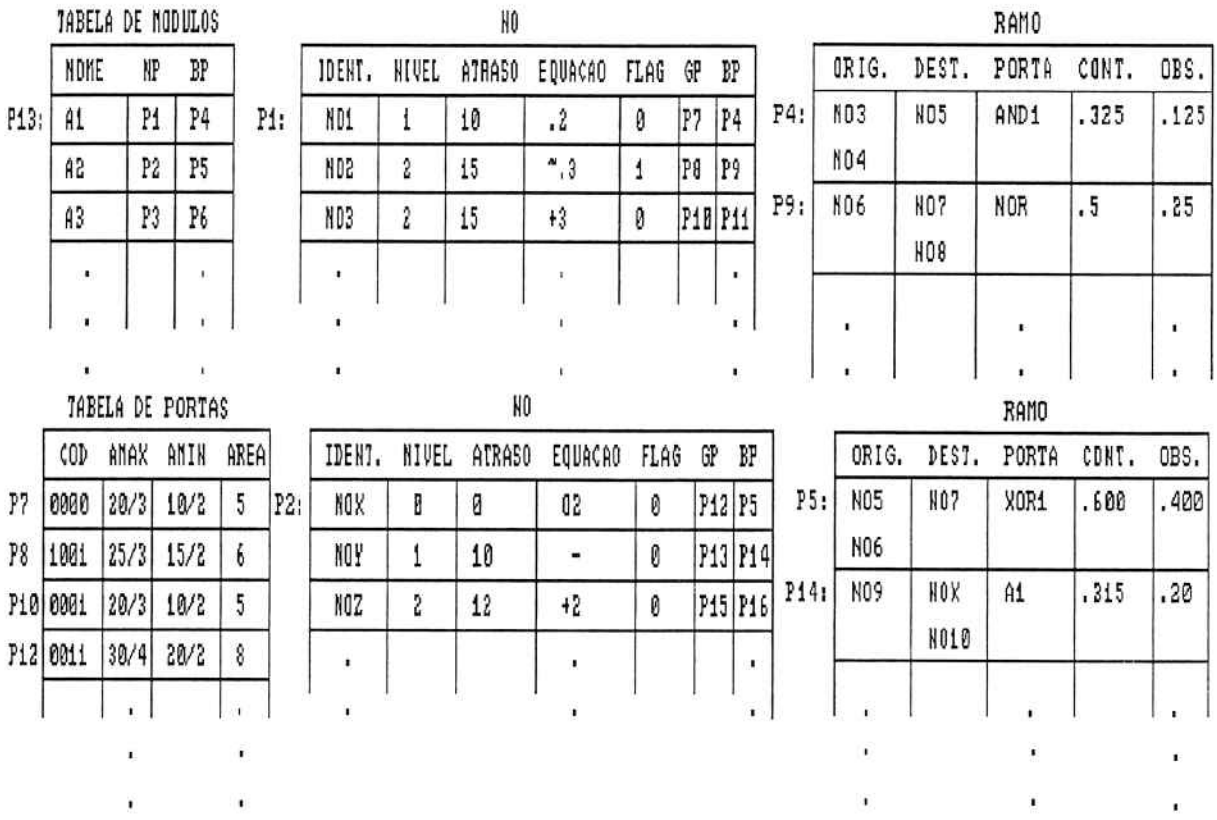

Figura 7 - Representação da Base de Dados

Alguns aspectos merecem ser destacados. A base proposta é hierárquica considerando-se que cada módulo não tem sua respectiva estrutura repetida a cada instância. A hierarquia completa de uma porta pode ser obtida pela combinação dos nomes dos módulos com o nome dos "gates", extraídos da TABELA DE MÓDULOS e da estrutura RAMO, para cada nó enfocado.

Da mesma forma, os níveis dos "gates" podem ser obtidos a partir dos níveis dentro de cada hierarquia.

A visão hierárquica é uma necessidade para a interface com o usuário e seu armazenamento pode ser muito dispendioso se não for feito também de forma hierárquica (é o que irá ocorrer quando a entrada do circuito a ser processado for um circuito expandido). Todavia, a base hierárquica suporta a expandida sem custo adicional e a recíproca não vale pois, para tanto, precisa-se consumir mais memória para guardar a informação de hierarquia.

A fragmentação de estruturas (TABELA DE PORTAS, NÓ e RAMO) advém do caráter distinto desses objetos e do fato do mapeador necessitar operar com todas essas informações ao longo de seu processamento. Procurou-se, contudo, minimizar a utilização de estruturas computacionais do tipo ponteiro, usualmente requisitadas quando se fragmenta muito uma base. Desta forma, a partir da TABELA DE MÓDULOS, tem-se dois ponteiros que permitem acessar a estrutura NÓ e RAMO diretamente. Da estrutura NÓ, também é possível acessar a estrutura RAMO e, nesse caso, há uma redundância que se crê justificada na medida em que pode ser interessante, dependendo do algoritmo, uma estrutura endereçada por linhas (ramos) ou uma estrutura endereçada por "gates" (nós).

Dentro da estrutura RAMO, a lista do campo origem é colocada em uma estrutura computacional do tipo vetor, uma vez que se pode determinar o número de elementos desse vetor com relativa facilidade, durante o processamento. Por outro lado, a lista do campo destino, que representa o "fan out" do nó, é uma estrutura computacional do tipo lista ligada, pois a determinação do número de saídas depende da conectividade do circuito, o que é mais difícil de ser obtido. As tabelas e as

estruturas RAMO estão em listas ligadas, uma vez que não se sabe de antemão o seu número de elementos. Da mesma forma para a estrutura NÓ; nesse caso, contudo, a lista estará em ordem alfabética (pois serão feitas buscas) e o processo de montagem em ordem é mais facilmente implementável via lista ligada.

A definição de ramo surge como alternativa para implementação das linhas, sem o custo adicional de se criar portas nos nós de ramificação ("fan out stem"). Na verdade, é apenas uma forma diferente de se encarar o grafo, que usualmente interpreta os "gates" como vértices e as conexões como linhas.

O campo equação pode conter qualquer tipo de representação da função lógica da célula.

4.2.2. Montagem da Base de Dados

A montagem da base a partir de um arquivo hierárquico segue os seguintes passos:

1. Atualizar TABELA DE MÓDULOS, criando as estruturas NÓ e RAMO respectivas ao módulo;
2. Preencher a estrutura NÓ e RAMO, da seguinte forma:
   2.1 Para cada nó, na ordem em que aparecem, a partir da definição do módulo, inseri-lo, em ordem alfabética, na lista de nós;
   2.2 Ir preenchendo os campos equação e os ponteiros da estrutura NÓ, porta e origem, da estrutura RAMO, à medida em que se lê a linha.
   Note que se deve saber o número de saídas de cada célula. Ao ler um nó de entrada, sabe-se que ele é origem do(s) nó(s) de saída e que estes são seu destino. Ao ler a linha de definição do módulo, marcam-se os nós como entradas primárias (PIs) e saídas primárias (POs), pois não se sabe quantas são as saídas e quantas as entradas de um módulo. Todavia, ao final do preenchimento das estruturas pode-se saber quais são as entradas e quais as saídas.
3. Ao terminar o módulo, proceder seu nivelamento da seguinte forma:
   3.1 Via "backtrack"[v] implementado por uma estrutura computacional de pilha, para cada PI atribuir nível 0.
   Tomar um caminho a partir da PI e ir determinando o nível dos nós seguintes, de acordo com a expressão de cálculo de nível. Nesse processo, quando se chega a um nó de nível já determinado, se este for maior que o que se pretende atribuir, já se abandona aquele caminho; caso contrário atualiza-se o nível. Quando se chega a um nó cujo destino é uma saída ou um elemento sequencial também se abandona o caminho, pois, no último caso, poder-se-ia entrar em "loop".
   Neste raciocínio, quando se chega ao mesmo nó do qual se partiu, detecta-se uma realimentação em lógica combinatória, podendo-se abortar o programa dependendo do nível de mapeamento em que se esteja.
4. Repetir os passos 1, 2 e 3 para todos os módulos.

5. CONCLUSÕES

5.1. Contribuições

O presente trabalho apresentou o problema do mapeamento tecnológico dentro do contexto de projetos de microeletrônica e sua relevância. Diversas deficiências foram identificadas nos mapeadores existentes, tais como:

a) ausência de uma base de dados adequada para suportar diversos pontos de vista de projeto: testabilidade, hierarquia, foco em portas, nós, linhas, etc.
b) ausência de mecanismos adequados para entrada de dados de biblioteca;
c) ausência de mecanismos de intervenção por parte do projetista;
d) ausência de mecanismos que possibilitem o uso de diferentes algoritmos.

---

[v] "Backtrack" é um processo algoritmo que implementa uma busca por tentativa e erro.

A partir da análise do universo existente e das necessidades reais de projeto, o presente trabalho contribuiu com o delineamento de requisitos necessários a um mapeador tecnológico que cubra um escopo maior de funcionalidades. Adicionalmente, foi apresenta uma alternativa de arquitetura para atender os requisitos identificados anteriormente. Finalmente, foi proposta uma implementação cobrindo um mapeamento em nível 0, isto é, a tradução de um circuito sem otimização.

6. BIBLIOGRAFIA:

[BER91] BERGAMASCHI, R. **"SKOL: A System for Logic Synthesis and Technology Mapping"** - IEEE Transactions on Computer-Aided Design, Vol. 10 N°11 - pp. 1342-1355 - November 1991

[BRA84] BRAYTON, R., et. al **"ESPRESSO-IIc: Logic Minimization Algorithms for VLSI Synthesis"** - Kluwer Academic Publishers - Holanda - 1984

[BRA87] BRAYTON, R., RUDELL, R. **"MIS: A multiple-level logic optimization system"** - IEEE Transactions on Computer-Aided Design, Vol. CAD 6 - pp. 1062-1081 - Novembro 1987

[BRG88] BRGLEZ,F. et. al. (Out. 1988). **"McMap: A Fast Technology Mapping Procedure for Multi-Level Logic Synthesis"** - 1988 IEEE International Conference on Computer Design: VLSI in Computers & Processors - ICCD '88

[CRA91] CRASTER, M. et. al. (1991). **"A Technology Mapping Method Based on Perfect And Semi-Perfect Matchings"** - 28th ACM/IEEE Design Automation Conference - Proceedings 1991

[FRA91] FRANCIS, R., ROSE, J., Vranesic, Z. **"Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs"** - 28th ACM/IEEE Design Automation Conference - Proceedings 1991 - pp.221-233

[GAJ88] GAJSKI, D. **"Silicon Compilation"** - Addison-Wesley - 1988

[GEU85] De GEUS, A., COHEN, W. **"A Rule-Based System for Optimizing Combinational Logic"** - Design and Test of Computers - Volume 2, n° 4 - pp.22-32 - Agosto 1985

[ISC89] ISCAS - International Sysmposium on Circuits and Systems - http://www.cbl.ncsu.edu/www/CBL_Docs/iscas89.html

[ISH88] ISHIKAWA, J. et al. **"A Rule Based Logic Reorganization System LORES/EX"** - 1988 IEEE International Conference on Computer Design: VLSI in Computers & Processors - ICCD '88 - 1988

[KEU87] KEUTZER, K. **"DAGON: Technology binding and local optimization by DAG matching"** 24th Design Automation Conference Proceedings - pp.341-347 - Junho 1987

[LEG88] LEGA, M.C. **"Mapping Properties of Multi-Level Logic Synthesis Operations"** - 1988 IEEE International Conference on Computer Design: VLSI in Computers & Processors - ICCD '88 - 1099

[LIS87] LISANKE, R., KEDEM G. & BRGLEZ, F. **"DECAF - Decomposition and Technology Mapping for Multi-level Logic Synthesis"** - Technical Report - Microelectronics Center of North Carolina, Research Triangle Park, NC - 1987

[MAI93] MAILHOT, F., De MICHELI, G. **"Algorithms for Technology Mapping Based on Binary Decision Diagrams and on Boolean Operations"** - IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems - Volume 12, n° 5 - Maio 1993

[MAS91] MASSOUD, P., BHAT, N. **"Layout Driven Technology Mapping"** - 28th ACM/IEEE Design Automation Conference - Proceedings 1991 - pp 99-105

[YOS91] YOSHIKAWA, K., et. al. **"Timing Optimization on Mapped Circuits"** - 28th ACM/IEEE Design Automation Conference - Proceedings 1991 - pp. 112-117

BOLETINS TÉCNICOS - TEXTOS PUBLICADOS

BT/PEE/9301 - Oscilador a HEMT - 10 GHz - FÁTIMA S. CORRERA, EDMAR CAMARGO
BT/PEE/9302 - Representação Senoidal da Voz através dos Polos do Filtro Preditor - MARCELO B. JOAQUIM, NORMONDS ALENS
BT/PEE/9303 - Blindagens por Grades Condutoras: Cálculo do Campo Próximo - LUIZ CEZAR TRINTINALIA, ANTONIO ROBERTO PANICALI
BT/PEE/9304 - Sistema de Otimização e Controle de Produção em Minas de Pequeno e Médio Porte - TSEN CHUNG KANG, VITOR MARQUES PINTO LEITE
BT/PEE/9401 - Determinação das Frases de Aplicação Forense para o projeto NESPER e Tese de Mestrado IME/94, com Base em Estudos Fonéticos - MARCONI DOS REIS BEZERRA, EUVALDO F. CABRAL JUNIOR
BT/PEE/9402 - Implementação e Teste de uma Rede Neural Artificial do Tipo KSON (Kohonen Self-Organizing Network) com Entradas Bidimensionais - MARCELO YASSUNORI MATUDA, EUVALDO F. CABRAL JR.
BT/PEE/9403 - Transformada de Walsh e Haar Aplicadas no Processamento de Voz - ALEXANDRE AUGUSTO OTTATI NOGUEIRA, THIAGO ANTONIO GRANDI DE TOLOSA, EUVALDO F. CABRAL JÚNIOR
BT/PEE/9404 - Aplicação de Redes Neurais ao Problema de Reconhecimento de Padrões por um Sonar Ativo - ALEXANDRE RIBEIRO MORRONE, CRISTINA COELHO DE ABREU, EDUARDO KOITI KIUKAWA, EUVALDO F. CABRAL JR.
BT/PEE/9405 - Tudo que se Precisa Saber sobre a Prática da FFT - Transformada Rápida de Fourier (Inclui Software) - ROGÉRIO CASAGRANDE, EUVALDO F. CABRAL JR.
BT/PEE/9406 - A Survey on Speech Enhancement Techniques of Interest to Speaker Recognition - CELSO S. KURASHIMA, EUVALDO F. CABRAL JR.
BT/PEE/9407 - Identificação de Pulsos Decádicos em Linhas Telefônicas - ANTONIO P. TIMOSZCZUK, MÁRCIO A. MATHIAS, EUVALDO F. CABRAL JR.
BT/PEE/9408 - Implementação e Teste de Filtros do Tipo Adaptativo e "Notch" para a Remoção de Interferência de 60 Hz em Sinais de Eletrocardiograma - FLÁVIO ANTÔNIO MENEGOLA, JOSÉ AUGUSTO DE MATTOS, JOSÉ GOMES G. FILHO, SIDNEY SILVA VIANA, EUVALDO F. CABRAL JR.
BT/PEE/9409 - Compressão de Sinais de Voz utilizando Transformadas de Karhunen-Loève, Fourier e Hadamard - IVAN LUIS VIEIRA, LUIZ FERNANDO STEIN WETZEL, EUVALDO F. CABRAL JR.
BT/PEE/9410 - "Ray Tracing" Paralelo - EDUARDO TOLEDO SANTOS, JOÃO ANTONIO ZUFFO
BT/PEE/9411 - Implementação de uma Ferramenta Posicionador para "Gate-Arrays" Tipo Mar de Portas - JORGE W. PERLAZA PRADO, WILHELMUS A. M. VAN NOIJE
BT/PEE/9412 - Tudo que se Precisa Saber Sobre a Teoria da FFT - Transformada Rápida de Fourier - FÁBIO LUÍS ROMÃO, REINALDO SILVEIRA, ROGÉRIO CASAGRANDE, EUVALDO CABRAL JR.
BT/PEE/9413 - Análise do Ruído Sonoro em uma Sala de Aquisição de Amostras de Som com Microcomputador - FÁBIO LUÍS ROMÃO, REINALDO SILVEIRA, EUVALDO CABRAL JR.
BT/PEE/9414 - Cor: Aspectos Relevantes para Visualização de Dados - SÍLVIA DELGADO OLABARRIAGA
BT/PEE/9415 - Projeto de Filtros Digitais IIR com Fase Aproximadamente Linear Utilizando Redução de Ordem - IVAN F. J. RODRIGUES, MAX GERKEN
BT/PEE/9416 - GERAFILTRO: Sistema para Projeto Automático de Filtros Digitais "IIR" (da especificação em alto nível ao leiaute do "ASIC") - RICARDO PIRES, JOSÉ VIEIRA DO VALE NETO
BT/PEE/9417 - Redes Neurais Artificiais Aplicadas à Identificação de Pulsos Decádicos em Linhas Telefônicas - ANTONIO P. TIMOSZCZUK, EUVALDO F. CABRAL JR.
BT/PEE/9501 - Estudo Comparativo de Métodos de Cálculo da Frequência Fundamental - MARCOS COSTA HUNOLD, EUVALDO F. CABRAL JR.
BT/PEE/9502 - Combinando Técnicas de Redes Neurais Artificiais e Informações de Excitação no Reconhecimento Automático do Locutor - ANDRÉ BORDIN MAGNI, EUVALDO F. CABRAL JR.
BT/PEE/9503 - Utilização de Redes Neurais Artificiais para Detecção e Identificação de Falhas em Circuitos - MÁRCIO YUKIO TERUYA, ROBERTO AMILTON BERNARDES SÓRIA, EUVALDO CABRAL JR.
BT/PEE/9504 - Uso de Redes Neurais Artiificiais no Reconhecimento de Locutores no Domínio Temporal - BENEDITO JOSÉ BARRETO FONSECA JÚNIOR, EUVALDO CABRAL JÚNIOR
BT/PEE/9505 - Projeto de Filtros Passivos e Ativos em Técnicas de Circuitos Integrados de Microondas - DAVID VIVEIROS JÚNIOR, DENISE CONSONNI
BT/PEE/9506 - Uma Análise de *Clustering* para as Frases de Projeto NESPER - RONALDO OLIVEIRA MESSINA, EUVALDO F. CABRAL JR.
BT/PEE/9507 - Controle com Estrutura Variável e Modos Deslizantes - Um Estudo para Aplicação em Controle Carga-frequência da Geração - JOSE PAULO F. GARCIA, JOCELYN FREITAS BENNATON
BT/PEE/9508 - Recuperação das Margens de Ganho e de Fase para Sistemas de Fase Não Mínima por Realimentação da Saída - MARCO H. TERRA, VITOR M. P. LEITE
BT/PEE/9509 - Sistema de Inspeção Óptica de Dispositivos Bi-Dimensionais - CASIMIRO DE ALMEIDA BARRETO, PEDRO LUÍS PRÓSPERO SANCHEZ
T/PEE/9510 - Sistema de Partículas Uma Poderosa Técnica de Animação em Computação Gráfica - RENATO CURTO RODRIGUES, JOÃO ANTÔNIO ZUFFO
BT/PEE/9511- Efeito de Ruídos em Sinais de Voz Visualizados em Trajetórias Neurais de Kohonen - CELSO S. KURASHIMA, EUVALDO F. CABRAL JR.
BT/PEE/9601 - "Um Reconhecedor de Sinais Sonoros Utilizando LVQ" - ALEXANDRE TORNICE, EUVALDO CABRAL JR.
BT/PEE/9602 - "Coleção *Artificial Neural Networks*: Uma Visão Geral dos Sistemas Neurais Artificais de Stephen Grossberg" - CHIU HSIUNG HUANG
BT/PEE/9603 - "Reactively-Sputtered TiN Formation Using a RF Magnetron System"- SÉRGIO PAULO AMARAL OSÓRIO, LUIZ SÉRGIO ZASNICOFF
BT/PEE/9604 - Aspectos em Tradução de Linguagens Naturais Através de Redes Neurais Artificiais - CARLOS EDUARDO DANTAS DE MENEZES, EUVALDO F. CABRAL JR.
BT/PEE/9605 - Implementação de Blocos Passa-Tudo Utilizando Realimentação de Erro - SÉRGIO JOSÉ CARNEIRO LEÃO, MAX GERKEN
BT/PEE/9606 - Coleção SANN group Redes Neurais Artificiais: A Rede Neural de Sakoe - ANDRÉ BORDIN MAGNI, EUVALDO F. CABRAL JR.

BT/PEE/9607 - Coleção SANN group Redes Neurais Artificiais: A Rede Neural de Steinbuch - ROBERTO AMILTON BERNARDES SÓRIA, EUVALDO F. CABRAL JR.
BT/PEE/9608 - Desenvolvimento de uma Estrutura de Duplo Nível de Metal para a Confecção de Interconexões em Circuitos Integrados - JOSÉ AUGUSTO DE ALENCAR PEREIRA, LUIZ CARLOS MOLINA TORRES
BT/PEE/9609 - Determinação de Parâmetros de Processo para Fotomáscara "Balzers" Utilizando Gerador de Padrões - JORGE SEKI, MEGUMI SAITO
BT/PEE/9610 - Um Ambiente para Desenvolvimento de Sistemas Distribuídos - PEDRO F. ROSA, JOÃO A. ZUFFO
BT/PEE/9611 - Interpretações Teóricas do Funcionamento Cerebelar: Uma Revisão - MARCUS FRAGA VIEIRA, ANDRÉ FÁBIO KOHN
BT/PEE/9612 - Marcapasso Cardíaco Temporário Microcontrolado de Demanda e Baixo Consumo - FLAVIO ANTONIO MENEGOLA, JOSÉ CARLOS TEIXEIRA DE BARROS MORAES
BT/PEE/9613 - Um Sistema de Planejamento de Ação Baseado em Casos para uma Célula Flexível de Manufatura - RICARDO LUÍS DE FREITAS, MÁRCIO RILLO
BT/PEE/9614 - Aplicações do Boundary-Scan para o Teste de Módulos Multichip - ROBERTO C. COSSI JR., JOSÉ ROBERTO DE A. AMAZONAS
BT/PEE/9615 - A 2.488 Gb/s GaAs 1:4/1:16 Demultiplexer IC with Skip Circuit for Sonet STS-12/48 Systems - TAUFIK ABRÃO, FATIMA S. CORRERA
BT/PEE/9616 - Uma Contribuição para a Construção de Algoritmos em Projetos de Redes - ALLAN DE SOUZA, JOSÉ ROBERTO CASTILHO PIQUEIRA
BT/PEE/9617 - Análise Crítica dos Métodos de Medição do Intervalo QT do Eletrocardiograma - SÍDNEY DA SILVA VIANA, JOSÉ CARLOS TEIXEIRA DE BARROS MORAES
BT/PEE/9618 - Deposição e Caracterização de Filmes de SiO_2 Crescidos pela Técnica de PECVD a Baixa Temperatura - MARCO ALAYO CHÁVEZ, INÉS PEREYRA
BT/PEE/9619 - PARSTOOL: Uma Ferramenta de Auxílio à Simulação de Sistemas Paralelos - LI KUAN CHING, LIRIA MATSUMOTO SATO
BT/PEE/9620 - Análise de um Método de Otimização por Malha no Treinamento de Robôs - OLÍMPIO MURILO CAPELI, JOSÉ CARLOS T. B. MORAES, SADAO ISOTANI
BT/PEE/9701 - Identification of Unstable Mechanical Systems - ROBERTO MOURA SALES, ANSELMO BITTAR, MICHAEL PORSCH, LAÉRCIO LUCCHESI
BT/PEE/9702 - Analysis of the Subthreshold Slope Transition Region in SOI nMOSFET - VICTOR SONNENBERG, JOÃO ANTONIO MARTINO
BT/PEE/9703 - Introduction of the SOI MOSFET Dimensions in the High-Temperature Leakage Drain Current Model - MARCELO BELLODI, JOÃO ANTONIO MARTINO, DENIS FLANDRE
BT/PEE/9704 - Controle de Largura de Banda Dinâmica para Transmissões *Multicast* para Redes de Alta Velocidade - SANG SOON LEE, SERGIO TAKEO KOFUJI
BT/PEE/9705 - Uma Modificação Proposta para o Controle Preditivo Generalizado com Filtro de Kalman - JAIME QUINTERO R., OSWALDO L. V. COSTA
BT/PEE/9706 - Aplicações de Redes Neurais em Previsões Financeiras - OLÍMPIO MURILO CAPELI, EUVALDO F. CABRAL JR.
BT/PEE/9707 - Sistema Microcontrolado, Multicanal e Portátil para Estimulação Neuromuscular Funcional - ROGÉRIO QUIARIM ZARZA, JOSÉ CARLOS TEIXEIRA DE BARROS MORAES