Capítulo 1

Introdução

> *Galilée - . . . ce que je sais, il faut que je le dise à d'autres. Comme un amoureux, comme un ivrogne, comme un trâitre. C'est um vice absolu, et qui mène au malheur. Combien de temps vais-je pouvoir le crier au braises de mon feu – c'est la question.*
>
> *Tradução:*
>
> *Galileu - . . . o que sei, é necessário que eu diga a outros. Como um amante, como um bêbado, como um traidor. É um vício absoluto, e que leva à infelicidade. Quanto tempo eu poderei gritá-lo nas brasas de minha fogueira – esta é a questão.*
>
> *Betholt Brecht,* La Vie de Galilée.

Sistemas digitais binários eletrônicos de muito alta escala de integração (do inglês, *Very Large Scale Integration*, ou VLSI) são hoje componentes indispensáveis a grande parte das atividades desempenhadas pelo ser humano, em particular aquelas de natureza econômica. Estes sistemas agilizam e, em muitos casos, viabilizam estas atividades. Os serviços bancários hoje disponíveis são impensáveis sem o concurso de produtos contendo sistemas digitais, o mesmo podendo ser dito das facilidades de transporte aéreo e dos recursos de telecomunicação, entre tantos exemplos. A presente obra endereça o estudo de métodos empregados para viabilizar o projeto moderno de sistemas digitais seqüenciais VLSI, com ênfase na disciplina de projeto lógico.

1.1 Definição de Sistemas Digitais

Inicia-se este Capítulo introduzindo uma definição estrutural precisa, embora simplificada, de sistema digital.

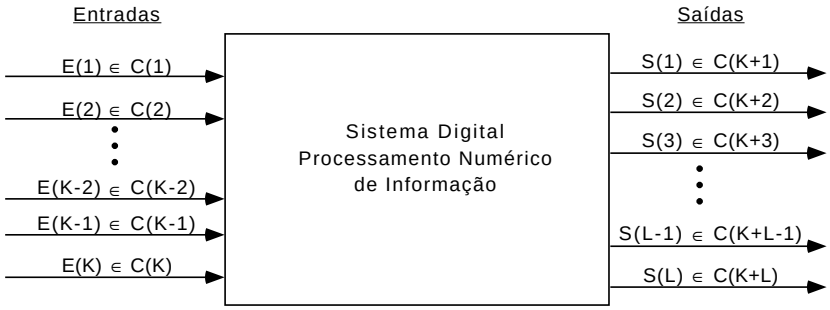


Figura 1.1: Um sistema digital visto como uma estrutura com entradas e saídas, e capaz de processar as primeiras, gerando as últimas. Cada entrada $E(i)$ ou saída $S(i)$ assume a cada instante de tempo um valor do conjunto $C(i)$ ou $C(K+i)$, respectivamente. Os conjuntos C(i) são conjuntos numéricos tais como $\{0,1\}$,$\{3,6,9\}$ ou $\{1,2,3,4,5,6,7,8,9,0\}$.

Definição 1.1 (Sistema digital e sistema digital binário) *Um **sistema digital** é um aparato concreto ou abstrato, dotado de um conjunto finito de entradas e um conjunto finito de saídas, e capaz de processar informação sob forma numérica. Para tanto, cada entrada e cada saída pode assumir ao longo do tempo valores de algum conjunto finito de números, denominado **domínio** daquela entrada ou saída. Se todas as entradas e saídas assumirem valores somente do domínio $\mathcal{B} = \{0,1\}$, o sistema é denominado **sistema digital binário**.*

Os termos sublinhados da definição acima denotam conceitos que necessitariam receber uma interpretação precisa para completá-la. Contudo, a quantidade de informação veiculada basta para a presente discussão.

Note-se que a definição é bastante geral. A restrição de entradas e saídas assumirem apenas valores numéricos não é limitante, pois quaisquer conjuntos finitos de entradas e saídas não-numéricas podem ser associados, via uma codificação adequada, a conjuntos numéricos que os representem, e um sistema digital pode então ser elaborado para processar a informação codificada. Sistemas de conversão não-digital para digital e digital para não-digital são empregados para prover a interface com o ambiente, sempre que necessário. Outra observação importante é que qualquer sistema digital pode ser implementado como um sistema digital binário, usando o mesmo raciocínio de codificações adequadas.

A restrição de empregar domínios finitos também pode quase sempre ser superada, desde que seja possível controlar a degradação do sistema que pressupõe o uso de domínios não-finitos ao ser implementado mediante o uso de

aproximações destes por domínios finitos. A teoria da informação desenvolvida por Shannon e o trabalho de Nyquist sobre a amostragem discretizada de sinais contínuos são excelentes exemplos da possibilidade efetiva de modelar sistemas físicos mediante sistemas digitais [165, 145].

A Figura 1.1 ilustra a definição, fornecendo a estrutura genérica de um sistema digital.

Tome-se como exemplo um leitor de discos ópticos de áudio, também chamados CDs (do inglês, *compact discs*). Som é um fenômeno físico que pode ser caracterizado por um conjunto de funções matemáticas *contínuas*, onde o domínio das *entradas e saídas* são conjuntos infinitos, tais como um intervalo de números reais. No caso de discos ópticos, a informação está armazenada sob forma digital, utilizando códigos numéricos em quantidade e número de ocorrências suficientes para garantir uma taxa de erro nas grandezas físicas associadas (freqüência, timbre, amplitude, etc) que o ouvido humano é incapaz de detectar a diferença entre o som original, contínuo e aquele, também contínuo, mas gerado a partir de informação numérica finita. O leitor consiste de um sistema óptico que recupera a informação armazenada sobre o disco, transformando esta informação em códigos numéricos binários que representam som via impulsos elétricos, sob uma codificação padronizada. Estes códigos binários são as entradas de um sistema digital que faz o processamento da informação, corrigindo erros, filtrando o som, etc. O leitor então fornece saídas digitais que representam o som "ideal" a ser reproduzido. Estas saídas passam a seguir por um sistema de conversão de digital para analógico, que gera ondas elétricas não-numéricas, a serem transformadas em som pelo conjunto amplificador/alto-falantes.

Trata-se neste trabalho de uma forma restrita de sistemas digitais. Assume-se sistemas digitais binários implementados sob a forma de circuitos eletrônicos adequadamente projetados, ou seja, capazes de representar e manipular informação sob forma numérica. A implementação em "hardware" de sistemas digitais baseia-se, de fato, na existência de dispositivos eletrônicos que podem estar em um de dois estados. Um transistor configurado para trabalhar nas regiões *saturação* e *corte* [190] provê um dispositivo elementar típico empregado em tais implementações. O bloco básico construtivo dos sistemas digitais em questão, o transistor, possui três terminais, e funciona como uma *chave*. No estado de saturação, dois dos terminais estão em contato e o elemento funciona como uma chave fechada. No estado de corte, os mesmos dois terminais estão isolados e o elemento funciona como uma chave aberta. O terceiro terminal é uma entrada que controla de maneira binária se o dispositivo está saturado ou cortado. Ou seja, o terceiro terminal funciona como o interruptor da chave. Doravante, usa-se o termo sistema digital no sentido da forma restrita estabelecida neste parágrafo.

1.2 Plano do Capítulo

Produzir sistemas digitais VLSI é uma tarefa extremamente complexa, sobretudo na fase de projeto. Nesta fase, grandes doses de criatividade humana e imensa quantidade de recursos materiais devem ser investidas para a obtenção de sistemas VLSI eficientes e baratos. O processo de projeto é obrigatoriamente subdividido em níveis de detalhamento, como única forma de gerenciar sua complexidade. Surgem assim as *disciplinas de projeto*, tais como o projeto sistêmico, o projeto lógico e o projeto físico. Em particular, o projeto lógico de sistemas digitais VLSI lida com duas grandes classes de sistemas, *combinacionais* e *seqüenciais* cada qual com um corpo distinto de métodos para sua elaboração.

O presente Capítulo possui a estrutura descrita a seguir. Na Seção 1.3, discute-se o projeto e a fabricação de sistemas digitais VLSI do estado da arte da tecnologia atual, instituindo o papel dos principais ramos do conhecimento científico envolvidos. A definição de, e a distinção entre sistemas digitais combinacionais e seqüenciais é o objeto de interesse da Seção 1.4. Em seguida, estudam-se classificações de sistemas digitais, com base na apresentação de critérios para estas classificações, na Seção 1.5. A Seção 1.6 consiste em explorar diferentes modelos para representar o processo de projeto de sistemas digitais. Sistemas de projeto auxiliado por computador para sistemas digitais são o tema da Seção 1.7. A seguir, a Seção 1.8 detalha o escopo da presente obra. As três seções finais fornecem conclusões, uma perspectiva histórica dos temas tratados no Capítulo e um resumo do conteúdo abordado.

1.3 Projeto e Fabricação de Sistemas Digitais

O termo ***projeto de sistemas digitais*** pode ser informalmente definido como a atividade de empregar um conjunto de métodos para desenvolver um *plano* detalhado do sistema, que pode ser então ser usado na fabricação automática (ou quase automática) do dispositivo eletrônico que implementa a funcionalidade desejada. Ao conjunto de métodos usado no projeto, dá-se o nome de ***estilo de projeto***. Normalmente, o processo de fabricação tem papel primordial em determinar o estilo de projeto, fornecendo as funções objetivo a serem otimizadas e estabelecendo restrições sobre a viabilidade de implementação de sistemas. Por exemplo, não é possível, devido a imposições da tecnologia atual, implementar um circuito integrado de memória com capacidade para armazenar 100 gigabytes.

Os métodos de *projeto* e de *fabricação* de sistemas digitais têm sua base teórica no tripé: Física de Semicondutores, Matemática Aplicada, Ciência da Computação. A Figura 1.2 ilustra como a interação destas ciências gera a disciplina de Engenharia de Sistemas Digitais VLSI. Como percebe-se na Figura,

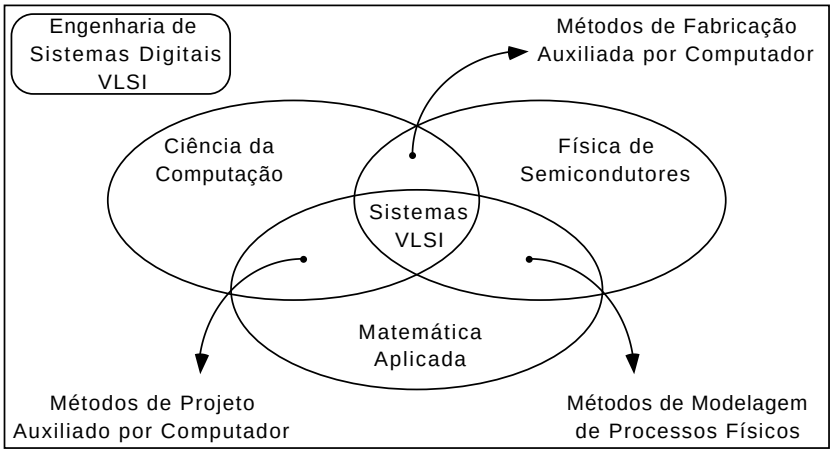


Figura 1.2: A relação entre Ciência da Computação, Física de Semicondutores e Matemática Aplicada na geração da disciplina de Engenharia de Sistemas Digitais VLSI.

na intersecção entre cada par de ciências encontram-se conjuntos de métodos aplicáveis a sistemas digitais.

A partir da Matemática e da Física, obtém-se métodos para modelar os processos inerentes a dispositivos eletrônicos. Assim, pode-se formalizar a quantificação dos fenômenos físicos envolvidos, visando controlar estes em um dispositivo produzido artificialmente. A partir desta modelagem e com o concurso da Computação, desenvolvem-se métodos de fabricação auxiliada por computador, para permitir a produção artificial automatizada em massa de dispositivos eletrônicos de comportamento previsível. Finalmente, a conjugação de Matemática e Computação permite a captura parcial da criatividade de projetistas humanos de sistemas digitais em métodos de projeto auxiliado por computador, capazes de acelerar a produção de planos de dispositivos altamente complexos, passíveis de serem fabricados em massa de forma automatizada. Hoje, os processos de maior sucesso em produzir sistemas digitais eletrônicos de alta complexidade são os chamados *processos planares*.

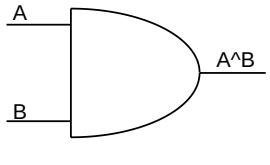
A partir dos processos planares de fabricação de circuitos integrados (ICs, do inglês *integrated circuits*), derivaram-se tecnologias que permitem produzir, *industrialmente*, sistemas eletrônicos digitais binários VLSI. Estas tecnologias habilitam o alcance de um baixo custo de comercialização por dispositivo eletrônico. Por industrial, entenda-se a produção de um grande volume de dispositivos idênticos. Por sistema eletrônico digital binário, entenda-se um sistema eletrônico onde a informação é representada e manipulada mediante uso de

uma codificação que faz uso de apenas dois valores distintos de uma grandeza elétrica, tipicamente tensão. Finalmente, circuitos VLSI do estado da arte são realizados como uma pastilha (em inglês, *dice*) de material semicondutor com cerca de $1cm^2$ de área, espessura inferior a $1mm$, e contendo um número de componentes eletrônicos interconectados que pode atingir a ordem da dezena de milhões. Mais curioso ainda, da espessura aproximada de $1mm$ da pastilha, menos de 1 milésimo é empregado como área ativa pelos dispositivos eletrônicos (donde o nome de processo planar de fabricação), o restante existindo primordialmente com finalidade de prover resistência mecânica.

O avanço dos processos planares de fabricação de sistemas digitais VLSI pode ser medido a partir da distância mínima entre duas linhas paralelas separadas que o processo permite traçar na superfície do semicondutor (em inglês, *min-feature size*). Para um processo de fabricação avançado no ano de 1996, um valor típico de distância era de $0,25\mu m$ [90], hoje já em superação por geometrias com o min-feature size de $0,18\mu m$. Ainda mais recentemente, foi anunciada pela IBM [84] a disponibilidade comercial do primeiro processo de fabricação empregando fios de cobre ao invés de alumínio, com um min-feature size de $0,12\mu m$! Tais valores de geometria explicam a possibilidade de dispor tamanha quantidade de componentes sobre área tão diminuta.

Dada a ordem de grandeza habilitada pelas tecnologias de fabricação VLSI, o problema de projeto de sistemas digitais pode ser enunciado, hoje, como aquele de controlar a complexidade de interconectar até dezenas de milhões de componentes, de forma a atender um conjunto de especificações. Tão crucial quanto este é o problema da *escalabilidade*[1] das técnicas de projeto devido ao avanço das tecnologias de fabricação. A título de exemplo, no final da década de 80, o *min-feature size* do estado da arte era de $6\mu m$, ou seja, 24 vezes maior que o valor de 1996, correspondendo a circuitos digitais com apenas dezenas de milhares de componentes dispostos no mesmo $1cm^2$ de área, respeitando a dependência quadrática entre o *min-feature size* e a área [39]. O ritmo de avanços da tecnologia de fabricação tem se mantido exponencial nas últimas décadas, como atesta a existência e observação da chamada *Lei de Moore*. Esta lei é devida a Gordon E. Moore, que em 1965 observou que a densidade de componentes em circuitos integrados dobrava a intervalos regulares, inferindo que este comportamento perduraria por muito tempo ainda. O intervalo medido por Moore para que a densidade média de ICs dobrasse foi de 18 meses, o que ainda hoje permanece uma taxa estável [162]. Assim, técnicas de projeto adequadas devem poder ser aplicadas dentro de cerca de 15 anos a problemas três ordens de grandeza maiores que os de hoje, uma vez que 180 meses correspon-

[1] O termo, aparentemente um neologismo em português, vem do inglês, *scalability*, e significa uma medida da capacidade de uma técnica (de projeto) poder lidar com instâncias cada vez maiores do mesmo problema. Em VLSI, isto significa lidar com instâncias de tamanho crescente de forma exponencial ao longo do tempo.



A	B	A^B
0	0	0
0	1	0
1	0	0
1	1	1

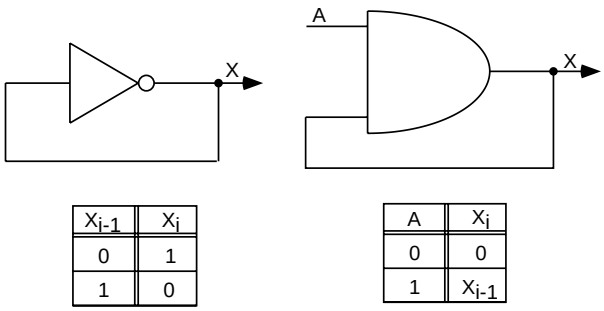
Figura 1.3: Uma porta lógica "E" e sua tabela verdade, um exemplo de circuito combinacional, interpretável como uma memória permanente.

dem, segundo a Lei de Moore, a uma multiplicação da densidade de circuitos por 2^{10}, ou seja, aproximadamente 1000 vezes o tamanho atual!

Além dos problemas de complexidade de projeto e escalabilidade, o tempo normal durante o qual um produto eletrônico gerava a maior parte dos lucros (em inglês, *revenue life*) foi reduzido de 3 a 5 anos, ao final dos anos 80, para 1 ano ou menos na atualidade. Como conseqüência, o tempo para um produto chegar ao mercado (em inglês, *time to market*) passa a ser muito mais relevante do que outros parâmetros tradicionalmente considerados, tais como o desempenho, ou mesmo os custos final e de produção.

1.4 Sistemas Combinacionais e Seqüenciais

Pode-se avançar uma primeira definição de sistema digital seqüencial baseado na relação entre entradas e saídas do sistema, e por oposição ao conceito de sistema digital combinacional. Informalmente, um *sistema digital combinacional* é qualquer sistema digital onde o comportamento de cada saída pode ser descrito como uma função que depende exclusivamente das combinações de valores instantâneos das entradas do sistema (donde o nome combinacional). Em outras palavras, um sistema digital combinacional pode ser completamente descrito por uma tabela verdade com tantas entradas quanto as entradas do sistema e com tantas saídas quanto as saídas do sistema. Esta tabela apresenta uma linha para cada combinação de valores de entradas distintos, e para cada linha as saídas recebem valores associados a esta combinação de entradas. Um *sistema digital seqüencial* é qualquer sistema que não possa, em geral, ser descrito desta maneira. O termo "em geral" é importante neste contexto, pois serve para garantir que sistemas combinacionais possam ser vistos como um caso especial de sistema seqüencial, como ver-se-á abaixo. Na Seção 1.4.1 discute-se a relação de sistemas digitais combinacionais e seqüenciais e o conceito de armazenamento de informação. Segue-se a Seção 1.4.2, onde os resultados da Seção 1.4.1 são usados para definir o conceito de *estado* em sistemas digitais. Finalmente, do conceito de estado deriva-se uma definição mais precisa

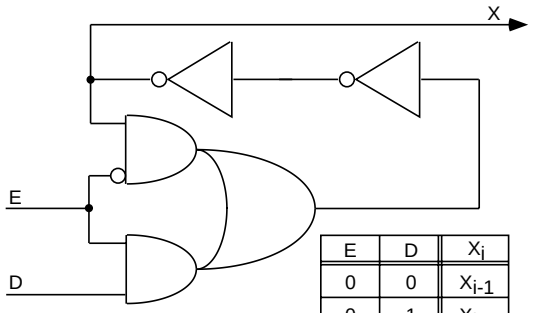


X_{i-1}	X_i
0	1
1	0

(a) Oscilador em anel

A	X_i
0	0
1	X_{i-1}

(b) Captura de zero



E	D	X_i
0	0	X_{i-1}
0	1	X_{i-1}
1	0	0
1	1	1

(c) Elemento biestável

Figura 1.4: Três circuitos digitais com capacidade de amazenar informações de forma transitória.

de sistemas digitais combinacionais e seqüenciais, na Seção 1.4.3.

1.4.1 Sistemas Digitais e Armazenamento de Informação

A capacidade de armazenar informação é uma característica típica de sistemas digitais seqüenciais. Contudo, pode-se distinguir duas formas de armazenamento de informação, *permanente* ou *transitória*. Como exemplo de armazenamento permanente de informação, a Figura 1.3 mostra um circuito digital amplamente conhecido, uma "porta lógica E", ou "E lógico de duas entradas"

e sua tabela verdade. Como a saída do circuito pode ser descrita como uma função que depende exclusivamente dos valores instantâneos das entradas, a porta E é claramente classificável como um circuito digital combinacional. Obviamente, o circuito também pode ser visto como uma memória permanente, onde as entradas indicam univocamente uma linha da tabela verdade (o *endereço* de memória) e a saída fornece o valor da função E para estas entradas (o *conteúdo* da posição de memória apontada pelo endereço). De fato, a possibilidade de interpretar qualquer circuito combinacional como uma memória permanente gera o princípio de construção de memórias de apenas leitura (do inglês, *read-only memories* ou ROMs).

A Figura 1.4, por outro lado, apresenta três circuitos digitais com diferentes graus de capacidade de armazenamento *transitório* de informação. Todos possuem pelo menos uma saída cujo comportamento não depende apenas do valor instantâneo das entradas do circuito. Logo, nenhum deles pode ser considerado um circuito digital combinacional. Um reflexo deste fato é a necessidade de expressar o comportamento através de uma versão modificada de tabela verdade, onde além dos valores das entradas, considera-se o *passado* do circuito, valores anteriores de saídas de portas lógicas, representados por identificadores de sinais com índices correspondendo a instantes de tempo relativos. Se X_i representa o valor de uma saída no momento atual, X_{i-1} representa o valor desta mesma saída no momento imediatamente anterior. Note-se que nesta representação está implícita a *discretização* do tempo, conceito importante para modelar circuitos digitais síncronos, como será visto no Capítulo 4. Estruturalmente, em circuitos digitais construídos a partir de portas lógicas, esta dependência do passado é obtida pela existência de pelo menos um ***laço de realimentação*** no circuito, definido informalmente como um caminho que conecta a saída de uma porta lógica à parte do circuito digital responsável pela sua geração[2].

O circuito da Figura 1.4(a) é um circuito sem entradas e apenas uma saída. A partir de sua tabela verdade, percebe-se que a saída deste circuito permanece em oscilação constante de 0 para 1 e vice-versa. Uma porta lógica, como qualquer dispositivo eletrônico real, possui um intervalo de tempo associado ao aparecimento das saídas após o estabelecimento das entradas que a produzem. Este intervalo de tempo denomina-se *atraso de propagação* (do inglês, *propagation delay time*, ou t_{pd}). No caso do oscilador, este intervalo de tempo determina a taxa ou freqüência de oscilação do circuito. Não sendo possível controlar a transitoriedade de armazenamento de informação, este circuito di-

[2]Em modelos de mais baixo nível de abstração de sistemas digitais, laços de realimentação não são a única maneira de criar efeitos de memória. Acumular carga elétrica em capacitâncias são uma maneira eficiente e barata de prover armazenamento transitório de informação [190], mas a discussão deste fenômeno, bem como dos métodos de projeto associados, está fora do escopo da presente obra.

gital tem pouca utilidade em sistemas de tratamento de informação.

Já o circuito da Figura 1.4(b), denominado "captura de zero", permite algum controle sobre a informação nele armazenada. Seu comportamento, expresso na tabela verdade modificada, é manter o valor da saída constante, até que surja um 0 na entrada A. A partir daí, a saída do circuito permanecerá sempre em 0 (donde a denominação captura de zero). Ou seja, a saída do circuito armazena uma informação binária, a ocorrência ou não de um 0 na entrada do circuito, em algum momento do passado, desde que iniciou sua operação. Novamente, existe um limite na transitoriedade da informação armazenável pelo circuito, limitando sua utilidade.

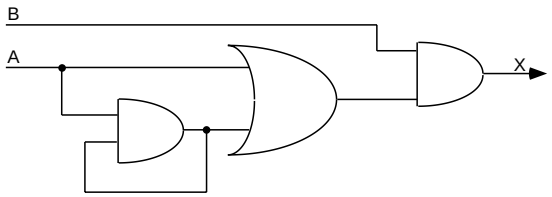
Finalmente, o circuito da Figura 1.4(c) permite controle total sobre a transitoriedade da informação armazenada no seu laço de realimentação. O circuito possui duas entradas e uma saída. A entrada D ("dado") contém uma informação binária candidata a armazenamento, enquanto a entrada E ("escreve") controla o armazenamento da informação em D. Se $E = 0$ a última informação armazenada permanece, e D não afeta a saída. Entretanto, enquanto $E = 1$, a saída é uma cópia de D. Quando E vai de 1 para 0, o último valor em D é armazenado. Logo, a qualquer momento é possível armazenar uma informação binária nova no circuito. O leitor com alguma experiência em lógica digital não terá dificuldade em identificar este circuito com a implementação de um bit de memória de escrita e leitura estática, onde os dois inversores, realimentados quando $E = 0$, são a célula de armazenamento, e as portas restantes implementam a lógica de controle de escrita.

1.4.2 Sistemas Digitais e o Conceito de Estado

Da Seção anterior, poder-se-ia inferir a definição de que sistemas combinacionais sejam aqueles que não possuem laços de realimentação, e que sistemas seqüenciais sejam os que apresentam esta estrutura. Esta definição é, estritamente falando, incorreta. O circuito digital da Figura 1.5 contraria esta definição, pois seu comportamento é combinacional, apesar de sua estrutura conter um laço de realimentação[3].

Da possibilidade de armazenamento de informação em sistemas digitais surge o problema do quanto de sua história passada um circuito digital é capaz de armazenar. Enquanto que circuitos combinacionais claramente não são capazes de armazenar qualquer parte desta história, circuitos seqüenciais podem fazê-lo, devido à existência dos laços de realimentação. Existe uma relação direta entre os laços de realimentação e a capacidade de armazenar o passado por um

[3] O leitor arguto perceberá que uma das leis de absorção Booleana (ver Página 58), $V \vee (V \wedge T) = V$, poderia ser aplicada neste circuito, para eliminar o laço de realimentação. Contudo, esta mudança não descarta a possibilidade do circuito inicial existir e ser implementado, contrariando a definição.



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Figura 1.5: Um exemplo de circuito com comportamento combinacional e com laço de realimentação.

circuito. Cada laço armazena exatamente um bit de informação. Um circuito com n laços armazena n bits, e o número de combinações destes bits, 2^n, dá a quantidade de informação total armazenável pelo circuito. Surge então a noção de estado de um sistema digital descrito de forma estrutural.

Definição 1.2 (Estado de um sistema digital) *Considere um sistema digital descrito estruturalmente como uma interconexão de portas lógicas. Para cada laço de realimentação, escolhe-se arbitrariamente um dos fios do laço. O valor binário instantâneo deste fio é o* ***valor do laço de realimentação****. Cada configuração distinta de todos os valores de laços de realimentação é denominada um* ***estado*** *do sistema digital. O fio cujo valor é escolhido para representar o estado é denominado* ***sinal de estado****.*

Note que a atribuição de valores aos laços é necessária, visto que os diferentes fios que compõem um laço podem ter valores opostos para um dado estado[4]. Mais importante ainda, o conceito de atribuição de valores deve ser estendido para lidar com atrasos em circuitos, pois fios que logicamente estão sempre com valores opostos, podem transitoriamente apresentar o mesmo valor, devido à distribuição dos atrasos do sistema. Alguns dos casos especiais de circuitos digitais descritos acima também devem ser discutidos à luz do conceito de estado. Circuitos como o oscilador da Figura 1.4(a) requerem a consideração de *estados instáveis*, de grande importância para o tratamento de circuitos assíncronos, como será visto no Capítulo 5. Circuitos como o da Figura 1.5, onde os laços de realimentação não contribuem para gerar o comportamento observável do circuito, possuem mais estados que os necessários para desempenhar a funcionalidade prevista, sendo então *redundantes*. Circuitos redundantes são mais difíceis de testar e maiores que o necessário.

As definições necessárias para diferenciar precisamente sistemas combinacionais e seqüenciais foram introduzidas. Resta a própria definição destes.

[4]Por exemplo, no caso de dois inversores realimentados formando um bit de memória, um fio contém sempre o complemento do outro, e escolhe-se qual deles corresponde ao sinal de estado.

Note-se que a definição de estado dada aqui é bastante limitada e um tanto informal. Uma forma mais precisa de apresentar os conceitos desta Seção seria modelar uma interconexão de portas lógicas como um grafo dirigido, conforme propõem Brzozowski e Seger na Seção 4.3 de [35]. Contudo, a precisão matemática pouco acrescentaria à discussão informal deste Capítulo, e foi por isso suprimida em favor da clareza.

1.4.3 Definição de Sistemas Combinacionais e Seqüenciais

A distinção de sistemas digitais combinacionais torna-se simples se feita com base no conceito de estado introduzido acima.

Definição 1.3 (Sistemas combinacionais e seqüenciais) *Qualquer sistema digital é um* ***sistema digital seqüencial****. Um* ***sistema digital combinacional*** *é um sistema digital onde, desconsiderando-se laços de realimentação que não contribuem para gerar qualquer das saídas, possui apenas* 1 *estado.* ***Sistema digital estritamente seqüencial*** *é aquele que sob as mesmas condições possui mais de* 1 *estado.*

Assim, sistema combinacional é definido como um caso especial de sistema seqüencial. No que segue, qualquer referência a um sistema digital seqüencial implica um sistema digital estritamente seqüencial.

1.5 Taxonomia de Sistemas Digitais

A compreensão aprofundada das alternativas existentes para a implementação de sistemas digitais pode ser apreendida a partir de um estudo das diferentes classes de dispositivos e sistemas disponíveis. O objetivo desta Seção é apresentar uma visão do estado da arte em classificações de sistemas digitais.

O primeiro passo para estabelecer uma taxonomia de sistemas digitais é escolher um conjunto adequado de *critérios de classificação*. A escolha de um conjunto de critérios de classificação deve buscar a *ortogonalidade* dos elementos do conjunto, ou seja, dois elementos do conjunto de critérios devem gerar classificações independentes entre si. A idéia da presente discussão não é atingir a seleção de um conjunto ótimo de critérios, e com isto atingir uma taxonomia completa e definitiva. Antes, o objetivo é expor alguns dos mais relevantes critérios de classificação de sistemas digitais e ilustrar a relação entre critérios, apresentando alguns estudos de caso de classificações. É importante perceber que o avanço tecnológico implica a revisão constante destas classificações, bem como o acréscimo de novos critérios e mudança na importância relativa de critérios estabelecidos. Discute-se a seguir sete critérios de classificação.

Um primeiro critério de classificação é a *personalizabilidade* do sistema digital, ou seja a capacidade de determinar o comportamento do sistema digital.

Dada esta definição informal, vários graus de personalização podem ser identificados. Inicialmente, sistemas digitais podem ser bipartidos em duas classes:

- sistemas *não personalizáveis* ou *fixos* - nesta primeira classe, incluem-se por exemplo os dispositivos da família TTL [177] e os microprocessadores atuais;
- sistemas *personalizáveis* - os ICs pertencentes à esta classe são às vezes denominados *circuitos integrados específicos para uma dada aplicação*, ou ASICs (do inglês, *Application Specific Integrated Circuits*). Desta classe, pode-se citar como exemplos os FPGAs (do inglês, *Field-Programmable Gate Arrays*), e os *gate arrays*.

Um FPGA é um IC cujo comportamento é determinado pelo preenchimento de uma memória interna. Dito de forma bastante simplificada, os bits desta memória de configuração comandam chaves eletrônicas. Estas chaves controlam a interconexão dos componentes ativos do IC, determinando assim o comportamento do circuito. Gate arrays, por outro lado, são ICs fabricados sob encomenda, mas onde a planta baixa está em grande parte pré-definida pelo fabricante, e as pastilhas já foram pré-fabricadas. Os componentes eletrônicos, normalmente transistores ou pares de transistores interconectados, estão todos prontos sobre o substrato semicondutor, mas a ou as camadas de interconexão destes componentes são especificadas pelo cliente que deseja um circuito com uma funcionalidade específica. Por este motivo, gate arrays são denominados também de ICs pré-difundidos, pois uma parte de sua fabricação está conceitualmente realizada antes do projeto da aplicação iniciar.

Da discussão acima nota-se que é possível fracionar os sistemas personalizáveis em duas sub-classes:

- os personalizáveis *pós-fabricação* - nesta sub-classe, encontram-se os dispositivos genericamente denominados *dispositivos lógicos programáveis*[5], ou PLDs (do inglês, *Programmable Logic Devices*), da qual fazem parte as diferentes memórias personalizáveis, tais como PROM, EPROM, EEPROM, memórias *flash* e dispositivos tais como PALs, CPLDs e os próprios FPGAs;
- os personalizáveis *por-fabricação* - na segunda sub-classe encontram-se os gate arrays.

Adicionalmente, os ICs personalizáveis por fabricação podem ser ainda subdivididos em duas sub-sub-classes:

[5] *a denominação* programável*, usada aqui é enganadora e seu significado está em desacordo com o uso introduzido nesta obra, mas é consagrada na indústria.*

- *parcialmente personalizáveis* (denominações alternativas são *semi-dedicados* e *pré-caracterizados*) - os gate arrays ou pré-difundidos não estão sós na primeira sub-sub-classe. Também existem os ICs pós-difundidos. A base de construção de ICs pós-difundidos é uma biblioteca de células pré-caracterizada, como no caso dos gate arrays, composta tipicamente por elementos do nível lógico de abstração, tais como portas lógicas e biestáveis. Estas células tem uma aparência física (ou geometria, em inglês *layout*) pré-definida, mas elas não estão fisicamente implementadas no início do projeto, como no caso dos gate arrays. Ou seja, o projetista da aplicação tem liberdade não apenas de especificar a interconexão de componentes, mas também a posição relativa de cada célula no IC. Contudo, o projetista da aplicação está restrito ao uso das células pré-caracterizadas da biblioteca. Uma classe importante de ICs pós-difundidos são os chamados *standard cells*, onde as células da biblioteca possuem um desenho retangular, onde a altura é fixa e a largura pode variar de célula para célula. Com esta restrição adicional, facilita-se a geração do IC, que é finalmente composto por uma alternância de tiras retangulares de células e canais. As primeiras são formadas por células colocadas lado a lado, e canais são são regiões usadas para interconectar as células entre si;

- *totalmente personalizáveis* - estes sistemas são aqueles em que não há restrições à geração do layout do IC. Obviamente, estes possibilitam obter o mais alto desempenho ao mais baixo custo por dispositivo, mas seu tempo de projeto é muito grande, quando comparado com gate arrays e standard cells.

Um segundo critério de classificação de sistemas digitais é a sua *programabilidade*, ou seja a capacidade do sistema digital de executar software. Como mudar o software executado pelo sistema equivale a mudar o comportamento do sistema, a distinção entre programabilidade e personalizabilidade é sutil. Considera-se aqui um sistema digital como sendo *programável* se ele possui estruturas específicas para implementar uma camada de abstração sobre o hardware, de forma a disponibilizar ao usuário do sistema uma *arquitetura* programável, ou seja uma abstração do funcionamento interno do sistema digital, cujo comportamento é determinado pela interpretação de instruções fornecidas em determinada ordem, com a natureza e a ordem das instruções interpretáveis pelo hardware podendo variar arbitrariamente ao longo do ciclo de vida do sistema digital. Caso estas estruturas não existam, diz-se que o sistema é *não-programável*.

Como terceiro critério a examinar, há a capacidade de *retenção da personalização* de um sistema digital. Deve estar claro que este não é um critério ortogonal ao critério personalizabilidade, pois não faz sentido falar em retenção da personalização para sistemas não-personalizáveis, mas a crescente importância

da personalização de hardware leva a propor este como um critério de classificação fundamental. Identifica-se quatro classes principais de sistemas digitais quanto a este critério, apresentadas a seguir, em ordem crescente de *volatilidade* da personalização:

- *personalização eterna pós-projeto* - sistemas personalizáveis por fabricação são imutáveis após fabricados, logo sua personalização é eterna após findo o seu projeto;
- *personalização eterna após a primeira personalização* - esta segunda classe é aquela que após fabricação permite uma única personalização, como as memórias PROM e os FPGAs baseados em antifusíveis [87];
- *personalização eterna, exceto pelo uso de procedimentos tecnológicos especiais* - esta terceira classe é aquela onde a personalização pode ser refeita mútltiplas vezes, mas para tanto necessita de procedimentos tecnológicos específicos. Por exemplo, antes de mudar a personalização de uma memória do tipo EPROM (sigla proveniente do inglês, *Erasable Programmable Read Only Memory*), é necessário "apagá-la", um procedimento que envolve submeter a pastilha semicondutora dentro do IC a luz ultravioleta, o que destrói a personalização anterior e prepara a memória para uma nova personalização;
- *personalização a cada entrada em funcionamento* - a última classe é a dos sistemas que mantém a personalização apenas enquanto alimentados por tensão elétrica. FPGAs baseados em RAM pertencem a esta classe.

Ilustrando os três critérios de classificação vistos até agora, a Figura 1.6 mostra uma primeira taxonomia de ICs. Cabe uma observação sobre esta taxonomia. Microprocessadores atuais como membros da família Alpha da DEC, o Pentium da Intel e o Power-PC do convênio IBM/Motorola/Apple são ICs fixos programáveis. O advento da computação reconfigurável [71] poderá estender esta taxonomia, criando sistemas que podem ser simultaneamente programáveis e personalizáveis, demonstrando a ortogonalidade dos critérios em questão, algo que não está claro na Figura 1.6.

O quarto critério a ser discutido é a *complexidade* do sistema digital. Inicialmente dispositivos digitais podem ser classificados como *discretos*, se existe uma correspondência entre um componente eletrônico e um encapsulamento. Por outro lado, circuitos digitais ou sistemas digitais *integrados* são aqueles onde cada encapsulamento possui mais de um componente eletrônico no seu interior. O número n de dispositivos eletrônicos dita uma sub-classificação dos integrados em:

- SSI - significa pequena escala de integração (do inglês, *Small Scale Integration*), quando $1 < n < 10^2$;

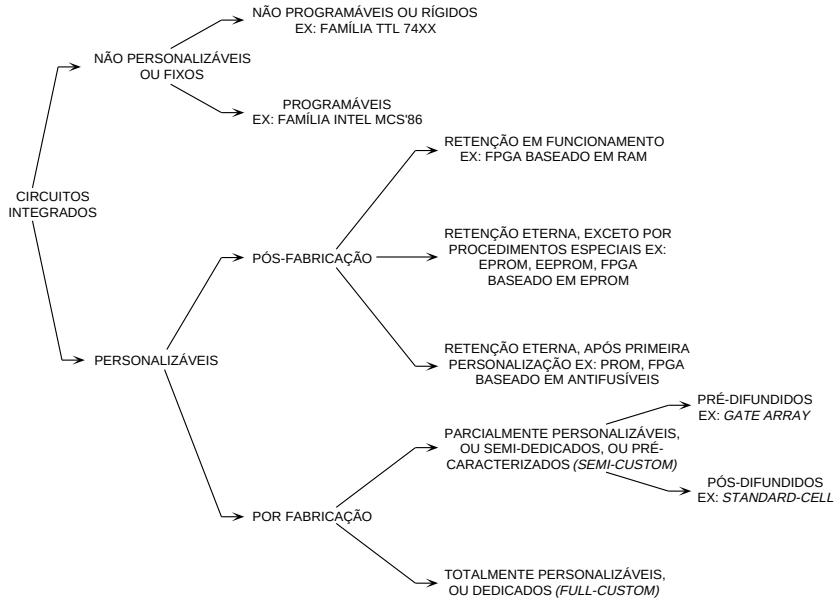


Figura 1.6: Uma taxonomia para ICs, construída segundo três critérios de classificação: personalizabilidade, programabilidade e retenção da personalização.

- MSI - significa média escala de integração (do inglês, *Medium Scale Integration*), quando $10^2 \leq n < 10^3$;
- LSI - significa grande escala de integração (do inglês, *Large Scale Integration*), quando $10^3 \leq n < 10^5$;
- VLSI - significa muito grande escala de integração (do inglês, *Very Large Scale Integration*), quando $10^5 < n < 10^6$;
- ULSI - significa ultra grande escala de integração (do inglês, *Small Scale Integration*), quando $10^6 < n$.

Os limites acima para cada classe são apenas indicativos de ordem de grandeza. Devido à velocidade de crescimento do tamanho dos ICs, esta classificação tende a perder significado, até pela falta de qualificativos de complexidade. Em 2015, dispositivos com 10^9 transistores serão classificados como? Malgrado esta observação, esta nomenclatura ainda é muito empregada informalmente.

Um quinto critério de classificação, exclusivo para ICs é a *forma de produção*. ICs produzidos em volume e disponíveis comercialmente são ditos *de prateleira*, enquanto que os restantes são classificados como ASICs, termo introduzido na Página 13.

Os dois últimos critérios são mais técnicos. O sexto critério é a *relação entre entradas e saídas* do sistema digital. As duas grandes classes de sistemas digitais segundo este critério são os sistemas combinacionais e os sistemas seqüenciais, assunto já discutido na Seção 1.4.

Finalmente, o sétimo critério são os *pressupostos de sincronismo* usados no projeto e na construção do sistema digital. Todo sistema digital que atende à Definição 1.1 pressupõe que as suas entradas e saídas trabalham com valores de um conjunto finito, ao contrário dos dispositivos físicos que implementam este sistema. Ou seja, um sistema digital é aquele que permite considerar suas entradas e saídas de forma abstrata, como operando sobre sinais discretos ao longo do tempo, e não sobre as grandezas físicas contínuas como as tensões e correntes elétricas que realmente são usadas. Além desta camada de abstração, comum a todos os sistemas digitais, pode-se ainda acrescentar uma camada de abstração com relação ao tempo. Se o tempo também é considerado como composto de um conjunto de momentos discretos, o sistema digital assim concebido é dito ***síncrono***. Caso contrário, o sistema digital é denominado ***assíncrono***. Cabe uma observação sobre estes conceitos. A maneira mais comum de implementar a discretização do tempo é utilizar um sinal de controle periódico que comanda todas as ações do sistema digital simultaneamente. Tal sinal normalmente é denominado *relógio* (do inglês, *clock*). Este sinal estabelece os momentos em que entradas podem ser utilizadas e sincroniza todas as operações do sistema. Em geral, o sinal de relógio possui freqüência, e portanto período, fixos. A cada ciclo do relógio, novas entradas são aceitas e novas saídas são geradas. Sistemas assíncronos ou não possuem relógio, ou não possuem um relógio único. Sistemas síncronos são o assunto do Capítulo 4, e sistemas assíncronos são o tema do Capítulo 5.

Vários outros critérios de classificação poderiam ser discutidos, tais como os que realizam a distinção entre sistemas digitais, sistemas analógicos e sistemas híbridos, ou a distinção entre sistemas de processamento simbólico, processamento numérico e processamento de sinais. Contudo, os critérios aqui discutidos são suficientes para os objetivos desta obra.

1.6 O Processo de Projeto de Sistemas Digitais

O ***processo de projeto*** pode ser definido informalmente como a transformação de uma *descrição inicial* do sistema, freqüentemente denominada *especificação*, em uma *descrição final*, também chamada de *projeto final* ou *projeto detalhado*[6]. A diferença primordial entre a descrição inicial e a descrição final

[6] A informação contida em uma especificação não difere fundamentalmente da informação contida no projeto detalhado, visto que ambas descrevem o sistema que se deseja fabricar. Apenas muda a quantidade de informação (grau de abstração) e a precisão destas. Assim,

está no fato desta última conter todas as informações necessárias à fabricação do sistema, ao contrário da primeira. Quando se trata de projetar sistemas complexos, a transformação dificilmente pode ser feita de maneira direta. Nestes casos, a atividade de projeto produz um *continuum* de descrições, obtidas pela sucessiva agregação de informações à descrição inicial, culminando com a descrição final.

Cabe salientar que a descrição final que atende aos requisitos da descrição inicial não é, em geral, única. Na maioria esmagadora dos casos de projeto de sistemas VLSI, é inviável enumerar todos os caminhos a seguir, mesmo que em alguns casos não existam infinitas possibilidades. Como conseqüência, o projeto dificilmente se realiza de forma linear, pois a cada transformação, decisões devem ser tomadas, sem que se saiba se estas são as que conduzem à melhor das descrições finais que satisfazem a descrição inicial. Além do mais, a cada transformação podem ser introduzidos erros, violados requisitos da descrição inicial, ou inviabilizada a obtenção da melhor solução possível para o processo como um todo.

Nesta Seção, discutem-se modelos para descrever o processo de projeto de sistemas digitais.

1.6.1 Uma Palavra sobre Modelos

Modelos são em geral abstrações de entidades do mundo real, destinadas a permitir a manipulação sistemática de parte dos conceitos destas entidades. Um bom modelo deve ser capaz de representar *todos* e *apenas* os conceitos relevantes a serem manipulados. Além do aspecto quantitativo de representação dos conceitos relevantes da entidade, é necessário que a esta permita as manipulações em vista. Por exemplo, embora a álgebra Booleana seja uma estrutura matemática extremamente útil na representação de modelos de sistemas digitais, ela é em si insuficiente para computar o desempenho temporal destes sistemas visto que não incorpora os aspectos de atrasos, intrínsecos aos dispositivos eletrônicos reais.

No caso do processo de projeto, a utilidade dos modelos a serem apresentados aqui jaz na capacidade de descrever o fluxo de informação empregado para construir uma descrição implementável do sistema digital em questão.

1.6.2 Modelos para o Processo de Projeto

Apresenta-se a seguir três modelos para representar o processo de projeto de sistemas digitais. O primeiro modelo é baseado em um critério de classificação de descrições de sistemas digitais, dando origem a uma classificação linear, ou

prefere-se aqui empregar o termo genérico descrição.

unidimensional. O segundo modelo é baseado em dois critérios de classificação, gerando um modelo bidimensional. O terceiro modelo utiliza três critérios de classificação de descrições, produzindo uma classificação tridimensional.

1.6.2.1 O Modelo de Suzim

Dada a complexidade dos sistemas digitais digitais VLSI modernos, impõe-se a decomposição de seu processo de projeto. Mencionou-se acima que este processo é em geral particionado em um conjunto de passos, com o objetivo de reduzir a complexidade de um tratamento global. Um primeiro critério a empregar para decompor o processo de projeto é fazer com que cada passo manipule descrições com um determinado *grau de detalhamento* de informações. Define-se assim ***nível de abstração*** como um conjunto de descrições de projeto com o mesmo grau de detalhamento. Por exemplo, para um mesmo sistema digital, um diagrama de portas lógicas, contém significativamente menos informação que uma descrição elétrica do tipo SPICE, estando portanto cada uma em um nível de abstração diferente. Outro exemplo de descrições com distinto nível de abstração seria uma descrição da arquitetura de um microprocessador como vista pelo programador de linguagem de montagem e o layout do mesmo microprocessador entregue para fabricação, a primeira descrição é muito mais abstrata que a segunda. Por outro lado, um conjunto de equações Booleanas pode transportar a mesma quantidade de informação que um diagrama de portas lógicas. Logo, estas descrições, apesar de distintas, pertencem ao mesmo nível de abstração. Qualitativamente, emprega-se a terminologia *alto nível de abstração* como sinônimo de pouca quantidade de informação. Neste sentido, a quantidade de informação é inversamente proporcional ao nível ou grau de abstração.

O modelo proposto por Suzim [172] é parcialmente ilustrado na Figura 1.7. Este modelo baseia-se na decomposição do processo de projeto segundo o critério de níveis de abstração de projeto e na classificação das operações de projeto segundo o tipo de transformação que estas realizam.

Claramente, a decomposição mostrada pelo modelo de Suzim estabelece uma hierarquia linear de conjuntos de descrições. A Figura 1.7 mostra dois níveis típicos e a relação entre níveis, supondo um processo de projeto com n níveis de abstração numerados de 1 (o mais abstrato, a descrição inicial) a n (o menos abstrato, a descrição final). Na Figura, elipses representam descrições e arcos correspondem a transformações, operações de projeto sobre as descrições. As operações são classificadas segundo a maneira como transformam descrições entre níveis de abstração. Antes de discutir a classificação de operações, ver-se-á um exemplo que serve para ilustrar os conceitos envolvidos.

Exemplo 1.1 (O Processo de Projeto Clássico) Considere-se dado o seguinte processo de projeto. A partir de uma especificação informal em por-

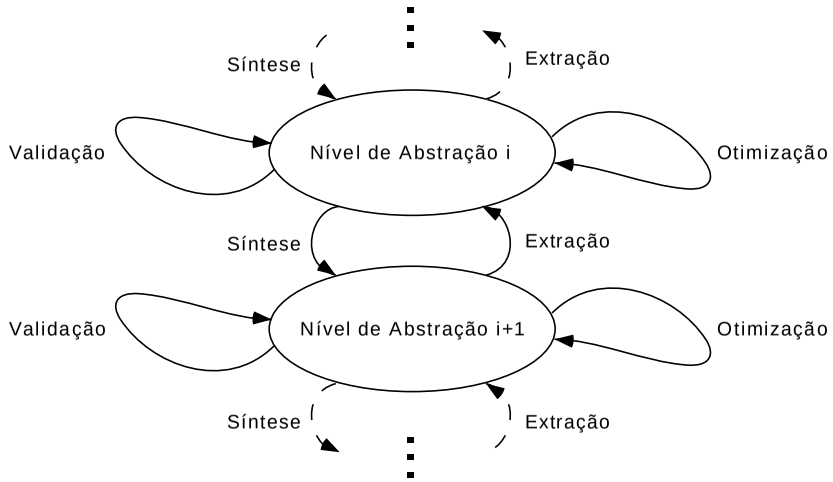


Figura 1.7: O modelo proposto por Suzim para representar o processo de projeto de sistemas digitais.

tuguês estruturado e diagramas de blocos e de tempos, gera-se uma diagrama hierárquico de portas lógicas, mediante uso de um editor de esquemáticos. A partir daí, procede-se à validação do diagrama de esquemáticos através de um simulador lógico com atrasos associados às portas. Após um número suficiente de iterações simulador-editor, obtém-se a validação neste nível. Este diagrama de portas lógicas, embora possivelmente sem erros, pode ainda conter mais portas que o número mínimo necessário. Assim, procede-se ao emprego de uma ferramenta automática de minimização de redundâncias, do tamanho do caminho crítico dos sinais e da estimativa de ocupação de área do sistema final. O novo diagrama de portas serve de entrada para um subsistema de ferramentas automáticas de síntese física (particionamento, posicionamento e traçado de rotas). Após a síntese física, uma ferramenta de "back annotation" é invocada para computar os atrasos associados a cada fio da implementação, e estes atrasos são alimentados no diagrama de portas lógicas obtido, viabilizando a posterior simulação lógica de temporização, cujos resultados fornecem uma garantia maior de funcionamento do sistema final que a primeira fase de simulação.

■

Pode-se agora definir as classes de operações do modelo de Suzim:

- *operações de síntese* - são as que acrescentam detalhes a uma descrição em um nível, gerando uma descrição de um nível menos abstrato. No

Exemplo 1.1, a edição de esquemáticos é uma operação de síntese manual, que parte das descrições iniciais[7]. As ferramentas de particionamento, posicionamento e traçado de rotas do Exemplo também proporcionam a realização de operações de síntese, em geral de forma automatizada [26];

- *operações de extração* - transformam a descrição lógica original, produzindo uma descrição mais abstrata que esta, ou seja, é a operação oposta da síntese, sendo útil para garantir a validade da síntese e/ou acrescentar detalhes obtidos em níveis inferiores a níveis superiores de abstração. A ferramenta de back annotation do Exemplo 1.1 realiza este tipo de operação.

- *operações de validação* - existem para garantir que a funcionalidade de uma descrição corresponde à esperada. Elas exercitam a descrição original, produzindo uma descrição do comportamento desta no mesmo nível de abstração, com vistas a comprovar que a funcionalidade da descrição corresponde à esperada no nível de abstração corrente. Os simuladores lógicos mencionados no Exemplo acima habilitam realizar validação de descrições. Outro exemplo são as ferramentas de verificação formal [133, 14, 24];

- *operações de otimização* - modificam a descrição de entrada, visando melhorar as figuras de mérito do sistema em algum nível de abstração. Transformam a descrição original em outra equivalente, em termos de nível de abstração, mas que melhora valores das figuras de mérito de projeto. O minimizador mencionado no Exemplo é claramente uma ferramenta que capacita executar operações de otimização. Outros exemplos seriam a minimização lógica combinacional [23, 59], e a minimização de estados[8] em máquinas de estados finitas (FSMs) [89, 12];

Note-se que operações de validação e otimização não geram descrições de níveis outros que o da(s) descrições de entrada. A otimização altera diretamente a descrição de entrada, e a validação gera uma descrição distinta da de entrada, mas no mesmo nível de abstração.

Na prática, as numerosas descrições de um sistema digital podem ser classificadas em quatro ou cinco níveis de abstração distintos. Nesta obra, emprega-se uma hierarquia de quatro níveis que são, em ordem descrescente de grau de abstração:

- nível sistêmico - aqui, o sistema digital é descrito como um conjunto de algoritmos e módulos capazes de executar estes, que podem ou não ter

[7] Note-se que uma ferramenta de projeto pode, em geral, receber várias descrições de entrada e gerar várias descrições de saída.

[8] Este problema específico será tratado em detalhe na Seção 4.1.

um mapeamento direto para um hardware existente; exemplos são processadores, memórias, componentes abstratos implementados no software básico do sistema tais como um pacote de ponto flutuante em um sistema sem coprocessador aritmético, e monitores de sistemas operacionais;

- nível arquitetural - onde os modelos de base são componentes complexos da teoria de circuitos digitais e as formas de descrevê-los: registradores, decodificadores, ULAs, multiplexadores, linguagens de transferência entre registradores, grafos de fluxo de dados e de controle, etc;
- nível lógico - onde o sistema digital é descrito a partir de noções elementares da teoria de circuitos digitais: portas lógicas, *flip-flops*, equações Booleanas, tabelas e grafos de transição, etc;
- nível elétrico - onde os modelos dos componentes provêm da teoria de circuitos elétricos e eletrônicos, e/ou da física de semicondutores, i.e. transistores, resistores, capacitores, equações diferenciais, diagramas elétricos, gráficos de resposta de circuitos (tensão versus tempo, tensão versus corrente, entre outros), etc.

Níveis de abstração distintos são seguidas vezes atacados por profissionais distintos, quando não por equipes de projeto distintas, ao longo do ciclo de projeto de um sistema complexo.

Finalmente, a presente descrição do modelo acima está incompleta. Foi realizada uma seleção dos aspectos mais importantes deste para o presente livro. Vale a pena citar aqui pelo menos mais uma característica relevante da proposta de Suzim, pois ela difere da definição dada aqui: trata-se do conceito de *nível de abstração*. ***Nível de abstração*** segundo Suzim é "um conjunto coerente de primitivas que possibilitam a descrição de um objeto" [172], ou mais sucintamente, uma linguagem. Aqui a definição foi estendida para compreender um conjunto de linguagens como caracterizando nível de abstração. Informações mais extensas sobre o modelo original podem ser obtidas na referência citada.

1.6.2.2 O Diagrama Y ou Diagrama de Gajski-Kuhn

Além da decomposição do processo de projeto em níveis de abstração, que pode-se classificar como *quantitativa* (pois esta se baseia no detalhamento ou quantidade de informação contida em uma descrição), é possível obter vantagens de uma decomposição *qualitativa* do processo de projeto. Uma ***visão***, ou ***domínio de descrição*** é um conjunto de descrições que compartilham um mesmo *tipo* de informação. Isto implica estabelecer um critério adicional de classificação de sistemas digitais, baseado na natureza da informação contida em uma descrição de projeto. Por exemplo, considere-se um sistema digital descrito por um diagrama de portas lógicas e também por um conjunto de

diagramas de tempo de sinais digitais, sem inclusão de informações de atraso. Embora ambas descrições pertençam estritamente ao nível lógico de abstração, a natureza da informação que elas veiculam é certamente distinguível. No primeiro caso, a *estrutura* do sistema digital está explicitada, enquanto que no segundo, apenas a *funcionalidade* ou o *comportamento* do sistema ao longo do tempo é expressa. Outro domínio fundamental é o da *geometria*, onde aspectos físicos do sistema sob projeto são expressos, como ocorre no caso de diagramas de placas de circuito impresso ou na representação de uma planta baixa de um IC.

Resumindo, pode-se identificar três domínios de descrição para sistemas digitais:

- físico ou geométrico - contém as descrições com informação geométrica sobre os componentes/módulos e/ou sobre a disposição espacial destes no sistema a ser fabricado;
- estrutural - contém as descrições com informação sobre como interconectar blocos de base de comportamento conhecido para realizar determinada funcionalidade, sem se ocupar com a disposição física destes no sistema, ou seja, representando apenas a *topologia* do sistema;
- comportamental ou funcional - envolve descrições com informação sobre o comportamento do sistema, sem se preocupar em como tal comportamento pode ser obtido, seja do ponto de vista físico, seja do ponto de vista estrutural.

Apresenta-se nesta Seção um modelo voltado para a representação do processo de projeto de sistemas digitais, proposto por Gajski e Kuhn [81], obtido pela combinação de níveis de abstração e domínios de descrição. Esta combinação provê um modelo ao mesmo tempo simples e poderoso para interpretar o processo de projeto de sistemas digitais. Tal modelo bidimensional, chamado diagrama Y, aparece na Figura 1.8. No diagrama Y, círculos concêntricos correspondem a níveis de abstração, enquanto que segmentos de reta radiais correspondem a domínios de descrição. Cada intersecção de um círculo com um segmento radial representa um tipo de descrição distinta do sistema digital. Por exemplo, um diagrama de esquemáticos de portas lógicas TTL é uma descrição estrutural lógica, estando portanto localizado na intersecção do eixo rotulado Domínio Estrutural com o círculo representando o nível lógico de abstração.

Dado o diagrama Y, como se representa um processo de projeto através dele? Considerando a definição do início da Seção 1.6, ilustra-se este processo como um grafo dirigido sobreposto ao diagrama Y, onde o conjunto de vértices do grafo é o conjunto de descrições de projeto (correspondendo a um subconjunto de pontos determinado pelas intersecções de círculos com segmentos de

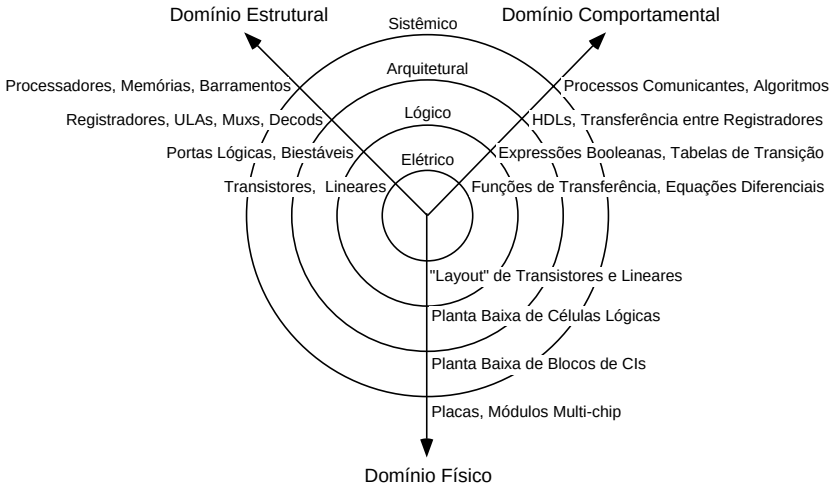


Figura 1.8: O diagrama Y ou modelo de Gajski-Kuhn para representar o processo de projeto de sistemas digitais. Círculos representam domínios de descrição e eixos radiais correspondem a domínios de descrição. Intersecções de círculos com eixos representam descrições de projeto.

reta). O conjunto de arestas é o conjunto de transformações aplicadas sobre estas descrições, ou seja, arestas são associadas a métodos e/ou a ferramentas empregados pelo projetista. Estas arestas conectam descrições de entrada de um método/ferramenta a descrições de saída geradas por este. Dado o grafo de um processo de projeto, um projeto de sistema digital corresponde a um ou a um conjunto de caminhos sobre este grafo, eventualmente com arestas repetidas, associadas a iterações de projeto.

Alguns comentários sobre o diagrama Y ajudam a compreender sua utilidade e emprego. Primeiro, o centro do diagrama corresponde conceitualmente à descrição final, ou seja, àquela que contém toda a informação necessária à fabricação do sistema digital. Segundo, ferramentas de projeto executam transformações em descrições, identificando-se então com as arestas do grafo que descreve o processo de projeto. Como ferramentas podem manipular mais de uma descrição de entrada distinta, pode-se eventualmente associá-las a um conjunto de arestas. Por outro lado, ferramentas podem gerar uma sucessão de descrições de saída. Terceiro, o tipo de arestas presentes no diagrama Y nos permite classificar as ferramentas de acordo com o tipo de transformação que estas realizam sobre suas descrições de entrada, como já foi feito no caso do modelo de Suzim na Seção 1.6.2.1. Por exemplo, ferramentas de análise produzem uma descrição de saída que não está mais próxima do centro do diagrama que a

descrição de entrada. Ferramentas de síntese no entanto, produzem descrições de saída que não estão mais afastadas do centro do diagrama que a descrição de entrada. Ferramentas de otimização e correspondem a arestas com origem e destino no mesmo vértice (em inglês, *self-loops*) do grafo do processo de projeto, enquanto que ferramentas de validação conectam vértices sobre um mesmo círculo do diagrama, eventualmente gerando self-loops. Quarto, o processo de projeto ideal seria linear, e lembraria um caminho em espiral, partindo da descrição comportamental de mais alto nível de abstração até o centro do diagrama, visitando todas as descrições de projeto possíveis. Neste caso ideal, a quantidade de informação agregada ao projeto cresceria monotonicamente do início ao fim do processo.

A título ilustrativo da representação do processo de projeto, a Figura 1.9 mostra o diagrama Y associado ao Exemplo 1.1.

Cabe neste ponto uma derradeira observação sobre o modelo apresentado. Os níveis de abstração e as visões aqui sugeridos não devem ser encarados como rígidos. A complexidade do processo de projeto causa a existência de uma multiplicidade de descrições. Um esquema rígido de posicionar cada descrição em um nível e em uma visão bem definidos faria níveis e visões se multiplicarem excessivamente, toldando a utilidade do modelo. Por exemplo, onde posicionar-se-ia um esquemático de chaves [94, 32] descrevendo um sistema digital? Trata-se de uma descrição claramente estrutural, mas os componentes desta descrição são transistores, encarados contudo como chaves, com um comportamento eminentemente lógico. Em nossa representação, posiciona-se tal esquemático sobre o eixo estrutural e *entre* os níveis lógico e elétrico de abstração. Um outro exemplo seria uma descrição em VHDL [108, 124], uma linguagem para descrever arquiteturas de sistemas digitais contendo modelos comportamentais e estruturais, que podem ser misturados em uma mesma descrição. Aqui, o nível de abstração está bem especificado, mas a visão é um misto de estrutura e comportamento. Posiciona-se tal descrição sobre o nível de abstração arquitetural mas entre os domínios estrutural e comportamental do diagrama Y. Mais importante ainda, alguns autores e.g. [109] apontam como uma deficiência do diagrama Y o fato deste não representar explicitamente a visão temporal de sistemas digitais, ao colocar sobre o eixo comportamental descrições de natureza tão dissimilar como a saída de um simulador lógico (um diagrama de tempos) e tabelas-verdade de circuitos combinacionais. Este autores freqüentemente sugerem a adição ao diagrama Y de um eixo associado à visão temporal de sistemas digitais, dada a importância deste aspecto do processo de projeto. Várias outras modificações, acréscimos e correções têm sido sugeridas no modelo, mas uma discussão detalhada destas foge ao escopo desta obra.

Independente das eventuais deficiências, o diagrama Y é um modelo que permite a visualização do processo de projeto de sistemas digitais, bem como a

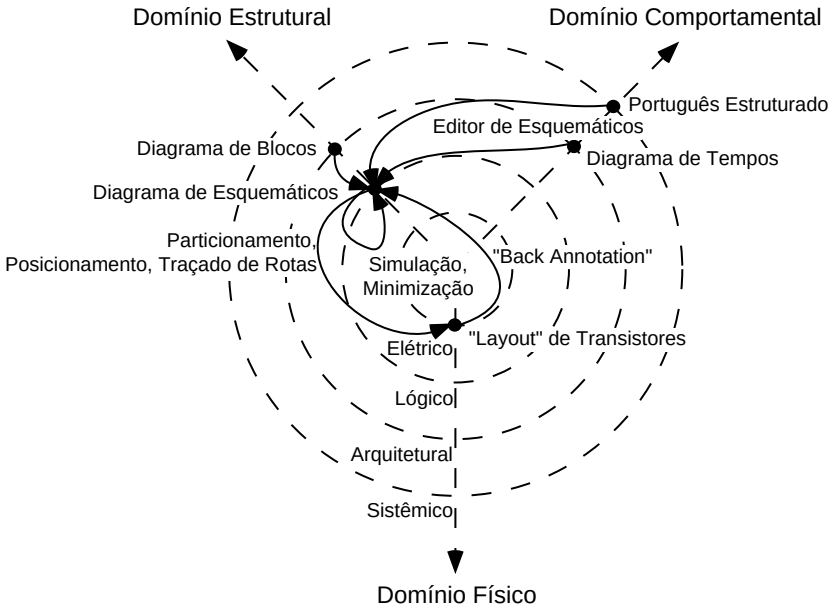


Figura 1.9: Representação do processo de projeto de sistemas digitais mediante uso do diagrama Y para o Exemplo 1.1.

apreensão de estilos de projeto particulares, dados um conjunto de ferramentas e um fluxo de seu emprego para a execução do projeto. O diagrama Y é o modelo que será prioritariamente empregado nesta obra para descrever processos de projeto.

1.6.2.3 O Modelo de Medland

Enquanto que o modelo de Suzim e o de Gajski-Kuhn foram elaborados no contexto de projeto de sistemas eletrônicos digitais VLSI, o modelo a ser estudado nesta Seção originou-se de autor que trabalha primordialmente com projeto mecânico auxiliado por computador. Medland propõe em [135] um modelo que se pretende de propósito mais geral, baseado em outros critérios de classificação. Para este autor, o processo de projeto é representado idealmente como o resultado da cooperação entre três subprocessos, evoluindo em paralelo: conceito, análise e detalhamento. A Figura 1.10 ilustra o modelo de Medland, mostrando os subprocessos e a relação entre eles.

Enquanto a fase de análise, em vista da discussão dos modelos anteriores, deve ter uma interpretação clara, é necessário diferenciar com cuidado as noções de conceito e detalhamento. Conceito envolve atividades destinadas a formular

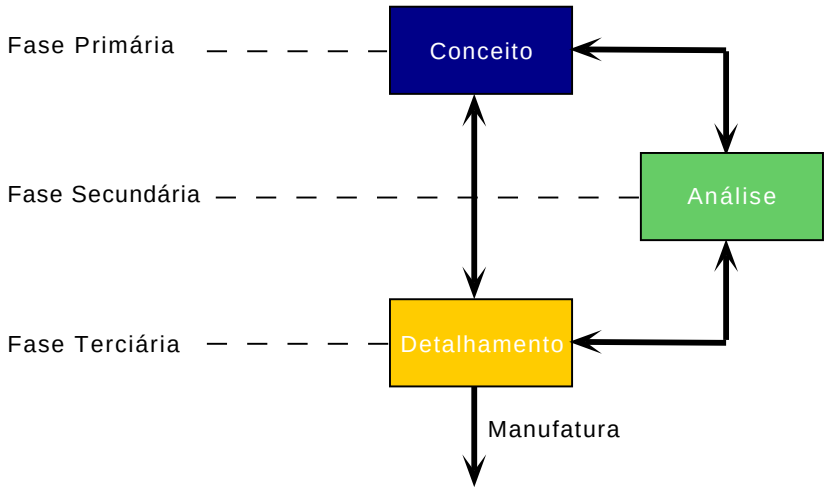


Figura 1.10: Uma visão geral do modelo de Medland para o processo de projeto em engenharia. Os três subprocessos (conceito, análise e detalhamento) correspondem às fases do projeto. Eles evoluem em paralelo, as setas indicando o fluxo de informações entre os subprocessos. Procede-se à manufatura após a iteração atingir detalhamento suficiente.

e limitar o escopo do problema de projeto, enquanto que detalhamento inclui atividades de teste de viabilidade de implementação dos resultados da fase conceitual e de implementação propriamente ditas.

Tome-se um exemplo prático de projeto: implementar um sistema digital capaz de realizar a soma de dois números representados em binário. Na fase primária avalia-se, conceitualmente, as alternativas de implementação, em vista das restrições de projeto. Se existem fortes restrições de área, é normal que se descarte aqui a possibilidade de usar esquemas de propagação avançada de vai-um (em inglês, *carry look-ahead generation*). O contrário pode ser decidido, caso os requisitos de desempenho sejam altos. Na fase terciária, uma vez decidido do esquema de implementação conceitual, realiza-se a construção de modelos detalhados que vão ajudar, mediante as ferramentas da fase de análise, a corroborar ou descartar as idéias selecionadas na fase conceitual. Nota-se então uma diferença real entre a natureza das atividades de seleção de alternativas (conceito) e de implementação da alternativa selecionada (detalhamento), intermediada a ação entre as fases pelo ferramental de análise abstrata (relação entre conceito e análise) e concreta (relação entre detalhamento e análise). Uma observação adicional sobre a Figura 1.10 é que a denominação de fases (primária, secundária, terciária) não implica uma ordem estrita de aplicação.

De fato, a fase primária inicia o projeto e o final da fase terciária determina o final da fase de projeto e a passagem à manufatura, mas durante o processo de projeto, todas as fases estão tipicamente ativas, determinando a já discutida natureza iterativa do processo como um todo. Esta idéia fica consubstanciada na Figura 1.11, onde cada fase é vista como uma sucessão de estados, e de cada estado de cada fase se pode passar a qualquer estado de qualquer outra fase.

Procurando aproximar o modelo de Medland dos anteriores, pode-se associar cada um dos estados de cada fase da Figura 1.11 com uma *descrição de projeto* dos modelos de Suzim e de Gajski-Kuhn. Na fase de análise, estes estados (s_1, s_2, etc) poderiam corresponder a descrições comportamentais de saída de ferramentas de análise, tais como simuladores, enquanto que os estados da fase terciária poderiam corresponder a descrições estruturais ou físicas de uma alternativa de projeto.

1.6.3 Projeto, Otimização e Complexidade

Um bom projetista é aquele capaz de elaborar uma descrição detalhada a partir de uma descrição abstrata, de tal forma que a funcionalidade desejada seja aquela do sistema final e que este sistema tenha custo mínimo e desempenho máximo. Assim, projeto *é* otimização. Este é um fato amplamente reconhecido.

Baseado nesta premissa, dois conceitos fundamentais a serem considerados durante o processo de projeto são os de *correção* ou *corretude* e *otimização* ou *otimalidade* do sistema a construir. Uma descrição final é *correta* se ela atende a todos os requisitos da descrição inicial e pode ser fabricada, ou seja, é *viável*. Uma descrição final é *ótima* se ela é correta e possui custo mais baixo e melhor desempenho que qualquer outra solução correta, de acordo com todos os possíveis *critérios de otimalidade*. Exprime-se critérios de otimalidade costumeiramente pelo estabelecimento de *funções objetivo*, maneiras de medir de forma precisa o grau de otimalidade de uma solução correta no atendimento de um (mais freqüentemente) ou de um subconjunto de critérios.

O processo de projeto deve ser guiado, obviamente, pela busca da ou de uma das soluções ótimas. Contudo, na maior parte dos casos, a solução ótima, como desrcita acima não existe, ou alcançá-la é uma tarefa que não compensa, devido ao alto grau de *complexidade* do problema a resolver. Assim, algum compromisso entre custo e desempenho deve ser obtido, normalmente pela valorização de certos critérios de otimalidade com relação a outros critérios considerados como menos importantes.

1.6.3.1 Funções Objetivo para Sistemas Digitais

Alguns dos principais critérios de otimalidade para sistemas digitais são:

- espaço - o sistema digital deve ser o menor possível;

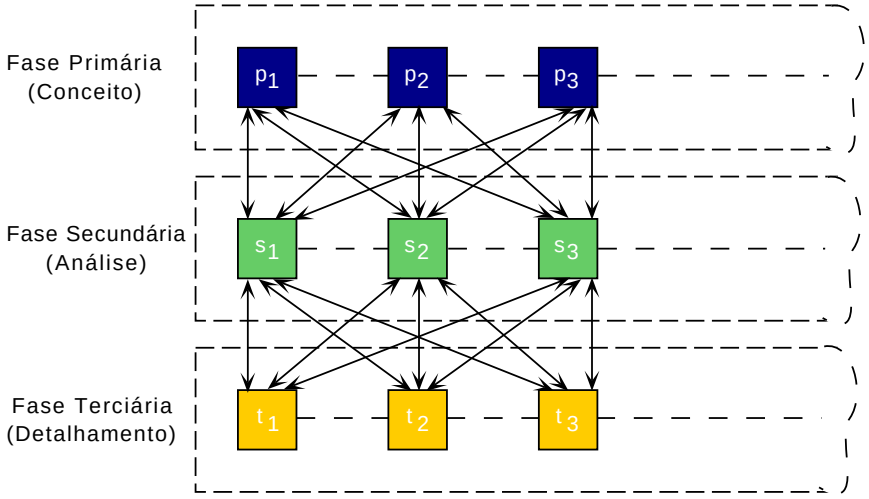


Figura 1.11: Uma visão detalhada do modelo de Medland como um processo de projeto iterativo.

- desempenho - o sistema digital deve ser o mais veloz possível;
- energia - o sistema digital deve consumir o mínimo de energia por unidade de tempo (potência mínima);
- robustez - o sistema digital não deve falhar.
- testabilidade - o sistema digital deve ser fácil de testar;
- custo - o sistema digital deve ser o mais barato possível. Trata-se de um critério especial, pois sua otimização normalmente é função de muitos fatores externos à fase de projeto, ao contrário dos critérios anteriores, eminentemente técnicos.

Como estes critérios são repetidas vezes conflitantes entre si, a cada projeto deve-se favorecer aqueles que concorram mais facilmente à satisfação dos requisitos particulares da descrição inicial. Por exemplo, durante o projeto de um telefone celular ou qualquer aparelho portátil operado a baterias, deve-se favorecer os critérios de espaço e energia, ficando o de tempo em segundo plano. Sendo um produto de consumo, o critério custo é talvez o mais fundamental neste caso. Por outro lado, o projeto de um super-computador deve favorecer em primeiro lugar o critério tempo, ficando os outros em um plano secundário. Por outro lado, aplicações espaciais devem favorecer a satisfação do critério de

robustez em primeiro lugar, sobretudo em se tratando de equipamentos para missões tripuladas.

1.6.4 Fases do Projeto de Sistemas Digitais

As descrições usadas durante o ciclo de projeto de um sistema digital, classificadas segundo o modelo bidimensional do diagrama Y, podem ser agrupadas de acordo com a similaridade das técnicas e da teoria subjacente empregadas na sua manipulação. Deste agrupamento surgem as diferentes disciplinas que compõem o espectro de projeto de sistemas digitais. Dentre estas, as mais proeminentes na atualidade são:

- *projeto sistêmico* - manipula descrições com altíssimo nível de abstração, contemplando técnicas tais como projeto integrado de software e hardware [18] e cossimulação, que permitem avaliar, por exemplo, os compromissos envolvidos na implementação de parte do sistema em hardware ou em software e o particionamento do sistema em diversos módulos de hardware cooperantes;

- *projeto comportamental*, *algorítmico* ou *de alto nível* - trabalha com descrições de nível de abstração ainda bastante alto, envolvendo técnicas tais como a alocação otimizada de recursos de hardware para a execução das funcionalidades desejadas, o escalonamento das tarefas no tempo de forma a acelerar a sistema resultante, a implementação otimizada de unidades de transformação de dados (*data-path*) e a simulação comportamental a partir de linguagens de descrição de "hardware" (*Hardware Description Languages* ou HDLs);

- *projeto lógico* - compreende descrições mais detalhadas que os itens anteriores, mas ainda abstratas o suficiente para serem consideradas independentes da tecnologia de fabricação subjacente, e inclui técnicas como a tradução de especificações funcionais de unidades de controle em portas lógicas e a minimização do número destas e do número de elementos biestáveis necessários, bem como sua correta localização, de forma a acelerar o sistema como um todo ou a reduzir a potência dissipada;

- *projeto físico* - promove a geração do conjunto completo de informações necessárias à construção do sistema digital, contando com técnicas tais como a determinação da planta baixa do sistema digital e da geometria das interconexões físicas entre os diferentes módulos, a tradução de descrições lógicas em descrições geométricas (máscaras de um IC) ou o mapeamento destas para um arquivo de configuração binária para dispositivos programáveis, a caracterização elétrica do sistema e a simulação de temporização.

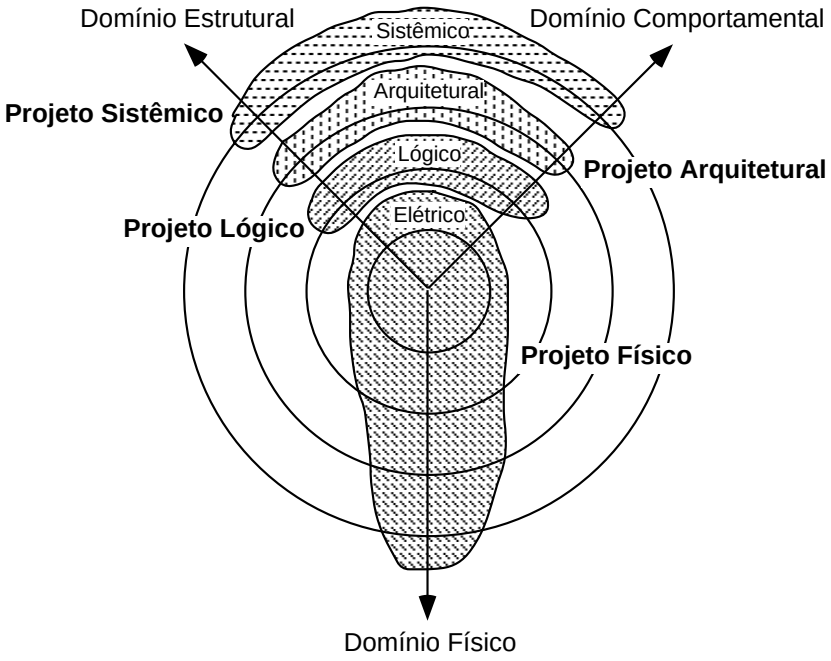


Figura 1.12: Fases do projeto de sistemas digitais sobre o diagrama Y.

A subdivisão em disciplinas de projeto pode variar de autor para autor. Por exemplo, é comum empregar nomenclaturas que incluem o *projeto no nível de chaves lógicas* [32], que se situa entre os níveis lógico e elétrico e o *projeto no nível de transferência entre registradores*, localizado entre os níveis algorítmico e lógico. Além destes, identifica-se em alguns casos um nível exclusivo para o *mapeamento tecnológico*, responsável pela transformação de descrições lógicas abstratas em descrições lógicas associadas a células de biblioteca específicas para um tipo de tecnologia de fabricação, tal como standard-cell ou gate-array [136].

Cada uma das disciplinas de projeto compõe-se de um conjunto de operações, cada uma classificável segundo a apresentação da Seção 1.6.2.1 como de síntese, extração, validação ou otimização. A Figura 1.12 ilustra as diferentes fases do projeto de sistemas digitais como regiões hachuradas no diagrama Y, compreendendo descrições e ferramentas de projeto contidas nas respectivas regiões.

1.7 Projeto Auxiliado por Computador

Já foi mencionado no Prefácio que sistemas de projeto auxiliado por computador para sistemas digitais, também denominados sistemas de CAD eletrônico são produtos caros e complexos, sendo compostos de grande quantidade de programas. A complexidade do projeto de hardware é a justificativa principal para o estudo de modelos de representação do processo de projeto empreendido na Seção 1.6.2. Nesta Seção, procura-se dar uma idéia da estrutura e composição geral de tais sistemas em suas versões modernas. Discute-se a estrutura geral de sistemas de sistemas de CAD eletrônico, na Seção 1.7.1. Segue-se uma comparação entre sistemas de CAD comerciais e acadêmicos ilustrada com exemplos, na Seção 1.7.2.

1.7.1 Estrutura Geral de Sistemas de CAD Eletrônico

A Figura 1.13 mostra de maneira simplificada a estrutura interna de um sistema de CAD eletrônico típico.

A interface gráfico-textual possui função óbvia. As descrições de projeto são manipuladas pelos programas, que implementam os métodos de projeto, cujos papéis foram discutidos durante a apresentação dos modelos para o processo de projeto, na Seção 1.6.2. As bibliotecas são repositórios contendo módulos pré-prontos para aumentar o grau de abstração de projeto e possibilitar o reuso de módulos já implementados.

Finalmente, nota-se na Figura a existência de um bloco denominado Arcabouço de Projeto, em inglês. CAD *Framework* [17]. A existência deste bloco é motivada pela necessidade de gerenciar a complexidade do projeto de sistemas digitais de maneira sistemática. A organização pública CAD Framework Initiative (CFI), responsável por definir padrões para estas partes de sistemas de CAD [170], define arcabouços de projeto como "... uma infraestrutura de software que provê um ambiente operacional unificado para ferramentas de CAD".

Mais especificamente, arcabouços de projeto devem prover serviços de gerenciamento de dados e gerenciamento do fluxo de projeto. O gerenciamento de dados é realizado pela organização das informações de projeto em um banco de dados de projeto, desenvolvido especialmente para dar suporte ao tipo de operações e informações envolvidas. Operações realizadas pelo arcabouço para gerenciar os dados de projeto incluem controle de versões, controle de visões[9], controle da hierarquia de projeto, e controle de acesso (quando o projeto é particionado entre equipes de projetistas com diferentes responsabilidades). O gerenciamento do fluxo de projeto [176] envolve operações como controle de atendimento das restrições (pelas descrições geradas), assistência (por exemplo

[9] *Visões* são descrições distintas de um mesmo projeto, tal como um diagrama de portas lógicas e uma tabela verdade com a mesma funcionalidade

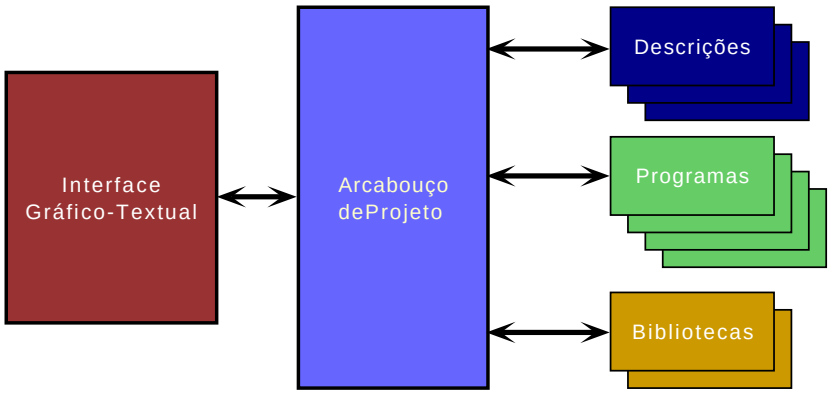


Figura 1.13: A estrutura interna genérica típica de um sistema de CAD eletrônico.

informando ao projetista que operações são possíveis ou aconselháveis a cada momento, ou controlando a parametrização dos programas), automação (executando uma seqüência de programas sob comando do projetista) e controle de andamento do projeto (identificando que operações já foram realizadas, quais falta realizar e determinando o próprio fluxo de operações).

Arcabouços de projeto para CAD formam uma área de pesquisa e desenvolvimento ativa e complexa. Uma discussão mais ampla do assunto está fora do escopo desta obra, e pode ser encontrada na bibliografia citada.

Algumas observações sobre a estrutura geral de CAD eletrônico são necessárias. Primeiro, as descrições de projeto manipuladas pelo CAD costumam apresentar-se em forma textual, o que facilita sua manipulação por seres humanos e facilita o intercâmbio de informações de projeto entre diferentes sistemas. De fato, existem padrões textuais definidos com o fim primário de propiciar o intercâmbio de informações de projeto em diversos níveis de abstração, exemplo do qual é a liguagem EDIF [53]. Segundo, a interface gráfica e o arcabouço de projeto normalmente provêem uma visão macroscópica e unificada do processo de projeto, agrupando procedimentos e gerenciando a informação de forma automatizada.

1.7.2 CAD Acadêmico versus CAD Comercial

Existe hoje abundância de sistemas de CAD de origem acadêmica e comercial. Contudo, várias diferenças fundamentais existem entre sistemas acadêmicos e comerciais.

Primeiro, sistemas acadêmicos costumam ser *abertos*, em vários sentidos.

Muitos destes são fornecidos com código fonte dos programas, possibilitando a total personalização do sistema. As ferramentas e bibliotecas usam descrições intermediárias documentadas e empregam formatos exclusivamente textuais. O usuário pode optar pelo uso da interface gráfico-textual e do arcabouço de projeto ou por controlar pessoalmente o fluxo de projeto, executando separadamente cada programa e gerenciando os dados de forma manual. Sistemas comerciais jamais fornecem o código fonte, pois este é a propriedade intelectual fonte do lucro da empresa. Assim, o usuário do CAD limita-se a equipamentos e configurações de software oferecidas pela empresa que implementa e mantém o sistema. As descrições intermediárias, embora quase sempre textuais, não são totalmente documentadas, pois o formato, muitas vezes proprietário, pode conter informações que a empresa não tem interesse em divulgar amplamente, principalmente nos níveis inferiores de abstração, onde informações de caráter estratégico do fabricante do circuito podem estar embutidas. Bibliotecas quase sempre são fornecidas em formato binário não-documentado. Normalmente o usuário não tem acesso facilitado para eliminar o uso do arcabouço de projeto e/ou a interface gráfica em sistemas comerciais, e quando o tem, falta informação de como fazê-lo adequadamente.

Segundo, o nível de acoplamento entre programas em sistemas acadêmicos é baixo, dificultando o fluxo de projeto. Isto resulta do fato destes sistemas derivarem em geral de trabalhos desenvolvidos por diferentes pessoas ao longo do tempo, cada uma com um objetivo específico em mente (tal como obter um título de mestre ou doutor), sem um grande controle sobre os objetivos do sistema como um todo. O contrário ocorre com sistemas comerciais, onde a integração do processo de projeto é um dos mais fortes argumentos de venda do produto.

Terceiro, sistemas acadêmicos costumam estar no estado da arte em termos de desempenho e qualidade dos resultados, pois os programas destes são resultado de pesquisa acadêmica, e o fato dos sistemas serem abertos permite que a comunidade acadêmica mundial tenha como comparar diferentes ferramentas de projeto que procuram resolver o mesmo problema, selecionando as de melhores resultados para compor o sistema próprio. Esta forma de construção de um sistema de CAD é normalmente inviável com sistemas comerciais, devido a considerções de custo e compatibilidade. A competição ferrenha entre empresas de CAD leva, muitas vezes, a comparações tendenciosas de desempenho entre competidores, além do segredo industrial impedir o compartilhamento de informações de diferentes sistemas entre fabricantes. Além destes pontos, a necessidade de manter um suporte constante e bem treinado aos usuários de sistemas comerciais reduz o ritmo de mudanças nestes, impedindo o acompanhamento imediato do estado da arte em desempenho e qualidade de resultados.

Finalmente, sistemas acadêmicos possuem alto grau de controlabilidade e observabilidade o que não ocorre com sistemas comeciais. Considere-se por

exemplo a síntese lógica. Enquanto que em um sistema acadêmico seria possível executar passo a passo transformações tais como conversão de um grafo de transição de estados para um grafo equivalente reduzido, codificação de estados, entradas e saídas, minimização lógica e mapeamento tecnológico, um sistema comercial tipicamente executaria todas estas tarefas em um único passo, sendo difícil medir a qualidade de cada ferramenta individual e mesmo saber se algum passo foi executado ou não, caso este seja opcional. Um exemplo desta última situação é a minimização de estados de máquinas estados finitas. Até o início da década de 90, os sistemas de CAD comerciais e acadêmicos ignoravam este passo opcional. A partir de trabalhos como o de Hachtel e colaboradores em [89], passou-se a agregar estes procedimentos em sistemas acadêmicos.

Muitos dos sistemas acadêmicos são publicamente disponíveis, e os comerciais podem ser obtidos a um custo nulo ou até duas ordens de grandeza abaixo do preço de mercado para uso em instituições de ensino. Isto pode significar que uma licença de um sistema de US$500.000,00 pode ser oferecida por US$5.000,00 para Universidades.

Um primeiro exemplo, talvez o mais conhecido e desenvolvido sistema acadêmico disponível sem custos é o sistema OCTTOOLS da Universidade de Berkeley na Califórnia. OCTTOOLS possui subsistemas de projeto físico e projeto lógico, um banco de dados de projeto e algum suporte para síntese arquitetural. Embora OCTTOOLS não seja totalmente domínio público (ele pode ser solicitado via Internet ao *Industrial Liaison Program*, da Universidade de Berkeley, a partir da página **http://hera.eecs.berkeley.edu/ software/index.html**), partes dele o são, tal o como o subsistema de síntese lógica MIS[26]. Mais recente é a versão seqüencial de MIS, SIS[164], disponível via Internet em **http://www-cad.eecs.berkeley.edu:80/Software/Software.html**. Também em Berkeley, foram desenvolvidos o subsistema de verificação formal VIS[188], e o subsistema de projeto integrado de software e hardware POLIS [16], igualmente disponíveis de forma gratuita a partir da última página citada. Para a síntese arquitetural, existe o subsistema Olympus [70], disponível via Internet em

http://akebono.stanford.edu/users/cad/synthesis/olympus/,

desenvolvido na Universidade de Stanford, Califórnia. Vários sistemas acadêmicos e programas isolados voltados para o projeto de sistemas assíncronos estão disponíveis gratuitamente a partir de

http://www.cs.man.ac.uk/amulet/async/async_tools.html.

Sistemas comerciais de CAD existem em grande número, sendo alguns vinculados a fabricantes de circuitos, como o sistema BOOLEDOZER da IBM, mas em sua maioria o mercado baseia-se em software independente de fabricante comercializado por *software houses* especializadas em CAD eletrônico, tais como

Cadence, Synopsis, Viewlogic e Menthor-Graphics.

1.8 Escopo da Obra

A partir das informações contidas nas Seções 1.3 a 1.7, pode-se definir tanto o tema como o escopo de abrangência do livro proposto.

Costuma-se descrever o comportamento de blocos de lógica combinacional através de formalismos da *álgebra de chaveamento* [61], uma classe de *álgebra Booleana*, enquanto que representam-se blocos de lógica seqüencial através da teoria de *estruturas de transição de estados finitas* [90, 115], que define modelos tais como *autômato finito* e *máquina de estados finita*.

O projeto lógico de sistemas digitais admite como entrada descrições estruturais, comportamentais ou combinações destas, como percebe-se na Figura 1.12. Exemplo das primeiras são as descrições de blocos de lógica combinacional intercalados com elementos de memória (registradores). Exemplos da segunda são as tabelas e os grafos de transição de estados de controladores e equações Booleanas para descrever blocos combinacionais.

A saída da fase de projeto lógico, por outro lado, consiste em geral de uma descrição estrutural, caracterizada como uma interconexão de portas lógicas e, eventualmente, elementos de memória biestáveis. A presença de biestáveis implica com freqüência ser o sistema digital em questão seqüencial, mas sua ausência não basta para caracterizar o sistema como combinacional. Sistemas digitais seqüenciais síncronos costumam empregar representações onde a natureza seqüencial do circuito está concentrada nos elementos biestáveis. Sistemas digitais seqüenciais também são caracterizados pela existência de laços de realimentação, como visto na Seção 1.4.1, embora sua existência seja uma condição necessária, mas não suficiente para garantir o comportamento seqüencial, segundo a discussão da Seção 1.4.2. Sistemas digitais seqüenciais *assíncronos* [182] costumam empregar representações onde a seqüencialidade no nível lógico é explicitamente determinada pelos laços de realimentação.

Qualquer bloco de lógica combinacional pode ser encarado como uma *máquina de estados finita* com apenas um estado, conforme a Definição 1.3 e [40]. Com raríssimas exceções, sistemas digitais práticos só podem ser considerados do ponto de vista externo como um bloco seqüencial com mais de um estado, inviáveis de serem tratados unicamente via técnicas de projeto de blocos combinacionais. Estes fatos *per se*, poderiam levar à conclusão errônea de que projeto lógico e projeto lógico seqüencial são sinônimos. Dois fatos atestam contra esta conclusão:

- qualquer bloco seqüencial pode ser representado como um bloco combinacional e um vetor de elementos de memória (o chamado *vetor de estado*) externo a este, o último normalmente de implementação trivial [40];

- as técnicas de projeto combinacional possuem com freqüência uma complexidade inferior às das técnicas de projeto seqüencial correspondentes, e são melhor dominadas por projetistas.

O primeiro fato demonstra que o problema de projeto de um sistema digital pode, em qualquer caso, ser reduzido ao problema de implementar um bloco combinacional e um vetor de elementos de memória. O segundo constitui uma justificativa de cunho prático para realizar esta redução sempre que possível. Contudo, esta possibilidade não permite descartar completamente as técnicas de projeto seqüencial. Primeiro, a existência do vetor de estado no sistema deve ser usada para conduzir a implementação do bloco combinacional, de forma a otimizá-lo ao máximo. A análise das características seqüenciais do sistema digital deve produzir um conjunto de *restrições* que a implementação do bloco combinacional deve obedecer, de forma a atingir valores ótimos para as figuras de mérito de projeto. Isto ocorre, por exemplo, nos problemas de minimização de estados e de codificação ótima de entradas, saídas e estados. Segundo, o modelo *bloco combinacional + vetor de estado* nem sempre permite explorar todas as possibilidades de otimização de descrições estruturais, tais como a retemporização de registradores [120, 121]. Como resultado, duas áreas distintas formam o que denomina-se de disciplina de projeto lógico: o *projeto lógico combinacional* e o *projeto lógico seqüencial*.

O tema do presente livro é a exploração da área de projeto lógico seqüencial, com ênfase nas atividades de síntese, otimização e análise. No nível lógico de abstração, a possibilidade de mapeamento direto entre descrições estruturais e comportamentais [136] faz com que o termo síntese seja indissociável do termo otimização, conforme discussão da Seção 1.6.3. Circuitos e sistemas síncronos têm sido o principal foco de pesquisa e desenvolvimento nas últimas três décadas. Contudo, sistemas assíncronos têm despertado renovado interesse de projetistas, dado seu crescente emprego em aplicações que exigem alto desempenho, tais como redes de alta velocidade, processadores construídos com *dispositivos de picossegundo* [143], e protocolos de comunicação [133], e às limitações dos métodos síncronos em atender requisitos gerados pelo aumento da escala de integração. Devido a esta tendência, é dada aqui atenção ao estudo de técnicas de projeto para sistemas seqüenciais síncronos e para sistemas seqüenciais assíncronos.

Após especificar sobre o que esta obra trata, vale a pena salientar alguns assuntos sobre os quais ela *não* trata. Embora os modelos adiantados aqui sobre sistemas digitais sejam gerais, modelos menos gerais podem ser muito mais úteis em contextos específicos. O caso mais relevante ocorre quando a informação a ser manipulada pelo sistema possui um alto grau de estruturação, como por exemplo números manipulados por operações aritméticas. A manipulação deste tipo de informação está fora do escopo da presente obra. Embora esta decisão possa parecer surpreendente para noviços na área, é necessário salientar

que trata-se de uma posição natural, pois a manipulação de informação não-estruturada, muitas vezes associada ao controle do sistema digital é a de mais difícil tratamento automatizado, e assim a que consome maior quantidade de recursos durante o projeto do sistema. Todo um corpo particular de técnicas é paralelamente desenvolvido para lidar com informação estruturada em projeto VLSI. Para exemplos, o leitor pode consultar obras como os livros editados por Swartzlander [173, 174].

Finalmente, adianta-se duas observações pertinentes sobre o termo *automatizado* no título desta obra. Primeiro, enfatiza-se a discussão de técnicas de projeto voltadas para a implementação de algoritmos computacionais eficientes, por oposição a técnicas de projeto manuais em uso por projetistas humanos (ou seja, projeto automatizado aqui significa projeto não-manual). Segundo, *projeto automatizado* pressupõe interação entre ferramentas computacionais e o projetista, por oposição ao projeto *automático*, onde há pouca ou nenhuma interação entre a ferramenta e o usuário. Destaca-se a necessidade da interação para garantir que o espaço de soluções seja melhor explorado, a partir do emprego iterativo das ferramentas computacionais de projeto com sucessivas e distintas parametrizações (ou seja, projeto automatizado significa projeto não-automático).

1.9 Conclusões do Capítulo

Como se pode depreender do exposto neste Capítulo, a implementação de sistemas digitais VLSI é uma atividade de alto grau de complexidade, a ponto de justificar a escrita de um livro como o presente, versando sobre apenas uma das fases de sua implementação (a fase de projeto), sobre apenas uma das disciplinas de projeto (projeto lógico), e sobre apenas uma das duas classes de sistemas que compõem esta disciplina (sistemas seqüenciais). Ainda assim, apenas uma pequena parte dos métodos atuais de projeto poderá ser abordada, dada a vastidão das proposições disponíveis e a quantidade de subproblemas a resolver.

1.10 Perspectiva Histórica e Bibliografia

Pode-se considerar que a era de projeto e fabricação de sistemas digitais modernos iniciou com a invenção do transistor eletrônico, devida a Bardeen, Brattain e Shockley, em 1947/48 [153, 154], após séculos de evolução de dispositivos computacionais operados manualmente, tais como o ábaco chinês e o quipu inca [97], e dispositivos mecânicos como as máquinas de diferenças e analítica, devidas ao cientista inglês Charles Babbage. O transistor foi o principal avanço tecnológico responsável por suplantar os relés eletromecânicos e as válvulas

eletrônicas, tecnologias então usadas na confecção de sistemas digitais.

O final da década de 50 viu o surgimento do circuito integrado, e as décadas de décadas de 60, 70 e 80 levaram ao aperfeiçoamento da tecnologia de integração, passando pela diversas gerações de ICs: SSI, MSI, LSI e VLSI. A década de 90 tem se caracterizado por dispositivos integrados com um número de transistores que excede o milhão, entre estes as memórias e processadores. Recentemente, a Associação de Indústrias de Semicondutores dos EUA previu que em torno do ano 2010, estarão sendo fabricados industrialmente processadores com 800 milhões de transistores, com milhares de pinos de entrada e saída, um barramento de dados de 1000 bits, e freqüências de chaveamento em torno de 2GHz [163]. Deve-se comparar estes valores com o estado da arte atual, de 10 a 20 milhões de componentes, centenas de pinos, barramentos de 64 bits e freqüências de 700MHz. Claramente, nem todos os parâmetros de projeto avançam no mesmo passo, o que deve paulatinamente mudar os compromissos de projeto. O desenvolvimento de cada sistema com estas características é uma tarefa que envolve equipes com *centenas* de projetistas [37].

A base teórica do projeto de sistemas digitais é bem mais antiga que o desenvolvimento da tecnologia de dispositivos eletrônicos a partir da física do estado sólido. Blaise Pascal estudou a construção de aparatos mecânicos de cálculo em 1640, base esta usada em 1800 por Babbage para construir suas máquinas. Ainda no século XIX, o matemático Georg Boole desenvolveu a hoje denominada álgebra Booleana e seu caso especial, a álgebra de chaveamento. A manipulação e a transmissão de informação sob forma discreta deve muito aos trabalhos de Nyquist em 1928 e Shannon em 1938 e 1948 [61]. Em particular, os métodos de projeto de sistemas digitais seqüenciais têm como pioneiros cientistas como Mealy [134] e Moore [138], que estabeleceram os primeiros modelos gerais para representar o comportamento seqüencial de sistemas digitais. Igualmente digno de nota são os trabalhos seminais de Huffman, sobre a síntese de sistemas digitais seqüenciais [104, 105].

Somente em passado recente foram desenvolvidos os modelos para representar o processo de projeto de sistemas digitais, quando a complexidade da tarefa de projeto começou a exigir a organização sistemática de métodos e ferramentas. O diagrama Y foi sugerido em 1983, e os outros modelos descritos são ainda mais recentes.

Os primeiros esforços na área de síntese lógica no campo teórico ocorreram antes do advento da integração em alta escala [66]. Assim, sua importância foi sobretudo na formulação dos problemas fundamentais e em técnicas de solução gerais, pois as funções objetivo empregadas nos trabalhos seminais têm com freqüência pouca relação com a tecnologia atual. O problema maior destes métodos acaba sendo a inadequação para lidar com o tamanho dos problemas a resolver.

O projeto manual no nível físico de abstração dominou a panorama de es-

força de projeto de sistemas digitais nos anos 60. Na década seguinte, floresceu a automatização das tarefas de posicionamento e traçado de rotas em ICs. Além destas primeiras ferramentas de CAD automatizado, o advento de tecnologias para obter imagens gráficas coloridas de alta resolução popularizou ferramentas de CAD manuais, tais como editores de layout de ICs. Uma boa obra de referência para ferramentas de síntese física é o livro de Preas e Lorenzetti [152].

Ainda nos anos 70, foi introduzido o emprego da teoria da intratabilidade de problemas [82] no estudo de questões de projeto de sistemas digitais, comprovando-se a necessidade de procurar não soluções exatas para os problemas, mas buscar soluções razoáveis obtidas em prazo curto, através do extenso emprego de heurísticas. Uma vez o problema de síntese física contando com um corpo de técnicas adequadas, a ênfase em pesquisa mudou para o projeto lógico. A motivação principal foi superar a geração manual do layout de ICs, altamente propensa ao erro humano. Os principais frutos desta fase foram as ferramentas de otimização lógica dois níveis, aplicáveis à síntese de PLAs (do inglês, *Programmable Logic Arrays*), tais como os minimizadores heurísticos MINI, desenvolvido pela IBM [101] e ESPRESSO, desenvolvido na Universidade de Berkeley [23].

Os anos 80 viram o advento de sistemas de síntese independentes de fabricantes e ferramentas automatizadas endereçando o nível arquitetural de abstração, tais como as que realizam a alocação de operadores e o escalonamento de tarefas e o início do uso extenso de HDLs. Juntamente, surgiram os geradores parametrizáveis de módulos complexos tais como RAM, ROM e Unidades Lógico-Aritméticas, como forma de acelerar a produção automatizada de layout para blocos regulares. Os anos 80 foram os anos da maturação da tecnologia de fabricação de circuitos VLSI [66]. A pesquisa conseguiu alcançar relativo avanço na configuração automatizada de circuitos lógicos multinível, adequados para estilos de projeto pré-caracterizados.

Os anos 90 poderiam inicialmente ser pensados como o ano da maturação das técnicas de projeto VLSI, mas uma série de fatos relacionados à evolução das tecnologias de fabricação mudou esta expectativa:

1. a progressiva importância de dispositivos portáteis tais como telefones celulares, computadores do tipo *laptop* e *pagers* provocou uma mudança igualmente progressiva no valor relativo das funções objetivo empregadas para medir a qualidade de implementações de sistemas digitais;

2. a comunidade de pesquisa em VLSI começa a se render ao fato de que o crescimento da escala de integração em breve inviabilizará o projeto de sistemas digitais totalmente síncronos como se conhece hoje;

3. a importância de dispositivos personalizáveis pós-fabricação, tais como FPGAs e CPLDs com relação a dispositivos personalizáveis por fabricação

vem crescendo dia a dia, devido ao avanço da tecnologia e das técnicas de projeto destes dispositivos.

O resultado líquido do primeiro item é a necessidade de revisão das técnicas de projeto e ferramentas para contemplar a redução de potência dissipada, ao invés de redução de área e/ou aumento da velocidade do dispositivo, gerando toda uma nova classe de algoritmos de síntese para baixa potência (em inglês, *low power*). O segundo item gerou um renovado interesse em projeto de circuitos assíncronos e sistemas digitais com múltiplos relógios desde o final da década de 80. Novamente, um corpo completamente distinto de técnicas de projeto de sistemas digitais ressurge a partir de técnicas que datam dos anos 60 [182], quando ainda não havia sido plenamente estabelecida a enorme vantagem de emprego de sistemas síncronos sobre assíncronos para a tecnologia VLSI da época. Finalmente, o advento de dispositivos personalizáveis no campo (sobretudo os FPGAs baseados em RAM) vem conseguindo atender o requisito de tempo de chegada ao mercado, em muitos casos com vantagens significativas sobre tecnologias personalizáveis por fabricação, como gate-array ou standard-cell. Além disto, a tecnologia de personalização de FPGAs avançou ao ponto de permitir a personalização parcial dos dispositivos *durante o funcionamento*, habilitando a consideração de métodos que prometem ser revolucionários, denominados genericamente de *computação reconfigurável* (do inglês, *reconfigurable computing*). Em resumo, a evolução da tecnologia vem obrigando, nesta década, a revisão constante dos métodos e ferramentas de projeto VLSI, em todos os níveis de abstração de projeto: físico, lógico, arquitetural e sistêmico.

1.11 Resumo do Capítulo

Após definir sistema digital, discutiu-se aqui o processo de projeto e fabricação de sistemas digiais e sua relação. Caracterizou-se sistemas digitais combinacionais e seqüenciais, com base na idéia de armazenamento de informação, geradora do conceito de estado. Após estas definições e difereciações, foi apresentada uma discussão de taxonomias para sistemas digitais, estabelecendo um conjunto relevante de critérios gerais de classificação, discutindo a ortogonalidade dos critérios e apresentando um exemplo de classificação. Em seguida, o processo de projeto de sistemas digitais foi apresentado em detalhe e estudou-se três modelos para representá-lo. Os modelos possuem diferentes capacidades de representação e empregam diferentes critérios para representar o processo de projeto. Com base em um dos modelos, o diagrama Y, foi sugerido um conjunto de fases distintas de projeto, uma forma de classificar conjuntos de descrições e ferramentas de transformação destas. A fase de projeto lógico foi definida como o objeto de interesse desta obra. Finalmente, caracterizou-se sistemas de projeto auxiliado por computador e estabeleceu-se o escopo do presente livro.