Capítulo 5

Síntese Assíncrona

Hamlet - ...
There are more things in heaven and earth, Horatio,
Than are dreamt of in your philosophy.
Tradução:
Hamlet - ...
Há mais coisas no céu e na terra, Horácio,
Que as sonhadas em sua filosofia.
William Shakespeare, Hamlet. I.5-165–167.

Não seria exagero afirmar que a maioria esmagadora dos projetos, projetistas, e sistemas de projeto de sistemas digitais VLSI da atualidade lidam tão somente com o paradigma síncrono, tendo pouca familiaridade, se alguma, com o paradigma assíncrono de projeto.

A diferença primordial entre sistemas síncronos e assíncronos foi introduzida ao final da Seção 1.5, junto com a discussão de critérios de classificação gerais de sistemas digitais. Lá, estabeleceu-se que a forma de tratamento da passagem do tempo é a base desta distinção. Sistemas digitais síncronos pressupõem um *modelo discreto de tempo*, enquanto sistemas digitais assíncronos assumem um *modelo contínuo de tempo*. Ambos partilham um *modelo discreto de sinais* (entradas e saídas), a característica distintiva de sistemas digitais. A diferenciação conceitual, embora simplesmente enunciável, possui profundas implicações no estilo de projeto adotado. Ela dirige qualquer proposta de métodos de síntese e de métodos de análise para estas classes de sistemas.

A primeira observação importante a ser feita sobre a distinção entre sistemas síncronos e assíncronos é que não se trata de *dois* estilos de projeto distintos. O projeto síncrono pode efetivamente ser considerado como *um* estilo de projeto, nos termos da definição fornecida na Seção 1.3, ou seja, dada uma tecnologia de implementação, o conjunto de métodos síncronos de projeto pode

ser considerado como único. Entretanto, ao assumir que um dado sistema digital não parte do pressuposto de tempo discretizado, uma grande restrição aos métodos de projeto é eliminada. O conseqüente aumento do número de graus de liberdade para implementar o sistema digital torna o espaço de soluções por demais amplo. Logo, costuma-se adicionar outras restrições para assim definir um estilo de projeto. Como não é consenso quais restrições aplicar neste caso, o termo *projeto assíncrono* engloba uma família de estilos de projeto, nominalmente um para cada conjunto de restrições impostas, em substituição ao tempo discreto. Pode-se categoricamente afirmar que nenhum destes alcançou até hoje o grau de desenvolvimento do estilo síncrono de projeto, mas vários estilos assíncronos são hoje empregados em aplicações específicas.

Apenas para dar uma idéia do estado de avanço atual dos estilos assíncronos de projeto, tome-se o levantamento realizado por Werner e Akella, de todos os processadores assíncronos já implementados após a retomada da pesquisa e desenvolvimento desta área, ao final da década de 80 [189]. São seis processadores, todos oriundos de esforços acadêmicos, e apenas um deles contando com uma versão comercial, a versão assíncrona da arquitetura comercial ARM [79]. Contudo, conforme documentado em [80], esta implementação já apresenta pelo menos uma figura de mérito com vantagem clara sobre uma versão síncrona equivalente em termos tecnológicos, a taxa de desempenho por potência dissipada, medida em MIPS/Watt (do inglês, Millions of Instructions Per Second).

Os objetivos deste Capítulo são:

1. analisar as conseqüências de eliminar a restrição de tempo discreto sobre métodos de projeto de sistemas digitais;

2. introduzir as principais considerações de projeto oriundas desta eliminação;

3. apresentar um exemplo de método de projeto assíncrono ilustrando o estilo de projeto.

No intuito de atender estes objetivos, o restante deste Capítulo inicia com uma discussão das vantagens e inconvenientes do estilo síncrono de projeto, na Seção 5.1. Os estilos de projeto assíncronos justificam-se na medida em que permitem a superação dos inconvenientes do estilo síncrono sem perda significativa das vantagens intrínsecas associadas ao último. O segundo objetivo pode ser alcançado pela discussão das formas de modelagem de fenômenos internos e externos ao sistema durante o projeto assíncrono, bem como dos problemas gerados pela não-discretização do tempo. Este é o tema da Seção 5.2. Segue-se a Seção 5.3, onde os modelos temporais, juntamente com técnicas de codificação são usados para definir critérios de classificação de sistemas e métodos de projetos, gerando uma taxonomia assíncrona. O problema mais importante gerado pela não-discretização do tempo em sistemas digitais é o controle de

eventos transitórios gerados nas saídas de sistemas digitais e aparecendo nas entradas dos mesmos. A Seção 5.4 explora a caracterização e classificações para este importante problema. A Seção 5.5 discute em algum detalhe um estilo de projeto assíncrono moderno. As três seções finais apresentam, como é praxe, conclusões, a perspectiva histórica e o resumo do Capítulo.

5.1 Síncronos: Vantagens e Inconvenientes

Embora a definição de sistemas digitais síncronos e assíncronos fornecida na Página 17 seja conceitualmente suficiente e completa, vale a pena introduzir uma definição estrutural dos circuitos digitais, mais instrumental para os assuntos a serem tratados nesta Seção. Note-se que na definição estrutural abaixo, um circuito digital síncrono nada mais é que um caso especial de circuito assíncrono. A definição a seguir foi extraída do livro de Lavagno e Sangiovanni-Vincentelli [117]. O termo circuito foi usado para manter a forma da definição original, sendo intercambiável no que segue pelo termo sistema. A definição é informal.

Definição 5.1 (Circuitos síncronos e assíncronos - visão estrutural) *Um* ***circuito digital assíncrono*** *é uma interconexão arbitrária de portas lógicas, com a restrição de que nenhum par de saídas de portas lógicas conecta-se entre si. Um* ***circuito digital síncrono*** *é um circuito digital assíncrono onde todos os laços de realimentação passam através de um elemento de memória controlado por um sinal de relógio do circuito. O comportamento do relógio deve ser tal que nenhum evento pode se propagar ao longo de um laço de realimentação sem "parar" em algum ponto deste pelo fato de encontrar um elemento de memória inativo, que o impede de ir adiante. Além disso, nenhum evento deve poder atingir a entrada de um elemento de memória fora de certas janelas de tempo pré-especificadas antes e depois de uma transição do relógio, denominadas* ***tempo de estabelecimento*** *(do inglês,* setup time*) e* ***tempo de manutenção*** *(do inglês,* hold time*), respectivamente. Isto vale tanto para o ambiente onde opera o sistema (que controla as entradas externas), quanto para os circuitos combinacionais internos percorridos pelos sinais.*

Note-se que a definição acima impõe restrições sobre a estrutura interna para circuitos assíncronos. Para circuitos síncronos, ela impõe restrições sobre a estrutura interna *e* sobre o ambiente onde o sistema opera. Os estilos de projeto assíncronos específicos inserem restrições adicionais sobre o ambiente e sobre a estrutura interna. Para ilustrar a Definição 5.1, a Figura 5.1 mostra um diagrama de blocos estrutural típico de um sistema digital síncrono. Note-se que qualquer dos laços de realimentação existentes na Figura depende do sinal de relógio para ser fechado, ou seja, os registradores controlados pelo relógio são *barreiras temporais* para os blocos combinacionais. Por outro lado,

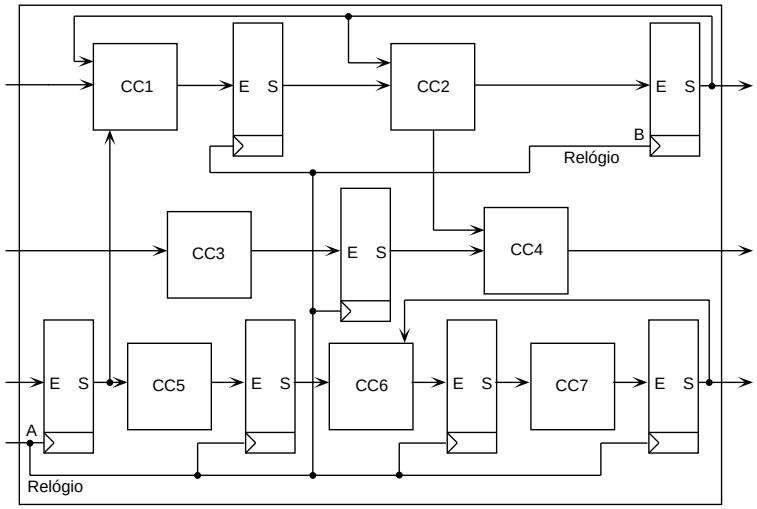


Figura 5.1: Um diagrama de blocos típico de um sistema digital síncrono. Entradas estão à direita e saídas à esquerda. Quadrados representam circuitos combinacionais, retângulos estreitos representam vetores de memória (registradores) de comprimento variável não explicitado. As conexões mostradas são conjuntos de fios, de cardinalidade variável e igualmente não explicitada. O sistema possui um relógio único, que comanda todos os vetores de memória.

a Figura 5.2 mostra exemplos das restrições que o sinais devem respeitar em relação ao relógio em um sistema síncrono, e dois exemplos de violação.

A partir da Figura 5.1, fica claro que o sinal de relógio é onipresente em um sistema síncrono. Onde existir um elemento de memória, deve chegar o relógio. Desta característica, derivam todas as virtudes e todos os defeitos do estilo síncrono, que são discutidos a seguir, com base no artigo de Hauck [93].

Uma primeira e a grande vantagem de sistemas síncronos é a abstração do tempo de propagação dos sinais durante o projeto. Cada bloco funcional, prefixado por CC na Figura 5.1, é elaborado a partir de sua relação entre entradas e saídas como um circuito combinacional, empregando técnicas derivadas unicamente da álgebra Booleana. A seguir, rodeia-se cada bloco de elementos de memória que o isolem do resto do sistema, enquanto o relógio está desativado. Uma segunda vantagem, decorrente da primeira, é que o projeto de sistemas digitais síncronos é fácilmente decomposto. A escolha do período do relógio é feita de forma que este seja o mais curto possível, contudo mais longo que o tempo necessário para permitir que a saída do bloco combinacional mais lento estabilize, uma vez suas entradas mudando.

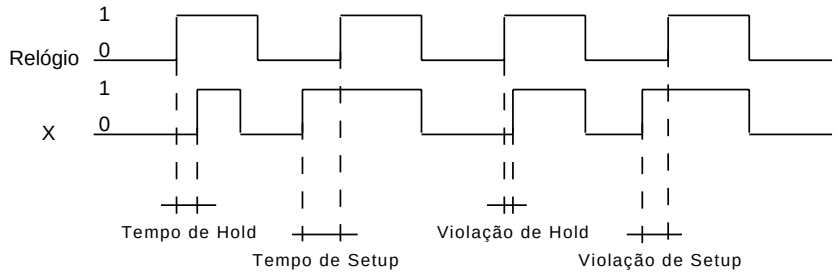


Figura 5.2: Diagrama de tempos mostrando a relação entre o sinal de relógio e qualquer outro sinal X de um sistema digital síncrono. Supõe-se o sinal de relógio ativo na transição de subida, mostram-se definições de tempos mínimos de hold e de setup, e ilustram-se violações de ambos.

Uma terceira vantagem é que as imposições sobre o ambiente e fronteiras de blocos funcionais da Definição 5.1 permitem abstrair quaisquer eventos espúrios resultantes de propagação diferenciada de sinais das entradas para as saídas dos blocos funcionais, fenômenos denominados *transitórios*. *Corridas* são um fenômeno indesejável resultante da propagação desigual de sinais ao longo de dois ou mais laços de realimentação. O estilo síncrono possui como quarta vantagem a total eliminação de corridas, pela obrigatória interposição de pelo menos uma barreira temporal (registrador controlado pelo relógio) em cada laço. Mais detalhes sobre transitórios e corridas serão fornecidos na Seção 5.2.4. Uma quinta vantagem é que circuitos síncronos permitem o emprego de técnicas de manipulação algébricas e Booleanas de decomposição em geral, enquanto que estilos assíncronos costumam limitar a aplicabilidade das manipulações Booleanas, as mais poderosas [93].

Do lado das desvantagens, a mais importante no contexto tecnológico atual e futuro é o chamado *escorregamento do relógio*. Trata-se da diferença no instante de chegada do sinal de relógio às diferentes partes do sistema. A Figura 5.3 caracteriza o conceito, mostrando no diagrama de tempos o sinal de relógio chegando nos pontos A e B do circuito da Figura 5.1. Os pontos em questão encontram-se nas entradas de vetores de memória supostos muito distantes no circuito. O problema ocorre quando o atraso do sinal de relógio não é acompanhado pelas entradas dos elementos de memória, mudando os valores relativos dos tempos de estabelecimento e manutenção. Esta diferença obriga à consideração precisa do escorregamento na temporização de sinais internos e externos em cada posição do sistema. Outro problema é que o escorregamento piora na medida em que a densidade dos sistemas integrados aumenta, pois uma diminuição do min-feature size (ver Seção 1.3 para definição do termo) leva a um maior atraso de propagação para todos os sinais. Uma conseqüência

de considerar o escorregamento em detalhe é assim reduzir consideravelmente o nível de abstração no momento de determinar o período do relógio, aumentando sobremaneira o tempo de projeto.

Uma segunda desvantagem do estilo síncrono de projeto é o potencial destes para desperdiçar energia. Como o sinal de relógio está presente em todos os pontos do sistema e é um sinal periódico, ele está constantemente sofrendo transições. A tecnologia de fabricação de ICs de maior sucesso na atualidade é CMOS, que consome uma quantidade de energia desprezível, exceto durante transições. Como um circuito síncrono possui o relógio que sofre transições periodicamente mesmo na ausência de atividade útil, isto leva a consumo desnecessário de potência elétrica. Além disso, a distribuição do sinal de relógio em sistemas complexos pode dispender uma porção significativa da potência total dissipada pelo sistema digital. Um caso extremo são os primeiros microprocessadores da família Alpha da DEC, com pelo menos um membro consumindo cerca de 30W, sendo cerca de metade destes devido à distribuição do relógio [76]. Prega-se a adoção de estilos assíncronos modernamente usando como principal argumento a potencial economia de energia que estes habilitam. Diversas técnicas têm sido propostas para amenizar o problema de transições inúteis do relógio e com isto diminuir o consumo desnecessário de energia, mas estas técnicas consistem na realidade na adição de algum assincronismo no estilo síncrono de projeto [117].

A terceira desvantagem de sistemas síncronos é o desempenho de pior caso. Este comportamento é decorrência do período do relógio ser determinado pelo bloco funcional mais lento. Bloco mais rápidos que este devem aguardar inutilmente um tempo adicional, após concluir a computação que lhes cabe, para garantir o sincronismo total do sistema. Assim, o estilo síncrono exige cuidado na decomposição do sistema para obter blocos funcionais de desempenho similar, evitando a degradação do desempenho global. Ora, tal decomposição, dirigida à implementação física e não aos requisitos da especificação, torna o projeto artificialmente mais complexo, e mais uma vez obriga a reduzir o nível de abstração das preocupações de projeto. Por outro lado, vários estilos assíncronos trabalham com *assinalamento de completude* explícito para operações entre blocos funcionais, garantindo um comportamento que depende do tempo médio de execução dos blocos. Embora esta comparação pareça indicar que estilos assíncronos levem a ganhos de desempenho maiores, não está claro na atualidade que circuitos síncronos sejam efetivamente mais rápidos. Como salienta Hauck em [93], as políticas de assinalamento de completude exigem tempo de comunicação adicional entre blocos, aumentando o tempo de computação total. O mesmo autor salienta, e este autor compartilha esta opinião, que mais pesquisa é necessária para esclarecer este ponto.

Uma quarta desvantagem de sistemas síncronos é sua baixa propensão para permitir a fácil migração tecnológica de projetos. Com a atual ênfase de projeto

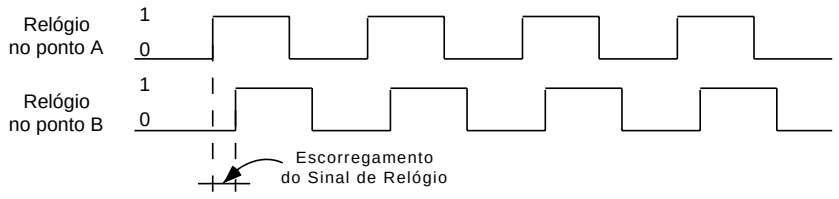


Figura 5.3: Diagrama de tempos mostrando o fenômeno de escorregamento do sinal de relógio para os pontos distantes A e B do sistema da Figura 5.1.

VLSI sendo colocada no tempo de chegada ao mercado do produto, técnicas de prototipação rápida e emulação tornam-se cada vez mais empregadas. Sistemas são implementados inicialmente em dispositivos personalizáveis pós-fabricação como FPGAs para fins de validação e mesmo para lançamento inicial do produto, após o que versões do mesmo sistema são implementadas como um ASIC, reduzindo custos para produção em volume. Migrar um projeto síncrono implementado sobre um FPGA para um ASIC pode tomar um tempo considerável de reprojeto, devido ao redimensionamento da distribuição de relógio e da necessidade de recálculo do escorregamento, para estabelecer a freqüência máxima de operação. Estilos assíncronos baseados em assinalamento de completude são migráveis de forma total ou quase totalmente automatizada. Além disto, um problema correlato é a realização de revisões de projeto, que exige os mesmos recálculos em sistemas síncronos, caso o componente mais crítico (o que gasta maior tempo de computação) seja reprojetado para aumentar o desempenho. Estilos assíncronos acomodam mais simplesmente a substituição modular de blocos funcionais.

A adaptabilidade a propriedades físicas variáveis tais como temperatura, tensão de alimentação e parâmetros de fabricação constituem a quinta desvantagem do estilo síncrono de projeto. Isto se deve ao fato que a pior combinação de fatores deve determinar o período do relógio, gerando mais uma vez um desempenho de pior caso. Estilos assíncronos baseados em assinalamento de completude de operações levam o tempo de computação que for necessário em cada situação de parâmetros alterados, executando computações tão rápido quanto os valores das grandezas físicas presentes permitam.

O tratamento de um fenômeno denominado *meta-estabilidade* [45] constitui a sexta desvantagem de estilos síncronos de projeto. Informalmente, meta-estabilidade é o equivalente elétrico de uma situação de equilíbrio instável em sistemas mecânicos, onde a composição de forças atuando sobre um corpo instantaneamente pode ser tal que este permanece em repouso, mas um distúrbio mínimo de qualquer das componentes leva o corpo a entrar em movimento. O exemplo clássico é o de um veículo no cume de uma montanha, onde uma

pequena força pode fazer o veículo mover-se montanha abaixo para um dos lados. Enquanto a força não atuar, o veículo equilibra-se instavelmente no topo da montanha. Eletricamente, alguns sinais podem estacionar a meio caminho entre as tensões correspondentes a 0 e 1 lógico [45]. Isto pode acontecer, por exemplo, em um par de inversores realimentados de um bit de memória, onde todo o circuito pode permanecer um tempo finito, mas indeterminado, na tensão média. Circuitos síncronos não admitem sinais que tomem tempo indeterminado para atingir um estado estável, gerando falhas de funcionamento. Circuitos assíncronos podem "aguardar" que a situação de equilíbrio instável se resolva, sendo então uma classe de estilos mais *robustos* de projeto.

Como conclusão, durante as décadas de 60, 70 e 80, o panorama de projeto de sistemas digitais complexos foi dominado pelo estilo síncrono de projeto, pois o contexto tecnológico apontava vantagens claras deste estilo. Neste contexto, sistemas síncronos eram mais fáceis de projetar e testar, exigiam menos hardware, e eram mais rápidos e mais confiáveis. As previsões de crescimento da escala de integração sugerem que esta situação não deva perdurar em futuro próximo, com a dificuldade de gerenciamento de redes de distribuição de relógio superando muitas das vantagens com relação a estilos assíncronos. Uma segunda tendência que aponta para o uso de estilos assíncronos de projeto é a possibilidade de separação dos aspectos de corretude lógica e de temporização global, impossível em um estilo síncrono puro [143]. Na prática, uma solução intermediária deverá servir de transição entre o uso do estilo síncrono puro e a adoção de estilos assíncronos, já praticada pela indústria de ponta: o uso de múltiplos sinais de relógio, um para cada grande bloco do sistema, e uso de técnicas de assinalamento de completude entre os grandes blocos. A partir deste ponto, deixa de existir o maniqueísmo da separação síncrono-assíncrono, passando o foco para a investigação da granularidade ideal onde realizar a troca de estilo.

5.2 Assíncronos: Modelos e Problemas

Esta Seção apresenta o conjunto de modelos disponível para uso em estilos assíncronos de projeto e para tratamento dos problemas de temporização passíveis de ocorrer em sistemas assíncronos. Os modelos são divididos em três classes:

- representação de informação - abordados na Seção 5.2.1, estes modelos sugerem diferentes formas de organizar e interpretar a informação digital para tirar partido de estilos assíncronos de projeto;
- ambiente - modelos de ambiente são o tema da Seção 5.2.2, que estabelece um conjunto de possíveis pressupostos de interação entre um sistema digital assíncrono e o meio em que este está colocado;

- fenômenos temporais - assunto da Seção 5.2.3, congrega os modelos de atrasos de sinais utilizáveis em diferentes estilos assíncronos, bem como modelos da estrutura de distribuição de atrasos no sistema.

Cada estilo de projeto assíncrono específico será visto aqui como determinado pelo pressuposto de escolha de um subconjunto de modelos das classes acima. O objetivo de cada estilo assim determinado é o de facilitar a tarefa de implementação de sistemas assíncronos que atendam as especificações funcionais, e que evitem os problemas temporais apresentados na Seção 5.2.4. Uma observação importante é que existe um sério problema de nomenclatura técnica para novatos na área de sistemas assíncronos, por motivos históricos. Termos como *insensibilidade a atrasos*, *independência de velocidade* e *auto-temporização* possuem significados precisos, mas dificilmente claros a partir da sintaxe das expressões que os definem, e sobretudo dificilmente diferenciados a partir do significado intuitivo desta terminologia. Procura-se aqui introduzir uma notação coerente, mantendo referências para a nomenclatura consagrada na literatura, na medida do possível.

5.2.1 Modelagem da Representação de Informação

Uma forma natural e direta de representar informação em sistemas digitais é empregar vetores binários de comprimento pré-fixado e tamanho suficiente para acomodar todas as informações distintas a serem manipuladas. Formalmente, seja $S = \{a, b, c, \ldots, n\}$, um conjunto de objetos, onde $|\,S\,|$ representa a cardinalidade (número de elementos) de S. É óbvio que se pode fixar o comprimento do vetor para codificar univocamente cada elemento de S em $m \geq \lceil \log_2(|\,S\,|) \rceil$ bits, onde a notação $\lceil\ \rceil$ designa o menor inteiro não menor que a expressão entre $\lceil$ e $\rceil$. Para obter uma codificação que identifique de forma única cada elemento de S, basta determinar uma injeção (Definições 2.30) de S no conjunto de vetores binários de comprimento m. A imagem de cada elemento x de S sob esta injeção é o *código* de x, conforme definido no Capítulo 4.

No nível físico de abstração, os valores binários são implementados no mais das vezes como um de dois níveis de tensão elétrica sobre um condutor. Esta é a forma mais comum, senão a única, de se representar informação no estilo síncrono de projeto. Estuda-se agora as alternativas a esta forma de representação, nas duas próximas Seções. Por último, a Seção 5.2.1.3 discute modelos de protocolos de comunicação de informação.

5.2.1.1 Alternativas para a Codificação Física de Informação

A implementação física de informação no estilo síncrono assume que cada bit é associado a um fio, onde o valor de tensão no fio determina o valor do bit, usando duas tensões distintas para codificar os valores 0 e 1, os chamados

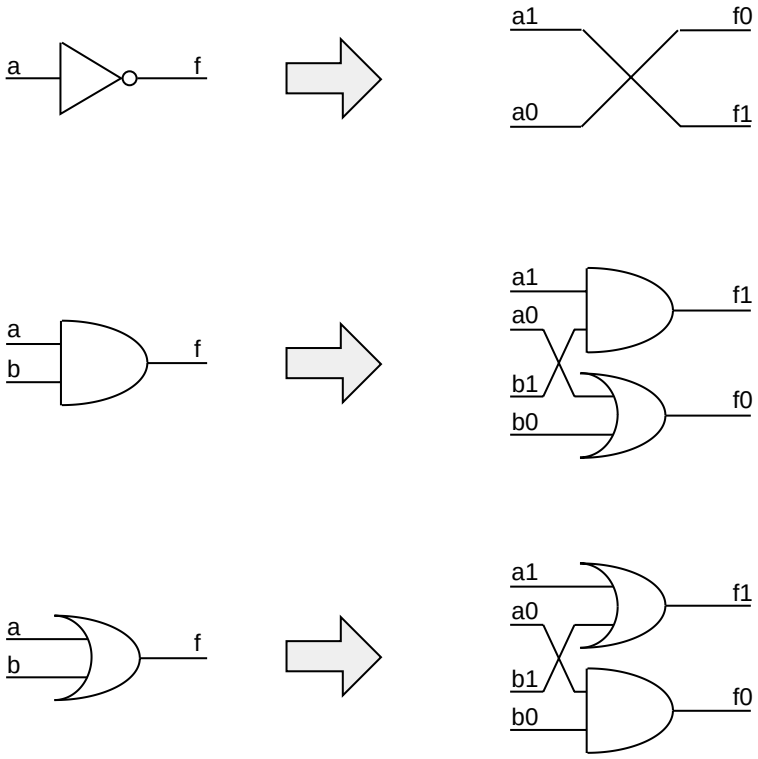


Figura 5.4: Correspondência entre funções desempenhadas por portas lógicas convencionais (à esquerda), implementando lógica com codificação em trilha única, e diagrama para a mesma função em trilha dupla (à direita). Exemplificado para portas NÃO, E e OU. Portas NÃO-E e NÃO-OU em trilha dupla podem obviamente ser obtidas pelo intercâmbio de posições dos sinais de saída $f1$ e $f0$ nas portas E e OU, respectivamente.

niíveis lógicos. Este pressuposto de projeto é denominado de ***codificação de trilha única*** (do inglês, *single-rail encoding*). A alternativa a esta é empregar *dois* fios por bit de informação, no que se denomina ***codificação de trilha dupla*** (do inglês, *dual-rail encoding*). Esta nova classe de codificações pode ser empregada para identificar situações que não se pode contemplar com técnicas de codificação convencionais de trilha única.

Por exemplo, seja um método de codificação onde dois fios $w1$ e $w0$ representam um bit em trilha dupla, tal que o nível lógico 1 em $w1$ e $w0$ representa os valores 1 e 0 para o bit, respectivamente. Seja ainda que a combinação $(w1 = 0, w0 = 0)$ indica a ausência de valor, batizada de ***espaçador*** (do in-

Valor do bit	classe 1	classe 2
0	01	00
1	10	11

Tabela 5.1: Técnica de codificação dupla trilha com conjuntos de dados alternantes. Dados sucessivos usam classes distintas para representar os valores sobre um mesmo sinal.

glês, *spacer*), e seja ($w1 = 1, w0 = 1$) uma combinação inválida. Um exemplo de seqüências válidas é o seguinte: supondo que um conjunto de sinais representa a informação binária `0110` nesta codificação, e que os sinais mudam sucessivamente para `1010` e `1001`, a seqüência de valores nos sinais é `01 10 10 01` seguido de `00 00 00 00`, `10 01 10 01`, `00 00 00 00` e `10 01 01 10`. Note-se os códigos espaçadores intermediários. A codificação em trilha dupla pressupõe uma nova classe de operadores para implementar as tradicionais funções Booleanas. A Figura 5.4 ilustra como algumas destas podem ser traduzidas do ponto de vista estrutural, usando portas lógicas tradicionais [143]. Uma observação imediata e interessante que salta aos olhos da Figura é que, em tese, inversores jamais são necessários para implementar circuitos de trilha dupla, sua implementação consistindo de um mero intercâmbio dos fios que transportam o valor do bit.

A codificação em trilha dupla permite construir *códigos auto-sincronizantes* [5], dos quais a técnica de codificação com espaçador citada acima é apenas um exemplo. Outro exemplo é técnica dos ***conjuntos de dados alternantes*** (do inglês, *alternating data sets*) [117], ilustrado na Tabela 5.1.

A vantagem desta técnica de trilha dupla é que transições sucessivas de valores de um sinal nunca implicam o chaveamento simultâneo de dois valores, como pode ser observado na Tabela. Supondo o sinal inicialmente na classe 1 da Tabela, para a mesma seqüência de valores acima os sinais são `01 10 10 01`, `11 00 11 00` e `10 01 01 10`.

A codificação de trilha dupla é muito empregada em representação de dados, por exemplo na implementação de bloco lógicos combinacionais como unidades lógico-aritméticas [83].

5.2.1.2 Alternativas de Assinalamento de Informação

A implementação física de informação no estilo síncrono usa *níveis* de tensão elétrica para determinar o valor de cada bit. Este pressuposto de projeto é denominado aqui de ***assinalamento por nível*** (do inglês, *level signalling*). A alternativa a este é empregar *transições* nos fios para determinar o valor do bit de informação, no que se denomina ***assinalamento por transição*** (do inglês,

transition signalling).

Esta nova classe de representações determina uma nova maneira de raciocinar em termos lógicos. A álgebra Booleana continua sendo a ferramenta principal para manipular sistemas no nível lógico de abstração, mas como no caso da codificação em trilha dupla, os dispositivos eletrônicos do nível lógico de abstração precisam ser repensados. De fato, ao se utilizar assinalamento por transição, os níveis lógicos podem ser considerados irrelevantes (ou não), interessando apenas as *mudanças* de nível lógico.

Quando nem mesmo o tipo de transição interessa, ou seja, quando uma transição $0 \rightarrow 1$ é considerada equivalente a uma transição $1 \rightarrow 0$, obtém-se a chamada ***lógica de eventos***, que será alvo de estudo mais detalhado na Seção 5.5. Na lógica de eventos, como aponta Sutherland [171], a maior parte dos blocos construtivos tem pouca relação com as portas lógicas usadas para manipular informação representada com assinalamento por nível. Uma exceção é a porta lógica OU-EXCLUSIVO, ou XOR (do inglês, *Exclusive-Or*).

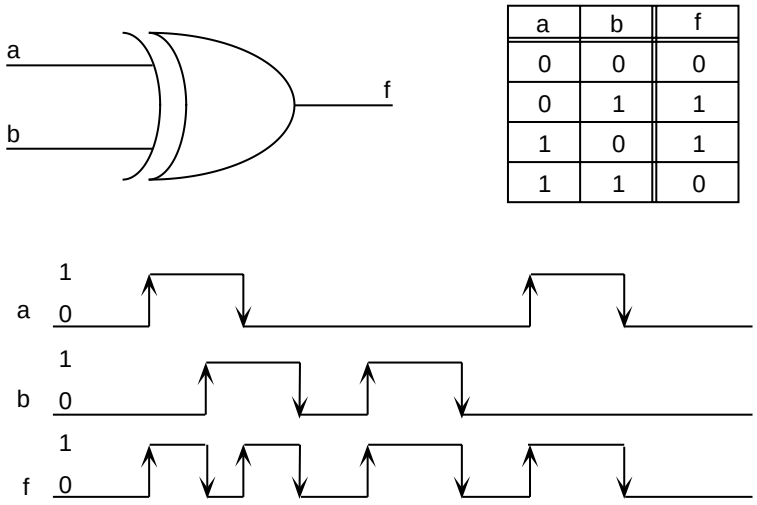
A Figura 5.5 mostra o símbolo da porta XOR e um exemplo de diagrama de tempos. Supondo que cada entrada transporta eventos ao longo do tempo, e que as entradas nunca mudam simultaneamente, nota-se que cada transição em alguma entrada gera uma transição na saída da porta. Na lógica de eventos, este componente é por este motivo denominado de MERGE (combinar, em inglês), pois ele combina dois fluxos de eventos em um único. Todos os outros blocos construtivos tais como o E de eventos, possuem um comportamento somente implementável com lógica seqüencial. Um estilo de projeto baseado na lógica de eventos e outros pressupostos será abordado na Seção 5.5

As principais vantagens do assinalamento por transição, sobretudo quando usando lógica de eventos, é o potencial para acelerar a computação e a facilidade de sua implementação na tecnologia CMOS, que domina o panorama de implementação de sistemas VLSI [171].

5.2.1.3 Alternativas de Protocolo de Comunicação de Informação

Além de representar informação, uma questão importantíssima em estilos assíncronos de projeto é a correta trasmissão de informação entre blocos do sistema. Note-se que a mesma discussão vale para o projeto de interfaces, mesmo que estas interfaces envolvam um componente síncrono, como é o caso do protocolo de comunicação clássico entre processador e memória principal, composto de barramentos de endereços e dados, controles de escrita e leitura e efetivação de operação [149].

O processo de controle de comunicação de informação em circuitos assíncronos deve ser extremamente simples e eficiente, uma vez que é utilizado muitas vezes no interior do sistema, como forma de garantir a sincronização entre blocos internos. Em circuitos síncronos, ao contrário, todas as ações internas são



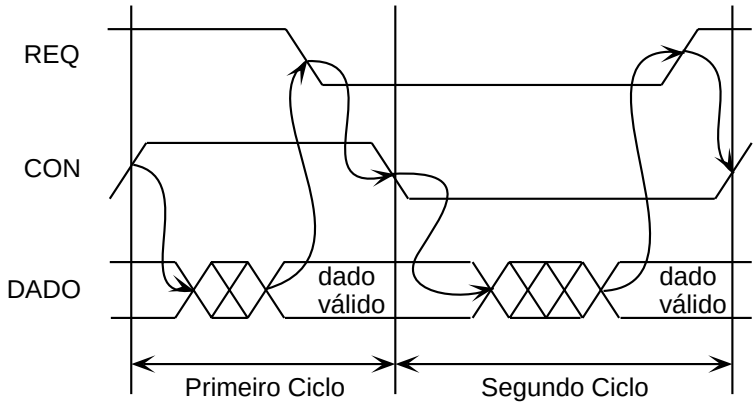
a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

Figura 5.5: A porta lógica XOR e seu comportamento como o OU de eventos.

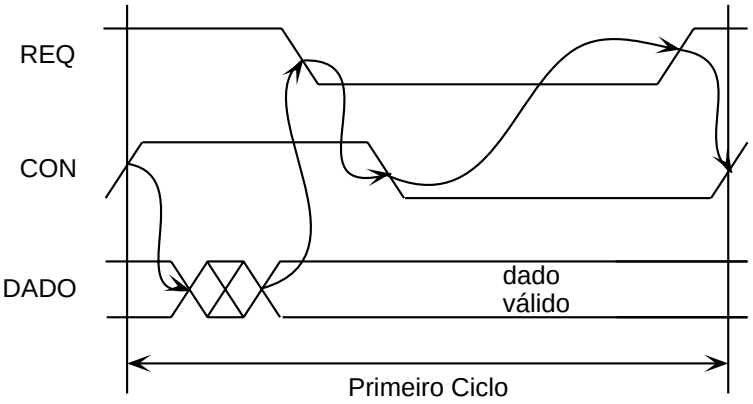
automaticamente ordenadas pelo sinal de relógio, e protocolos de intercâmbio de informação têm lugar apenas na interface com o mundo externo.

Supondo um fluxo unidirecional de informação entre dois blocos, apenas dois sinais são necessários para comandar a troca de dados, em adição aos sinais que transportam a própria informação. A Figura 5.6 ilustra dois tipos de protocolos simples, denominados respectivamente de *protocolo de duas fases* e *protocolo de quatro fases*. O protocolo de quatro fases exige que a cada início de ciclo os sinais de controle estejam em um nível lógico determinado. Este protocolo exige menos hardware para ser implementado, mas é mais lento que o protocolo de duas fases, como observa Hauck [93]. A natureza assíncrona da comunicação é facilmente observada na Figura 5.6(a), onde os dois ciclos possuem durações nitidamente diferentes.

Como comentário final, a composição de escolhas de codificação, tipo de assinalamento e protocolo de comunicação determina em grande parte o estilo assíncrono de projeto adotado. Note-se que mais de uma de tais composições pode ser usada em diferentes partes de um mesmo projeto, e até uma mistura de blocos assíncronos usando diferentes escolhas com blocos síncronos pode ocorrer em um sistema digital complexo. Um dos principais fatores para decidir por tamanha mistura é a adequação da técnica de projeto ao subsistema em construção. Por exemplo, a codificação em duas trilhas é útil para viabilizar a insensibilidade à atrasos no tratamento de dados, mas onera sobremaneira o sistema quando usada para implementar circuitos de controle.



(a) Duas fases



(b) Quatro fases

Figura 5.6: Tipos gerais de protocolos assíncronos de comunicação. Mostram-se dois ciclos completos do protocolo de duas fases e um ciclo completo do protocolo de quatro fases. O sinal REQ corresponde a uma requisição de um recurso, de acesso ou de envio de informação, o sinal CON é a concessão do ou o acesso ao recurso, ou a recepção da informação. O conjunto de sinais denominados DADO corresponde ao recurso, à informação a ser enviada ou a qual se faz acesso. As setas indicam a ordem de ocorrência de eventos, quando respeitado o protocolo.

5.2.2 Modelagem do Ambiente

A modelagem de um sistema físico passa pela seleção de um certo número de variáveis, que fornecem uma abstração do sistema em questão. Estas variáveis são denominadas *variáveis de estado*. As variáveis escolhidas devem ser suficientes para descrever aqueles aspectos do comportamento do sistema que são de interesse, caso contrário é necessário adicionar elementos ao conjunto de variáveis. Utiliza-se eventualmente o termo *sinal* como alternativa para variável, embora o primeiro seja normalmente usado em um contexto mais restrito de variáveis associadas a grandezas físicas variando sobre componentes eletrônicos como fios.

Todo sistema digital opera em um *ambiente* que fornece as entradas do sistema e recebe suas saídas. Observe-se que o conjunto sistema digital-ambiente pode ser visto como um sistema digital autônomo, formado por dois módulos que se comunicam através das entradas e saídas do sistema digital alvo do processo de projeto. Neste caso, as entradas do sistema digital são vistas como saídas do ambiente e vice-versa. Este tipo de modelagem é adotado, por exemplo, no trabalho de Hoare sobre processos seqüenciais comunicantes (do inglês, *communicating sequential processes* ou CSP) [100], um modelo hoje sugerido como adequado para descrever sistemas digitais em alto nível de abstração [178]. O conceito de uma mudança na entrada de um sistema digital assíncrono será de grande importância neste Capítulo, por isso cabe formalizá-lo adequadamente.

Definição 5.2 (Mudança na entrada de sistema assíncrono) *Considera-se que uma variável $v(t)$, que assume valores de um domínio finito e variando ao longo do tempo* **muda** *no instante τ se $v(\tau) = \beta$, e existe um $\delta > 0$ tal que $v(t) = \alpha \neq \beta$ para $\tau - \delta \leq t < \tau$. Dito em palavras, v deve ter um novo valor β no instante τ e deve ter assumido um valor antigo α ao longo de todo um intervalo de tempo imediatamente anterior a τ, e de duração não nula δ.*

Dada esta definição precisa de mudança de valor de entrada, cabe ainda uma definição de quando dois sinais (de entrada ou não) mudam *simultaneamente* em um sistema digital.

Definição 5.3 (Mudanças simultâneas em um sistema digital) *Considere-se um par de variáveis $v(t)$ e $w(t)$, cada uma assumindo valores em domínios finitos e variando ao longo do tempo. Sejam dois intervalos de tempo, $\delta 1$ e $\delta 2$, com $\delta 1 > \delta 2$ denominados de* **períodos críticos**. *Se $v(t)$ e $w(t)$ mudam em instantes que distam menos de $\delta 2$ unidades de tempo, assume-se que eles mudam* **simultaneamente** *de valor. Se a mudança ocorre com uma diferença temporal maior que $\delta 1$, diz-se que as variáveis mudaram de valor* **separadamente**. *Se o intervalo entre as mudanças ficar entre $\delta 1$ e $\delta 2$, nada se pode afirmar sobre a temporização relativa das mudanças.*

Obviamente, a terceira situação da Definição 5.3 deve ser evitada a todo custo, sob pena de tornar impossível controlar o comportamento do sistema digital. De fato, a diferença entre $\delta 1$ e $\delta 2$ dá uma noção de quão robusta é a tecnologia de implementação. Quanto mais próxima de zero esta diferença, menos risco há de o sistema digital apresentar problemas de comportamento imprevisível. Em circuitos síncronos, as mesmas restrições existem, mas dizem respeito ao relacionamento de cada variável com o sinal de relógio. Para estilos assíncronos, a extensão da Definição 5.3 para simultaneidade de mudança de um conjunto de variáveis é óbvia.

Estuda-se nas Seções a seguir os modelos de ambiente, doravante denominados *modos de ambiente*. O estudo é feito com base nos pressupostos sobre o relacionamento entre ambiente e sistema digital colocado neste. A discussão segue a linha de Brzozowski e Seger em [35], enriquecida com detalhamento da obra de Lavagno e Sangiovanni-Vincentelli [117].

5.2.2.1 Modo Completamente Irrestrito

Neste modo, o ambiente pode mudar as entradas do sistema a qualquer momento, sem se preocupar com o estado do sistema. Apesar de ser o modo que mais precisamente reflete toda e qualquer parte do mundo físico, sistemas digitais podem falhar se suas entradas mudam rápido demais ou no momento errado. Mesmo ignorando as limitações devidas ao próprio circuito, o modo completamente irrestrito pode levar a sérias dificuldades. Por exemplo, se uma variável muda nos seguintes instantes de tempo: $0, 1/2, 3/4, 7/8 \ldots$, existirão infinitas mudanças em apenas uma unidade de tempo. Sabemos que fenômenos físicos de interesse nunca se comportam assim, então este modo é menos restrito que o necessário na prática.

Logo, daqui por diante assume-se que todo ambiente satisfaz a seguinte condição.

Definição 5.4 (Condição de Finitude de Ambientes) *Em qualquer ambiente, apenas um número finito de mudanças de sinal pode ocorrer em qualquer intervalo de tempo finito. Este número, conquanto finito, pode ser arbitrária e ilimitadamente grande.*

Um ambiente que satisfaz *apenas* a condição de finitude é dito trabalhando no modo ***irrestrito***. Se o ambiente satisfaz a condição de finitude e mais alguma restrição, o ambiente trabalha no modo ***restrito***. Duas classificações ortogonais de modos restritos são discutidas a seguir, cada uma com dois modos.

5.2.2.2 Modos da Relação Entrada-Saída

Uma primeira classificação de ambientes é obtida ao considerar as possíveis restrições impostas à relação entre entradas e saídas, visando reduzir a complexidade de projeto de sistemas assíncronos.

O primeiro modo a considerar é o ***modo fundamental***. Neste modo, assume-se que o sistema assíncrono inicia em algum estado total estável (Definição 3.10), i.e. em um estado no qual suas entradas, sinais internos e saídas estão todos com valores fixos e nenhum destes apresenta tendência para mudar. Por definição, um estado total estável persiste permanentemente, a menos que as entradas do sistema mudem. Neste estado estável, permite-se ao ambiente mudar as entradas do circuito. Contudo, após todas as mudanças simultâneas, não é permitido ao ambiente mudar as entradas novamente, até que um novo estado total estável seja atingido. Claro, está implícito que o circuito de fato vai estabilizar, o que vale em praticamente qualquer sistema útil. O modo fundamental é o pressuposto básico da maioria dos estilos assíncronos clássicos, tais como o trabalho de Huffman [106] e Unger [182].

No modo fundamental, os parâmetros que estabelecem a simultaneidade de mudanças da Definição 5.3 possuem uma interpretação precisa [117]:

- $\delta 1$ é o tempo de estabelecimento (setup) máximo para todo o circuito, ou seja, o tempo que deve ser aguardado para que todos os elementos estabilizem, uma vez que suas entradas tenham estabilizado;
- $\delta 2$ é o atraso de propagação t_{pd} mínimo da lógica combinacional.

Na prática, este modo é realizado seguindo um procedimento direto. Estima-se o tempo necessário para o sistema assíncrono estabilizar, no pior caso. A seguir, assegura-se que as entradas permaneçam constantes pelo menos por este tempo. Desta maneira, a definição do modo fundamental somente faz sentido quando assume-se que os atrasos envolvidos possuem um limite superior. Os estilos assíncronos clássicos normalmente assumem o ambiente operando no modo fundamental [106, 137, 182].

Um segundo modo de ambiente é o ***modo entrada-saída***. Como no modo fundamental, o ponto de partida deste é um estado total estável do sistema assíncrono. Neste ponto, o ambiente pode alterar as entradas do sistema. A seguir, o ambiente poderá voltar a alterar as entradas, mas somente após o sistema ter produzido pelo menos uma mudança em uma de suas saídas, ou se nenhuma mudança é esperada. Note-se que isto *não* implica que o sistema inteiro deva estar estável, pois alguns sinais internos podem ainda estar mudando.

O modo entrada-saída é menos restrito que o modo fundamental, tendo sido por este motivo utilizado em estilos assíncronos mais modernos, capazes de lidar com sua maior complexidade de controle e implementação.

5.2.2.3 Modos de Simultaneidade de Entradas

A segunda classificação de ambientes é obtida ao considerar a cardinalidade de entradas que são permitidas mudar simultaneamente. Neste sentido, os ambientes podem ser classificados em duas categorias [117]:

- o ***modo mudança única de entrada*** implica que somente uma entrada pode mudar a cada instante dado (ou seja, dentro de um período $\delta 2$). Se combinado com o modo fundamental, este modo produz o ***modo fundamental normal*** de ambiente;
- o ***modo múltipla mudança de entradas*** implica que qualquer número de entradas pode mudar a cada instante.

5.2.3 Modelagem de Fenômenos Temporais

Na Seção 5.2.2 foram estudadas as classes de restrições atribuíveis a ambientes para viabilizar estilos assíncronos de projeto, os chamados modos de ambiente. Agora, deve-se abordar a modelagem de fenômenos temporais inerentes a dispositivos eletrônicos usados em projeto VLSI. Modelos para estes fenômenos, juntamente com a álgebra Booleana e os modelos de ambiente, constituem um arcabouço para a introdução e discussão de estilos de projeto assíncronos. Nesta Seção, aborda-se o fenômeno de atraso *per se*, e somente na Seção 5.2.4 o alvo serão os problemas resultantes da propagação diferenciada de atrasos, denominados genericamente de corridas e transitórios.

Os modelos de atraso são divididos em modelos para componentes e modelos para circuitos.

5.2.3.1 Modelagem de Atrasos para Componentes

Uma representação estrutural no nível lógico de abstração de um sistema assíncrono é em geral composta de:

- ***portas lógicas***, ou seja, componentes que calculam um conjunto (possivelmente unitário) de valores para variáveis de saída como funções discretas gerais, no sentido das Definição 2.32. Estas funções podem ser funções de chaveamento (Definição 2.39) ou funções ternárias, um tipo de função inteira (Definição 2.36) onde a cardinalidade do codomínio é 3;
- ***elementos atraso***, um componente que posui uma única entrada e uma única saída, esta última uma função discreta da entrada e do tempo[1]. Um

[1] É necessário lembrar que o conceito de função discreta geral definido aqui pressupõe o domínio como o produto cartesiano de conjuntos discretos e finitos. Logo, o uso do tempo como variável independente nestas funções assume *tempo discretizado* e a consideração de uma quantidade finita de instantes de tempo.

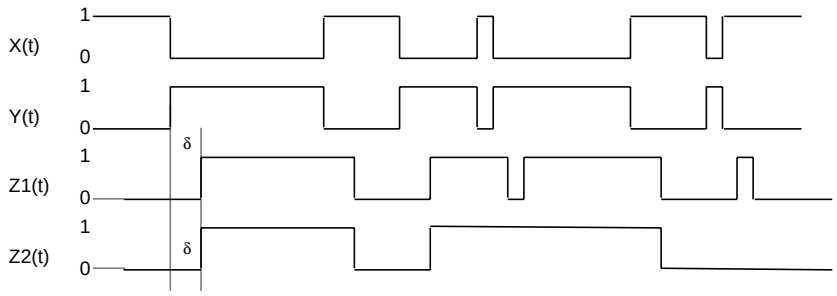


Figura 5.7: Dois modelos de comportamento de atraso para o inversor da Figura 5.8: atraso ideal ($Z1(t)$) e atraso inercial ($Z2(t)$).

elemento atraso está ***excitado*** se alguma transição está se propagando através dele, senão o elemento está ***estável***;

- ***fios***, condutores ideais que conectam a saída de uma porta lógica ou a saída de um elemento atraso a um subconjunto de entradas de portas e/ou atrasos. As entradas e saídas primárias (aquelas que constituem a interface com o ambiente) do sistema são normalmente consideradas como portas que computam uma função de chaveamento de entradas desconhecidas.

Tomando como exemplo um inversor, a álgebra Booleana modela tal componente como um elemento capaz de implementar exatamente a função complemento na sua saída com relação a sua entrada. Um modelo mais preciso é representar um inversor como uma conexão série do componente derivado da álgebra Booleana (porta lógica ideal, sem atraso) e de um elemento atraso, conforme mostrado na Figura 5.8.

O comportamento do inversor físico é então dado pela relação entre os sinais X e Z. O sinal Y é parte do modelo, mas seu comportamento não é observável, ele resulta da decomposição do comportamento na parte Booleana e na parte atraso. A posição relativa da porta ideal e do atraso é uma convenção. A Figura 5.7 mostra um exemplo de diagramas de tempo para os sinais X, Y e Z.

Assume-se que todos os sinais são binários e que as transições de valor são instantâneas. Assume-se também que as transições do sinal $X(t)$, a entrada primária, podem ocorrer a qualquer instante, gerando *pulsos* em 0 e em 1 de duração arbitrária. O ambiente é irrestrito, mas satisfaz a condição de finitude. O valor do sinal Y é o complemento de X a todo instante. O atraso é assumido como uma constante real δ. A Figura ilustra dois tipos de comportamento

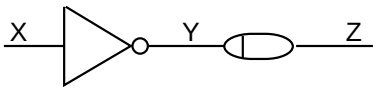


Figura 5.8: Um modelo fisicamente mais preciso de um inversor, como a conexão série de uma porta lógica ideal (sem atraso), que calcula a função Booleana complemento, e um elemento atraso.

possíveis para o elemento atraso. O primeiro, $Z1(t)$ é o chamado ***atraso ideal***[2], onde a saída consiste simplesmente na translação temporal da entrada, ou seja, a saída é uma cópia fiel da entrada δ instantes de tempo deslocada para o futuro. Contudo, os atrasos reais normalmente não se comportam assim. O tempo δ é uma medida não apenas do intervalo necessário para que uma mudança na entrada provoque algum efeito na saída, mas também do tempo de permanência mínima de um valor estável na entrada para que este afete a saída. Este comportamento pode ser modelado com o elemento atraso não transferindo para sua saída o efeito de pulsos de duração menor que δ. Este comportamento aparece na Figura 5.7 como a onda $Z2(t)$. Tal modelo é denominado ***atraso inercial***.

Além da classificação de atrasos em ideal e inercial, é possível estabelecer uma classificação com base na magnitude δ. Um atraso é:

- ***ilimitado*** (do inglês, *unbounded*), se nenhum limite para a magnitude do atraso é conhecido, exceto que esta é finita e positiva;
- ***limitado*** (do inglês, *bounded*), se limites superior e inferior para a magnitude do atraso são conhecidos ao longo do projeto do sistema[3]. Neste caso, deriva-se uma subclassificação:
  - ***atraso inteiro*** ou ***atraso discreto***, no qual a magnitude é sempre um múltiplo de alguma unidade;
  - ***atraso real*** ou ***atraso contínuo***, onde a magnitude é restrita apenas pelos limites superior e inferior.

Uma terceira classificação ortogonal de atrasos diz respeito à modelagem da transição de valor na saída do elemento atraso. Um atraso é:

[2] Alguns autores denominam este modelo de ***atraso puro*** [117].

[3] Brzozowski e Seger propõe um modelo adicional, onde apenas o limite superior é conhecido, e denominam-no *atraso limitado por cima*, do inglês, *up-bounded delay*, enquanto que o modelo aqui citado é por eles batizado de *atraso duplamente limitado*, do inglês, *bi-bounded delay* [35]. Como o atraso duplamente limitado pode ser pessimista na previsão do comportamento assíncrono, o atraso limitado por cima fornece freqüentemente um compromisso interessante entre custo de processamento e qualidade de resultados na modelagem.

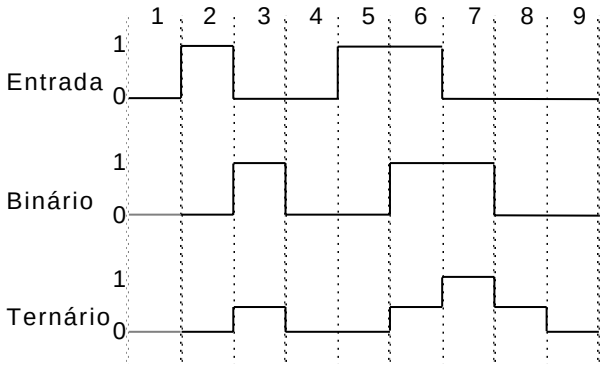


Figura 5.9: Comparação de atrasos inerciais inteiros binário e ternário. Assume-se magnitude de atraso unitária em ambos casos, e mostra-se o valor intermediário ternário como um nível entre 0 e 1.

- ***binário***, se as transições de valor na sua saída são modeladas como eventos instantâneos;
- ***ternário***, se entre os limites superior e inferior da magnitude do atraso, o valor da saída é *indefinido*, ou seja, um terceiro valor lógico, diferente de 0 e de 1, é inserido na representação dos sinais.

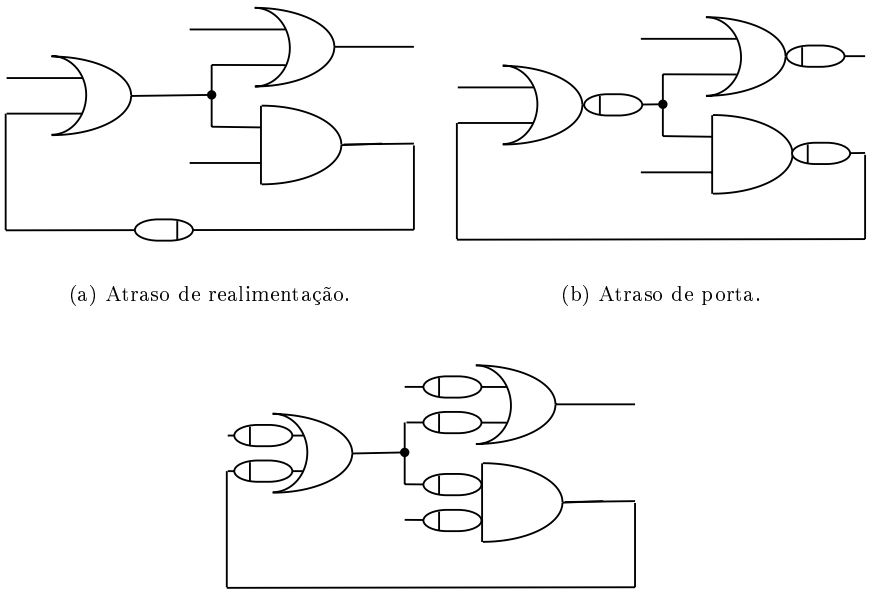
A Figura 5.9 compara o comportamento de atrasos inerciais binário e ternário, assumindo atrasos inteiros e magnitude δ unitária.

Note-se na Figura 5.9 que o atraso inercial ternário altera significativamente a forma de onda de entrada, em particular a largura relativa dos pulsos. A Figura também serve para ilustrar os conceitos de atraso excitado e estável introduzidos acima. O atraso binário está excitado nos intervalos 2, 3, 5 e 7, e o atraso ternário nos momentos 2, 3, 5, 6, 7 e 8.

5.2.3.2 Modelagem de Atrasos para Circuitos

Pressupostos para a distribuição de elementos atraso em uma descrição de um sistema digital assíncrono estabelecem modelos que influenciam diretamente o processo de projeto. A Figura 5.10, extraída de [117] ilustra os modelos de atraso de circuitos para descrições estruturais no nível lógico de abstração. Os tipos de modelo de atraso de circuito são:

- ***atraso de realimentação***, onde:
  - todo laço de realimentação contém pelo menos um elemento atraso;



(a) Atraso de realimentação.

(b) Atraso de porta.

(c) Atraso de fio.

Figura 5.10: Modelos de distribuição de atrasos em circuitos assíncronos.

  - substituir qualquer elemento atraso por um fio produz um circuito onde algum laço de realimentação não contém nenhum elemento atraso;

- ***atraso de porta***, onde existe exatamente um elemento atraso por saída de porta lógica;
- ***atraso de fio***, onde existe exatamente um elemento atraso por entrada de porta lógica.

A partir das cinco classificações de atrasos e das classificações de ambiente, pode-se combinar escolhas para determinar de forma unívoca um estilo assíncrono de projeto. Por exemplo, o trabalho seminal de Muller e Bartky [139] estabeleceu um estilo batizado de *circuitos independentes de velocidade* (do inglês, *speed-independent circuits*), que corresponde à combinação de três escolhas de modelo de atraso, gerando o *modelo de porta ilimitado inercial* (do inglês, *inertial unbounded gate delay model*). Várias outras combinações são obviamente possíveis, embora nem todas levem a um modelo coerente para dar origem a um estilo assíncrono relevante. A taxonomia da Seção 5.3 explora

algumas das combinações de modelos de atraso e de ambiente que conduzem a estilos de projeto de algum sucesso.

5.2.4 Problemas Temporais

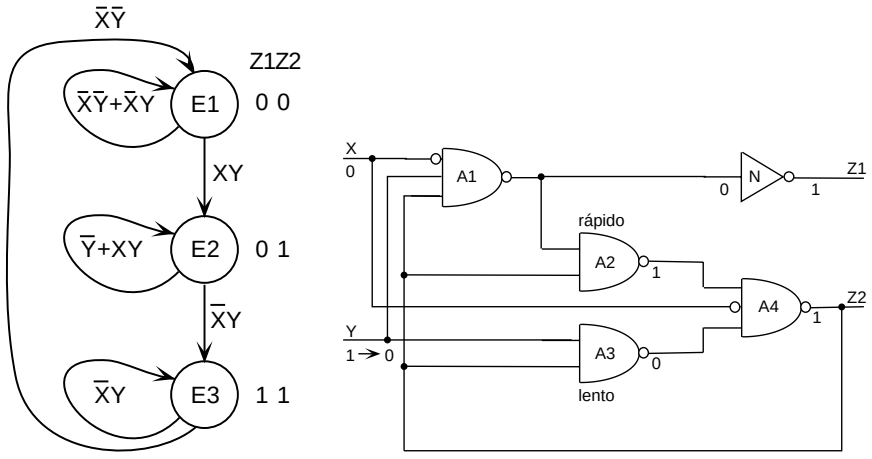
Nesta Seção, adianta-se a apresentação dos dois problemas temporais fundamentais afetando o projeto de sistemas digitais assíncronos, *corridas* e *transitórios*. Fornecemos aqui apenas a definição geral destes, deixando para seções posteriores seu tratamento e classificação. Corridas são problemas associados a laços de realimentação de um sistema, enquanto transitórios são falhas de funcionamento devidas à parte combinacional do sistema ou a blocos responsáveis por realizar tarefas de *arbitração* [196].

Definição 5.5 (Corridas) *Uma* ***corrida*** *entre dois ou mais sinais de estado (ver Definição 1.2) de um sistema digital assíncrono é uma mudança simultânea dos valores destes sinais devida a uma única transição de estado. Uma* ***corrida crítica*** *é aquela onde o estado estável alcançado pelo circuito após uma corrida depende do vencedor da corrida, ou seja, da ordem em que os sinais mudam. Usa-se também diretamente o termo inglês* race *para designar corridas.*

A Figura 5.11, retirada do livro de Hill e Peterson [98], ilustra um comportamento e uma implementação deste que pode gerar uma corrida.

A especificação da Figura 5.11(a) é um grafo de transição com três estados: $E1$, $E2$ e $E3$. Pode-se verificar facilmente que o circuito da Figura 5.11(b) está no estado estável $E3$, dadas as entradas $X = 0$, $Y = 1$ e as saídas $Z1 = 1$ e $Z2 = 1$. De acordo com a especificação, se ocorre a transição de entrada indicada na Figura, Y mudando para 0, o próximo estado estável é $E1$, e as saídas devem ser $Z1 = 0$ e $Z2 = 0$. Contudo, suponha a existência de atrasos desiguais, $A2$ sendo rápido e $A3$ sendo lento para efetuarem a mudanças de suas respectivas saídas. Neste caso, é possível que as portas $A1$ e $A2$ mudem suas saídas antes da porta $A3$, atingindo o estado estável $E2$ ($Z1 = 0$ e $Z2 = 1$), contrariando a especificação. Logo, a corrida em questão é crítica.

Definição 5.6 (Transitórios) *Um* ***transitório*** *ocorre quando a saída de uma parte combinacional de um sistema digital deveria, após uma transição de entrada, atingir um determinado nível lógico, mas devido a alguma combinação de atrasos a saída não se comporta de forma monotônica para esta entrada, produzindo transições indesejáveis de sinais. Alternativamente, pode-se definir transitório como qualquer mudança de valor de algum sinal de saída ou próximo estado que não ocorreria se os atrasos estivessem concentrados apenas nos laços de realimentação. Usa-se também diretamente o termo inglês* hazard *para designar transitórios.*



(a) Grafo de Transição (b) Circuito

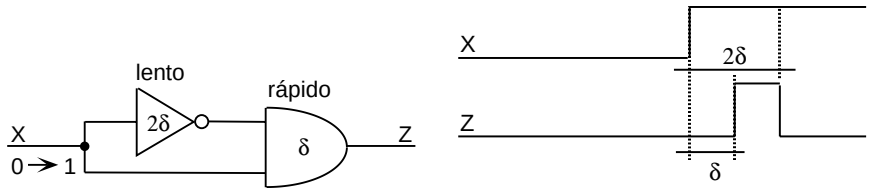
Figura 5.11: Uma especificação de comportamento e uma implementação capaz de gerar um problema de corrida.

Um exemplo simples de circuito passível de apresentar um transitório aparece na Figura 5.12. Obviamente, o nível lógico da saída Z jamais deveria ser diferente de 0. Entretanto, supondo que o inversor da Figura possui um atraso de magnitude 2δ o dobro da do atraso δ da porta E, o comportamento após uma transição de 0 para 1 como a indicada na Figura 5.12(a) gera o diagrama de tempos com um transitório de duração δ na saída Z.

Um estudo mais detalhado e uma classificação dos diferentes tipos de transitórios serão vistos na Seção 5.4.

5.3 Taxonomia Assíncronas

A multiplicidade de estilos assíncronos dificulta em muito as tentativas de estabelecer uma sistemática unificada de classificação de métodos de projeto. Embora praticamente toda classificação de métodos assíncronos tenha como base os pressupostos estabelecidos para o ambiente e os modelos de atraso, e embora existam classes consagradas de métodos, a forma de agrupar os métodos e a própria nomenclatura, bem como sua interpretação, variam enormemente. Alguns exemplos de classificação são resumidos a seguir, visando ilustrar este ponto. A revisão de Brzozowski, Seger e Hauck propõe a divisão de estilos assíncronos em seis classes de métodos [34]:



(a) Circuito (b) Diagrama de Tempos

Figura 5.12: Um circuito combinacional sujeito a transitórios.

- métodos que assumem o modelo de *atraso limitado* (*bounded delay*), conforme descrito na Página 236;
- a classe de métodos *independentes de velocidade* (*speed-independent*), já citados na Página 238, onde assume-se atrasos finitos mas de magnitude arbitrária nas portas lógicas, e os fios como elementos ideais;
- métodos *insensíveis a atrasos* (do inglês, *delay-insensitive*), onde tanto portas quanto fios podem apresentar atrasos finitos de magnitude arbitrária;
- métodos que empregam *especificações com alto nível de abstração*, tais como CSP [100] e a teoria de *traces* [183];
- métodos miscelâneos, incluindo como exemplos os *micropipelines* de Sutherland [171] e métodos *auto-temporizados* de projeto (do inglês, *self-timed*), onde sinais de relógio são gerados localmente nos blocos do sistema, que por sua vez se comunicam de forma assíncrona.

Por outro lado, Lavagno e Sangiovanni-Vincentelli sugerem uma taxonomia similar, composta de quatro classes [117]:

- métodos que assumem o *modelo de Huffman*. Estes incluem os métodos de atraso limitado e outros, como os métodos *auto-sincronizados* [156];
- micropipelines;
- métodos independentes de velocidade;
- métodos insensíveis a atrasos.

As três últimas classes são equivalentes às classes ou subclasses de mesmo nome da taxonomia anterior. Esta classificação inclui os métodos que utilizam especificações de alto nível no grupo que mais se adapta ao modelo de atrasos

subjacente, freqüentemente o de sistemas insensíveis a atrasos. Os métodos auto-temporizados não são mencionados em nenhuma das classes.

Uma terceira classificação é proposta por Staunstrup em [168]. Ainda bastante similar às anteriores em termos funcionais, a nomenclatura aqui é bastante distinta, dando diferente interpretação para alguns dos termos usados nas anteriores. Staunstrup inicialmente não emprega o termo assíncrono, consagrado na maioria dos trabalhos, optando por batizá-los de sistemas digitais auto-temporizados (*self-timed*), o que entra em conflito direto com trabalhos que utilizam esta terminologia para classificar classes restritas de métodos assíncronos. Rosenblum e Yakovlev por exemplo, usam o termo self-timed como sinônimo do que acima foi definido como speed-independent [157]. Embora seja admissível considerar que o termo "assíncrono" transmite uma idéia falaciosa da real natureza do comportamento dos sistemas digitais alvo de estudo deste Capítulo, outros termos sugeridos também têm limitações na expressão semântica dos comportamentos envolvidos. Adicionalmente, estas novas propostas de terminologia não são de uso difundido, gerando mais confusão que esclarecimento. Hill e Peterson, por exemplo, usam a denominação *circuitos seqüenciais em modo nível* (em inglês, *level-mode sequential circuits*) [99], o que parece simplesmente bizarro. Voltando à classificação de Staunstrup, são também quatro as classes propostas por este autor para regrupar sistemas assíncronos:

- métodos que empregam relógios locais, ou seja, equivalente a uma das subclasses do modelo de Huffman da proposta anterior;
- métodos de atrasos limitados, idêntico à primeira classe da primeira taxonomia acima;
- métodos independentes de velocidade, que Staunstrup batiza, de acordo com o modelo de atraso fundamental, de *gate-delay independent*;
- métodos insensíveis a atrasos, batizados tabém de acordo com o modelo de atraso, de *gate-delay/wire-delay independent*.

Outras classificações mudam radicalmente de enfoque, considerando em primeiro lugar a forma de especificação do sistema assíncrono, e de maneira marginal o modelo de atraso subjacente.

Um primeiro exemplo destas novas classificações é a proposta de Myers e Meng [142], que separa métodos de projeto assíncronos em duas grandes classes:

- métodos e sistemas *temporizados*, onde restrições quantitativas precisas sobre os valores de atraso são usados ao longo do processo de projeto;
- métodos e sistemas independentes de velocidade, todos os demais.

Note-se que nesta classificação, o conceito anteriormente introduzido de independência de velocidade é expandido para compreender o conceito de insensibilidade a atrasos, a auto-temporização e a auto-sincronização, entre outros métodos.

Por sua vez, Vanbekbergen, Goossens e Lin produziram uma classificação similar, com nomenclatura distinta [185]. Para eles, métodos e sistemas assíncronos podem se classificar em:

- métodos e sistemas temporizados, com a mesma interpretação dada por Myers e Meng;
- métodos e sistemas auto-temporizados, fornecendo mais uma interpretação distinta para o termo *self-timed*.

Das cinco classificações acima, extrai-se uma conclusão simples: não há consenso quanto a como classificar métodos de projeto assíncronos e os sistemas implementados a partir destes. Procura-se agora estabelecer não uma classificação nova, mas listar os *critérios* de classificação mais relevantes e com base neles derivar exemplos de classificações parciais, ortogonais e coerentes.

5.3.1 Uma Proposta de Critérios e Classificações

Lembrando a Definição 5.1 de circuitos digitais assíncronos da Página 219, é importante salientar que esta Seção aplica-se igualmente a sistemas e métodos síncronos, embora este não seja no momento o foco da discussão. Empregam-se quatro critérios para classificar métodos e sistemas assíncronos:

- o modelo de atrasos, o critério mais importante;
- a quantificação de atrasos;
- a forma de especificação do projeto;
- a quantidade de sinais de relógio.

O modelo de atrasos condiciona as escolhas descritas na Seção 5.2 para representar informação e escolher o modelo de ambiente. Quanto à combinação destas escolhas com o modelo de atrasos, sistemas digitais podem ser classificados em:

- sistemas de *atrasos limitados* - são projetados via métodos que envolvem a consideração de limites para as magnitudes dos atrasos, limites que determinam a corretude de uma implementação quando respeitados. Neste grupo, incluem-se os métodos clássicos, tais como os baseados no modelo de Huffman [182, 106] e os sistemas auto-sincronizantes, entre outros. Aqui pode-se também colocar os métodos síncronos, pois a determinação

da freqüência de operação do relógio depende de uma análise cuidadosa do caminho crítico de atrasos do sistema digital;

- sistemas *micropipelines* [171] - são na realidade a combinação de uma parte de dados projetada com um modelo de atraso limitado e uma parte de controle engenhosamente projetada para ser independente de valores relativos ou absolutos de atraso;

- sistemas *independentes de atrasos* nos elementos de processamento - equivalente à classificação de métodos e sistemas speed-independent citados anteriormente, considera que atrasos de portas são ilimitados, embora finitos. Atrasos nos condutores são considerados nulos, o que não os torna atraentes para tecnologias futuras, onde se prevê a dominância de atrasos de fios sobre os atrasos nos elementos de processamento;

- sistemas *totalmente insensíveis a atrasos* - assumem que todos os elementos do sistema podem ter qualquer valor finito de atraso. Embora sendo a classe mais robusta de métodos, equivalente aos anteriormente citados sistemas delay-insensitive, nem todas especificações de sistemas digitais possuem uma implementação totalmente insensível a atrasos. Na verdade, a classe de sistemas que possuem tal implementação é muito limitada, como demonstram os resultados de Unger em [182] (Se uma tabela de fluxo possui um transitório essencial[4], ela não admite uma uma implementação totalmente insensível a atrasos) de Patil e Dennis em [148] (em um sistema totalmente insensível a atrasos construído apenas com portas lógicas convencionais, todas as portas devem participar em todos os laços de realimentação do sistema) e Martin em [129] (todo sistema totalmente insensível a atrasos que executa computações não-triviais só pode conter inversores e os chamados elementos-C[5]).

A quantificação de atrasos, quando disponível, leva normalmente a implementações mais eficientes que métodos que não empregam quantificação [142], mas que implicam trabalhar em níveis de abstração mais baixos, e que são conseqüentemente mais complexos e menos escaláveis. Quanto à quantificação de atrasos, classificam-se sistemas digitais em:

- sistemas temporizados - aqueles onde o método de projeto pressupõe a especificação e controle de valores precisos para os atrasos, seja para os associados a transições de sinais, seja aos relacionados à propagação de valores da entrada para a saída do sistema;

[4] Transitórios essenciais são descritos na Seção 5.4.
[5] Elementos-C são descritos na Seção 5.5.

- auto-temporizados - aqueles onde o método de projeto, quando aplicado, é capaz de estruturar a implementação de forma a respeitá-la independente dos valores de atrasos relativos ou absolutos. Sistemas digitais síncronos caem nesta categoria, pois durante o projeto não é necessário se preocupar com com atrasos relativos ou absolutos de sinais, exceto para fins de aumento de desempenho e determinação da freqüência do relógio.

A forma de especificação do sistema digital envolve os usuais estudos de compromissos de desempenho, custo e tempo de projeto. Quanto a este critério, classifica-se sistemas digitais como especificados via:

- *linguagens* - a exemplo do que já ocorre há algum tempo com estilos síncronos de projeto, esta é a classe vista como mais promissora para lidar com sistemas assíncronos VLSI do futuro, seja na originalmente síncrona VHDL [175] ou em linguagens mais adequadas como Occam [178] e Synchronized Transitions [168];

- *máquinas de estados finitas* - a forma mais tradicional, constituída de representações baseadas em estados, como as tabelas de fluxos AFTs;

- *modelos baseados em eventos* - mais modernamente, modelos baseados em eventos têm sido empregados para explicitamente representar além de estados, os eventos separados associados às transições entre estes. Exemplos são as redes de Petri ou sua versão para estilos assíncronos, os Grafos de Transição de Sinais (ASTGs) [117], o Sistema de Regras e Eventos (do inglês, *event-rule system*) de Burns [38, 142] e representações usando *traces* [183].

Finalmente, o critério quantidade de sinais de relógio estabelece o grau de sincronismo interno do sistema, sendo o estilo síncrono de projeto o caso extremo. Sistemas digitais podem ter:

- nenhum sinal de relógio - esta é a característica distintiva dos sistemas assíncronos *puros*;

- um sinal de relógio único - obviamente aqui se encontram os sistemas e métodos síncronos, mesmo aqueles compostos por relógios de múltiplas fases, pois estes podem ser considerados como um entrelaçamento de diversas instâncias do mesmo sinal transladado no tempo por diversos intervalos distintos;

- múltiplos sinais de relógio - caracterizando os métodos de interconexão assíncrona de blocos localmente síncronos (denominados em inglês *self-clocked*).

Com esta Seção , conclui-se a exposição do amplo quadro de modelos, e características de sistemas digitais assíncronos. Com tal contexto estabelecido, pode-se então explorar métodos de projeto específicos. Antes contudo, é necessário caracterizar o principal problema associado a sistemas sem sinal global de relógio, os transitórios ou hazards.

5.4 Caracterização de Transitórios

A Definição 5.6 estabeleceu uma primeira abordagem do problema de sinais espúrios em um sistema digital. Aqui, detalha-se o conceito e apresenta-se uma classificação destes fenômenos. Na realidade, sob a denominação de transitório existem diversos fenômenos, que podem possuir diversas origens e exigem diferentes técnicas para serem evitados, e quando isto não for possível, viável ou computacionalmente factível, diversas formas de contornar seus efeitos maléficos.

Informalmente, um transitório ou hazard é qualquer conjunto de transições inesperadas de sinais internos ou de saída de um sistema digital. Pode-se avançar dois critérios de classificação de transitórios:

- o número de transições envolvidas no hazard - duas situações são possíveis, um número par ou um número ímpar de transições, gerando os seguintes tipos de transitório:
  - ***transitório estático*** - ocorre se um sinal deveria permanecer constante, mas muda duas vezes em direções opostas. Estes podem ser subdivididos em duas outras classes:
    - ***0-hazard*** - se o valor constante do sinal deveria ser 0;
    - ***1-hazard*** - se o valor constante do sinal deveria ser 1.
  - ***transitório dinâmico*** - ocorre se um sinal deveria mudar uma vez, mas muda um número maior, ímpar de vezes (no mínimo três).
- a parte do sistema que gera o transitório - duas situações são possíveis, geração associada à parte combinacional ou às linhas de realimentação:
  - ***transitório combinacional*** - é um transitório que pode ou não ocorrer, dependendo da distribuição de atrasos no sistema. Estes podem ser subdivididos em duas outras classes:
    - ***hazard lógico/estrutural/de escopo/ocasional*** - são aqueles que dependem da implementação particular da função lógica. As quatro denominações são sinônimos;
    - ***hazard de função ou intrínseco*** - são os que não podem ser eliminados, a despeito de atrasos, alterando-se a implementação.

- ***transitório seqüencial*** - são os que inerentemente envolvem os laços de realimentação. Existem aqui duas subclassificações. A primeira considera a possibilidade de eliminação do transitório:
  - ***hazard essencial*** - são aqueles inerentes à AFT (tabela de fluxo, Definição 3.11, Página 124) que descreve o sistema;
  - ***hazard não-essencial*** - aqueles não-inerentes à AFT, tais como os que ocorrem devido a corridas críticas, e que podem ser eliminados por uma codificação de estados adequada.

  A segunda subclassificação diz respeito ao estado final alcançado quando da ocorrência do transitório em questão:
  - ***hazard transiente*** - aquele hazard seqüencial cujo efeito não muda o estado final total do sistema, após a estabilização;
  - ***hazard de regime*** - ao contrário do anterior, quando presente, muda o estado estável com relação à especificação da AFT.

Como exemplo desta classificação, o transitório mostrado na Figura 5.12 é um transitório estático, do tipo 0-hazard.

Algumas observações sobre as propriedades dos diferentes tipos de transitórios devem ser colocadas. Primeiro, transitórios combinacionais possuem várias propriedades bem estabelecidas [182]. Um implementação em lógica dois níveis de qualquer função lógica pode ser tornada livre de 1-hazards estáticos sob o modo mudança única de entrada, simplesmente incluindo todos os implicantes primos na implementação. Note-se que esta abordagem é muito cara em termos de recursos, uma vez que no caso geral, o número de implicantes primos de uma função de n variáveis é $\mathcal{O}(3^n)$ [61]. Existe uma série de transformações da lógica do sistema que não incluem novos efeitos transitórios assumindo operação no modo fundamental. Exemplos são as leis de de Morgan, as leis associativa, distributiva e de absorção. Finalmente, quase nenhuma função lógica possui uma implementação sem transitórios combinacionais sob o modo múltipla mudança de entradas.

Quanto aos transitórios seqüenciais, também eles tiveram várias de suas propriedades demonstradas por Unger, em [182]. Por exemplo, qualquer AFT sem transitórios essenciais admite uma implementação livre de hazards de regime, sem necessidade de acrescentar elementos de atraso, caso opere no modo mudança única de entrada. Nenhuma AFT com transitórios essenciais possui uma implementação sem hazards de regime, independente dos atrasos de realimentação. Este último resultado é a primeira prova de que algum comportamento não possui *nenhuma* implementação insensível a atrasos.

Como conclusão, transitórios são o principal e mais difícil problema a ser evitado durante a síntese de sistemas digitais sem relógio.

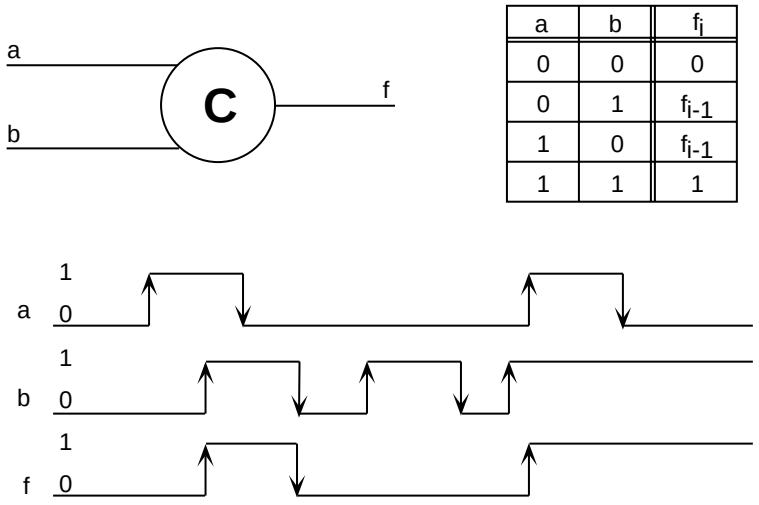


Figura 5.13: O elemento-C e seu comportamento como o E de eventos. Quando as entradas do elemento-C são iguais, a saída apresenta o mesmo valor que as entradas, caso contrário, a saída mantém seu valor anterior.

5.5 Um Estilo de Projeto Assíncrono Moderno

Escolheu-se explorar nesta Seção o estilo de projeto denominado micropipelines, devido a Ivan Sutherland [171]. A escolha deriva de uma série de motivações. A principal destas é o atual sucesso deste estilo, com vários trabalhos relatando a implementação de sistemas digitais mediante seu uso. O mais significativo destes esforços é a implementação do atualmente único microprocessador assíncrono comercial, o AMULET2e [80]. Além disto, micropipelines misturam com sucesso técnicas distintas de projeto assíncrono. Seu projeto baseia-se no tipo de informação a manipular, com insensibilidade a atrasos na parte de controle e manipulações eficientes de pacotes de dados através do uso de protocolos que seguem o modelo de atraso limitado.

A apresentação do método está dividida em três partes. A primeira apresenta, na Seção 5.5.1, a escolha de restrições a aplicar ao projeto de sistemas digitais para caracterizar o estilo micropipeline. Em seguida, a Seção 5.5.2 introduz as técnicas de implementação de controle em micropipelines. Finalmente, discute-se a construção de sistemas digitais micropipeline, na Seção 5.5.3.

5.5.1 Pressupostos do Estilo Micropipeline

Como já foi mencionado na Seção 5.2.1, a modelagem da representação de informação determina em grande parte um estilo de projeto assíncrono. Este é o caso de micropipelines. Primeiro, micropipelines assumem uma codificação física do tipo trilha única na parte de controle. Dados podem ser manipulados seja em trilha única, seja em trilha dupla. Segundo, o assinalamento de informação é por transição, usando uma lógica de eventos. Finalmente, o protocolo de comunicação pressuposto por micropipelines é do tipo duas fases, denominado por Sutherland de duas fases com dados empacotados (em inglês, *two-phase bundled data*). Estas escolhas foram devidamente exploradas ao longo da Seção 5.2.1, à exceção de um maior detalhamento dos operadores da lógica de eventos e sua implementação em hardware. Explora-se agora este tópico em mais detalhe.

A Figura 5.5 ilustrou um primeiro operador da lógica de eventos, o OU de eventos, cuja implementação é realizada mediante uso da porta OU-exclusivo tradicionalmente empregada em assinalamento de informação por nível. Este é um operador que produz um evento de saída cada vez que algum evento ocorre em alguma de suas duas (ou mais) entradas (por isto mesmo, o operador possui o nome alternativo de combinador de eventos ou MERGE). Além deste, é útil dispor de um operador dual, capaz de gerar um evento de saída somente após ter havido a ocorrência de eventos em todas suas entradas (duas ou mais). Tal operador recebe adequadamente o nome "E de eventos". A implementação deste operador não é tão simples como no caso do OU de eventos, pois seu comportamento não pode ser implementado como um circuito combinacional. De fato, o OU de eventos é o único dos operadores desta lógica que possui uma implementação combinacional. A implementação do E de eventos é feita através do chamado elemento-C de Miller (em inglês, Miller C-element) [137]. A Figura 5.13 mostra o símbolo, a tabela verdade modificada e um exemplo de diagrama de tempos do comportamento deste componente, assumindo uma instância com duas entradas.

Note-se que o comportamento do componente E pressupõe a alternância de eventos nas entradas. Se esta alternância não se verifica, ou seja, dois eventos consecutivos ocorrem na mesma entrada, estes cancelam-se em termos de efeito sobre a saída do elemento-C. O elemento-C, sendo seqüencial, necessita de inicialização explícita para garantir seu funcionamento de acordo com a especificação. Ou seja, uma entrada externa adicional de inicialização deve ser provida no componente, sendo sua representação quase sempre omitida.

Os operadores restantes da lógica de eventos sugeridos por Sutherland em [171] são ilustrados na Figura 5.14.

Perceba-se que todos estes novos operadores, a exemplo do elemento-C possuem comportamento seqüencial, pois necessitam memorizar alguma informação do passado do sistema. Note-se a semelhança de alguns operadores

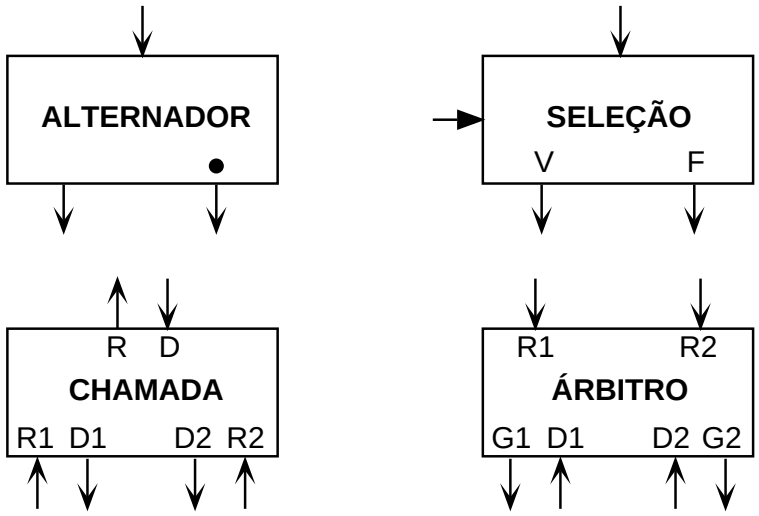
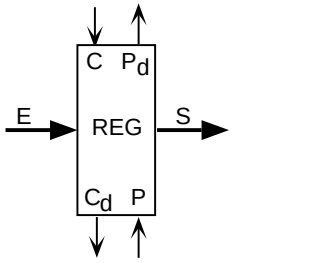


Figura 5.14: Outros operadores da lógica de eventos. O ALTERNADOR (em inglês, *TOGGLE*) transmite eventos nas suas entradas alternativamente para cada uma das saídas, iniciando com a saída marcada. SELEÇÃO (em inglês, *SELECT*) transmite eventos de sua entrada superior para uma das saídas, baseado no nível lógico da entrada de controle lateral. CHAMADA (em inglês, *CALL*) lembra-se de qual cliente chamou o procedimento R, R1 ou R2, e após o término do procedimento, indicado por evento em D, retorna um evento em D1 ou D2, avisando o procedimento correspondente. ÁRBITRO (em inglês, *ARBITER*) fornece serviço G1 ou G2 a apenas uma requisição de cada vez, R1 ou R2, atrasando o fornecimento de outro(s) serviço(s) até a ocorrência de um evento indicando término de serviço fornecido, D1 ou D2.

com construções de linguagem de programação convencionais (CHAMADA) e paralelas (ÁRBITRO). Sutherland mostra implementações CMOS estática e dinâmica para o elemento-C, donde conclui-se que a complexidade de implementações de E e OU de eventos são similares, equivalentes cada a algo como 2 portas AND de lógica convencional ou 3 portas NAND (12 transistores em lógica estática). Os operadores restantes são mais complexos, mas Sutherland afirma que cada um dos operadores pode ser implementado em CMOS com um circuito contando com não mais que uma centena de transistores. Assim, estes operadores pode implicar na sua construção o uso de até uma ordem de grandeza a mais de transistores que os componentes E e OU de eventos.

Os seis componentes da lógica de eventos constituem um conjunto suficiente para representar qualquer comportamento de circuitos (assíncronos) no



C	P	Função
0	0	Transparente baixo
0	1	Armazena alto
1	0	Armazena baixo
1	1	Transparente alto

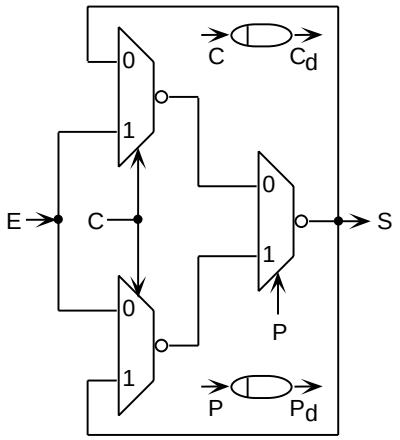


Figura 5.15: Estrutura de um elemento de memória controlado por eventos. O símbolo é similar ao de um registrador convencional, mas possui duas entradas de controle e duas saídas de controle, ao invés de um único sinal de relógio. O comportamento é dado pela tabela, em função dos sinais de controle C e P. Alto e baixo na tabela referem-se aos laços de realimentação formados pelo multiplexador de saída e um dos outros dois. As saídas C_de P_d assinalam com um evento o término de operações solicitadas nas entradas C e P, respectivamente. E e S são entrada e saída de dados. O diagrama à esquerda mostra uma implementação de um bit de memória usando multiplexadores 2:1 e inversores, além de elementos atraso.

nível lógico de abstração, a exemplo do que ocorre com as portas lógicas E/OU/Inversor da lógica Booleana tradicional. Digna de nota é a ausência de um elemento "inversor" de eventos, pois não faz sentido falar de "não-eventos". Não obstante, inversores são usados na lógica de eventos para alterar as condições de funcionamento de algum componente, por exemplo em uma das entradas ou na saída de um elemento-C.

Além de operadores, o estilo de projeto micropipeline necessita de um dispositivo adequado para armazenar informação. Embora os componentes da lógica de eventos sejam seqüenciais em sua maioria, isto é uma conseqüência de sua funcionalidade, não de uma estratégia explícita de armazenamento de informação. O armazenamento de dados em micropipelines necessita de um tipo especial de dispositivo, com funcionalidade equivalente a registradores da lógica síncrona, mas com estrutura de controle distinta. Sutherland sugere implementações de tal elemento, que ele denomina de elemento de memória controlado

por eventos (do inglês, *event-controlled storage element*). A Figura 5.15 mostra o símbolo deste componente, uma tabela que descreve seu comportamento e a estrutura interna de um bit de memória.

A Figura usa multiplexadores 2:1 para simbolizar chaves controladas. Círculos nas saídas destes representam inversores. Note-se que este elemento ora funciona ora como um *latch* transparente, ora como um elemento de armazenamento, dependendo da combinação de valores dos sinais de Captura e Passagem (C e P). No símbolo mostrado, C_d e P_d são saídas, assinalando o término da operação solicitada em C e P, respectivamente. A implementação de C_d e P_d pode consistir de apenas um elemento atraso aplicado sobre C e P, respectivamente, conforme mostrado na Figura 5.15. Estes sinais são necessários ao correto funcionamento da sincronização de eventos em micropipelines, como será visto na Seção 5.5.3. Sutherland explora outras estruturas de implementação destes componentes apresentando diferentes compromissos de custo e desempenho.

5.5.2 Implementação de Controle em Micropipelines

A composição de elementos-C em cascata, com a interposição de inversores é a construção de controle fundamental para micropipelines. Um exemplo de implementação é mostrada na Figura 5.16.

O circuito da Figura 5.16 funciona como duas filas entrelaçadas e em sentidos opostos de eventos (implementados como transições de valores Booleanos). Veja-se um exemplo que explica o funcionamento desta fila. Suponha todos os fios da Figura 5.16 inicialmente em 0. Esta situação é estável, pois todos os elementos-C possuem entradas distintas (devido ao inversor em uma de suas entradas), mantendo o valor anterior, no caso o valor inicial 0. Suponha agora a chegada de uma requisição de serviço do transmissor, sob a forma de uma transição de 0 para 1 na entrada R_e. Esta transição vai gerar uma transição sucessivamente em todos os fios, a última sendo na entrada do receptor, R_s (vide comportamento do elemento-C). Este último evento faz com que o receptor inicie a prestação do serviço segundo o protocolo de comunicação duas fases (Figura 5.6(a)). Existem duas possibilidades para o próximo evento, um novo pedido de serviço pelo transmissor em R_e ou o aviso de prestação de serviço concluída. Suponha que o primeiro evento ocorre. O leitor pode verificar que neste caso, todos os fios sofrem nova transição, exceto um, pois o último elemento-C antes do receptor não poderá transicionar enquanto o reconhecimento de serviço prestado não chegar, sob a forma de uma transição em A_s.

Ao todo, a estrutura pode armazenar $(n+1)$ requisições de serviço, sem receber um reconhecimento de prestação, antes de começar a perder informações, onde n é o número de elementos-C (estágios) do micropipeline. A saída A_e avisa ao transmissor, ao não transicionar após uma requisição chegar em R_e, que

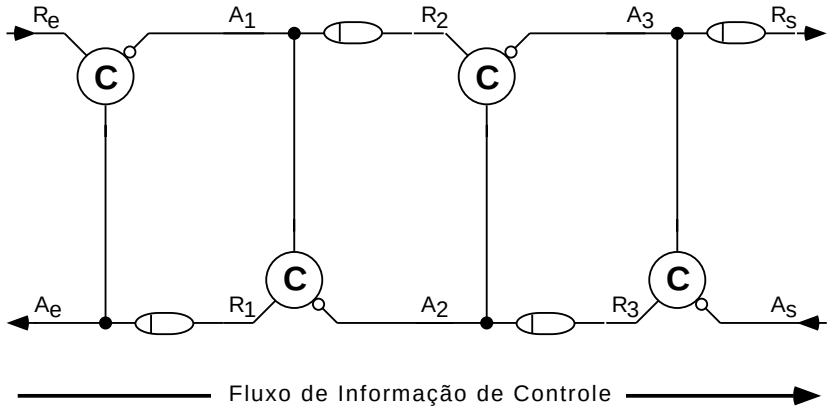


Figura 5.16: Estrutura de controle fundamental em micropipelines. À direita da estrutura está o transmissor de informação de controle, à esquerda o receptor. A estrutura funciona como duas filas: uma de requisições de serviço (cadeia de R_is) e uma de reconhecimento de serviço prestado (cadeia de A_is). Os elementos atraso podem ser dispensados se a manipulação de dados (não mostrada) for simples. Cada elemento-C possui a saída de um inversor conectado à uma de susas entradas, indicado pelo círculo vazado.

próximas requisições serão perdidas. Assim, ambos os lados da estrutura podem controlar o fluxo de informação, sem necessidade de qualquer sincronização externa.

O número de estágios no micropipeline dá o tamanho da fila entre transmissor e receptor, e pode ser calculado para prover uma adaptação entre dispositivos de desempenhos distintos.

5.5.3 Estrutura de Sistemas Digitais Micropipeline

A arquitetura mostrada na Seção 5.5.2 permite modelar uma variedade de implementações do tipo pipeline. Claramente, filas elásticas (com quantidade variável de elementos) são uma aplicação imediata e muito útil, mas existem outras. Existem construídos com micropipelines sistemas tais como multiplicadores, decodificadores, controladores de memória [171], unidades lógico-aritméticas [83] e até um microprocessador completo [80].

Aqui, mostra-se apenas a estrutura geral de uma implementação micropipeline. O leitor interessado pode consultar as referências sobre o tema. A estrutura em questão emprega como componentes apenas elementos-C, inversores e elementos atraso para a lógica de controle, e elementos de memória controlados por eventos e portas lógicas convencionais para a parte de processamento de da-

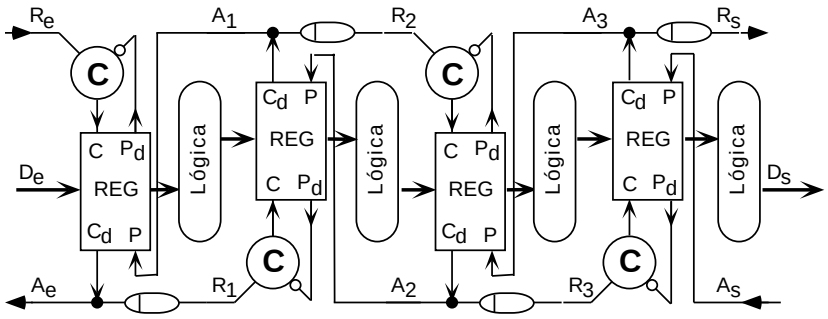


Figura 5.17: Estrutura genérica de um sistema digital micropipeline. Todos os estágios são idênticos, contendo memória e lógica para processar os dados armazenados nesta, e gerando novos dados para o próximo estágio. Uma fila elástica seria implementada com a lógica sendo a função identidade, implementada como um conjunto de fios. Note que quando a fila está vazia, a produção de novo dado em D_e implica que este atravessará toda a estrutura (memórias funcionando como *latches* transparentes, exceto a última) e será alojado na memória mais próxima do consumidor.

dos. Funções de controle mais complexas podem ser implementadas usando os módulos mostrados na Figura 5.14 e portas OU-exclusivo além das já citadas. A Figura 5.17 mostra a estrutura genérica de um sistema digital micropipeline.

Um ponto importante em implementações micropipeline é que nem todos os estágios do micropipeline precisam conter elementos de memória. Pode-se computar um compromisso entre a complexidade da lógica entre elementos de memória e o número de estágios, estabelecendo um balanço entre latência do pipeline (latência é proporcional ao número de elementos de memória que os dados devem atravessar) e taxa de produção de resultados (em inglês, *throughput*, proporcional ao tempo de computação da lógica mais lenta entre dois elementos de memória consecutivos).

Entre as vantagens deste estilo assíncrono de projeto encontram-se o fato de se tratar de uma poderosa maneira de se implementar pipelines [96], um grande problema quando não se emprega relógios. Contudo, o desempenho ainda é, como no caso de circuitos síncronos, de pior caso, pois se adicionam atrasos em vários pontos para considerar atrasos grandes. Além disso, como existem vários pressupostos sobre atrasos, um teste por falhas de temporização deve ser feito, reduzindo o grau de abstração de projeto [93].

5.6 Conclusões do Capítulo

A principal conclusão estratégica resultante da leitura deste Capítulo deve ser que a conjuntura atual de desenvolvimento de sistemas VLSI não permite mais prescindir da consideração de técnicas oriundas de estilos de projeto assíncronos. Ou seja, conclui-se especificamente que os projetista VLSI precisam hoje ser formados em estilos assíncronos de projeto, não apenas no estilo síncrono. O objetivo é capacitá-lo a reconhecer as situações em que um determinado estilo de projeto assíncrono pode ser uma alternativa atraente, ou mesmo aquela que melhor satisfaz a especificação.

Além da necessidade de resolver o complexo problema de formação de recursos humanos competentes em estilos de projeto assíncronos, outra conclusão é que existe também uma carência enorme de métodos e ferramentas para lidar com os fenômenos assíncronos. Embora algum ferramental de análise e projeto já exista para os níveis de abstração lógico e físico, pouco foi feito nos níveis superiores de abstração. Enquanto esta situação perdurar, dificilmente os métodos assíncronos penetrarão no meio industrial, onde o projeto síncrono conta já com um corpo formidável de métodos altamente eficientes para realizar projeto em níveis de abstração bastante altos.

Conclui-se também aqui que métodos de síntese e análise assíncronos são intrinsecamente mais complexos que os correspondentes síncronos, devido ao tempo não ser discretizado nos primeiros. Logo, não faz sentido esperar que a comunidade de projeto opte por métodos assíncronos enquanto estes não fornecerem uma relação custo-benefício significativamente maior que os métodos do estilo síncrono. Esta situação pode ocorrer em breve com o aumento da densidade de integração causando a impossibilidade de manter o escorregamento do sinal de relógio dentro de valores aceitáveis.

Não obstante, os estilos assíncronos de projeto possuem atrativos inegáveis no contexto tecnológico atual, e há indicações que estes atrativos se tornarão cada vez mais relevantes a curto e médio prazo. A curto prazo, a possibilidade já demonstrada de economizar energia pelo uso de um estilo assíncrono de projeto promete fomentar ainda mais a pesquisa no tema, o mesmo ocorrendo com o aumento de desempenho dos sistemas, como resultado da superação do comportamento de pior caso típico do estilo síncrono. Contudo, as tendências apontam não para o abandono do estilo síncrono, mas para a mistura deste com técnicas assíncronas de projeto em algum grau, pelo menos a curto prazo.

Da maior importância são os aspectos que indicam serem alguns estilos assíncronos de projeto mais propensos que o estilo síncrono à escalabilidade (para uma definição de escalabilidade, ver nota de rodapé na Página 6), à migração tecnológica e ao projeto em altos níveis de abstração, esta última característica devida à possibilidade de adaptacção automática de sistemas assíncronos a propriedades físicas mutantes. Estes aspectos estão relacionados à facilidade de realização de um projeto de sistema digital, à portabilidade do

projeto, e ao esforço necessário para revisar um projeto existente.

A generalização do emprego industrial de sistemas digitais sob a forma de propriedade intelectual (IP, do inglês, *Intellectual Property*) também pode vir a favorecer estilos assíncronos. Trata-se de sistemas digitais cujo projeto está pré-validado em um ou mais níveis de abstração. Aqueles que estão validados em altos níveis de abstração chamam-se *soft cores*, e necessitam passar tipicamente por fases de síntese lógica e física. Aqueles validados em praticamente todos os níveis de abstração são chamados *hard cores*. Os primeiros são mais flexíveis, os últimos de mais alto desempenho. Cores são módulos pré-projetados de alta complexidade utilizáveis como blocos construtivos de aplicações específicas. Exemplos de cores hoje disponíveis são interfaces para barramentos como PCI e PCMCIA, microprocessadores e processadores digitais de sinais ou DSPs.

5.7 Perspectiva Histórica e Bibliografia

Após um período de florescimento nos anos 50 e 60, estilos assíncronos de projeto de sistemas digitais caíram no esquecimento por cerca de 20 anos. Os trabalhos clássicos que lançaram as bases destes estilos datam daquele primeiro período. Unger [182] desenvolveu diversas técnicas de projeto baseadas em AFTs e gerou um corpo de resultados teóricos gerais que até hoje influenciam a proposta de métodos. Miller e Huffman [106, 105] foram outros pesquisadores que trouxeram significativas contribuições, seja no campo de métodos, seja no de propostas de componentes e modelos teórico-práticos de representação. Tracey publicou um importante trabalho que mais tarde serviu como elo de ligação entre problemas de síntese assíncrona e síntese síncrona [180].

A era moderna de pesquisa em circuitos assíncronos teve origem em uma razoável quantidade de locais nos Estados Unidos, Japão e Europa. Digno de nota são o trabalho de Alain J. Martin e sua equipe no California Institute of Technology (Caltech), que no final dos anos 80 implementou um microprocessador assíncrono, atraindo atenção para o tema [130]. Furber e seus colaboradores na Universidade de Manchester na Inglaterra têm trabalhado por mais de 10 anos na implementação de microprocessadores, sendo responsáveis pelo projeto do único microprocessador assíncrono comercialmente disponível, após várias versões acadêmicas. O trabalho desenvolvido em conjunto pela Universidade de Eindhoven e a Philips produziu um grande número de resultados teóricos na modelagem de sistemas digitais assíncronos baseado em paradigmas de processamento paralelo como CSP [100] e a teoria de traces [183]. Luciano Lavagno e colaboradores [117] na Universidade de Berkeley desenvolveram ferramentas baseadas na manipulação de descrições baseadas em Redes de Petri de sistemas assíncronos, obtendo métodos eficientes para o projeto em baixo nível de abstração de assíncronos, trabalho mais tarde melhorado por pesquisadores do IMEC, na Bélgica, [196]. A lista é muito longa para enumerar todos os

atuais esforços no sentido de estudar e propor métodos assíncronos de projeto de sistemas digitais.

Mais importante que listar os diferentes trabalhos em andamento é avaliar a tendência de quais aspectos de projeto assíncrono devem ser enfatizados. Métodos baseados em descrições com alto nível de abstração e alto desempenho ainda são muito raros em estilos assíncronos de projeto. Praticamente inexistem ferramentas e bibliotecas de dispositivos para o projeto assíncrono. Modelos como total insensibilidade a atrasos seriam ideais para viabilizar o projeto abstrato, mas são fortemente limitados. Falta estudos teóricos e experimentação prática antes de viabilizar plenamente estilos assíncronos competitivos.

5.8 Resumo do Capítulo

O Capítulo iniciou com a apresentação das vantagens e desvantagens do estilo síncrono de projeto com relação aos estilos assíncronos, avaliando uma grande quantidade de critérios a considerar. Seguiu-se um estudo aprofundado de modelos e problemas subjacentes aos estilos assíncronos de projeto, dividido em seus quatro aspectos mais relevantes:

- os modelos de representação de informação - subdivididos em codificações físicas, formas de assinalamento e protocolos de comunicação;
- os modelos de ambientes - são os pressupostos fundamentais sobre o mundo externo que permeiam cada estilo assíncrono específico (e o estilo síncrono). Estabeleceu-se aqui os modos completamente irrestrito e os modos restritos, que respeitam a condição de finitude. Dentre os modos restritos, duas subclassificações despontaram, os modos da relação entrada-saída (compreendendo o modo fundamental e o modo entrada-saída) e os modos de simultaneidade de entradas (com dois membros, mudança única e mudança múltipla;
- modelagem de fenômenos temporais - subdividida em duas classes, os modelos de atraso para componentes e os modelos de atrasos para circuitos;
- problemas temporais - aqui definiram-se corridas e transitórios.

Dado o resultado do estudo de modelos e problemas para estilos assíncronos, atacou-se classificações de sistemas assíncronos, revisando propostas da literatura e propondo um conjunto de critérios de classificação e exercitando estes critérios para construir classificações parciais.

Dada a importância de transitórios (hazards) dedicou-se então uma Seção a estudar sua caracterização e propostas de taxonomia para transitórios.

Finalmente, conclui-se com a apresentação sucinta de um estilo de projeto assíncrono moderno, os micropipelines.