Capítulo 7

Básculos e *Flip-Flops*

We will perform in measure, time and place.
William Shakespeare, Macbeth, V,6.

Até aqui vimos circuitos que são ditos *combinacionais* porque os valores nos sinais em suas saídas resultam de combinações dos valores nos sinais de suas entradas. Para um conjunto de entradas $\mathcal{E}$, as saídas do circuito $\mathcal{S}$ serão *sempre* $\mathcal{S} = \mathcal{F}(\mathcal{E})$, com valores determinados pelas funções lógicas $\mathcal{F}$ que definem o comportamento de cada membro de $\mathcal{S}$. Circuitos combinacionais somente implementam funções simples, tais como tabelas verdade.

Na Seção 4.3.4, e com mais detalhes na Seção 5.6.2, vimos um circuito simples que tem a capacidade de memorizar um bit e portanto pode permanecer no estado 1 ou no estado 0. Circuitos com memória são ditos *sequenciais* porque o comportamento das saídas depende tanto dos valores nas entradas quanto da *sequência* em que os valores foram armazenados na memória. Circuitos sequenciais são um tanto mais úteis porque estes nos permitem implementar algoritmos cujos comportamentos dependem das sequências de valores apresentados as suas entradas.

Neste capítulo[1] retornamos ao tema *memória*, estudando dois circuitos com memória, os básculos e os *flip-flop*s. Fazendo uso de circuitos combinacionais e de circuitos de memória podemos implementar desde máquinas simples como sinaleiros de trânsito até o processador em que o autor digita este texto.

Iniciamos, na Seção 7.2, com circuitos que, independentemente do que ocorre em suas entradas, mantêm o valor de um bit enquanto a fonte de alimentação fornecer energia. A Seção 7.3 introduz os *básculos*[2], que são circuitos que podem capturar um bit em sua entrada e memorizá-lo, define dois novos parâmetros para o comportamento temporal de circuitos com memória, e com base neles, define uma abstração para *tempo discreto*. A Seção 7.4 apresenta um circuito composto de dois básculos, chamado de *flip-flop*, cuja temporização é mais simples do que a de básculos. A Seção 7.5 define circuitos sequenciais síncronos (CSS) e o que se entende por "comportamento apropriado".

[1] © Roberto André Hexsel, 2012-2021. Versão de 1 de fevereiro de 2021.

[2] O termo em Inglês, *latch*, denota uma trava de porta que pode, ou estar aberta, ou estar fechada. Segundo Houaiss, a tradução correta é 'basculador', um cacófato assaz deselegante. O autor é um admirador do *über*-dicionarista mas prefere empregar, desavergonhadamente, a forma incorreta 'básculo'. Atravessemos o Rubicão sem mais delongas.

7.1 Estado e Tempo Discretizado

Neste texto empregamos uma noção estreita e estrita de *estado*, que é distinta do conceito empregado em nas Ciências Humanas. Vejamos a noção de estado num jogo de xadrez. Num dado instante, o *estado* de uma partida é a configuração das peças no tabuleiro e qual o jogador com direito a mover uma peça. A partida evolui – troca de estado – de acordo com as regras de movimentação para cada peça – um cavalo se move "em L", um bispo na diagonal, *et cetera* – e da alternância entre os jogadores. O *estado atual* do jogo é completamente definido pelo par tabuleiro–jogador e as *mudanças de estado* são rigidamente definidas pelas regras de movimentação das peças. Uma partida evolui ao longo de uma sequência de estados, que se inicia com 16 pedras brancas e 16 pedras pretas no tabuleiro, e que termina com um único rei "de pé". Cada um dos estados da sequência é atingido após a movimentação de uma única peça. Esta é uma ideia central para o que se segue: *os estados são discretos e representados univocamente por uma configuração de peças no tabuleiro.*

Encontramos *estados discretos* em jogos de baralho, na maioria dos jogos de tabuleiro e nas variáveis de um programa. Em todos estes exemplos, existe uma maneira de descrever a configuração atual do jogo ou programa, e é possível apontar qual a regra que deve ser aplicada para atingir uma nova configuração modificando-se a atual.

Recordemos a abstração de voltagem discretizada:

1. os níveis de tensão nos circuitos são separados em três faixas, uma chamada de 0 (zero), outra chamada de 1 (um), e valores na terceira faixa são ditos indefinidos. Os valores válidos definem o conjunto $\mathbb{B} = \{0, 1\}$;
2. emprega-se um conjunto fechado de operadores $\mathbb{B}^n \mapsto \mathbb{B}$;
3. um *dispositivo combinacional* é definido por uma função $\mathbb{B}^n \mapsto \mathbb{B}$ e pelo seu tempo de propagação;
4. um *circuito combinacional* é uma composição de dispositivos combinacionais, com entradas e saídas de tipo $\mathbb{B}^m$, não há curto-circuitos, entradas abertas e nem ciclos.

Essa abstração simplifica enormemente a especificação, projeto, implementação e verificação de circuitos *sem* estado.

Neste capítulo são definidos os *circuitos com estado* e uma abstração que *discretiza o tempo* de tal forma que as mudanças de estado ocorram somente em instantes determinados. Estes instantes são definidos por um *sinal de relógio* que sincroniza as mudanças de estado em todos os componentes *com* estado. O comportamento dos circuitos no intervalo entre os instantes de sincronização é indefinido. A Figura 7.1 identifica as duas abstrações.

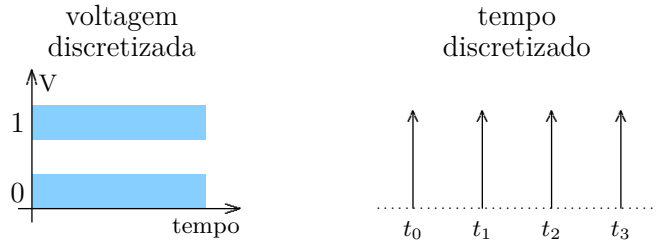


Figura 7.1: Abstrações: voltagem e tempo discretizados.

7.2 Circuitos com Memória

O tempo de contaminação das portas lógicas é, em essência, uma forma de memória: enquanto as alterações provocadas pelos novos valores nas entradas de um circuito não se propagarem através de todos os nós internos, sua saída 'memoriza' o valor anterior.

O circuito da Figura 7.2 mantém um valor estável e definido no sinal a' enquanto a fonte de alimentação estiver ligada. Este comportamento decorre do tempo de contaminação dos inversores e da realimentação da saída de um inversor para a entrada do outro. O tempo de contaminação finito dos inversores possibilita que os níveis lógicos nos dois nós do laço, a e b, estabilizem, e que um nível lógico constante se mantenha nas saídas dos inversores.

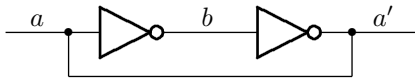


Figura 7.2: Circuito com memória.

Este circuito tem três modos de operação. No primeiro, quando a fica em 0 durante um intervalo maior que $2 \times T_I$, para que os sinais se propaguem através dos dois inversores, b fica em 1, e a saída do segundo inversor a' fica em 0, que por sua vez realimenta o sinal a com 0, 'capturando' o bit em 0 no circuito. No segundo modo de operação, quando a fica em 1 por um intervalo maior do que $2 \times T_I$, a' manterá o valor 1.

Estes dois modos de operação são *estáveis* porque o laço de realimentação no caminho a-b-a' mantém os valores nas entradas dos inversores estáveis, se a entrada se mantiver estável por $2 \times T_I$.

O terceiro modo de operação é dito *metaestável*, e a metaestabilidade pode ocorrer se o nível em a ficar indeterminado por, ao menos T_I. Neste caso, o sinal em b também fica indeterminado e o valor armazenado no laço pode não ser nem 0 e nem 1, ou pode oscilar entre 0 e 1. Em qualquer destes casos, a abstração de sinais como bits é violada.

Não há como determinar a duração do intervalo metaestável: (i) os inversores podem regenerar os sinais imediatamente, ou após o decurso de um intervalo arbitrariamente longo; (ii) ruído elétrico na vizinhança pode interferir positiva ou negativamente para a regeneração do sinal; e (iii) as tensões de saída dos inversores podem oscilar durante um intervalo arbitrariamente longo em níveis abaixo dos limiares para 0 ou para 1.

Não existe um parâmetro similar a T_I para definir a duração da metaestabilidade; o que se pode fazer é tentar evitá-la, ao impedir que a condição se estabeleça. Veremos na Seção 7.4 como evitar que transições nos sinais de controle do circuito de memória o levem à metaestabilidade.

Da forma como está desenhado na Figura 7.2, pode não ser fácil alterar o valor memorizado. A Figura 7.3 mostra o circuito de memória com duas chaves que permitem alterar o valor armazenado. Na esquerda, o laço de realimentação está fechado e assim mantém o valor do bit; no lado direito, o laço está aberto e um novo valor pode ser propagado de a para a' através dos dois inversores. A posição das duas chaves deve ser trocada simultaneamente para que o circuito se comporte como o desejado.

Com circuitos CMOS, as chaves são implementadas com portas de transmissão, ou transistores de passagem, como vimos na Seção 5.6. Circuitos com memória podem ser implementados com portas lógicas ao invés de inversores e portas de transmissão. Estes circuitos são chamados de *básculos* porque sua saída normalmente está em um de dois estados possíveis. A Figura 7.4

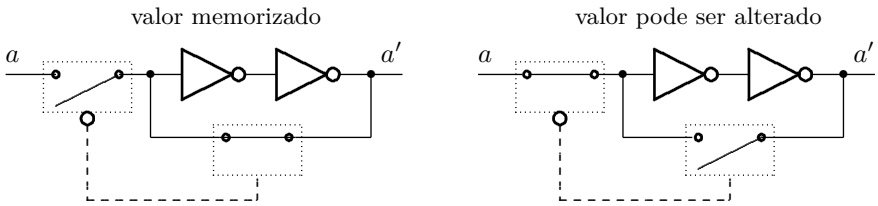


Figura 7.3: Circuito com memória programável.

mostra duas formas de desenhar o circuito de um básculo que é implementada com portas *nor*, e chamado de *básculo SR*. Normalmente as entradas r e s estão em 0. Se a entrada s (*set*) mudar para 1, a saída q muda para 1. Se a entrada r (*reset*) mudar para 1, a saída q muda para 0. Se as duas entradas mudarem de 1 para 0 simultaneamente, o básculo pode entrar em metaestabilidade. Básculos podem ser implementados com um par de portas *nand*, e neste caso o nível ativo das entradas *set* e *reset* é 0.

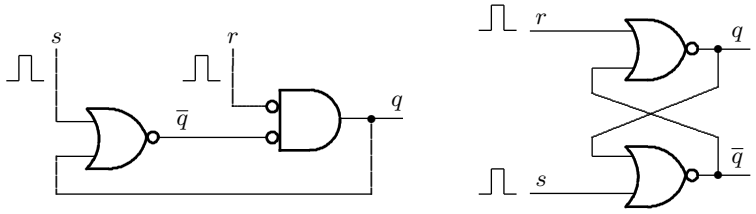


Figura 7.4: Básculo SR com portas *nor*.

O diagrama no lado esquerdo da Figura 7.4 mostra o básculo de forma similar ao laço com dois inversores, e o do lado direito mostra a forma convencional de representação. Os pulsos (0-1-0) nas entradas s e r indicam os níveis lógicos necessários para mudar o estado do básculo: um pulso em s coloca a saída q em 1 (*set*), e um pulso em r coloca a saída q em 0 (*reset*).

No que se segue, empregaremos básculos ligeiramente mais complexos do que aqueles com um par de portas lógicas, embora sejam funcionalmente equivalentes. Tal escolha favorece o aprendizado sobre o realismo na implementação dos circuitos[3].

7.3 Básculos

Como vimos na Seção 5.4, o comportamento dinâmico de um circuito combinacional pode ser distinto daquele determinado por sua especificação lógica, por causa da contaminação dos sinais em sua saída. Para minimizar as perturbações causadas por pulsos espúrios, usaremos um multiplexador bem comportado para implementar circuitos que podem armazenar um bit, tal como aquele o mostrado na Figura 7.5.

Um *básculo* (*latch*) é um circuito que armazena um bit e o valor do bit pode ser facilmente alterado ativando-se um sinal de controle, que permite carregar o básculo com um novo valor, ou manter o valor carregado previamente.

[3]Grande parte do material deste capítulo é baseado em conteúdos da disciplina 6.004 – *Computation Structures* do MIT, versão de 2013, que é, com uma larga vantagem, a melhor explanação sobre básculos de que o autor tem conhecimento. Material acessado em setembro de 2015.

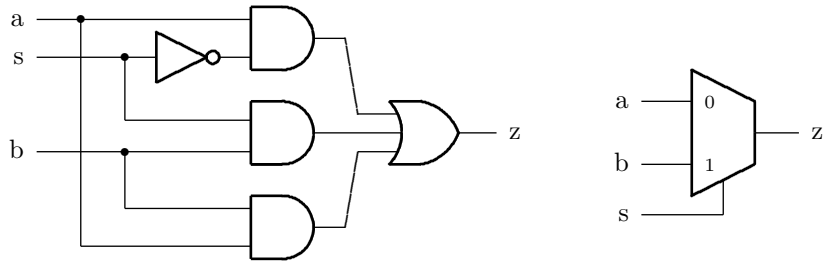
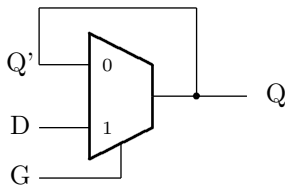


Figura 7.5: Multiplexador com saída bem comportada.

O básculo mostrado na Figura 7.6 tem três terminais: uma entrada D (*datum*), uma entrada Q' que é também a saída Q (*quiescent*), e uma entrada de 'gatilho' G. A entrada D determina o valor por armazenar quando $G = 1$. Quando $G = 0$, o básculo mantém o valor armazenado no terminal Q. O terminal de controle é chamado de 'gatilho' pois a transição de 1 para 0 em G é o que permite capturar um novo valor para armazená-lo. Note que o caminho com realimentação Q-Q' é similar àquele do circuito com os dois inversores. Observe na Figura 7.5 que a saída do *or* mantém o sinal na entrada a do *and*, cuja saída é uma das entradas do *or*.



G	D	Q'	Q	
0	x	0	0	Q mantém valor
0	x	1	1	
1	0	x	0	Q transparente
1	1	x	1	

Figura 7.6: Básculo implementado com um multiplexador.

O multiplexador implementado com dois implicativos primários – sem o implicativo $a \wedge b$ – produz um pulso espúrio em sua saída quando o sinal de controle passa de 1 para 0. Este pulso contaminaria o caminho de realimentação, e com alta probabilidade alteraria o bit armazenado em Q-Q'. Usamos o multiplexador bem comportado para evitar a contaminação da saída.

Como no caso dos dois inversores, no básculo é necessário usar um circuito que viola a restrição aos ciclos em circuitos combinacionais. A violação é necessária para obtermos a função de memória. Como no caso da abstração que faz uso de elementos de $\mathbb{B}$ para representar sinais elétricos contínuos, aqui usaremos uma nova abstração para discretizar o tempo contínuo.

Nossa abstração para *tempo discreto* é uma sequência discreta de instantes, determinada por um sinal periódico que alterna entre 1 e 0. Aqui é necessária alguma paciência porque convém estudarmos o comportamento dinâmico do básculo com mais atenção antes que possamos definir "tempo discreto".

A especificação temporal para o básculo é determinada pelo tempo de propagação T_M do multiplexador. O diagrama de tempos na Figura 7.7 mostra o comportamento dinâmico do básculo. Quando a entrada $G = 1$, o básculo está em modo transparente e a saída é uma cópia da entrada, a menos do atraso causado pelo tempo de propagação. Quando $G = 0$, a saída do básculo mantém seu valor enquanto G for 0.

Espaço em branco proposital.

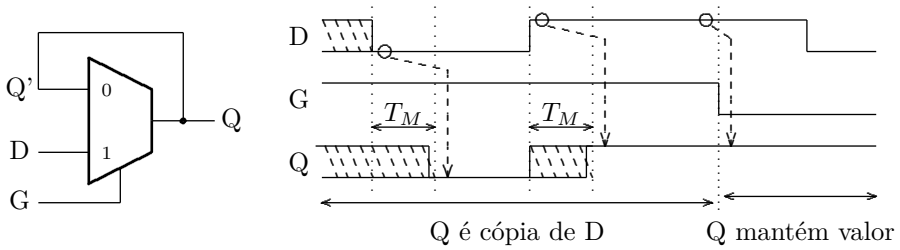


Figura 7.7: Comportamento dinâmico do básculo.

Considere as três combinações de entradas que definem o comportamento dinâmico do básculo, mostradas na Figura 7.8. Aqui, t é um intervalo arbitrário, e $\beta \in \mathbb{B}$.

(1) *Cópia de um novo valor.* Se $G = 1$, então Q é uma cópia de D. A entrada D é selecionada e se o gatilho G e a entrada $D = \beta$ permanecem estáveis por $t \geq T_M$ então Q assume o valor β independentemente do que ocorra com Q'.

(2) *Captura.* Alteração em G sem contaminar a saída Q. Se as entradas $D = Q' = \beta$ durante $t \geq T_M$, então $Q = \beta$ independentemente da alteração em G porque a escolha entre dois valores iguais nas entradas não altera a saída.

(3) *Q mantém seu valor enquanto $G = 0$.* Se $G = 0$ e $Q' = \beta$ por $t \geq T_M$, então $Q = \beta$ independentemente do valor em D porque Q' foi selecionado como a entrada do multiplexador por $G = 0$.

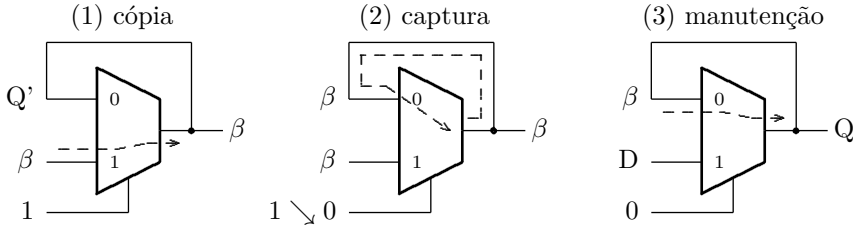


Figura 7.8: Combinações de entradas do básculo.

Antes da mudança de 1 para 0 em G, a entrada D deve ficar estável por $2 \times T_M$ para garantir que β seja capturado de forma confiável. O primeiro T_M garante a estabilização dos nós no caminho $D \rightsquigarrow Q$, enquanto que o segundo T_M garante a estabilização no caminho $Q' \rightsquigarrow Q$. Estes caminhos estão mostrados como linhas tracejadas na Figura 7.9.

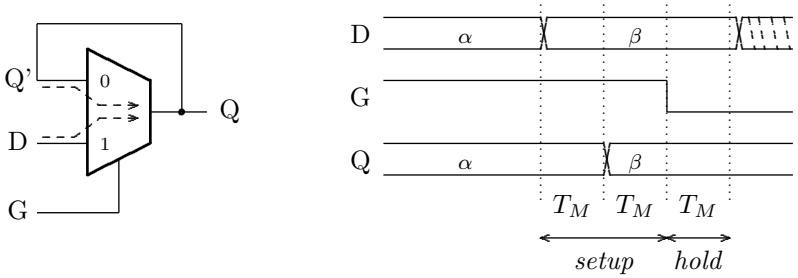


Figura 7.9: Comportamento dinâmico do básculo – transições em G.

A entrada D deve permanecer estável por $2 \times T_M$ *antes* de uma alteração em G de 1 para 0. Este intervalo é necessário para garantir a captura de um novo valor antes da borda descendente

no gatilho. O intervalo de $2 \times T_M$ antes da borda em G é chamado de *tempo de estabilização das entradas* (*setup time*), e é denotado por T_{su}.

A entrada D deve permanecer estável por T_M *após* a borda de descida em G para garantir que o valor capturado no laço de realimentação não seja contaminado por uma mudança em D. O intervalo T_M após a borda no gatilho é o *tempo de manutenção das entradas* (*hold time*), e é denotado por T_h.

Se os tempos de estabilização, ou de manutenção, do básculo são violados, seu comportamento pode tornar-se metaestável e a saída do básculo pode manter um valor indefinido por um tempo arbitrariamente longo.

7.3.1 Abstração do Tempo Discretizado

Para que possamos discretizar o tempo, e assim abstrair a complexidade do comportamento temporal dos básculos, é necessário definirmos limites para os intervalos de tempo de estabilidade dos sinais imediatamente antes e imediatamente depois das bordas descendentes no sinal de gatilho. Estes limites são usados como a especificação temporal do comportamento de básculos, em adição aos tempos de propagação e de contaminação:

> *o tempo de estabilização das entradas (setup time) é o intervalo em que as entradas estão válidas e estáveis* antes *da transição no sinal de gatilho; e*
>
> *o tempo de manutenção das entradas (hold time) é o intervalo em que as entradas estão válidas e estáveis* após *a transição no sinal de gatilho.*

Nos resta um pequeno porém incômodo problema: nossa definição para circuitos combinacionais proíbe ciclos, tais como os caminhos de realimentação nos básculos. O problema decorre de que, em circuitos com ciclos, os sinais percorrem o circuito combinacional e computam um novo valor, e por causa do ciclo, o novo valor percorre o circuito e computa mais um novo valor, *ad aeternum*. Dependendo da complexidade da parte combinacional do circuito, os sinais se propagam a taxas incontroláveis, e não há como prever se e quando estabilizarão. A parte 'pequena' do problema é que a sua solução é relativamente simples: empregam-se dois básculos para quebrar os ciclos.

7.4 *Flip-Flops*

Básculos possuem características temporais que tornam os projetos complexos porque sua temporização tem uma certa tendência à metaestabilidade. A combinação de dois básculos elimina o problema causado pelos ciclos combinacionais e ainda permite armazenar bits com uma especificação temporal que é relativamente simples de satisfazer e implementar. O circuito com dois básculos é chamado de *flip-flop*.

Os instantes em que os dados são capturados pelos *flip-flops* são determinados pelas bordas ascendentes do *sinal de relógio*, que é um sinal periódico que alterna entre 0 e 1. Este sinal é, tipicamente, uma *onda quadrada*, na qual os intervalos em 0 têm a mesma duração que os intervalos em 1. No que se segue, o sinal de relógio é chamado de *clock* e frequentemente abreviado como *clk*.

Se usarmos dois básculos ligados em série, com seus gatilhos em níveis distintos do sinal de relógio, temos o que se chama de *flip-flop mestre-escravo*. A Figura 7.10 mostra o *flip-flop* e um diagrama de tempo que relaciona entradas com a saída Q. Note que as entradas de gatilho tem efeitos complementares no mestre e no escravo: quando $clock = 0$, o mestre fica transparente e o escravo mantém seu valor.

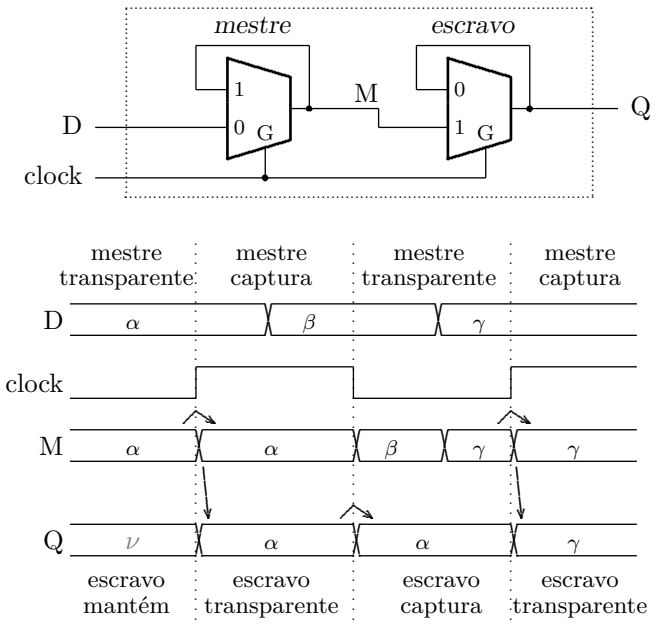


Figura 7.10: *Flip-flop* mestre-escravo.

No *borda ascendente*, na qual *clock* muda de 0 para 1, o mestre captura a entrada $D = \alpha$, enquanto que o escravo fica 'transparente' porque sua saída segue $M = \alpha$ e portanto $Q = \alpha$.

No *borda descendente*, na qual *clock* muda de 1 para 0, o escravo captura a entrada $M = \alpha$, e assim a saída se mantém inalterada, em $Q = \alpha$. Neste semiciclo o mestre está transparente e sua saída segue $D = \beta$ e $D = \gamma$.

No primeiro semiciclo, o mestre captura a entrada D e o escravo, que está transparente, transfere sua entrada para a saída Q. No segundo semiciclo, o escravo captura a saída do mestre e a transfere para sua saída Q, que permanece inalterada. Neste semiciclo o mestre está transparente e transfere a entrada que será capturada pelo escravo na próxima borda ascendente. O comportamento do *flip-flop* mestre-escravo é o desejado: a cada borda ascendente do relógio, a entrada é capturada e permanece estável até a próxima borda ascendente.

A Figura 7.11 mostra os símbolos que representam um básculo e um *flip-flop*. A diferença entre os dois símbolos é o triângulo no *flip-flop*, que representa a entrada de relógio que é sensível à borda ascendente, enquanto que a entrada *gate* do básculo é sensível ao nível do sinal de sincronismo.

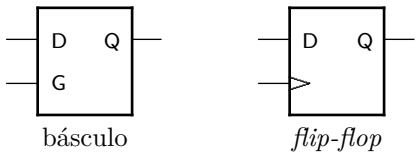


Figura 7.11: Símbolos para básculos e *flip-flops*.

A diferença de comportamento entre básculos e *flip-flops* é explicitada no diagrama de tempo da Figura 7.12. No semiciclo em que o sinal de sincronismo é 1, o básculo fica transparente e a saída Q segue a entrada D, e isso é indicado pela faixa hachurada no sinal de entrada. Antes do início da observação dos sinais, no ciclo 1, os valores em B e em FF são desconhecidos. No ciclo 1, básculo e *flip-flop* têm o mesmo comportamento porque a entrada permanece estável ao longo do primeiro semiciclo.

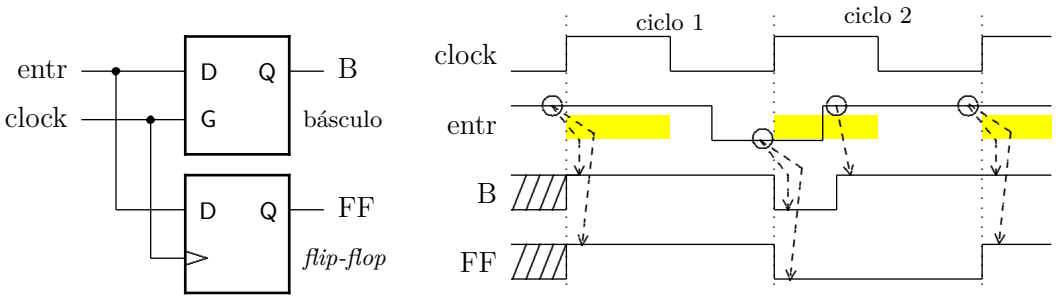


Figura 7.12: Comportamento de básculos e *flip-flops*.

No ciclo 2, a entrada muda no primeiro semiciclo e a saída do básculo segue a entrada. O valor na entrada do *flip-flop* é capturado na borda do sinal de relógio e este valor não se altera até a próxima borda.

Exemplo 7.1 Vejamos uma aplicação simples porém importante de *flip-flops*. Suponha que a saída de um circuito combinacional seja 'ruidosa', porque contenha muitos pulsos espúrios, o que pode prejudicar a operação do(s) circuito(s) que fazem uso deste sinal. O circuito da Figura 7.13 mostra uma forma de filtrar os pulsos indesejados: o sinal ruidoso R é ligado à entrada de um *flip-flop* (Q1), e a saída deste é ligada à entrada de um segundo *flip-flop* (Q2).

Antes do ciclo 1, o estado de Q1 (sinal X) é indeterminado, e antes do ciclo 2 o estado de Q2 (Y) ainda é indeterminado. No ciclo 1 o valor de R é capturado por Q1 e este valor é copiado para Q2 no ciclo 2. Os pulsos em R não se refletem na saída de Q1, e a entrada de Q2 recebe uma versão 'limpa', porém atrasada, do sinal R.

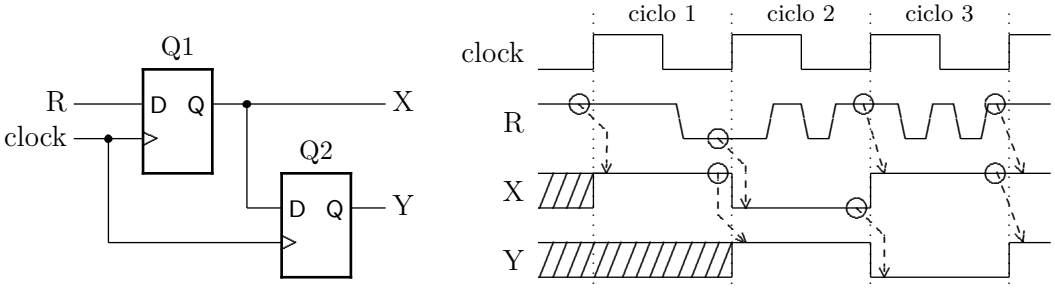


Figura 7.13: Filtro de ruído com dois *flip-flops*.

O valor de Y está dois períodos do relógio atrasado em relação a R, e este é um ponto importante: ao final do ciclo 1, o nível 0 em R é capturado por Q1, e X=0 ao longo do ciclo 2. Somente *após* a borda do ciclo 3 é que Q2 passa ao nível 0 que R exibira no ciclo 1. ◁

Exemplo 7.2 O circuito do Ex. 7.1 tem mais uma utilidade, que é a de eliminar o comportamento metaestável que pode ocorrer em *flip-flops*. O *flip-flop* Q1 é suficiente para filtrar o ruído no sinal R, embora Q1 possa incorrer em metaestabilidade caso uma transição em R ocorra suficientemente perto da borda do relógio. Isso é mostrado no diagrama de tempos da Figura 7.14, com a transição no sinal R junto à borda do ciclo 2, o que faz com que a saída de Q1 fique metaestável durante quase todo aquele ciclo.

A probabilidade de que a saída de Q1 ainda esteja metaestável na próxima borda é pequena. O *flip-flop* Q2 reduz significativamente a probabilidade de que um sinal metaestável se propague a partir do sinal Y. Neste caso, ambos Q1 e Q2 teriam que permanecer metaestáveis durante dois ciclos completos, o que é possível mas altamente improvável. Comportamento metaestável pode ser eliminado se nossos projetos aderem rigidamente à abstração de tempo discretizado. ◁

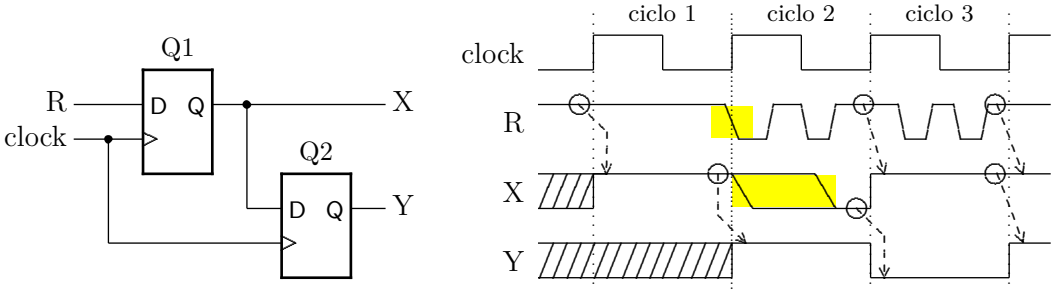


Figura 7.14: Defesa contra metaestabilidade com dois *flip-flops*.

Exemplo 7.3 O circuito da Figura 7.15 mostra um *flip-flop* cuja saída se inverte ($1 \to 0$ ou $0 \to 1$) sempre que sua entrada for 1. Pode não ser óbvio, mas este circuito conta, em módulo-2, o número de vezes em que o sinal de entrada trocou de nível. Voltamos a este assunto na Seção 8.2.1.

Como mostra a tabela verdade do *xor*, se $T = 0$ então $R = S$, e mais ainda, se $T = 1$ então $R = \overline{S}$. A porta *xor* se comporta como um inversor controlado pelo sinal T. Suponha que, inicialmente, $S = 1$. Com $T = 0$ a saída do *flip-flop* é uma cópia da sua entrada e assim, no ciclo 2 temos $S = 1$. Quando $T = 1$, R se inverte e no ciclo 3 S muda para 0. Enquanto $T = 1$, na próxima borda do relógio, o valor de S muda. Este circuito é chamado de *flip-flop toggle* porque sua saída se inverte sempre que a entrada de controle for 1. ◁

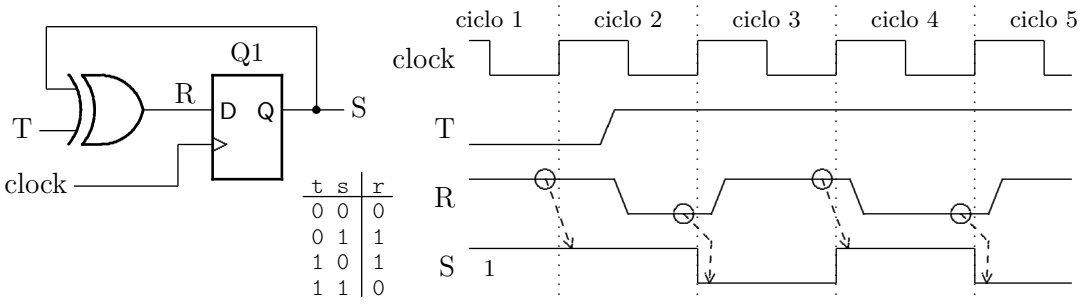


Figura 7.15: *Flip-flop* que troca de estado, ao invés de mantê-lo.

7.4.1 *Flip-flops* e Tempo Discretizado

Para garantir que a abstração de tempo discretizado seja válida, e que o comportamento dinâmico do *flip-flop* é o desejado, sua especificação temporal deve ser respeitada. O diagrama de tempo com as restrições na temporização da entrada de dados, e a especificação do comportamento da saída é mostrada na Figura 7.16.

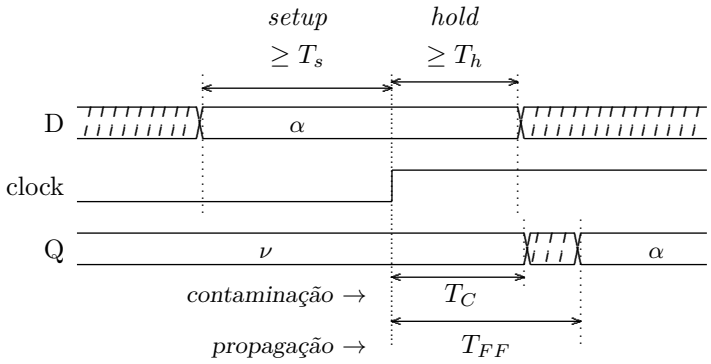


Figura 7.16: Temporização do *flip-flop*.

Os tempos de estabilização, de manutenção, de propagação e de contaminação, de *flip-flops* são *sempre* definidos com relação à borda de subida do relógio. Frequentemente são empregadas as expressões *clock-to-Q* ou *clock-to-output* para explicitar que o tempo de propagação do *flip-flop* é contado a partir da borda do sinal de relógio.

A entrada D deve permanecer estável *antes* da borda do relógio, por um intervalo maior do que o tempo de estabilização T_{su}. A entrada deve permanecer estável, *após* a borda do relógio, por um intervalo maior do que o tempo de manutenção T_h. Se as restrições quanto à entrada forem respeitadas, então a saída Q do *flip-flop* mantém seu valor durante o tempo de contaminação $T_{C,F}$ após a borda do relógio, e a saída se torna definida e válida após o tempo de propagação T_{FF}.

O tempo de propagação T_{FF} é a soma do tempo de propagação dos dois básculos – após a borda do relógio, um novo valor é capturado pelo mestre e se propaga através do escravo até a saída $T_{FF} = 2 \times T_B$. O tempo de contaminação T_C do *flip-flop* é determinado pelo básculo escravo – ao ficar transparente, o escravo mantém a saída do *flip-flop* como estava durante $T_{C,E}$. Os tempos de estabilização T_{su} e de manutenção T_h são determinados pelo básculo mestre – que mantém a entrada do *flip-flop* após a borda de subida do relógio.

O leitor atento deve ter percebido um possível problema com o tempo de manutenção do escravo. Se o tempo de contaminação T_C, M do mestre for menor do que o tempo de manutenção T_h do escravo, então o escravo pode capturar um valor incorreto na borda de descida do relógio. Isso pode ser contornado pelo acréscimo de um *buffer* entre a saída do mestre e a entrada do escravo, para elongar o $T_{C,M}$ do mestre.

7.4.2 Tipos de *Flip-Flops*

Além do *flip-flop* tipo D (D de 'direto' ou '*data*'), outros *flip-flop*s são úteis. A Figura 7.17 mostra os símbolos para os *flip-flop*s tipo D, tipo T (de '*toggle*') e JK. Estes *flip-flop*s podem ser implementados com um par de saídas complementares, o que é indicado pela negação na

segunda saída, que é chamada de $\overline{Q}$.

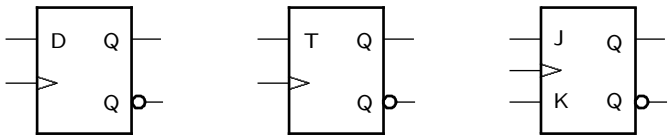


Figura 7.17: *Flip-flops* tipo D, T e JK.

O comportamento destes *flip-flops* (FFs) é definido pelas suas *equações características* mostradas na Equação 7.1, ou alternativamente, pelas suas tabelas de excitação, na Tabela 7.1. Nesta tabela, Q^+ é o estado do *flip-flop* após a borda do relógio. Na Equação 7.1 o símbolo $\leftarrow$ denota a atribuição do novo estado ao FF. A equação característica do FF T pode ser reescrita como $Q \leftarrow T \oplus Q$.

$$\begin{array}{l} D, T, J, K, Q : \mathbb{B} \\ \text{FF D:} \quad Q \leftarrow D \\ \text{FF T:} \quad Q \leftarrow T\overline{Q} \vee \overline{T}Q \\ \text{FF JK:} \quad Q \leftarrow J\overline{Q} \vee \overline{K}Q \end{array} \tag{7.1}$$

Tabela 7.1: Tabela de excitação dos *flip-flops* JK, T e D.

Q	Q^+	JK	T	D
0	0	$0X$	0	0
0	1	$1X$	1	1
1	0	$X1$	1	0
1	1	$X0$	0	1

Existem situações nas quais é necessário inicializar os *flip-flops* com um estado definido. Por exemplo, frequentemente se deseja inicializar uma contagem em zero, e para isso todos os *flip-flops* do circuito devem iniciar no estado $Q = 0$. O sinal de inicialização é tipicamente chamado de *reset*, e quando este sinal está ativo, é atribuído um estado definido aos *flip-flops*.

A equação característica do FF D com sinais de inicialização *set* ($Q \leftarrow 1$) e *reset* ($Q \leftarrow 0$) é mostrada na Equação 7.2. O estado do *flip-flop* é indeterminado quando ambos *set* e *reset* são ativados ao mesmo tempo.

$$\text{FF D:} \quad \begin{cases} set \wedge reset & \Rightarrow & Q \leftarrow D \\ \overline{set} & \Rightarrow & Q \leftarrow 1 \\ \overline{reset} & \Rightarrow & Q \leftarrow 0 \end{cases} \tag{7.2}$$

O símbolo para o *flip-flop* D 'completo' é mostrado na Figura 7.18. Quando ativo, o sinal *set* coloca imediatamente o FF em 1, e o sinal *reset* força o estado em 0. Note que os sinais *set* e *reset* são, geralmente, ativos em 0, o que é indicado pelo círculo de negação junto à entrada. Estes dois sinais são assíncronos e atuam independentemente do sinal de relógio.

Os *flip-flops* tipo T e JK também podem ser implementados com as entradas assíncronas *set* e *reset*, que têm os mesmos efeitos que no FF D. Quando estas entradas são desnecessárias, elas são omitidas dos diagramas, embora não devam ficar desligadas em nenhuma implementação.

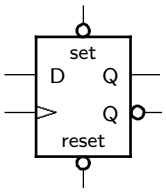


Figura 7.18: *Flip-flop* tipo D com entradas *set* e *reset*.

Exemplo 7.4 Quando se emprega dispositivos com lógica programável (*Field Programmable Gate Arrays*, ou FPGA), o compilador de VHDL impõe uma série de restrições aos tipos de construções que podem ser empregadas nos modelos dos circuitos. Por exemplo, o compilador de um dos grandes fabricantes de FPGAs proíbe a utilização de básculos porque estes tornam a temporização do circuito imprevisível, por causa dos ciclos.

Contudo, básculos são úteis e necessários em muitas aplicações. O circuito da Figura 7.19 contém um *flip-flop* e seu comportamento imita aquele de um básculo SR. Um pulso em 1, na entrada *set* coloca a saída Q em 1. Um pulso em 0, na entrada *clr* (*clear*), coloca a saída em 0.

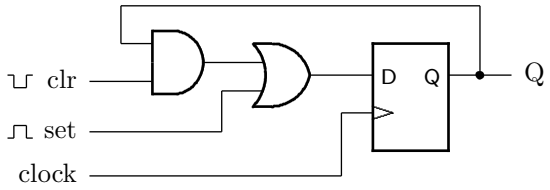


Figura 7.19: Circuito com *flip-flop* que se comporta como básculo SR.

O comportamento do circuito da Figura 7.19 é mostrado no diagrama de tempo da Figura 7.20, para as quatro combinações de *clr* e *set*. Existe alguma combinação das entradas que torna o circuito metaestável? Considere que todos os parâmetros de temporização sejam atendidos com folga.

Se as entradas forem *clr*=0 e *set*=1, Q ficará num "estado 'preferencial", quer dizer, uma das entradas tem mais prioridade que a outra. Altere o circuito para inverter a prioridade das entradas. O novo circuito é imune à metaestabilidade? ◁

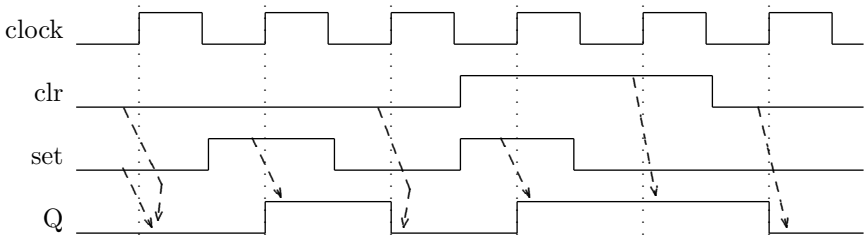


Figura 7.20: Temporização do circuito que se comporta como básculo SR.

7.4.3 Registradores

Um *registrador* é um circuito com um certo número de *flip-flops* que compartilham o sinal de relógio, como mostrado na Figura 7.21. Tipicamente todos os *flip-flops* são atualizados simultaneamente e o registrador armazena o equivalente a uma variável numa linguagem de

programação de alto nível. Apesar da sua simplicidade, este componente é usado extensivamente no projeto de sistemas digitais e em breve veremos vários exemplos de seu uso.

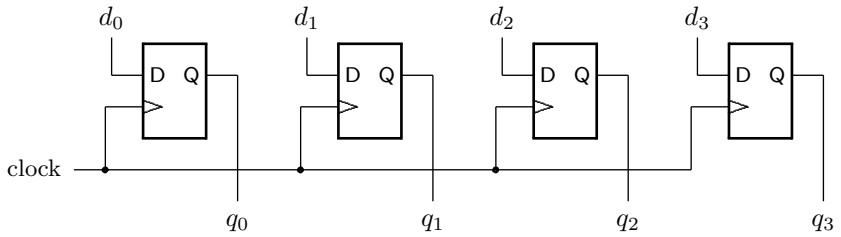


Figura 7.21: Registrador simples de quatro bits.

Um *registrador simples*, que é aquele da Figura 7.21, é atualizado a cada borda ascendente do relógio. Há aplicações nas quais deseja-se atualizar o conteúdo do registrador somente em determinados instantes, mantendo seu conteúdo inalterado no restante do tempo. O registrador da Figura 7.22 permite que um novo valor seja carregado, na borda do relógio em que o sinal *carga* esteja ativo (*carga=1*), enquanto que seu valor é preservado quando *carga* está inativo. Este registrador é chamado *registrador com carga paralela*.

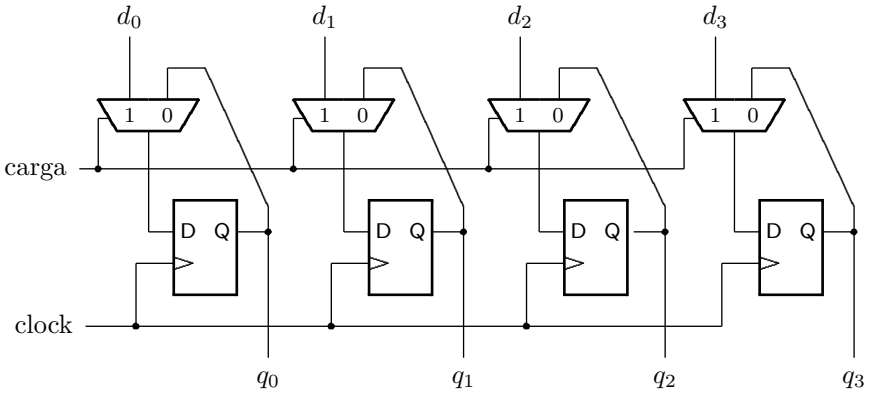


Figura 7.22: Registrador com carga paralela.

Os diagramas das Figuras 7.21 e 7.22 são demasiado detalhados para uso em desenhos/diagramas de circuitos complexos. As versões simplificadas dos símbolos para registradores com n bits de largura – n *flip-flops* – são mostradas na Figura 7.23.

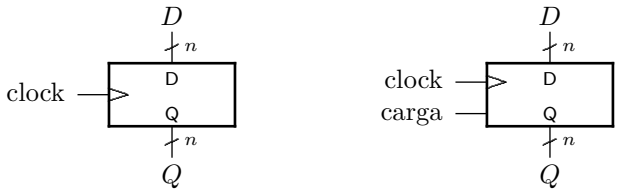


Figura 7.23: Símbolos para registradores de n bits.

Em geral, emprega-se um sinal de *reset* em circuitos complexos para inicializar todos os *flip-flops* e registradores para um estado conhecido. Um circuito dedicado ativa o *reset* por um intervalo suficiente para garantir que todos os registradores tenham sido inicializados. Depois disso, este sinal permanece inativo até que o circuito seja desligado ou reinicializado.

7.5 Circuitos Sequenciais Síncronos

Agora que fomos apresentados a todos os componentes interessantes, podemos definir o que se entende por *circuito sequencial síncrono*[4] (CSS). Estes são circuitos compostos por um componente com memória para armazenar seu *estado atual*, além de lógica combinacional que computa o *próximo estado* como uma função do estado atual e das entradas, bem como uma função para computar as *saídas* do circuito. O 'síncrono' indica que as mudanças de estado ocorrem sincronicamente ao sinal de relógio.

Um *circuito sequencial síncrono* é composto por:

1. um sinal de relógio que sincroniza a operação do CSS;
2. zero ou mais *entradas* digitais;
3. uma ou mais *saídas* digitais;
4. um *registrador de estado*;
5. uma *função de próximo estado*; e
6. uma *função de saída*.

A Figura 7.24 mostra um modelo para um circuito sequencial síncrono. O *registrador de estado* (RE) tem ϵ bits de largura e portanto o CSS pode ter até 2^ϵ estados. O CSS tem λ bits de entrada, e o circuito combinacional que implementa a *função de próximo estado* deve computar o próximo estado PE com a função f

$$PE = f(\text{estado atual}, \text{entradas}).$$

A saída S tem σ bits e é computada pela *função de saída* g

$$S = g(\text{estado atual}, \text{entradas}).$$

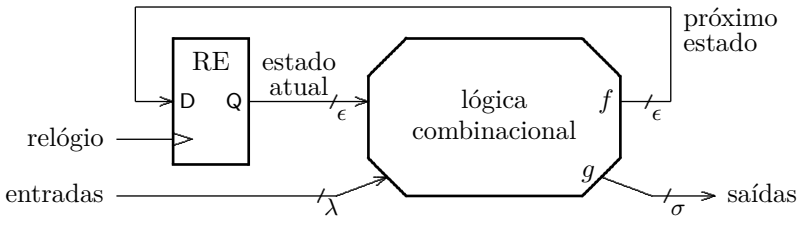


Figura 7.24: Circuito sequencial síncrono.

Se a máxima "tempo é dinheiro" pode ser considerada válida, então nos interessa computar a velocidade máxima de operação dos nossos CSSs. Esta é determinada pela frequência máxima do relógio na qual o circuito pode operar e produzir os resultados desejados.

7.5.1 Operação Apropriada – *Setup* e *Hold*

Neste contexto é desejável e conveniente introduzir o conceito de *operação apropriada*, em contraste ao conceito de *operação correta*. Um circuito opera corretamente se seu funcionamento

[4] A definição de *circuito sequencial síncrono* desta seção é baseada em conteúdos da disciplina 6.004 – *Computation Structures* do MIT, versão de 2013. Material acessado em setembro de 2015.

satisfaz à sua especificação: se uma sequência de valores aceitáveis é apresentada nas suas entradas então o circuito produz uma sequência de valores nas suas saídas que são exatamente aqueles valores determinados pela sua especificação.

Um circuito opera apropriadamente se nenhuma das restrições impostas pela tecnologia de implementação é violada. Por exemplo, a fonte de alimentação mantém um fluxo de corrente estável enquanto opera na sua tensão nominal; as restrições de temperatura e umidade são atendidas. O que nos interessa aqui é verificar se as restrições de *setup* e *hold* dos *flip-flops* são atendidas pelo projeto do CSS.

Se, e somente se, as restrições de *setup* e *hold* são atendidas, então podemos fazer uso da abstração de *tempo discretizado*.

A *operação apropriada* de um CSS é definida como:

> *para cada evento de sincronização, todos os flip-flops controlados pelo mesmo sinal de relógio examinam suas entradas e determinam seus próximos estados. Para tanto, duas condições devem ser satisfeitas:*
> *(1) as entradas ficam estáveis e determinadas* antes *do evento de sincronização;*
> *e*
> *(2) nenhum flip-flop pode mudar de estado mais do que uma vez em cada evento de sincronização.*

Setup ou Tempo de Estabilização

A primeira condição é satisfeita se o tempo de estabilização (*setup*) dos *flip-flops* é respeitado. O tempo de estabilização é uma especificação temporal definida em função de dois eventos consecutivas de sincronização, ou de duas bordas consecutivas do relógio.

O relacionamento entre as bordas do relógio e a especificação para *setup* é mostrada na Figura 7.25. Dois *flip-flops* são ditos *adjacentes* se estes compartilham o sinal de relógio e são interligados somente por lógica combinacional. A temporização é definida entre dois *flip-flops* adjacentes, sendo FF1 o *flip-flop* fonte do valor e FF2 o destino. No circuito da Figura 7.24, 'fonte' e 'destino' são a saída e a entrada do mesmo *flip-flop*.

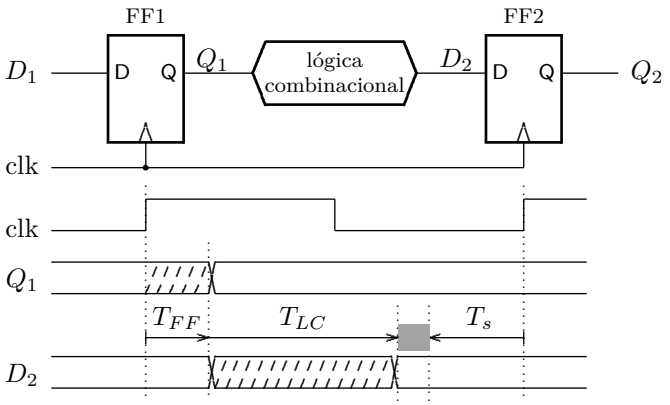


Figura 7.25: Operação apropriada: relação entre borda do relógio e *setup*.

A especificação de *setup* envolve duas bordas consecutivas do relógio: a soma dos tempos de propagação do *flip-flop* T_{FF} e do circuito combinacional T_{LC}, mais o *setup* (T_{su}), devem ser

menores do que o intervalo entre duas bordas, que é o período do relógio T_{clk}. A 'folga' para a estabilização na entrada do FF2 é indicada pela faixa cinzenta no diagrama.

Hold ou Tempo de Manutenção

A segunda condição é satisfeita se o tempo de manutenção (*hold*) dos *flip-flops* é respeitado. O tempo de manutenção é definido em função de um único evento de sincronização que é observado em dois locais diferentes: a saída de um *flip-flop* que produz o valor α e o mantém estável por T_C segundos, e a entrada do *flip-flop* adjacente que deve observar aquele mesmo valor α, estável durante T_h segundos.

A Figura 7.26. mostra o relacionamento entre a borda do relógio e a especificação para *hold* entre dois *flip-flops* adjacentes. A especificação de *hold* é satisfeita quando a soma dos tempos de contaminação do *flip-flop* $T_{C,F}$ e do circuito combinacional $T_{C,C}$ são maiores do que o tempo de manutenção T_h. A 'folga' para o *hold* é indicada pela faixa cinzenta no diagrama de tempos.

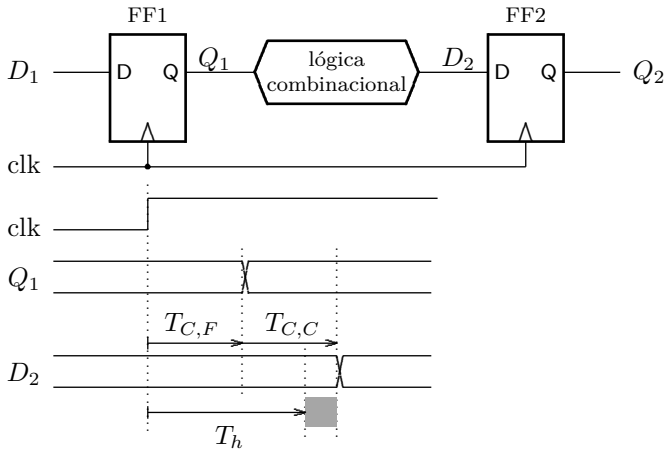


Figura 7.26: Operação apropriada: relação entre borda do relógio e *hold*.

Período Mínimo, *Setup* e *Hold*

Para que um CSS opere adequadamente, sua frequência máxima de operação é computada em função do período mínimo do relógio

$$T_{min} = 1/f_{\max} \tag{7.3}$$

que é determinado pelo tempo de propagação do *flip-flop* ou registrador (T_{FF}), pelo tempo de propagação do circuito combinacional (T_{LC}) e pelo tempo de estabilização do *flip-flop* (T_{su})

$$T_{min} \geq T_{FF} + T_{LC} + T_{su}\,. \tag{7.4}$$

O cálculo do período mínimo do relógio atende à primeira condição para a operação apropriada se o intervalo T_{su} (*setup*) dos *flip-flops* é respeitado.

O tempo de manutenção deve ser respeitado para que a segunda condição seja atendida. A soma do tempo de contaminação do *flip-flop* ou registrador ($T_{C,F}$) e do circuito combinacional

($T_{C,C}$) deve ser maior do que o tempo de manutenção (T_h)

$$T_{C,F} + T_{C,C} \geq T_h \,. \tag{7.5}$$

As entradas de um CSS devem satisfazer à especificação do tempo de estabilização dos *flip-flops*

$$T_{s,E} \geq T_{LC} + T_{su} \,. \tag{7.6}$$

Mudanças nas entradas devem se propagar através do circuito combinacional com alguma folga para respeitar o *setup* dos *flip-flops*.

As entradas de um CSS devem também satisfazer à especificação do tempo de manutenção dos *flip-flops*

$$T_{h,E} \geq T_h - T_{C,C} \,. \tag{7.7}$$

As entradas devem permanecer estáveis após a borda do relógio para que seja possível garantir que as consequências da mudança sejam capturadas pelo registrador de estado.

As Equações 7.4 a 7.7 definem o período mínimo do relógio e os intervalos de *setup* e *hold* como desigualdades. Os valores exatos, tais como

$$T_{min} = T_{FF} + T_{LC} + T_{su} \tag{7.8}$$

definem um período mínimo em condições ideais, que dificilmente são aquelas atingidas na fabricação em larga escala de circuitos integrados. Por isso, os engenheiros de fabricação projetam os circuitos de temporização com alguma folga, para acomodar as diferenças de construção de cada cópia do circuito.

Exemplo 7.5 Considere o circuito da Figura 7.26. A especificação temporal do registrador de estado é: $T_{FF} = 2\,\text{ns}$, $T_{C,F} = 0,5\,\text{ns}$, $T_{su} = 2\,\text{ns}$, $T_h = 1\,\text{ns}$. A especificação temporal do circuito combinacional é: $T_{LC} = 10\,\text{ns}$, $T_{C,C} = 2\,\text{ns}$.

Qual é o período mínimo do relógio?

$$T_{min} = T_{FF} + T_{LC} + T_{su} = 2 + 10 + 2 = 14\,\text{ns}.$$

Há folga no tempo de manutenção do registrador?

$$T_{C,F} + T_{C,C} = 0,5 + 2 = 2,5\,\text{ns} \geq 1\,\text{ns} = T_h.$$

A folga é de $1,5\,\text{ns}$. Qual deve ser o *setup* mínimo nas entradas?

$$T_{s,E} \geq T_{LC} + T_{su} = 10 + 2 = 12\,\text{ns}.$$

Qual deve ser o *hold* mínimo nas entradas?

$$T_{h,E} \geq T_h - T_{C,C} = 1 - 2 = -1\,\text{ns}.$$

Há uma folga de $1\,\text{ns}$ para acomodar a especificação de *hold* porque o tempo de contaminação do circuito combinacional $T_{C,C}$ é maior do que T_h. ◁

Exemplo 7.6 Considere o circuito da Figura 7.26. A especificação temporal do registrador de estado é: $T_{FF} = 3\,\text{ns}$, $T_{C,F} = 1\,\text{ns}$, $T_{su} = 2\,\text{ns}$, $T_h = 2\,\text{ns}$. A especificação temporal do circuito combinacional é: $T_{LC} = 5\,\text{ns}$.

Qual deve ser o tempo mínimo de contaminação do circuito combinacional?

$$T_{C,F} + T_{C,C} = 1\,\text{ns} + T_{C,C} \geq T_h = 2\,\text{ns};\ \text{portanto}\ T_{C,C} \geq 1\,\text{ns}.$$

Qual é o período mínimo do relógio?

$$T_{min} \geq T_{FF} + T_{LC} + T_{su} = 3 + 5 + 2 = 10\,\text{ns}.$$

Qual deve ser o *setup* mínimo nas entradas?

$$T_{s,E} \geq T_{LC} + T_{su} = 5 + 2 = 7\,\text{ns}.$$

Qual deve ser o *hold* mínimo nas entradas, considerando-se o $T_{C,C}$ computado?

$$T_{h,E} \geq T_h - T_{C,C} = 2 - 1 = 1\,\text{ns}.$$

As entradas devem permanecer definidas e estáveis 1 ns após a borda do relógio. ◁

Exemplo 7.7 Considere o circuito que efetua a operação ternária *multiply-add* (MADD), que multiplica dois valores e adiciona o produto a um terceiro valor. Esta operação é usada em vários algoritmos de processamento digital de sinais. O circuito da Figura 7.27 mostra uma possível implementação: o operando B é multiplicado por 2^D, e o produto é adicionado ao operando A. A multiplicação por 2^D é obtida com um deslocador exponencial como aquele descrito na Seção 6.4.1, e neste exemplo B pode ser multiplicado por 1, 2, 4, ou 8. O produto de um número de 4 bits (B) por um número de 3 bits ($\ll 3$) produz um número de 7 bits. Na entrada α do somador, A é estendido para um número de 7 bits pela concatenação de três zeros nas posições mais significativas. A soma de dois números de 7 bits é representada em 8 bits.

O circuito é segmentado em quatro estágios pelos três registradores Γ_1, Γ_2 e Γ_3. Há um circuito ligado ao registrador Γ_1 que alimenta o MADD com operandos, e um outro ligado a Γ_3, que consome os resultados R. A cada ciclo, uma nova tripla $\langle \text{A}, \text{B}, \text{D} \rangle$ é transferida para o segmento com o deslocador, entre Γ_1 e Γ_2, e no ciclo seguinte a dupla $\langle \text{A}, \text{C} \rangle$ é transferida para o segmento com o somador. No terceiro ciclo, o resultado R é apresentado nas saídas do registrador Γ_3. Um novo resultado R é produzido a cada ciclo, embora decorram dois ciclos desde a captura de uma nova tripla em Γ_1 até que o resultado para aqueles valores seja capturado em Γ_3.

A sequência de eventos descrita acima é mostrada no diagrama de tempos da Figura 7.28. As partes hachuradas nos sinais C e S correspondem aos intervalos em que os sinais nas saídas do deslocador e do somador estão instáveis porque seus nós internos ainda não estabilizaram, e os X são valores irrelevantes.

Um diagrama de tempos detalhado para os sinais C e S é mostrado na Figura 7.29. No ciclo 1, os tempos de propagação do registrador e do deslocador são mostrados como T_{r+d} e este é ligeiramente menor que um semiciclo do relógio. No ciclo 2, os tempos de propagação do registrador e somador (T_{r+a}) são tais que há um curto intervalo entre a estabilização da soma e o *setup*.

A Equação 7.4 é uma desigualdade: o período deve ser maior ou igual que a soma dos tempos de propagação do registrador, do circuito combinacional, mais o tempo de estabilização. O período do relógio está portanto bem dimensionado para este circuito. Alguma tolerância é necessária para acomodar variações que ocorram na fabricação e/ou pequenos desvios na frequência do relógio. ◁

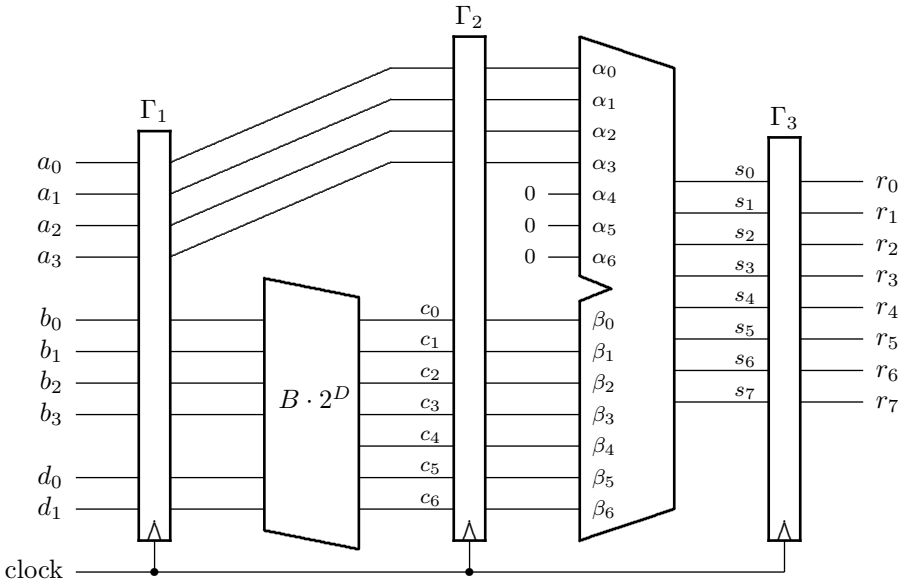


Figura 7.27: Implementação do circuito MADD.

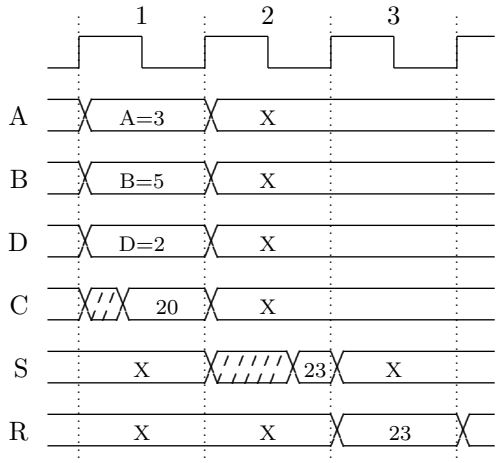


Figura 7.28: Temporização do circuito MADD.

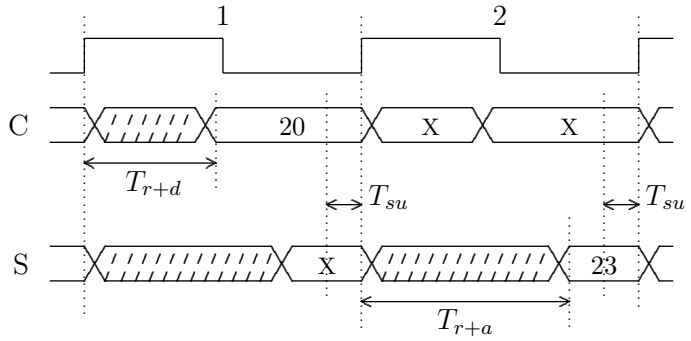


Figura 7.29: Limite na frequência máxima do relógio do MADD.

A segmentação de circuitos combinacionais é uma técnica amplamente usada para aumentar a vazão de circuitos que efetuam cálculos repetitivos. Em particular, a segmentação de pro-

cessadores é uma técnica muito eficaz para aumentar a capacidade de processamento. Esta técnica é conhecida como *pipelining* porque os registradores de segmento lembram as flanges que interligam os segmentos de tubulações num reator químico.

O circuito do Exemplo 7.7 – com blocos combinacionais separados por registradores – pode ser modelado como $R \leftarrow f(A, B)$, sendo a função $f(\,)$ implementada por um bloco combinacional com entradas A e B, com o valor da função atribuído ao registrador R no final do ciclo de relógio. Estes circuitos é que dão origem ao que se chama de "linguagem de transferência entre registradores" (*register transfer language*, ou RTL), porque os valores computados pelos circuitos combinacionais nos estágios são transferidos de registrador em registrador.

7.5.2 Operação Apropriada – *Skew*

Quando jogamos uma pedra num lago, as frentes de onda se afastam do local onde a pedra afundou em círculos concêntricos. Se traçarmos uma reta que corta estes círculos, como mostra a Figura 7.30, percebe-se que uma mesma onda atinge a reta em instantes distintos (a, a'), bem como que mais de uma onda corta a reta ao mesmo tempo, como nos pontos a, b e c.

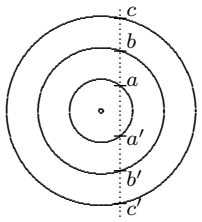


Figura 7.30: Propagação de uma frente de onda.

A analogia da onda no lago se aplica à propagação do sinal de relógio num sistema digital. O local de onde as ondas se propagam é a fonte do sinal de relógio, e as frentes de onda são as bordas do relógio, avançando com uma velocidade entre $0,6c$ e $0,7c$, que é aproximadamente um palmo por nanosegundo. Infelizmente, os efeitos das bordas de sinais digitais viajam a velocidades um tanto menores que $0,7c$ porque, embora a frente da onda eletromagnética se desloque a $0,7c$, a determinação dos níveis lógicos depende da resistência e da capacitância distribuída ao longo do trajeto dos sinais, o que acentua o problema. Se a forma das ondas for aproximada por losangos, elas avançam de forma enviesada pelo circuito, e este comportamento é batizado de *clock skew*, por causa do viés na propagação do sinal de relógio.

O circuito da Figura 7.31 explicita os atrasos a_1 e a_2 nos sinais de relógio dos dois *flip-flops* clk_1 e clk_2, com relação ao sinal original de relógio.

Para cada par de *flip-flops* adjacentes, nos interessa o atraso relativo entre os dois, e não o atraso com relação ao relógio original – a uma certa distância do centro do lago, num setor com mesmo raio, todos os pontos daquela região estão mais ou menos próximos de uma mesma frente de onda. O intervalo de enviesamento (*skew*) $T_{skew,1,2}$ entre os *flip-flops* FF1 e FF2 é dado pela diferença entre os instantes em que o sinal de relógio atinge o FF2 e o FF1, como define a Equação 7.9.

$$T_{skew,1,2} = t_{FF2} - t_{FF1} \tag{7.9}$$

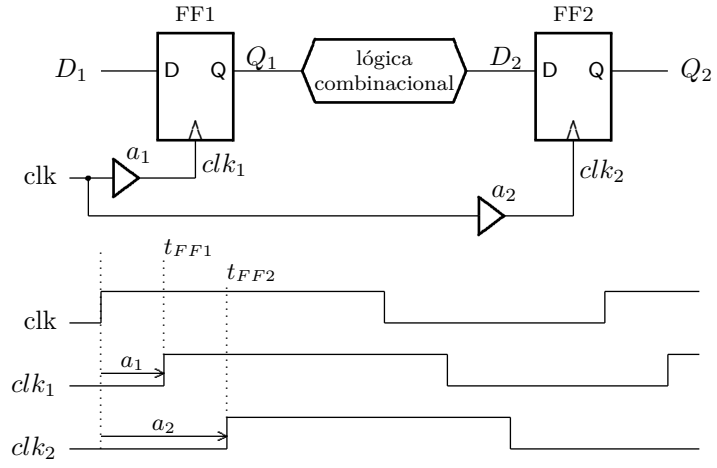


Figura 7.31: Enviesamento entre os sinais de relógio.

Skew Positivo

Se a diferença entre os tempos de chegada das bordas de relógio é positiva, $t_{FF2} - t_{FF1} > 0$, então o *skew* é positivo, e neste caso as restrições quanto a *setup* melhoram, enquanto que as do *hold* pioram, como indica a Figura 7.32.

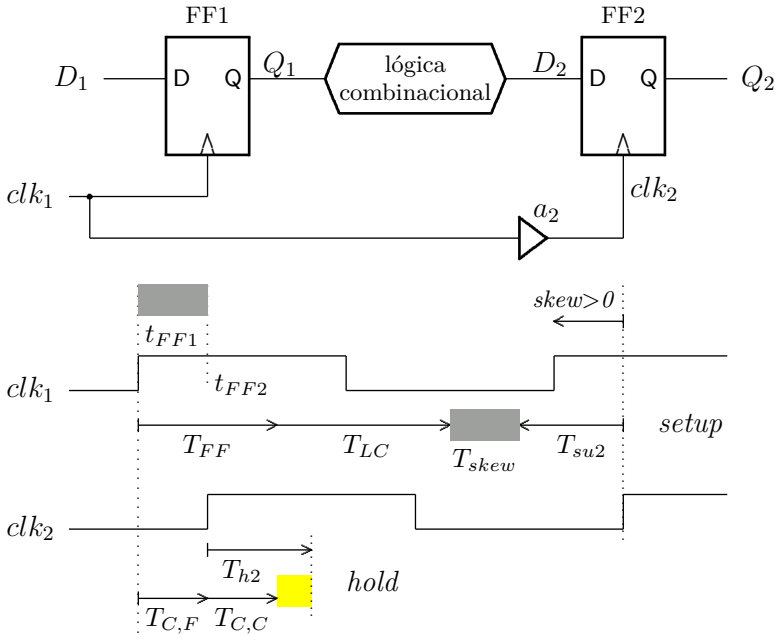


Figura 7.32: *Skew* positivo.

A folga no *setup* decorre do atraso da borda de subida em clk_2 com relação a clk_1, o que aumenta o intervalo para a propagação através do *flip-flop* (T_{FF}) e da lógica combinacional (T_{LC}) mais o *setup* no *flip-flop* destino (T_{su2}).

Por outro lado, o atraso na borda do relógio em FF2 faz aumentar a exigência de contaminação no sinal D_2, reduzindo a folga no *hold*. A soma da contaminação do FF1 ($T_{C,F}$) com a contaminação do circuito combinacional ($T_{C,C}$), descontando-se o *skew*, pode tornar-se me-

nor do que o intervalo de manutenção no FF2 (T_{h2}). Estas condições estão explicitadas na Equação 7.10.

$$\begin{aligned} \text{Folga } \textit{setup} &= T_{clk} - T_{FF} - T_{LC} - T_{su2} + T_{skew} \\ \text{Folga } \textit{hold} &= T_{C,F} + T_{C,C} - T_{h2} - T_{skew} \end{aligned} \tag{7.10}$$

A contaminação das saídas é gerada no FF1 pela borda em clk_1 e é observada no FF2 na borda em $clk2$.

Skew Negativo

Se a diferença entre os tempos de chegada das bordas de relógio é negativa, $t_{FF2} - t_{FF1} < 0$, então o *skew* é negativo. No exemplo mostrado na Figura 7.33, a restrição quanto a *setup* é violada porque o período do relógio é curto demais para acomodar a propagação dos sinais e o intervalo de *setup*:

$$T_{min} < T_{FF} + T_{LC} + T_{su2}\,.$$

A restrição do *hold* melhora porque a contaminação inicia com atraso, o que aumenta o intervalo de estabilidade no sinal D_2 antes da borda em clk_2.

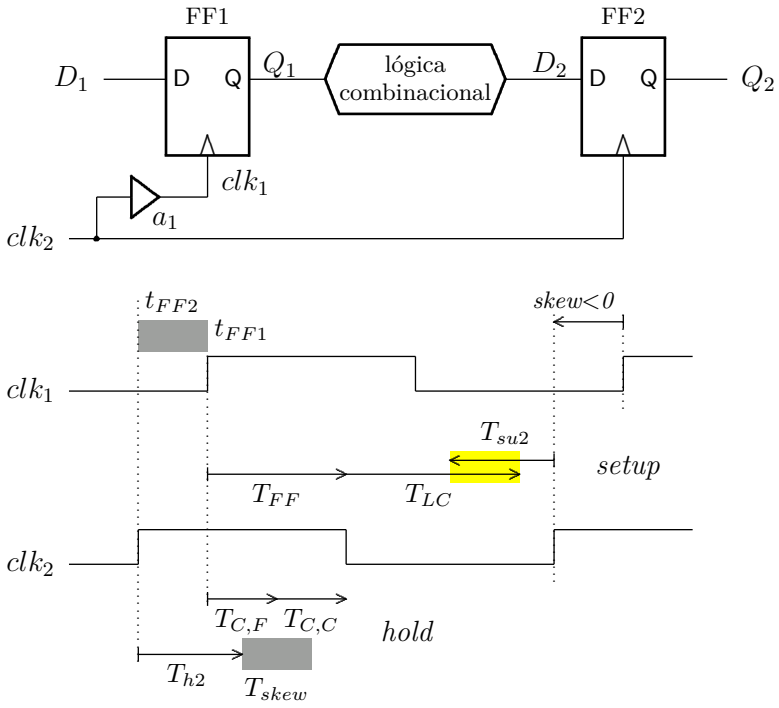


Figura 7.33: *Skew* negativo.

Por causa do atraso na borda de clk_1, o intervalo entre as bordas do relógio nos *flip-flops* fonte e destino diminui – as bordas se aproximam – o que reduz a folga no cálculo do *setup*. O atraso na chegada do relógio em FF1 diminui a exigência de contaminação no sinal D_2, melhorando a folga para a restrição de *hold*. Estas condições são as mesmas que na Equação 7.10, exceto que $T_{skew} < 0$.

Período Mínimo e Tempo de Manutenção na Presença de *Skew*

Quando há enviesamento nos sinais de relógio, a equação que computa o período mínimo do relógio deve ser adaptada para acomodar as diferenças entre as bordas dos sinais de relógio. O *skew* deve ser subtraído ao período mínimo T_{min}, enquanto que o *skew* é adicionado no cálculo da contaminação, como mostram as Equações 7.11 e 7.12. Lembre que $T_{skew} = t_{FF2} - t_{FF1}$.

$$T_{min} \geq T_{FF} + T_{LC} + T_{su} - T_{skew} \tag{7.11}$$

$$T_{C,F} + T_{C,C} - T_{skew} \geq T_h \tag{7.12}$$

Exemplo 7.8 Considere o circuito da Figura 7.26, na página 195. A especificação temporal dos *flip-flops* é: $T_{FF} = 10\,\text{ns}$, $T_{C,F} = 5\,\text{ns}$, $T_{su} = 4\,\text{ns}$, $T_h = 2\,\text{ns}$. A especificação temporal do circuito combinacional é: $T_{LC} = 6\,\text{ns}$, $T_{C,C} = 2\,\text{ns}$. O *skew* é $T_{skew} = -5\,\text{ns}$.

Qual é o período mínimo do relógio?

$$T_{min} = T_{FF} + T_{LC} + T_{su} - T_{skew} = 10 + 6 + 4 - (-5) = 25\,\text{ns}.$$

Há folga no tempo de manutenção do FF2?

$$\text{Folga } \textit{hold} = T_{C,F} + T_{C,C} - T_{h2} - T_{skew} = 5 + 2 - 2 - (-5) = 10\,\text{ns}.$$

Há folga no tempo de estabilização do FF2?

$$\text{Folga } \textit{setup} = T_{clk} - T_{FF} - T_C - T_{su2} + T_{skew} = 25 - 10 - 6 - 4 - 5 = 0$$

Neste caso não há folga porque empregou-se $T_{clk} = T_{min}$. ◁

Exemplo 7.9 Considere o circuito da Figura 7.26, com a seguinte especificação temporal: *flip-flops* com $T_{FF} = 6\,\text{ns}$, $T_h = 3\,\text{ns}$, $T_{su} = 6\,\text{ns}$, $T_{C,F} = 1\,\text{ns}$, lógica combinacional com $T_{C,C} = 4\,\text{ns}$ e $T_{LC} = 25\,\text{ns}$. O *skew* é $T_{skew} = +1\,\text{ns}$.

Qual é o período mínimo do relógio?

$$T_{min} = T_{FF} + T_{LC} + T_{su} - T_{skew} = 6 + 25 + 6 - 1 = 36\,\text{ns}.$$

Há folga no tempo de manutenção do FF2?

$$T_h = 3\,\text{ns} \not\geq T_{C,F} + T_{C,C} - T_{skew} = 1 + 4 - 1 = 4\,\text{ns}.$$

Não há folga, e portanto é necessário acrescentar um *buffer* com $T_{C,b} \geq 1\,\text{ns}$ para atender à especificação de *hold* do FF2. O tempo de propagação deste buffer provavelmente aumentará o período mínimo T_{min}. Uma alternativa é acrescentar um *buffer* com tempo de propagação de $T_b = 1\,\text{ns}$ na entrada de relógio do FF1, o que eliminaria o *skew* mas acrescentaria 1 ns a T_{min}. ◁

Definição de Tempo de Propagação Revisitada

Finalmente podemos explicitar uma premissa escondida na definição de *tempo de propagação*. A definição da Seção 5.4 reza que o intervalo de propagação se inicia quando *todas as entradas ficam estáveis*. O que está escondido na definição é que o modelo de circuito que se emprega, na grande maioria dos casos, é aquele da Figura 7.25, no qual as entradas do circuito combinacional provêm de um registrador, e suas saídas alimentam outro registrador. Esta metodologia de projeto é descrita no Exemplo 7.7.

Se todas as entradas do circuito combinacional provêm da saída de um ou mais registradores então estes sinais estabilizam aproximadamente ao mesmo tempo. Todas as entradas são atualizadas na mesma borda do relógio, e o circuito que conduz o sinal de relógio é otimizado para minimizar o *skew* entre todos os registradores fonte e todos os registradores destino.

Se o circuito sequencial é uma máquina de estados implementada segundo o modelo da Figura 7.24, o registrador de estado pode ser encarado como se fosse dividido em dois, no qual as saídas – registrador fonte – são consideradas separadamente das entradas – registrador destino.

"Não Se Faz Lógica Com O Relógio"

A máxima de projeto "não se faz lógica com o relógio" pretende defender o projetista contra uma tentação comum, que é de acrescentar lógica (combinacional) ao sinal de relógio[5]. Uma porta lógica aqui e outra acolá reduzem dramaticamente qualquer possibilidade de que o *skew* seja eliminado do sinal de relógio. Por "lógica com o relógio" não me refiro aos atrasos introduzidos para compensar as diferenças no tempo de propagação do sinal pelo circuito, mas sim a tentativas preguiçosas de inserir ou remover bordas no sinal de relógio.

Se o que se deseja é controlar a atualização do conteúdo de *flip-flops* ou registradores, deve-se empregar registradores como aquele da Figura 7.22, no qual a lógica de decisão é inserida exclusivamente no caminho de dados, e nunca no caminho de sincronização. Além do problema com *skew*, o circuito combinacional pode introduzir transitórios na entrada de relógio, que causam erros que são particularmente difíceis de detectar.

Sistemas de síntese proíbem lógica com o sinal de relógio porque isso inviabiliza a otimização do *skew* na rede de distribuição daquele sinal.

Exercícios

Ex. 7.1 Desenhe um diagrama de tempos similar ao da Figura 7.28 para a sequência de entradas $\langle 1, 2, 0\rangle$, $\langle 4, 5, 1\rangle$, $\langle 7, 9, 2\rangle$, $\langle 15, 15, 3\rangle$. Considerando que a cada ciclo uma das triplas na sequência é amostrada, quantos ciclos são necessários para computar os quatro resultados?

[5] O autor escutou essa frase, pela primeira vez, dita pelo Professor Anatólio Laschuck, na UFRGS, no início dos anos 1980.

Ex. 7.2 Considere o circuito da Figura 7.34. A especificação temporal dos *flip-flops* é: $T_{FF} = 3\,\text{ns}$, $T_{C,F} = 1\,\text{ns}$, $T_{su} = 2\,\text{ns}$, $T_h = 2\,\text{ns}$, e o *skew* é zero. A especificação temporal do circuito combinacional é: $T_{LC} = 10\,\text{ns}$, $T_{C,C} = 3\,\text{ns}$. (a) Qual é o período mínimo do relógio? (b) Há folga no tempo de *hold* do FF2? (c) Qual deve ser o *setup* mínimo nas entradas? (d) Qual deve ser o *hold* mínimo nas entradas?

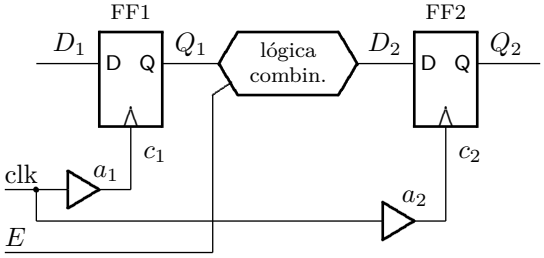


Figura 7.34: Circuito com entradas e *skew*.

Ex. 7.3 Considere o circuito da Figura 7.34. A especificação temporal dos *flip-flops* é: $T_{FF} = 4\,\text{ns}$, $T_{C,F} = 1\,\text{ns}$, $T_{su} = 2\,\text{ns}$, $T_h = 2\,\text{ns}$, $T_{skew} = -3\,\text{ns}$. A especificação temporal do circuito combinacional é: $T_{LC} = 12\,\text{ns}$, $T_{C,C} = 4\,\text{ns}$. (a) Qual é o período mínimo do relógio? (b) Há folga no tempo de *hold* do FF2? (c) Qual deve ser o *setup* mínimo nas entradas? (d) Qual deve ser o *hold* mínimo nas entradas?

Índice Remissivo