Latches, Flip-Flops e Registradores

6.1 Introdução

Um circuito combinacional, conforme visto nos capítulos anteriores, é um circuito digital cujos sinais de saída são função unicamente dos atuais valores dos sinais de entrada. Quando uma entrada tem seu valor modificado, este evento pode causar uma alteração praticamente imediata em uma ou mais saídas, dependendo obviamente das funções lógicas implementadas e desprezados os atrasos de propagação dos sinais. Os valores das saídas não dependem dos valores passados das entradas, ou mesmo dos valores apresentados anteriormente nas saídas, o que significa dizer que o circuito não tem capacidade de memorização ou registro de dados. Exemplos de circuitos combinacionais são multiplexadores, decodificadores, geradores de paridade, somadores e multiplicadores, apresentados nos Capítulos 4 e 5, entre outros.

Um circuito seqüencial, por sua vez, é aquele cujas saídas são função de uma seqüência de valores das entradas e não apenas dos valores atuais das mesmas. Para que o circuito possa exibir este comportamento, ele precisa possuir capacidade de armazenamento de valores.

Exemplo 6.1. Seja um somador serial de 4 bits que efetua a soma de 2 operandos 'A' e 'B' em 4 tempos consecutivos, através da seguinte seqüência de passos:

- tempo 1 - $A_0 + B_0$: calcula 'S_0', gera e armazena 'C_1' para o tempo seguinte;
- tempo 2 - $A_1 + B_1 + C_1$: calcula 'S_1', gera e armazena 'C_2' para o tempo seguinte;
- tempo 3 - $A_2 + B_2 + C_2$: calcula 'S_2', gera e armazena 'C_3' para o tempo seguinte;
- tempo 4 - $A_3 + B_3 + C_3$: calcula 'S_3' e gera 'C_4', ou o sinal *carry-out* do somador.

Este é um circuito seqüencial, pois ele executa sua função em uma seqüência de passos de tempo e depende da capacidade de memorização para transferir o valor do sinal *vai-um* 'C_n', calculado em um passo de tempo, para o passo seguinte.

Exemplo 6.2. Seja um circuito que conta N ocorrências de um determinado evento. Um sinal binário de entrada 'E' informa cada ocorrência do evento através de um rápido pulso com valor '1', ficando com o valor '0' enquanto não há ocorrência de eventos. O circuito tem uma única saída 'S', cujo valor é

- '0', enquanto não chegar o n-ésimo pulso de contagem de eventos;
- '1', quando chegar o n-ésimo pulso de contagem de eventos; e
- '0', novamente, no pulso de contagem subseqüente ao n-ésimo pulso.

Este ciclo de contagem de pulsos até N se repete indefinidamente.

Este também é um circuito seqüencial, pois ele executa sua função em uma seqüência de passos de tempo e depende da capacidade de memorização para determinar em que momento chegou o n-ésimo pulso de contagem dos eventos.

Exemplo 6.3. Seja um circuito cuja função é detectar a seqüência de valores ‘0110’ em um sinal de entrada binário ‘E’. O circuito examina indefinidamente o vetor de valores formado pelo valor atual da entrada ‘E’ e pelos seus 3 valores imediatamente anteriores, comparando o mesmo com o código ‘0110’ (onde o bit mais significativo do vetor corresponde ao valor atual de ‘E’). O circuito tem uma única saída ‘S’, cujo valor é

- ‘0’, no início do funcionamento do circuito;
- ‘1’, se os 4 últimos valores da entrada ‘E’ formarem o vetor ‘0110’; e
- ‘0’, se os 4 últimos valores da entrada ‘E’ não formarem o vetor ‘0110’.

A Figura 6.1 ilustra uma seqüência de valores na entrada ‘E’, com os correspondentes valores de ‘S’. Neste caso, cada período de 30ns representa uma nova leitura do valor de entrada.

Este também é um circuito seqüencial, pois ele executa sua função em uma seqüência de passos de tempo e depende da capacidade de memorização para determinar se os 4 últimos valores da entrada ‘E’ são iguais ao padrão ‘0110’, como acontece nos intervalos de tempo de 60ns a 180ns e de 330ns a 450ns.

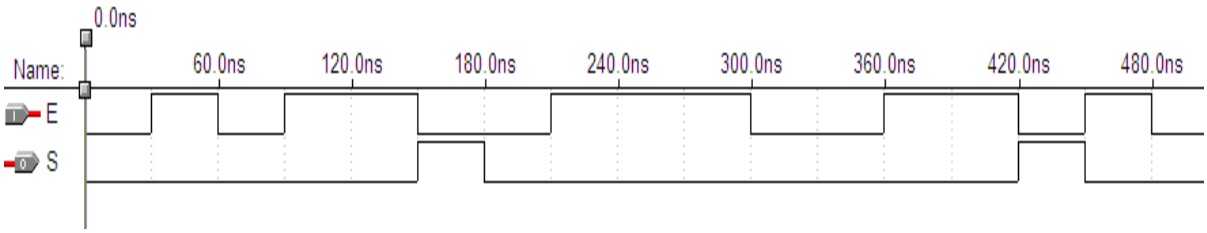


Figura 6.1 – Seqüência ilustrativa de valores para o Exemplo 6.3.

Um circuito seqüencial é implementado através do arranjo de uma ou mais lógicas combinacionais com elementos de memória, conforme a representação da Figura 6.2. Esta representação corresponde a uma *máquina de estados*, que será estudada detalhadamente no próximo capítulo. Neste capítulo serão apresentados os elementos de armazenagem de dados, chamados de *latch*es e *flip-flops*, assim como estruturas clássicas que utilizam tais elementos para a implementação de funções como registradores de vetores binários, registradores de deslocamento, contadores e divisores de freqüência.

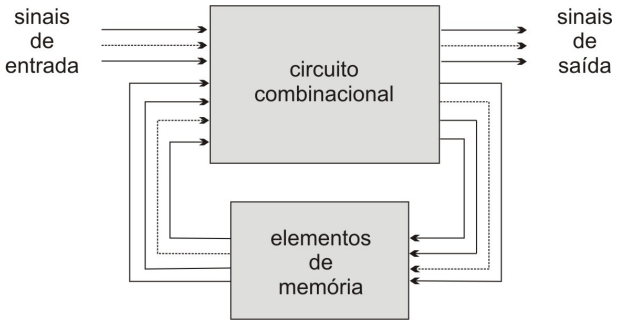


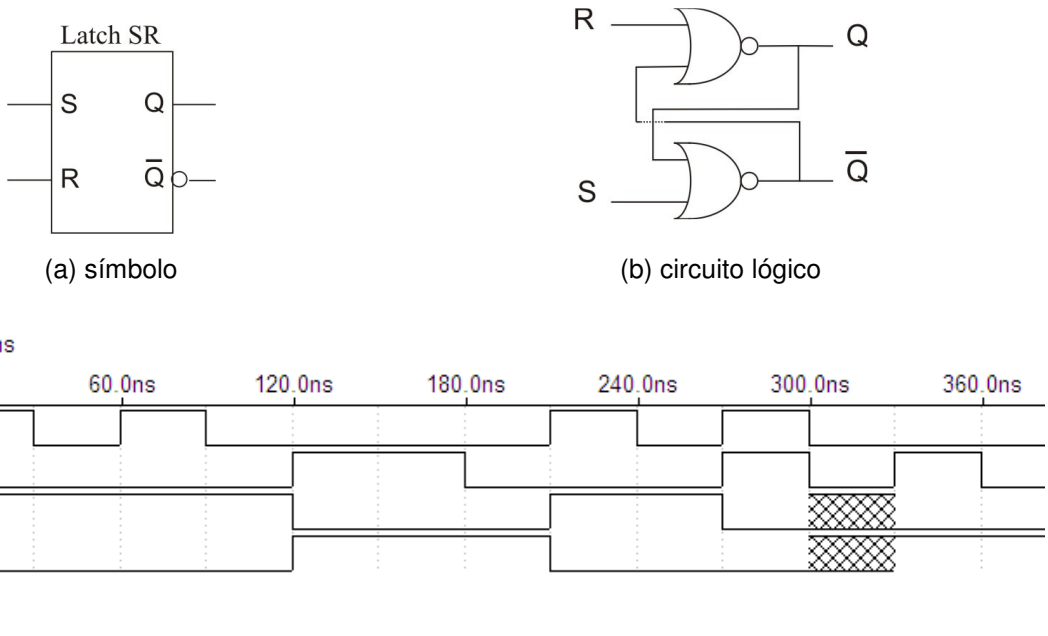
Figura 6.2 – Representação canônica de um circuito seqüencial.

Os elementos de memória têm o objetivo de armazenar temporariamente o valor de um sinal binário, ou 1 bit. Estes elementos podem ser vistos como portas lógicas ou circuitos lógicos cujas saídas são modificadas para certas combinações dos sinais de entrada e permanecem inalteradas para outras.

6.2 Latch SR

Um dos elementos de memorização mais simples de ser construído, a partir de portas lógicas básicas, é o *latch* SR. Este circuito apresenta dois sinais de entrada 'S' (*set*) e 'R' (*reset*) e o sinal de saída 'Q' (um segundo sinal de saída '$\overline{Q}$', complementar ao sinal 'Q', geralmente também é disponibilizado). Os sinais de entrada definem se a saída será *ligada* (receberá o valor '1') a partir da ativação do sinal de entrada 'S', ou *desligada* (receberá o valor '0') a partir da ativação do sinal de entrada 'R'. Caso nenhuma das duas entradas seja ativada, o valor de saída deve permanecer inalterado. Parece evidente que os dois sinais 'S' e 'R' não devam ser ativados simultaneamente, pois não tem sentido querer colocar dois níveis lógicos na saída ao mesmo tempo. Os sinais de entrada podem ser considerados ativados quando apresentarem o valor lógico '1' (neste caso diz-se que o sinal é ativo em nível lógico alto) ou o valor '0' (sinal ativo em nível lógico baixo), dependendo do circuito lógico desenvolvido.

Na Figura 6.3 tem-se um latch SR formado por duas portas NOR de duas entradas cada, sendo que uma das entradas de cada porta NOR corresponde ao sinal de saída da outra porta NOR, em configuração de realimentação ou *feedback*. Neste caso, os sinais 'S' e 'R' são ativos em nível lógico alto ou '1', pois este valor lógico na entrada da porta NOR garante o valor conhecido '0' na sua saída ('Q' ou '$\overline{Q}$'), independentemente do valor das demais entradas. Sendo portanto R = 1, então tem-se Q = 0 e, sendo necessariamente S = 0, tem-se $\overline{Q} = 1$. Do contrário, no caso de S = 1 tem-se $\overline{Q} = 0$ e, conseqüentemente, Q = 1 uma vez que necessariamente deve-se ter R = 0.



(a) símbolo (b) circuito lógico

(c) formas de onda

Figura 6.3 – Latch SR com portas NOR.

Nas formas de onda apresentadas na Figura 6.3, o sinal '$\overline{S}$' apresenta o valor '1' até o tempo de 30ns. Por este motivo, o sinal de saída complementar '$\overline{Q}$' apresenta obrigatoriamente o valor '0' resultante da função lógica NOR. Em conseqüência, a porta NOR superior gera o sinal de saída 'Q' com valor '1', visto que os respectivos sinais de entrada apresentam o valor '0'. O sinal 'Q', então com valor '1', reforça o valor do sinal '$\overline{Q}$' igual a '0' através da porta NOR inferior, onde ele é utilizado como entrada.

No tempo de 30ns, o sinal ‘S’ passa a ter o valor ‘0’. A saída ‘$\overline{Q}$’ não altera seu valor, pois o sinal ‘Q’ está com valor ‘1’ e mantém a lógica da porta NOR inferior. Uma vez que ‘R’ e ‘$\overline{Q}$’ não modificaram seus valores, não há alteração do sinal ‘Q’ na respectiva porta NOR.

No tempo de 60ns o sinal ‘S’ retorna ao valor ‘1’. Porém nenhum outro sinal é modificado: o sinal ‘$\overline{Q}$’ permanece em ‘0’ pois tanto ‘Q’ quanto ‘S’ agora possuem o valor ‘1’. Não havendo modificação em ‘$\overline{Q}$’ e ‘R’, não há modificação no sinal ‘Q’. No tempo de 90ns, quando ‘S’ passa novamente ao valor ‘0’, tem-se a mesma análise do tempo de 30ns, sem nenhuma modificação dos demais sinais.

No tempo de 120ns, o sinal ‘R’ recebe o valor lógico ‘1’ e com isso o sinal ‘Q’ forçosamente passa ao valor ‘0’, segundo a função lógica NOR. Com ambos os sinais ‘Q’ e ‘$\overline{S}$’ com valor ‘0’ a porta NOR inferior gera na sua saída o valor ‘1’, correspondente ao sinal ‘$\overline{Q}$’. Na sequência, estando o sinal ‘$\overline{Q}$’ em ‘1’, o sinal ‘Q’ é mantido em ‘0’, mesmo com o retorno da entrada ‘R’ para ‘0’, em 180ns. O valor de ‘Q’ só pode ser modificado, ou levado para ‘1’, caso ambas as entradas da porta NOR superior apresentarem o valor ‘0’. Isso ocorrerá novamente no tempo de 210ns, quando o sinal ‘S’ recebe o valor ‘1’, levando o sinal ‘$\overline{Q}$’ para ‘0’.

Por fim, no tempo de 270ns tem-se uma situação de conflito quanto à lógica do latch SR: ambos os sinais ‘S’ e ‘R’ estão ativados, ou com valor ‘1’. O que se quer, ‘ligar’ ou ‘desligar’ o sinal de saída ‘Q’? Neste caso em que tal latch é construído a partir de duas portas NOR tem-se ambos os sinais de saída ‘Q’ e ‘$\overline{Q}$’ com o valor ‘0’. No momento em que os sinais de entrada retornam a ‘0’, é praticamente impossível determinar qual das portas NOR apresentará na sua saída o valor ‘1’. Essa definição ocorrerá quase que aleatoriamente em razão de pequenos detalhes elétricos de construção deste latch, e que não serão abordados neste livro. As saídas só voltarão a ter uma definição lógica previsível no momento em que um dos sinais ‘S’ ou ‘R’ for ativado, o que ocorre no tempo de 330ns.

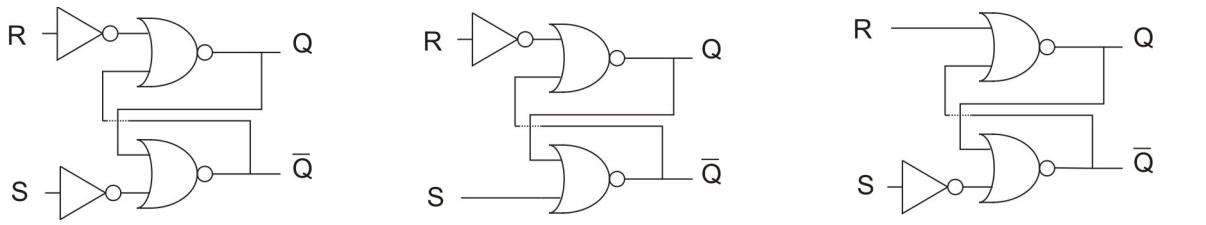
Cabe observar que se a análise com as formas de onda fosse iniciada com ambos os sinais de entrada ‘S’ e ‘R’ iguais a ‘0’, seria difícil determinar o valor inicial dos sinais de saída ‘Q’ e ‘$\overline{Q}$’, por não serem conhecidos os valores anteriores armazenados no latch. Por este motivo, sugere-se que a análise de um latch SR inicie sempre com um dos sinais de entrada ativado para estabelecimento de um valor inicial conhecido nas saídas.

Como conclusão desta análise, chega-se à constatação de que ‘Q’ e ‘$\overline{Q}$’ são sinais que apresentam sempre valores complementares, exceto na situação inesperada S = R = 1. Constata-se também que ‘S’ e ‘R’ são entradas de controle que estão ativadas quando apresentam o valor ‘1’. Quando ‘S’ está ativado, o latch tem em sua saída ‘Q’ o valor ‘1’, enquanto ‘$\overline{Q}$’ apresenta o valor complementado ‘0’. A entrada ‘S’ é então um sinal que *liga* o latch. Quando ‘R’ está ativado, o latch tem em sua saída ‘Q’ o valor ‘0’, e consequentemente ‘$\overline{Q}$’ apresenta o valor complementado ‘1’. A entrada ‘R’ é portanto um sinal que *desliga* o latch. Enquanto tanto ‘S’ como ‘R’ estiverem inativos, neste caso com valor ‘0’, o latch mantém armazenado o valor anterior e suas saídas permanecem inalteradas.

A Tabela 6.1 mostra a tabela-verdade do latch SR. Ao contrário dos circuitos combinacionais, onde as saídas são função apenas dos valores atuais das entradas, a tabela-verdade de um circuito seqüencial precisa refletir o fato de que as saídas dependem também de valores anteriores dos sinais. Na tabela-verdade do latch SR a notação ‘Q_i’ identifica o valor de ‘Q’ no tempo t_i, enquanto ‘Q_{i-1}’ identifica o valor de ‘Q’ em um intervalo de tempo imediatamente anterior t_{i-1}.

Tabela 6.1 – Tabela-verdade do latch SR, com os sinais ‘S’ e ‘R’ ativos em nível lógico alto.

S	R	Q_i
0	0	Q_{i-1}
0	1	0
1	0	1
1	1	X



(a) ‘S’ e ‘R’ ativos em ‘0’ (b) ‘S’ ativo em ‘1’, ‘R’ ativo em ‘0’ (c) ‘S’ ativo em ‘0’, ‘R’ ativo em ‘1’

Figura 6.4 – Latches SR com sinais de entrada ativos em diferentes níveis lógicos.

Caso se queira construir latches SR cujos sinais de entrada são ativos em nível lógico baixo, ou mesmo diferenciados, um ativo em alto e outro em baixo, basta acrescentar uma porta lógica inversora na respectiva entrada, conforme ilustrado na Figura 6.4. Diz-se que a entrada ativada em nível lógico baixo opera segundo uma lógica *negada*. Para fins de símbolo lógico, utiliza-se um circulo de negação para indicar quando o sinal de entrada é ativado com valor ‘0’, conforme ilustrado na Figura 6.5(a).

A Figura 6.5 apresenta um latch SR implementado com portas lógicas NAND de duas entradas cada. Da análise das formas de onda na Figura 6.5(c) conclui-se que:

- ‘Q’ e ‘$\overline{Q}$’ têm valores complementares, exceto na situação em 270ns, em que ‘R’ e ‘S’ são simultaneamente iguais a ‘0’ (ativos em nível lógico baixo), o que deve ser evitado;
- quando R = 1 e S = 1, os valores de ‘Q’ e ‘$\overline{Q}$’ permanecem como no estado anterior;
- quando R = 0 (ativo) e S = 1, as saídas são $Q = 0$ e $\overline{Q} = 1$; e
- quando R = 1 e S = 0 (ativo), as saídas são $Q = 1$ e $\overline{Q} = 0$.

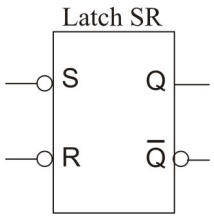
Comparando-se a seqüência de valores para o latch SR implementado com portas NAND com aquela obtida para o latch com portas NOR, constata-se que, ao serem utilizadas portas NAND, inverteu-se a lógica dos sinais de controle ‘S’ e ‘R’. Estes sinais de controle agora estão ativos quando apresentam o valor ‘0’. Caso queira-se alterar o nível de ativação dos sinais de entrada para o latch com portas NAND, basta utilizar portas NOT (inversores) da mesma forma mostrada na Figura 6.4.

6.3 Registradores sensíveis ao nível (latches)

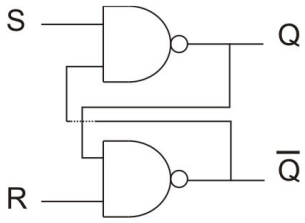
Os elementos de memória, de forma geral, possuem sinal de habilitação ou sincronismo para atualização dos sinais de saída. Distinguem-se claramente dois grupos de elementos lógicos seqüenciais controlados por sinal de habilitação: registradores sensíveis ao nível e registradores sensíveis à borda.

Os registradores sensíveis ao nível, tratados neste livro pela denominação *latches*, possuem um sinal de habilitação que, quando ativado, permite a atuação direta dos sinais de entrada sobre os sinais de saída, segundo a lógica correspondente. Caso o sinal de habilitação esteja no nível lógico de desativação, não há influência alguma dos valores de entrada sobre os valores de saída, e mesmo sobre os sinais memorizados internamente na estrutura lógica. Os

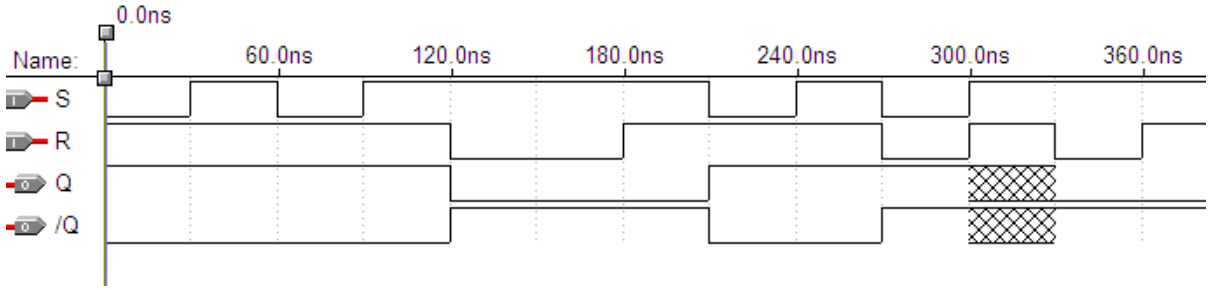
registradores sensíveis à borda, por sua vez, neste livro tratados pelo termo *flip-flops*, serão discutidos mais adiante no capítulo.



(a) símbolo



(b) circuito lógico



(c) formas de onda

Figura 6.5 – Latch RS com portas NAND.

6.3.1 Registradores SR

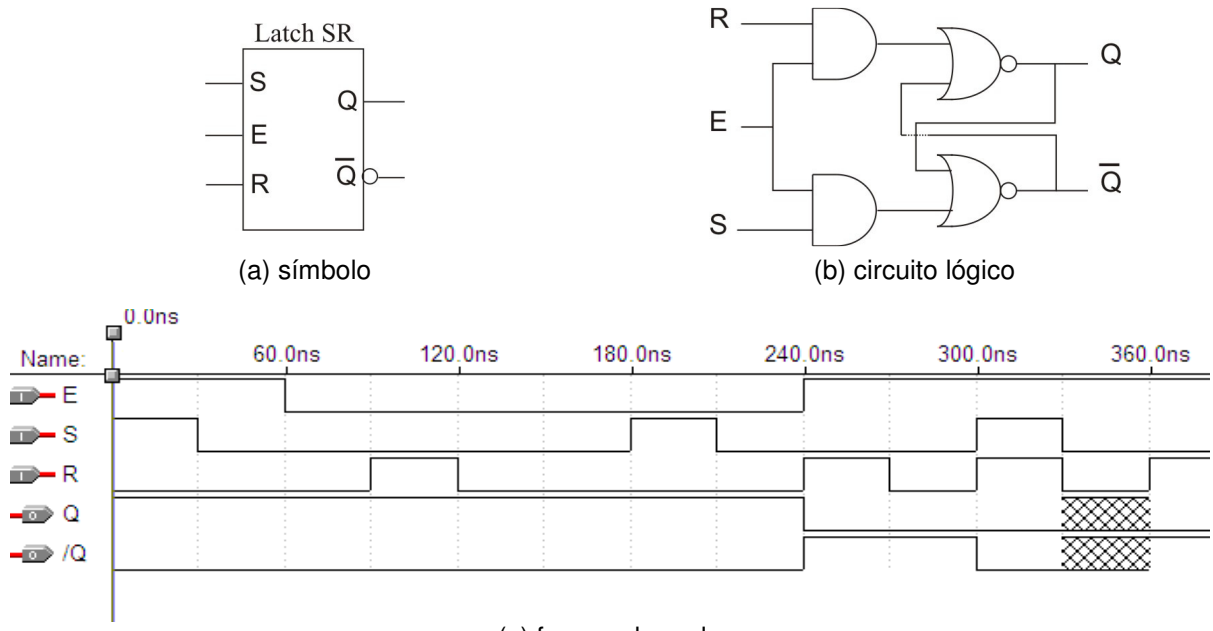
A Figura 6.6 mostra um latch SR implementado com portas NOR ao qual foi acrescentada uma lógica de entrada destinada a condicionar a atuação dos sinais 'S' e 'R'. Esta lógica é baseada em um sinal de *habilitação* 'E' (*enable*). Neste caso, quando o sinal 'E' tem o valor '1', os valores de 'S' e 'R' passam inalterados para as saídas das portas AND, acrescentadas ao latch, fazendo o mesmo operar normalmente de acordo com a tabela-verdade da Tabela 6.1. No entanto, quando o sinal 'E' apresenta o valor '0', as duas portas AND terão obrigatoriamente o valor '0' em suas saídas, mantendo inalterados os valores de saída do latch SR. Portanto, E = 0 desabilita a ação dos sinais 'S' e 'R', não importando os valores dos mesmos, conforme ilustrado na Figura 6.6(c) para o intervalo de tempo de 60ns a 240ns.

Na discussão sobre o funcionamento dos latches SR, apresentados anteriormente, os sinais de entrada 'S' e 'R' foram apresentados como sinais de 'controle'. Com a introdução do sinal de habilitação 'E' no latch da Figura 6.6, os sinais 'S' e 'R' devem ser pensados agora como sinais de 'dados', ficando o sinal 'E' com a função de controle, motivo pelo qual este latch é dito 'controlado'. Assim, se os dados estão habilitados por E = 1, o latch armazena o valor '1' quando 'S' está ativo e o valor '0' quando 'R' está ativo. Caso ambos estejam inativos o valor do latch não é alterado, mesmo estando o sinal 'E' habilitado.

Em outras palavras, pode-se dizer também que quando o latch está com seu controle desabilitado o mesmo encontra-se em modo de memorização. Do contrário, com o controle habilitado, o latch é dependente de toda e qualquer variação dos sinais de entrada.

Diz-se que este tipo de registrador é sensível ao nível, pois é o nível lógico da entrada de controle 'E' que habilita a leitura das entradas de dados do circuito. Pode-se alterar o nível de habilitação do latch para E = 0 apenas invertendo este sinal. As configurações apresentadas nas Figuras 6.4 e 6.5 também são facilmente modificadas para incluir o sinal de habilitação ativo em nível lógico alto ou baixo, como nos exemplos apresentados na Figura 6.7. Em tais exemplos

tem-se diferentes níveis de ativação para o sinal de habilitação ‘E’, bem como para os sinais de dados ‘S’ e ‘R’; o círculo nos sinais do símbolo lógico indica quando o valor de ativação é baixo ou ‘0’.

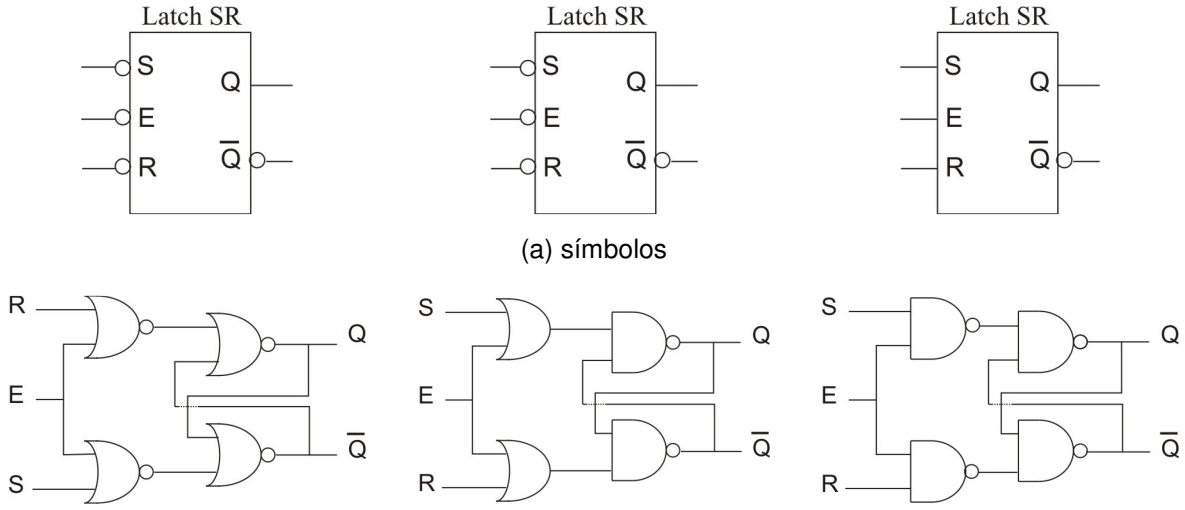


(a) símbolo (b) circuito lógico

(c) formas de onda

Figura 6.6 – Latch SR com sinal de habilitação.

com portas NOR com portas OR e NAND com portas NAND



(a) símbolos

(b) circuitos lógicos

E	S	R	Q_i
0	0	0	X
0	0	1	1
0	1	0	0
0	1	1	Q_{i-1}
1	X	X	Q_{i-1}

E	S	R	Q_i
0	0	0	X
0	0	1	1
0	1	0	0
0	1	1	Q_{i-1}
1	X	X	Q_{i-1}

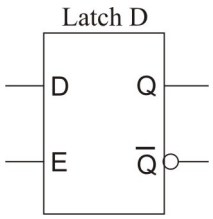
E	S	R	Q_i
0	X	X	Q_{i-1}
1	0	0	Q_{i-1}
1	0	1	0
1	1	0	1
1	1	1	X

(c) tabelas-verdade

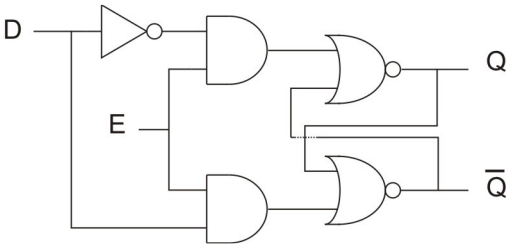
Figura 6.7 – Outros latches SR com sinal de habilitação.

6.3.2 Registradores D

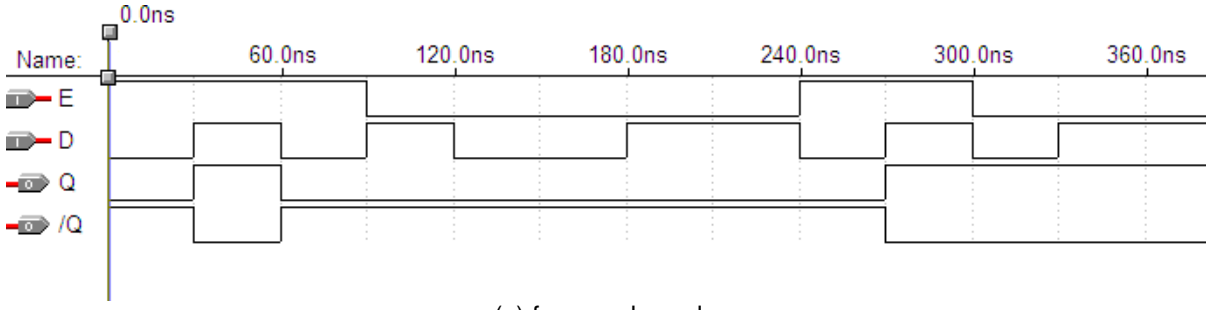
Os latches SR apresentados até agora possuem dois sinais de dados ‘S’ e ‘R’ e um eventual sinal de controle ‘E’. Para armazenar um determinado valor, deve-se colocar valores adequados nos dois sinais de dados ‘S’ e ‘R’. Além disto, deve-se evitar a combinação indesejável na qual ambos os sinais de dados estão ativos simultaneamente. Uma outra configuração de latch, que elimina esta inconveniência, é apresentada na Figura 6.8. Este novo latch possui uma única entrada de dados, indicada por ‘D’, por isso ele é conhecido como latch tipo ‘D’, ou simplesmente latch ‘D’. Na comparação com o circuito apresentado na Figura 6.6(b), observa-se que o sinal ‘D’ está diretamente ligado ao sinal que antes era identificado como ‘S’, enquanto o sinal ‘R’ original recebe o valor complementado de ‘D’. Com isto garante-se que ‘S’ e ‘R’ sempre terão valores complementares.



(a) símbolo



(b) circuito lógico



(c) formas de onda

Figura 6.8 – Latch D.

A tabela-verdade do latch D é mostrada na Tabela 6.2. Seu funcionamento é bastante simples: quando se quer armazenar um novo valor no latch este deve estar habilitado e a entrada ‘D’ deve possuir o valor lógico que será armazenado e amostrado na saída ‘Q’. Apenas por curiosidade, perceba que a versão do latch D sem sinal de habilitação não tem sentido de ser construída, pois neste caso teríamos um circuito lógico sem função lógica alguma e sem capacidade de memorização: se a entrada ‘D’ fosse ‘1’ a saída ‘Q’ também seria ‘1’, acompanhando o valor de ‘D’; o mesmo aconteceria para D = 0 fazendo Q = 0.

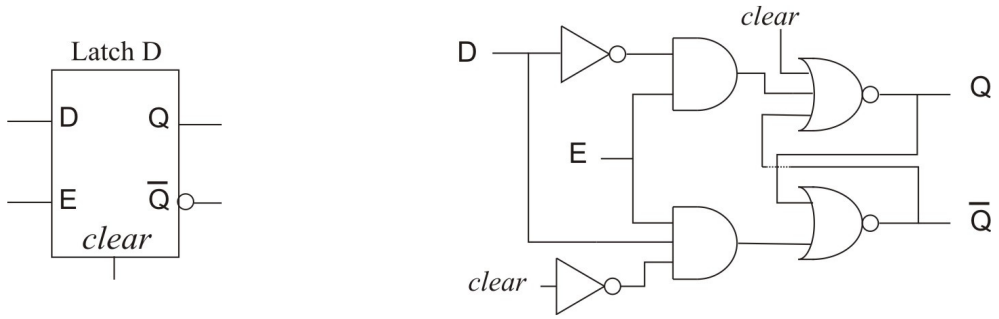
Tabela 6.2 – Tabela-verdade do latch D com sinal de controle ‘E’ ativo em nível lógico alto.

E	D	Q_i
0	X	Q_{i-1}
1	0	0
1	1	1

O latch D é um dos mais utilizados na construção de sistemas digitais. Outras configurações de latch D podem ser feitas utilizando-se por base, por exemplo, os latches SR mostrados na Figura 6.7.

A análise apresentada nas formas de onda da Figura 6.8(c) inicia com o latch D habilitado (E = 1); do contrário não se saberia que valores iniciais colocar nos sinais de saída 'Q' e '$\overline{Q}$'. Esta dificuldade ocorre na prática quando um circuito é formado por este tipo de latch e o seu sinal de habilitação encontra-se inicialmente inativo. Neste caso, o valor lógico na saída do latch, bem como no seu laço de realimentação que permite a memorização do dado, será definido de forma aleatória segundo detalhes na construção física do circuito. Este fato muitas vezes é indesejável e o usuário do circuito gostaria de inicializar o latch com certo valor lógico. Para realizar esta inicialização do latch D, acrescenta-se os sinais de entrada *clear* ou *preset*, ou mesmo ambos, no circuito. O sinal *clear* quando ativado garante na saída 'Q' o valor '0' e na saída '$\overline{Q}$' o valor '1', independentemente dos valores das entradas 'E' e 'D'. O sinal *preset*, por sua vez, faz Q = 1 e $\overline{Q} = 0$, sem também tomar conhecimento dos valores de 'E' e 'D'. Esses novos sinais de entrada *preset* e *clear* são também denominados, respectivamente, sinais de *set* (S) e *reset* (R) assíncronos, visto que os mesmos independem do estado do sinal de habilitação 'E' para atuarem nos sinais de saída do circuito.

Um exemplo de latch D com sinal *clear* ('CL', ou *reset* assíncrono), ativo em nível lógico alto, é apresentado na Figura 6.9. Neste exemplo, o sinal *clear* na entrada da porta NOR garante o valor '0' na saída 'Q' quando tiver o valor '1'. O sinal *clear* complementado colocado na entrada da porta AND garante que esta mesma porta não terá valor de saída igual a '1', o que colocaria em '$\overline{Q}$' também o valor '0'. Isso ocorreria caso ambas as entradas 'E' e 'D' apresentassem o valor '1' durante a ativação do sinal *clear*.



(a) símbolo (b) circuito lógico

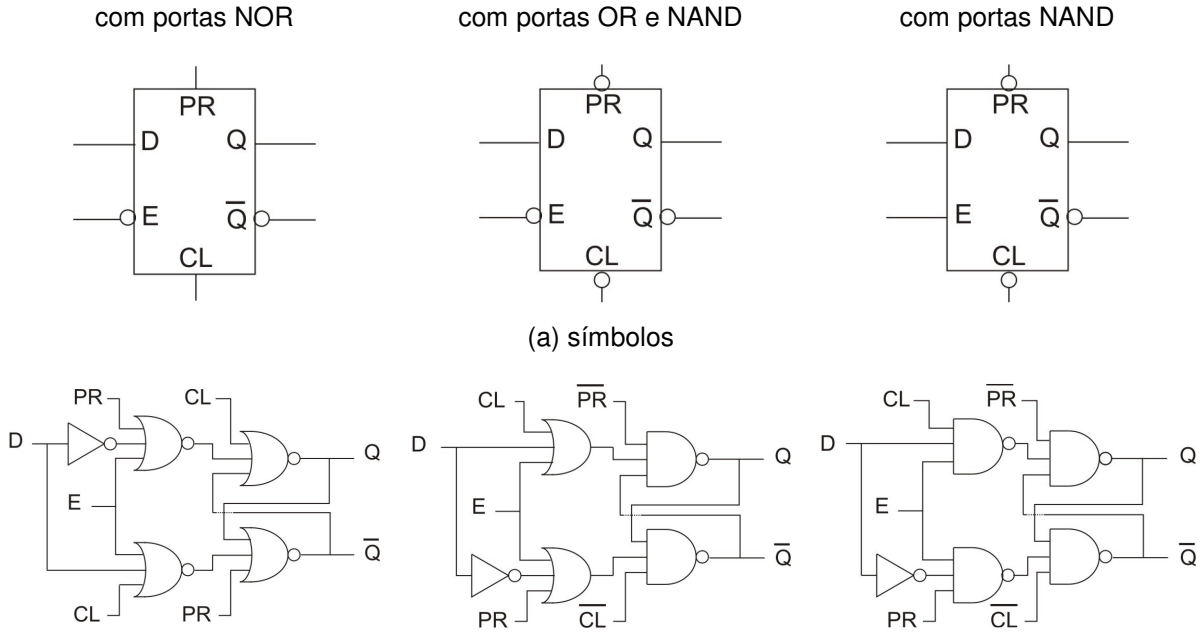
Figura 6.9 – Latch D com sinal de *clear*.

Para acrescentar um sinal *preset* ('PR'), que inicializa a saída 'Q' com valor '1' e a saída '$\overline{Q}$' com valor '0', basta conectar este sinal na porta NOR que gera o sinal '$\overline{Q}$' e o seu complementar na porta AND onde está ligado o sinal complementar de 'D'. Este sinal *preset* também seria ativo em nível alto ou '1'. Para construir um latch D com sinais *clear* e *preset* ativos em '0' é suficiente inverter os sinais descritos acima.

Estruturas alternativas de latch D com sinais *clear* ('CL') e *preset* ('PR') são vistas na Figura 6.10. Pode ser encontrado também na literatura o sinal de *clear* representado pela letra R em alusão ao termo *reset*, e o sinal *preset* indicado pela letra S referente ao termo *set*.

A diferença dos sinais 'PR' e 'CL' utilizados para inicializar o latch D com valor de 'Q' igual a '1' e '0', respectivamente, para os sinais 'S' e 'R' utilizados no latch SR é bastante sutil mas importante: os sinais 'PR' e 'CL' no latch D servem em geral apenas para a inicialização da estrutura de memorização, enquanto que os sinais 'S' e 'R' no latch SR são usados para a definição do valor do dado armazenado no latch em regime normal de funcionamento. Conforme já foi discutido, havendo o sinal de habilitação 'E' no latch SR, os sinais 'S' e 'R' só têm atuação caso este sinal de controle 'E' esteja habilitado. No caso dos sinais de inicialização 'PR' e 'CL' presentes no latch D, estes cumprem suas tarefas independentemente do estado do sinal de habilitação 'E'. Assim como no latch SR, os sinais 'PR' e 'CL' nunca devem ser ativados

simultaneamente no latch D, pois não é coerente desejar colocar os valores ‘1’ e ‘0’ ao mesmo tempo na saída ‘Q’.



PR	CL	E	D	Q_i
1	X	X	X	1
0	1	X	X	0
0	0	0	0	0
0	0	0	1	1
0	0	1	X	Q_{i-1}

PR	CL	E	D	Q_i
0	X	X	X	1
1	0	X	X	0
1	1	0	0	0
1	1	0	1	1
1	1	1	X	Q_{i-1}

PR	CL	E	D	Q_i
0	X	X	X	1
1	0	X	X	0
1	1	1	0	0
1	1	1	1	1
1	1	0	X	Q_{i-1}

(c) tabelas-verdade

Figura 6.10 – Latches D com sinais de *preset* (‘PR’) e *clear* (‘CL’).

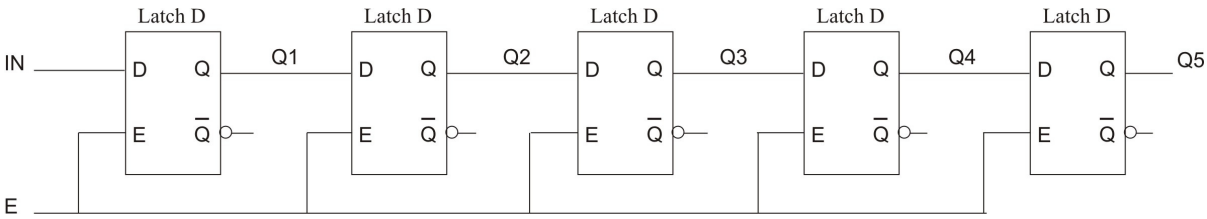
6.3.3 Transparência dos latches

O latch D da Figura 6.8, enquanto estiver habilitado com E = 1, responde imediatamente a qualquer variação no sinal de dados ‘D’ (desprezados obviamente os atrasos de propagação das portas que o implementam), armazenando o valor de ‘D’ e mostrando o mesmo na saída. Diz-se que, nesta situação, o latch está ‘aberto’ e a saída segue o valor da entrada. Por outro lado, o latch está ‘fechado’ quando E = 0; neste caso, variações no sinal ‘D’ não são sentidas. Conclui-se que a função de armazenamento do latch é sensível ao nível do sinal de controle: o latch está aberto e copia o valor de ‘D’ quando o nível do sinal de habilitação é ‘1’ e está fechado quando o nível do mesmo é ‘0’. Simplificadamente, afirma-se que o latch é ‘sensível ao nível’ do sinal de controle.

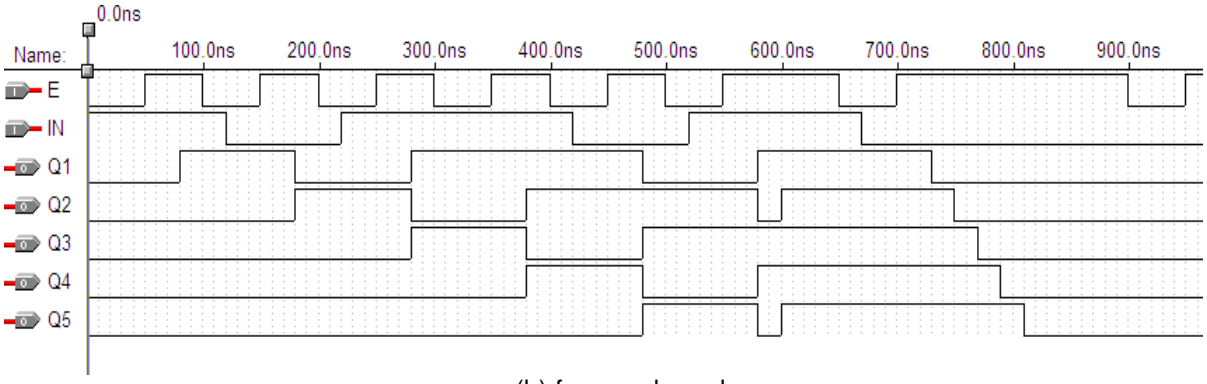
Outra forma de se interpretar o funcionamento do latch é imaginar que o mesmo, enquanto estiver habilitado, se comporta como um ‘fio’, pois qualquer variação no valor de ‘D’ se propaga automaticamente para a saída ‘Q’. Diz-se, então, que o latch é ‘transparente’ enquanto está habilitado. Quando o sinal de habilitação é desativado, o latch armazena e retém o último valor presente na entrada ‘D’.

A transparência dos latches pode causar certos problemas quando deseja-se apenas atualizar o valor memorizado com o valor disponível neste momento na entrada ‘D’. Caso o latch seja mantido habilitado, alterações na entrada ‘D’ terão influência no valor memorizado e nas saídas ‘Q’ e ‘$\overline{Q}$’.

Um exemplo típico de tal dificuldade é a construção de um registrador de deslocamento (*shift register*, em inglês) utilizando-se latches D. O princípio de funcionamento do registrador de deslocamento é que um vetor binário seja deslocado serialmente através de uma cadeia de elementos de memória a cada novo comando de habilitação, como ilustrado na Figura 6.11. O que se espera com esta configuração de circuito é que a cada pulso de habilitação (E = 1) o valor armazenado em cada latch seja deslocado para o latch à sua direita. Nas formas de onda da Figura 6.11(b), supõe-se que o atraso de propagação entre a entrada ‘D’ e a saída ‘Q’ de cada latch é de cerca de 30ns.



(a) circuito lógico



(b) formas de onda

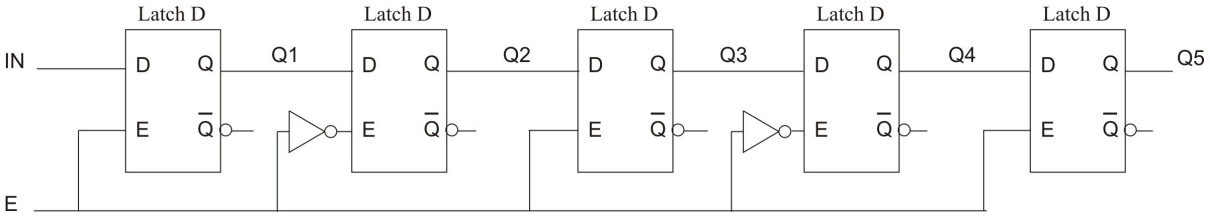
Figura 6.11 – Registrador de deslocamento com latches D controlados pelo mesmo sinal de habilitação.

Porém, uma vez atualizada a saída deste latch à direita e estando o latch seguinte habilitado, tal sinal continua a se propagar atualizando os demais estágios com o mesmo valor. O correto funcionamento do registrador de deslocamento só seria possível se o sinal de habilitação fosse mantido ativado apenas durante o tempo necessário para atualização da saída de cada estágio, o que corresponde ao tempo de propagação do sinal ‘D’ através do latch devido aos atrasos de resposta das portas lógicas que o compõem, ou seja, um valor pouco maior do que os 30ns de atraso sugeridos na Figura 6.11(b). Nesta figura adotou-se um tempo de ativação de 50ns para o sinal de habilitação ‘E’. Este tempo de habilitação é muito difícil de ser controlado, especialmente em circuitos que operam em altas freqüências, pois se for mais curto que o tempo ótimo talvez a atualização do sinal de saída não seja garantida. Se, por outro lado, ele for um pouco mais longo que o tempo ótimo, o risco deste sinal influenciar o estágio seguinte é grande, como se vê entre os tempos de 700ns e 900ns, onde o valor ‘0’ da entrada ‘IN’ se propagou para a saída de todos os latches. As características temporais dos latches serão discutidas em detalhe mais adiante neste capítulo.

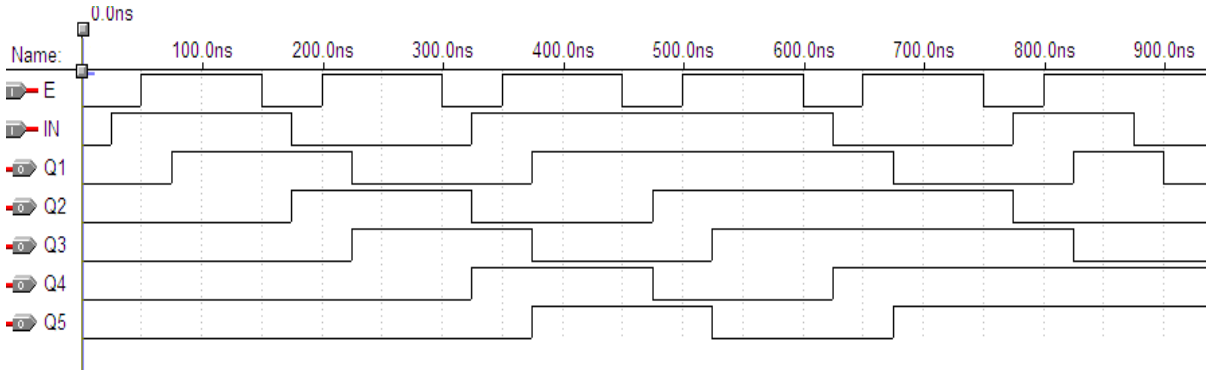
6.4 Registradores sensíveis à borda (flip-flops)

6.4.1 Flip-flop D

O registrador de deslocamento da seção anterior pode ser construído utilizando-se, ao invés de um, dois latches D em cada estágio, com seus sinais de habilitação complementados, conforme ilustrado na Figura 6.12. Desta forma quando um latch está habilitado os latches vizinhos estão desabilitados.



(a) circuito lógico



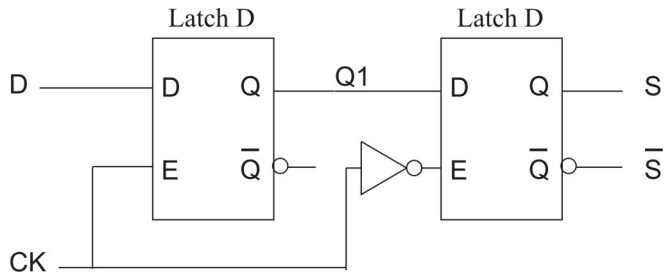
(b) formas de onda

Figura 6.12 – Registrador de deslocamento usando latches D controlados por sinais de habilitação complementares ou alternados.

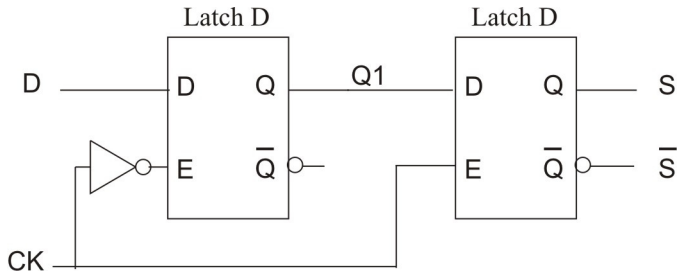
Se for analisado cada par de latches D, com sinais de habilitação complementares, como na Figura 6.13, percebe-se que o sinal na entrada de cada par só é atualizado na saída do mesmo quando o sinal de habilitação apresentar uma certa transição lógica, mudando do valor ‘0’ para o valor ‘1’, ou no caso contrário de ‘1’ para ‘0’. Esta é a definição do bloco lógico flip-flop tipo D, também conhecido por latch D sensível à borda, e não ao nível como tem sido tratado o latch D individual.

No flip-flop da Figura 6.13(a), o primeiro latch registra em sua saída ‘Q1’ o valor da entrada ‘D’ quando CK = 1, enquanto o segundo latch mostra este valor na saída ‘S’ do flip-flop quando CK = 0. Ignorando-se a construção interna do flip-flop, seu comportamento externo revela que a saída ‘S’ só mostra o valor da entrada ‘D’ quando há a transição de ‘1’ para ‘0’ no valor do ‘CK’.

Nos flip-flops utiliza-se geralmente o termo *clock* ‘CK’ (ou ‘relógio’) para indicar o sinal de habilitação. A construção do flip-flop D a partir de dois latches D, com sinais de habilitação complementares, é conhecida como configuração ‘mestre-escravo’. O primeiro latch age como o elemento ‘mestre’, registrando o sinal disponibilizado na entrada, e o segundo latch representa o ‘escravo’, atualizando no momento certo o valor da saída.



(a) sensível à borda de descida



(b) sensível à borda de subida

Figura 6.13 – Par de latches D sensíveis à transição do sinal de controle: flip-flop D.

Pode-se construir o mesmo flip-flop D com latches SR, sem a distinção entre estágio mestre e estágio escravo, conforme apresentado na Figura 6.14. A análise deste flip-flop é melhor compreendida iniciando-se com sinais de entrada que garantam uma certa condição estável nas realimentações existentes. Esta análise é deixada como exercício para o leitor.

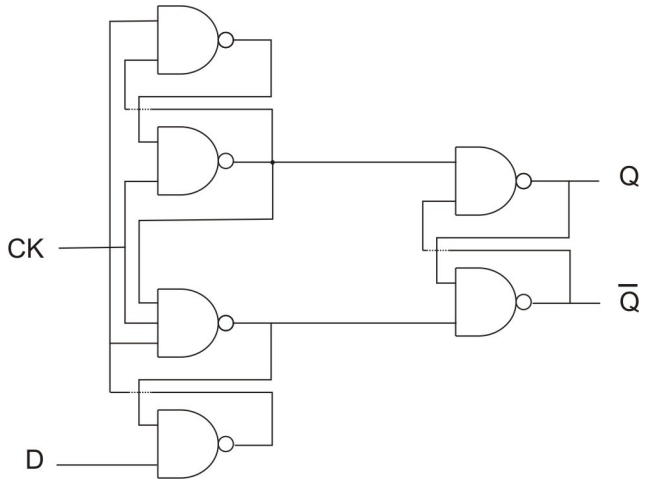


Figura 6.14 – Flip-flop D sem a distinção entre estágio mestre e estágio escravo.

Perceba que para iniciar a análise dos flip-flops, assim como acontece com os latches, não se sabe qual o valor presente na saída. Este passa a ser conhecido à medida que o sinal de habilitação, ou relógio, passa a atualizar a saída com o valor lido da entrada. Este comportamento ocorre em um circuito real quando, ao iniciar o uso do mesmo, os valores nas saídas dos flip-flops são desconhecidos. Neste caso, fala-se em inicializar o circuito, ou mais propriamente tais blocos registradores. Para isso, utiliza-se sinais de *set* (S) e *reset* (R) assíncronos, que independem do valor ou comportamento dos demais sinais de entrada, para colocar os valores ‘1’ e ‘0’ na saída do flip-flop, da mesma forma que foi discutida para os latches anteriormente. No caso dos flip-flops mestre-escravo, construídos a partir de latches D,

para obter-se um flip-flop D com sinais de ‘PR’ e ‘CL’ basta utilizar tais latches D com sinais de *preset* e/ou *clear* em ambos os estágios mestre e escravo. No caso do flip-flop D que não apresenta esta distinção entre estágio mestre e estágio escravo, deve-se avaliar como inicializar as realimentações internas. A Figura 6.15 dá um exemplo.

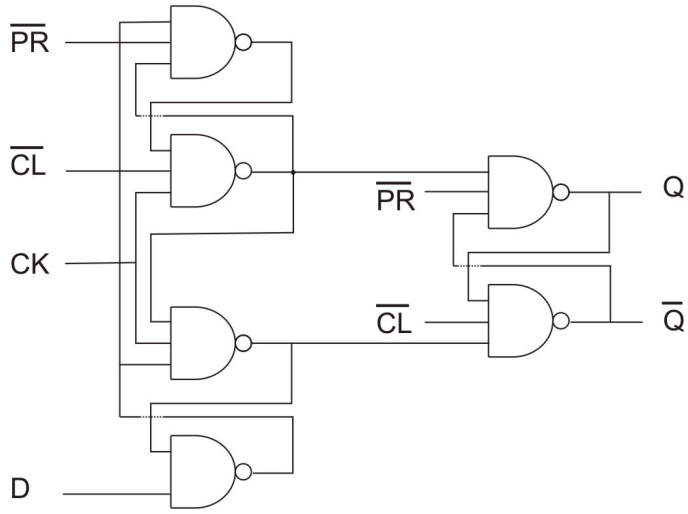


Figura 6.15 – Flip-flop D sem a distinção entre estágio mestre e estágio escravo, com sinais de *preset* (‘PR’) e *clear* (‘CL’) ativos em nível lógico baixo.

6.4.2 Flip-flops SR, JK e T

Diferentemente do flip-flop D, onde o sinal de saída recebe o valor da entrada ‘D’, há flip-flops cujo valor de saída não corresponde necessariamente ao próprio valor da entrada, mas é resultante de uma certa lógica definida através de um ou mais sinais de entrada.

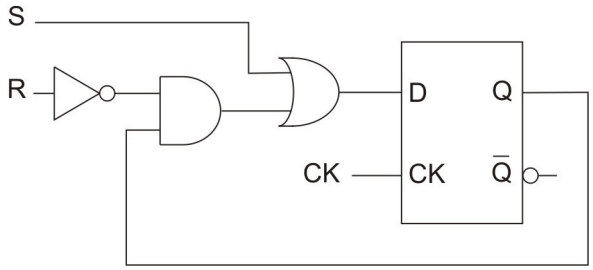
Observe o flip-flop SR cuja tabela-verdade é ilustrada na Figura 6.16(a). Este flip-flop SR tem comportamento semelhante ao latch SR, ou seja, quando a entrada ‘S’ está ativa o sinal de saída recebe o valor lógico ‘1’, caso seja a entrada ‘R’ que esteja ativa então a saída recebe o valor ‘0’. Não havendo nenhuma das duas entradas ‘S’ e ‘R’ ativas então não há alteração no valor do sinal de saída, seja ele qual for. E não tem sentido ativar ambas as entradas pois não é possível colocar ao mesmo tempo na saída os valores ‘0’ e ‘1’. No que o flip-flop SR difere do latch SR, obviamente, é o momento em que saída é atualizada: seja quando o sinal de *enable* está ativo no caso do latch, seja quando ocorre a transição de ativação do sinal de *clock* no caso do flip-flop.

A construção do flip-flop SR pode ser realizada aproveitando-se um flip-flop D ao qual se acrescenta um circuito lógico na entrada ‘D’ comandado pelas variáveis ‘S’ e ‘R’, conforme observado na Figura 6.16(b). A maneira como este flip-flop D é implementado não interfere no comportamento lógico do flip-flop SR.

Facilmente pode se criar muita confusão sobre o comportamento das entradas ‘S’ e ‘R’ e os sinais *preset* e *clear*, principalmente porque esses últimos são algumas vezes tratados na literatura também através dos termos *set* (‘S’) e *reset* (‘R’), respectivamente. A diferença está no sincronismo com o sinal de relógio. No caso das entradas ‘S’ e ‘R’, o sinal de saída só é atualizado no momento de chegada da borda de ativação do sinal de relógio (*clock*), enquanto que os sinais *preset* e *clear* realizam sua função de colocar ‘1’ e ‘0’ na saída, respectivamente, no momento em que foram ativados e independentemente do estado ou comportamento do sinal de relógio.

S	R	Q_i
0	0	Q_{i-1}
0	1	0
1	0	1
1	1	X

(a) tabela-verdade



(b) circuito

Figura 6.16 – Flip-flop SR.

Outra classe de flip-flop é o tipo JK, ou simplesmente flip-flop JK. Este flip-flop apresenta duas entradas de dados 'J' e 'K', além do sinal de relógio ('CK') e eventuais sinais de *preset* ('PR') e *clear* ('CL'). As entradas 'J' e 'K' definem o valor que será colocado na saída a partir da lógica apresentada na Tabela 6.3. O comportamento do flip-flop JK pode ser visto da seguinte maneira:

- a saída só é atualizada na borda de ativação do relógio ('CK');
- caso as entradas 'J' e 'K' sejam iguais a '0', o valor da saída não é alterado;
- caso as entradas 'J' e 'K' sejam iguais a '1', o valor de saída é alterado, independentemente do seu valor (ou seja, ele tem seu valor complementado); e
- caso as entradas 'J' e 'K' sejam diferentes, o valor de saída 'Q' recebe o valor da entrada 'J'.

Tabela 6.3 – Flip-flop JK.

J	K	Q_i
0	0	Q_{i-1}
0	1	0
1	0	1
1	1	$\overline{Q}_{i-1}$

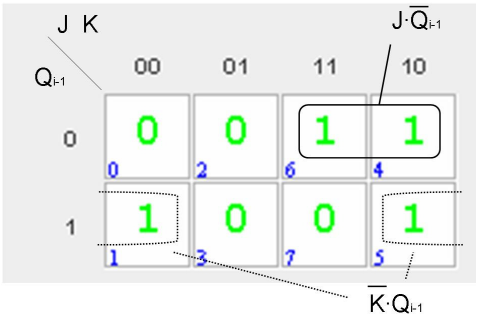
Em relação à construção do flip-flop JK, observa-se que este representa um flip-flop D com uma certa lógica de entrada controlada pelos sinais 'J' e 'K' e pelo sinal de saída atual 'Q'. Esta lógica pode ser construída a partir dos métodos apresentados no Capítulo 3 deste livro. Isto é ilustrado na Figura 6.17, através do uso do método de mapa de Karnaugh.

Uma quarta classe de flip-flop é o tipo T (do inglês *toggle*), mostrado na Figura 6.18. O flip-flop T apresenta uma única entrada de dados 'T' que indica se a saída será mantida inalterada quanto ocorrer a transição de ativação do sinal de relógio 'CK' (quando T = 0), ou se a saída será alterada ou complementada (quando T = 1). Este controle da entrada 'T' é semelhante

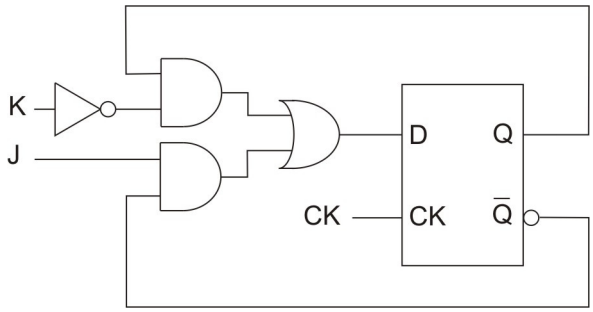
ao comportamento do flip-flop JK quando ambas as entradas ‘J’ e ‘K’ são iguais. Portanto, para a construção do flip-flop T poder-se-ia, por exemplo, alternativamente, tomar uma implementação de um flip-flop JK e conectar as entradas ‘J’ e ‘K’ uma à outra.

J	K	Q_{i-1}	D
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(a) tabela-verdade



(b) mapa de Karnaugh

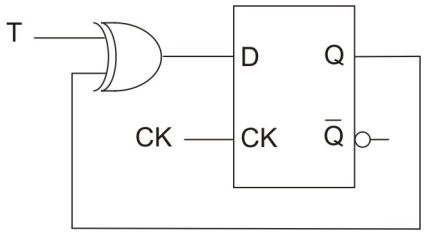


(c) circuito

Figura 6.17 – Implementação do flip-flop JK por mapa de Karnaugh.

T	**Q_i**
0	Q_{i-1}
1	$\overline{Q}_{i-1}$

(a) tabela-verdade



(b) circuito

Figura 6.18 – Flip-flop T.

6.5 Características temporais de latches e flip-flops

Da mesma forma que ocorre para as demais portas lógicas e blocos funcionais, estudados nos capítulos anteriores, os circuitos registradores, tanto latches como flip-flops, também apresentam características temporais como o atraso de propagação do sinal de entrada até o nodo de saída. Isso é naturalmente esperado, pois nenhum circuito lógico ou elétrico permite resposta imediata aos estímulos de entrada.

Lembre-se das definições de 'td_{lh}' e 'td_{hl}' apresentadas no Capítulo 2:

- td_{lh} é o atraso de atualização do sinal de saída, quando este passa do valor lógico '0' (baixo ou *low*) para o valor lógico '1' (alto ou *high*), em relação a um certo sinal de entrada que provocou esta mudança;
- td_{hl} é o atraso de atualização do sinal de saída, quando este passa do valor '1' (*high*) para o valor '0' (*low*), em relação a um certo sinal de entrada que provocou esta mudança.

Portanto, no caso do latch SR sem sinal de habilitação tem-se os atrasos td_{lh} e td_{hl} das saídas 'Q' e '$\overline{Q}$' em relação a cada sinal de entrada 'S' e 'R'. Havendo um sinal de habilitação 'E', os atrasos em função das entradas 'S' e 'R' continuam válidos caso o sinal 'E' esteja habilitado, conforme ilustrado na Figura 6.19 para o sinal de saída '$\overline{Q}$' (neste exemplo mais lento que o sinal Q), entre os tempos 0ns e 20ns. Tem-se ainda o atraso das saídas em relação ao sinal de habilitação 'E' quando sua própria ativação provocar uma alteração lógica nos sinais de saída, conforme ilustrado na Figura 6.19, entre os tempos 20ns e 50ns, também para o sinal de saída '$\overline{Q}$'. Esta mesma análise é válida para o latch D.

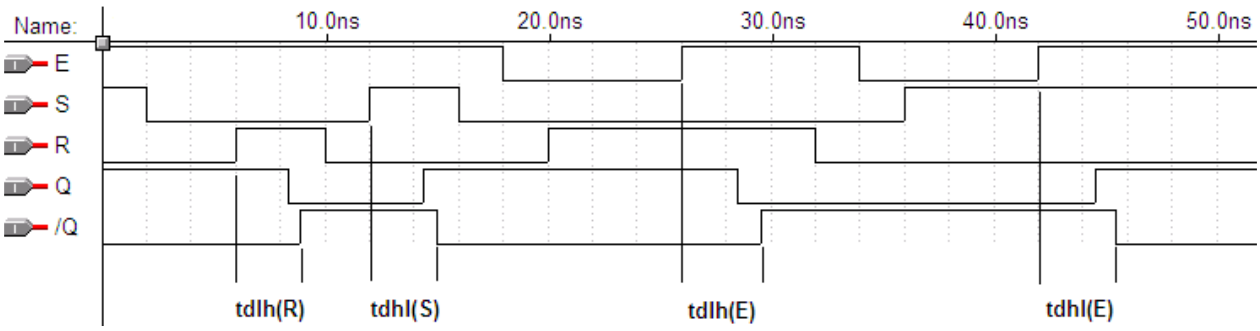


Figura 6.19 – Atrasos do latch SR.

No caso do latch D com sinais *preset* e *clear* (ou respectivamente *set* e *reset* assíncronos), tem-se ainda os atrasos para atualização dos sinais de saída quando tais sinais são ativados, e neste caso independemente do sinal de habilitação 'E'.

Uma observação interessante é que os atrasos relacionados com a saída 'Q' são, muito provavelmente, diferentes dos atrasos relacionados com a saída complementar '$\overline{Q}$'. Para entender este fato basta se observar os diferentes caminhos lógicos para as saídas 'Q' e '$\overline{Q}$' nas estruturas lógicas dos latches, apresentadas anteriormente.

Além dos atrasos de propagação dos sinais de saída, tem-se uma outra característica temporal que é a *largura mínima de pulso* de certos sinais de entrada. No caso do sinal de habilitação 'E', por exemplo, tem-se um tempo mínimo em que este deve ser mantido ativado, suficiente para permitir a passagem do dado de entrada e a atualização do valor da saída. Do contrário, se este período de ativação do sinal 'E' for muito curto, o latch será desabilitado não ocorrendo talvez mudança correta dos sinais de saída.

Este conceito de largura mínima de pulso de sinal de entrada é aplicado também aos sinais *preset* e *clear* de latches e flip-flops. Ou seja, há um período de tempo mínimo em que tais sinais devem permanecer ativados de forma que a sua atuação seja efetivada. A Figura 6.20 ilustra o período mínimo destes sinais no latch D.

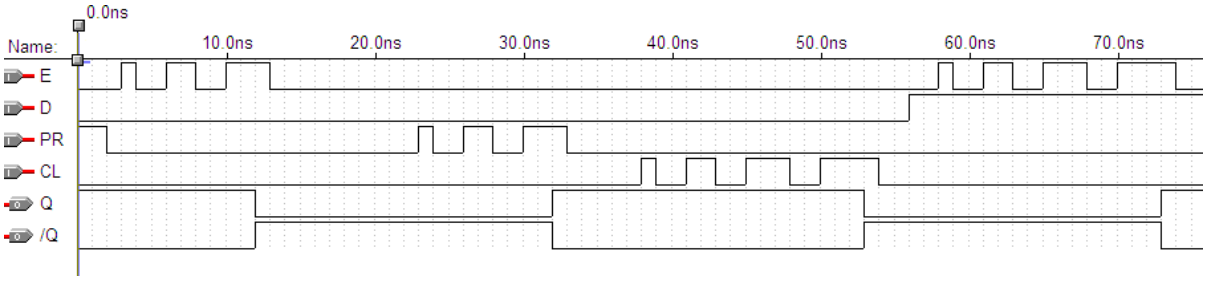


Figura 6.20 – Largura mínima de pulso dos sinais de habilitação ('E'), preset ('PR') e clear ('CL').

Os flip-flops também apresentam as características temporais descritas acima:

- atrasos de propagação td_{lh} e td_{hl} dos sinais de saída 'Q' e '$\overline{Q}$' em relação à borda de ativação do sinal de relógio, responsável pela atualização desses sinais;
- largura mínima de pulso dos sinais de *preset* e *clear*.

Os flip-flops apresentam, porém, uma particularidade em relação aos latches e às portas lógicas combinacionais que são os tempos de *setup* e *hold*. Esses tempos estão relacionados com a definição, estabilidade e manutenção dos sinais de entrada em relação ao momento em que ocorre a transição de ativação do sinal de relógio (veja ilustração apresentada na Figura 6.21):

- tempo de *setup* – período em que os sinais lógicos de entrada (D, J e K, S e R, T) devem estar definidos e se manter inalterados antes da ocorrência da borda de ativação do sinal de relógio, seja esta de subida (relógio transitando de '0' para '1') ou de descida (relógio indo de '1' para '0'), para que os sinais de saída recebam garantidamente o valor correto.
- tempo de *hold* - período em que os sinais lógicos de entrada (D, J e K, S e R, T) devem ser mantidos inalterados após a ocorrência da borda de ativação do sinal de relógio, de forma que a atualização dos sinais de saída seja confirmada.

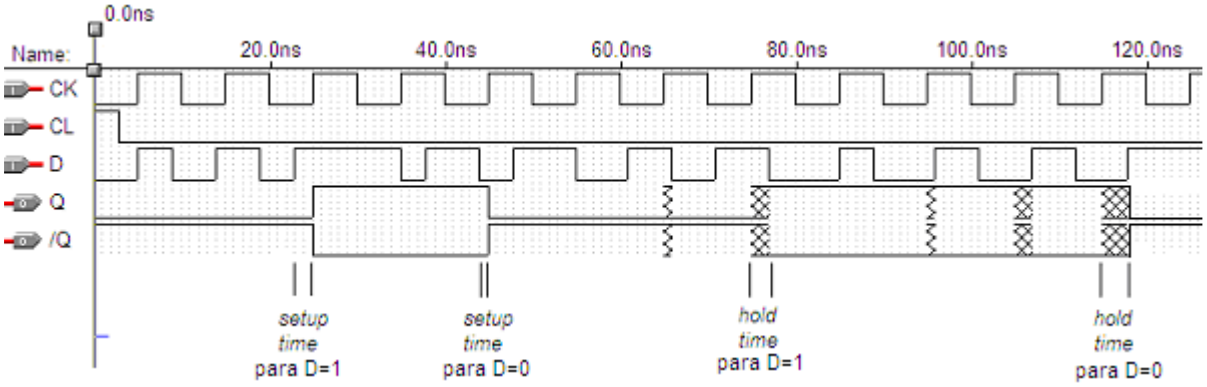
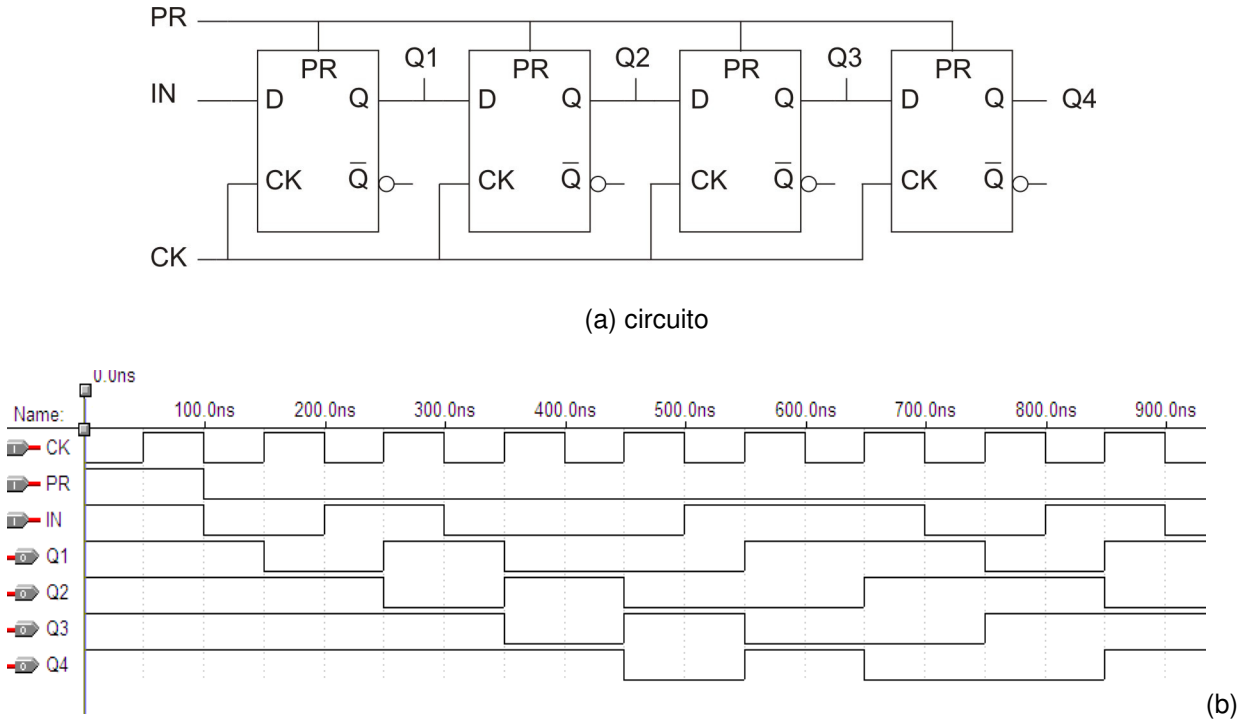


Figura 6.21 – Tempos de *setup* e *hold* em flip-flop D sensível à borda de subida do relógio.

6.6 Registradores de deslocamento

O registrador de deslocamento, ou *shift register* (em inglês), como a própria denominação sugere, tem por objetivo deslocar um vetor binário através de uma cadeia de flip-flops, sem que um bit interfira em outro (anterior ou posterior). Observando a Figura 6.22(a) percebe-se a presença de quatro estágios registradores cujos dados são enviados ao estágio seguinte no momento do sinal de habilitação ou relógio. O primeiro estágio recebe novo dado, enquanto que o último dado é descartado ou perdido, dando lugar ao penúltimo dado da cadeia.



(a) circuito

(b)

comportamento temporal

Figura 6.22 – Registrador de deslocamento.

Este deslocamento deve ser de apenas uma posição a cada comando de habilitação. Conforme discutido anteriormente, a construção de registradores de deslocamento utilizando latches tipo D, como ilustrado na Figura 6.11(a), é muito difícil, pois o sinal de habilitação 'E' deve permanecer no nível de transparência no tempo exato para que o sinal de entrada de cada latch seja propagado para a saída.

A implementação do registrador de deslocamento utilizando flip-flops D permite um bom funcionamento, uma vez que as saídas são atualizadas em um determinado instante bem definido (borda de ativação do sinal de relógio), e não durante um período de tempo como ocorre com os latches. A Figura 6.22(b) ilustra este comportamento, onde supõe-se o uso de flip-flops sensíveis à borda de subida.

Com a utilização de blocos funcionais multiplexadores, é possível construir registradores de deslocamento mais elaborados, conforme ilustrado na Figura 6.23. Nesta arquitetura temos as seguintes configurações de funcionamento, segundo os sinais de controle dos multiplexadores 'S_1' e 'S_0':

- sendo S_1=S_0=0, no momento da borda de ativação do sinal de relógio as entradas $D_{[3..0]}$ são armazenadas nos flip-flops, ficando disponíveis nas saídas $Q_{[3..0]}$;

- sendo S_1=0 e S_0=1, tem-se a mesma configuração do registrador de deslocamento da Figura 6.22(a);
- sendo S_1=1 e S_0=0, tem-se uma configuração semelhante à anterior, porém com deslocamento do vetor binário no sentido contrário;
- sendo S_1=S_0=1, tem-se uma configuração semelhante à segunda opção, porém a entrada do primeiro estágio é fornecida pelo último estágio, configurando uma operação de rotação do vetor binário, mais conhecida como *rotate* (em inglês).

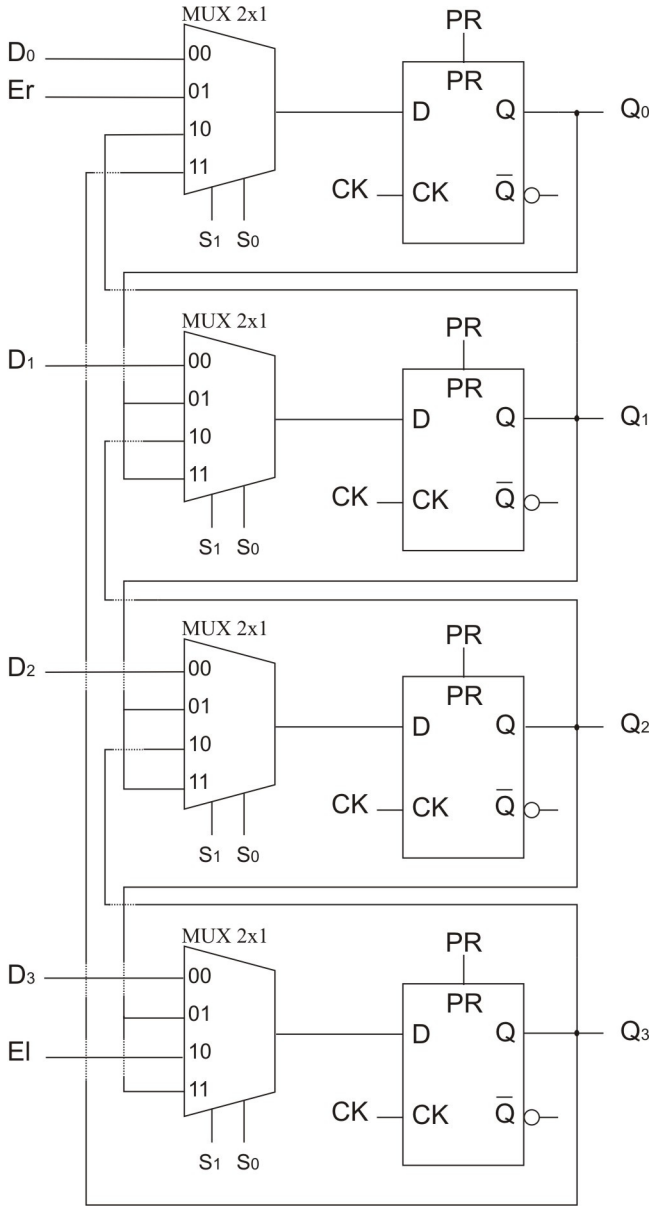


Figura 6.23 - Registrador de deslocamento complexo.

Este registrador mais elaborado, que serve como registrador para armazenagem de dados em paralelo (S_1=S_0=0), deslocamento do vetor binário à esquerda (S_1=0 e S_0=1), deslocamento do vetor binário à direita (S_1=1 e S_0=0), e rotação à esquerda (S_1=S_0=1), pode também ser usado para comunicação série-paralelo e paralelo-série:

- Comunicação série-paralelo – com os sinais S_1=1 e S_0=0 o vetor binário entra no registrador de forma serial, através do ponto de entrada 'D_0', e completa seu carregamento após quatro pulsos de sinal de relógio, ficando disponível de forma paralela através das saídas $Q_{[3..0]}$;
- Comunicação paralelo-série – inicialmente os sinais de controle 'S_0' e 'S_1' estão em '0', ou seja, há o carregamento paralelo dos flip-flops em um pulso de relógio. A seguir faz-se S_0=1 e a saída 'Q_3' fornece serialmente o vetor binário armazenado, em quatro pulsos de relógio.

6.7 Divisor de frequência e contadores

Uma outra aplicação intuitiva dos flip-flops são os contadores. Observe inicialmente a configuração ilustrada na Figura 6.24 e imagine um flip-flop sensível à borda positiva do sinal de relógio. Nesta configuração, a cada borda positiva do relógio tem-se a mudança dos valores dos sinais de saída 'Q' e '$\overline{Q}$'. Efeito semelhante é obtido com o flip-flop T, estando a entrada 'T' em '1', ou com o flip-flop JK, tendo-se J=K=1. Como resultado deste comportamento observa-se que o sinal de saída 'Q' (ou '$\overline{Q}$') altera seu valor em um período duas vezes maior que o sinal de relógio 'CK', ou possui a metade da freqüência de 'CK'. Construiu-se um divisor de freqüência por 2. Se o sinal de saída 'Q' for utilizado como sinal de relógio em um outro flip-flop com configuração semelhante, obtém-se mais uma divisão por 2 da freqüência deste sinal, ou seja, um sinal com uma freqüência 4 vezes menor que a freqüência do sinal de relógio original (do estágio anterior). Esta configuração em cascata pode ser ampliada para obter sinais com freqüências menores, mas sempre divididas por 2, conforme ilustrado na Figura 6.25.

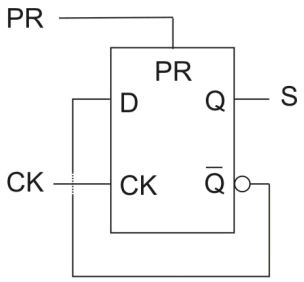


Figura 6.24 – Célula básica de um divisor de freqüência.

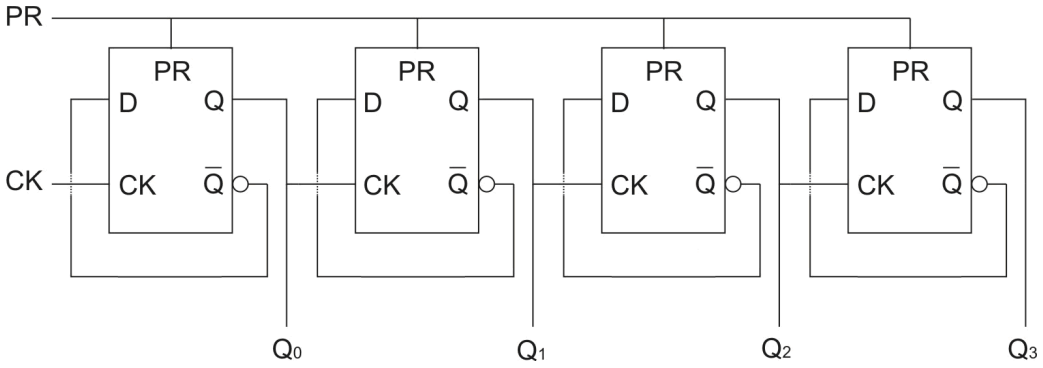


Figura 6.25 – Divisor de freqüência.

A configuração da Figura 6.25 pode ser vista também como um contador progressivo ou regressivo, como ilustrado na Figura 6.26. Iniciando todos os estados com valor de saída '1'

(através do uso do sinal de *preset*), após o primeiro pulso de relógio o sinal 'Q_0' vai para '0' e os demais $Q_{[3..1]}$ permanecem em '1'. O segundo estágio terá sua saída 'Q_1' alterada para '0' quando a saída '$\overline{Q}_0$' for para '1', no próximo pulso de relógio. Se forem observados os sinais de saída negados $\overline{Q}_{[3..0]}$, percebe-se a mudança deste vetor partindo do valor '0000' (decimal 0), e passando pelos valores '0001', '0010', '0011', '0100', etc., até chegar ao valor '1111' (decimal 15), retornando ao valor '0000' no pulso de relógio seguinte. Obteve-se um contador progressivo (em inglês, *up counter* ou, de forma mais usual, contador *up*). Se for observado o vetor de saída $Q_{[3..0]}$ percebe-se que foi obtida uma contagem regressiva, seguindo a freqüência '1111', '1110', '1101', '1100', '1011',..., '0000', '1111', '1110',.... Ou seja, com a mesma configuração tem-se um contador regressivo (em inglês, *down counter* ou, de forma mais usual, contador *down*).

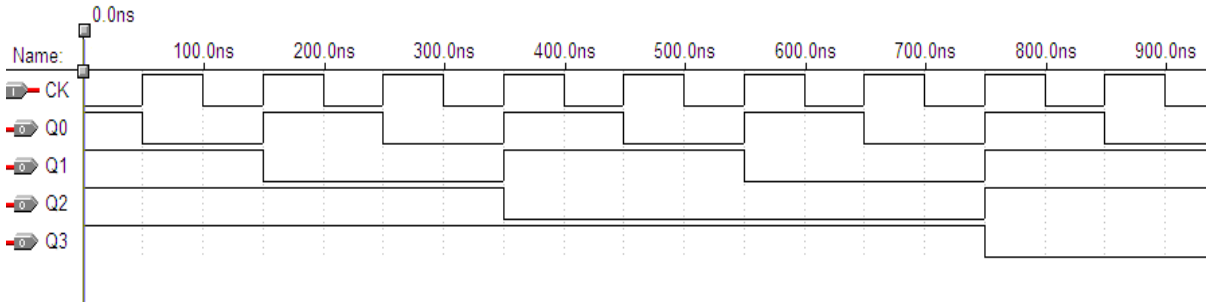


Figura 6.26 – Comportamento do contador *ripple*.

Este tipo de contador é conhecido como contador assíncrono, ou *ripple counter*, uma vez que o sinal de relógio externo controla diretamente apenas o primeiro estágio, sendo os demais comandados pelos estágios precedentes. Isso causa uma defasagem dos sinais de saída dos estágios causada pelos atrasos de propagação ou atualização de cada sinal de saída anterior.

Outros contadores, conhecidos por *contadores síncronos*, pois todos os flip-flops são controlados pelo mesmo sinal de relógio, serão vistos no capítulo seguinte onde será estudada a construção de máquinas de estados finitos.