Sistemas Digitais

Aula 09 - Circuitos Sequenciais (Parte II)

Apresentação

Na aula anterior, você estudou os circuitos sequenciais, que são circuitos com memória, uma vez que eles conseguem memorizar os valores existentes no instante anterior. Você descobriu que esses circuitos lógicos sequenciais são formados por blocos básicos, que chamamos *flip-flop*.

Nesta aula, estudaremos como construir **registradores** e, também, **contadores** com esses *flip-flops*, já que eles conseguem memorizar valores. Assim, veremos que esses *flip-flops* podem ser utilizados em uma ampla variedade de aplicações como: contagem, armazenamento binário de dados, deslocamento e transferência de dados.

Objetivos

Ao final desta aula, você será capaz de:

- Entender o uso de *flip-flops* em circuitos de sincronização.
- Compreender as principais diferenças entre as transferências serial e paralela de dados.
- Saber conectar registradores de deslocamento para formar circuitos de transferência de dados.
- Saber usar *flip-flops* em circuitos divisores de frequência e contadores.

Registradores

Na aula anterior, você estudou os *flip-flops*, que são os blocos básicos dos circuitos lógicos sequenciais, responsáveis pelo armazenamento dos *bits* para que fiquem disponíveis em instantes posteriores, ou seja, correspondem à memória do circuito.

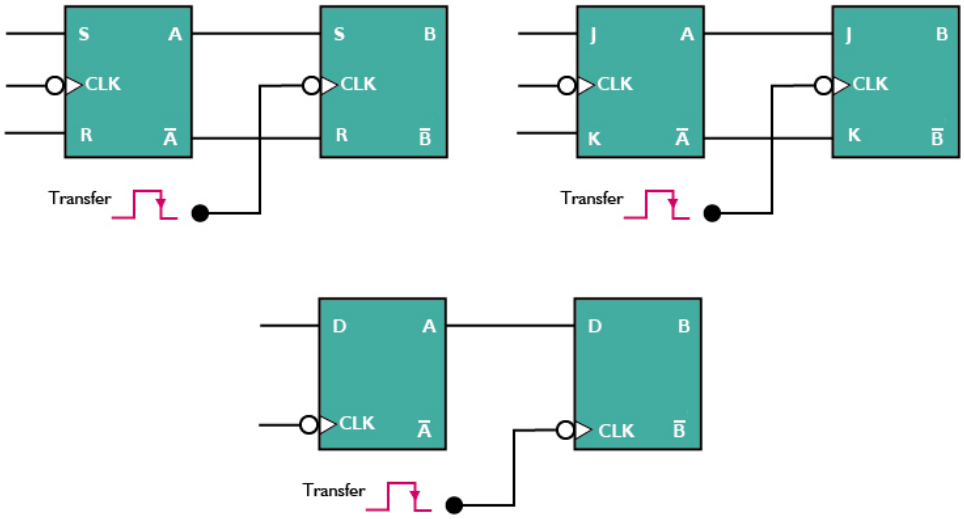
Esses elementos de memória, os *flip-flops*, podem ser utilizados em uma ampla variedade de aplicações como: contagem, armazenamento binário de dados, deslocamento e transferência de dados.

O uso mais comum de *flip-flops* é o armazenamento de dados, que podem representar valores numéricos (números binários ou outros codificados em binário). Dados são, geralmente, armazenados em grupos de *flip-flops* denominados **registradores**.

A operação mais comum realizada sobre os dados armazenados em *flip-flops* ou registradores é a operação de transferência de dados. Essa operação envolve a transferência de dados de um registrador (*flip-flop*) para outro.

Na **Figura 1**, você pode ver como a transferência de dados pode ser implementada entre dois registradores usando *flip-flops* SR, JK e D. Em cada caso, o valor lógico atual armazenado em um *flip-flop* A é transferido para um *flip-flop* B na borda de descida do pulso TRANSFER. Dessa forma, após a borda de descida, a saída B terá o mesmo valor que a saída A.

Figura 01 - Operação de transferência síncrona de dados realizada por diversos tipos de flip-flops com clock.

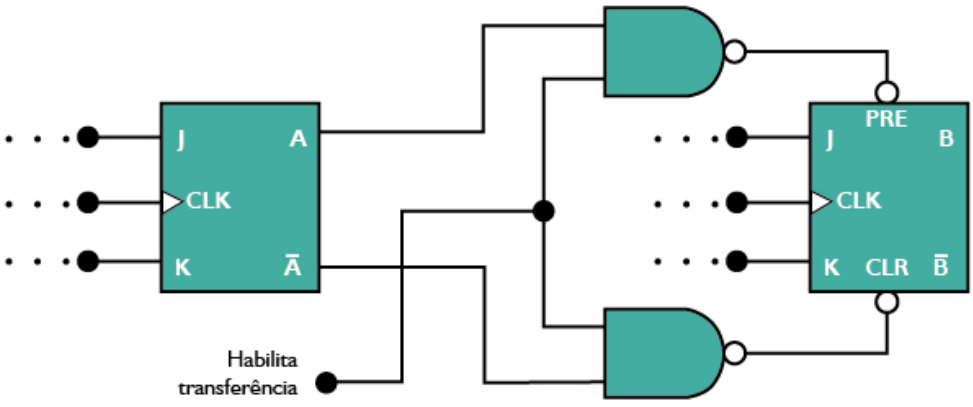


Fonte: Tocci, Widmer, Moss (2007).

As operações de transferência da **Figura 1** são **transferências síncronas**, pois as entradas de controle síncronas e a entrada de *clock* (CLK) foram usadas para realizar a transferência.

Operações de transferência também podem ser obtidas usando as entradas assíncronas de um *flip-flop*. A **Figura 2** mostra como **transferências assíncronas** podem ser implementadas usando as entradas assíncronas PRESET e CLEAR de um *flip-flop* qualquer.

Figura 02 - Operação de transferência assíncrona de dados.



Fonte: Tocci, Widmer, Moss (2007).

Na **Figura 2**, as entradas assíncronas preset (PRE) e clear (CLR) são ativadas em '0'. Quando a entrada “habilita transferência” é colocada em '1', uma das saídas das portas NAND vai para '0', dependendo do estado das saídas A e $\overline{A}$. Esse nível '0' vai **ativar** (set) ou **desativar** (reset) o *flip-flop* B para o mesmo estado do *flip-flop* A.

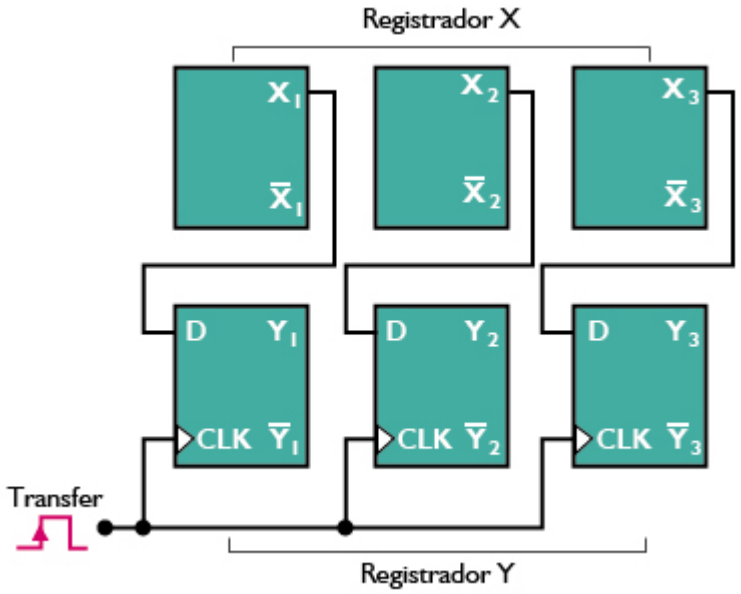
Transferência Paralela de Dados

Vimos como é feita a transferência de dados de maneira síncrona e de maneira assíncrona. Agora, vamos ver a diferença entre a transferência paralela e a transferência serial de dados.

Na **Figura 3**, vemos a transferência de dados de um registrador para outro usando *flip-flops* D. O registrador X é formado dos *flip-flops* X_2, X_1 e X_0 e o registrador Y pelos *flip-flops* Y_2, Y_1 e Y_0. Quando ocorre a borda de subida do pulso TRANSFER, o valor de X_2 é transferido para Y_2, o de X_1 para Y_1 e o de X_0 para Y_0.

Essa transferência síncrona do conteúdo do registrador X para o registrador Y é uma **transferência paralela**, pois os conteúdos de X_2, X_1 e X_0 são transferidos, simultaneamente, para Y_2, Y_1 e Y_0. É importante notar que o conteúdo do registrador X não é alterado.

Figura 03 - Transferência paralela do conteúdo do registrador X para o registrador Y.



Fonte: Tocci, Widmer e Moss (2007)

Transferência Serial de Dados

Vimos que na transferência paralela todos os dados são transferidos, simultaneamente, em um único pulso de *clock*.

Na **transferência serial**, quando chega o primeiro pulso de *clock*, os dados são transferidos de um *flip-flop* para o seguinte e somente quando chega um novo pulso de *clock* é que o dado vai ser enviado desse segundo *flip-flop* para um terceiro, e assim por diante.

Esse arranjo é chamado de registrador de deslocamento.

Um **registrador de deslocamento** é um grupo de *flip-flops* organizados, de modo que os números binários armazenados nos *flip-flops* sejam deslocados de um *flip-flop* para o seguinte a cada pulso de *clock*>.

Você já deve ter visto registradores de deslocamento em dispositivos como uma calculadora, em que os dígitos mostrados no *display* são deslocados cada vez que você tecla um novo dígito. Essa operação é equivalente ao que acontece em um registrador de deslocamento

A **Figura 4(a)** mostra uma forma de usar *flip-flops* JK para construir um registrador de deslocamento de 4 *bits*. Observe que os *flip-flops* estão conectados de maneira que o valor da saída X_3 é transferido para X_2, o valor de X_2 para X_1 e o de X_1 para X_0.

Dessa forma, quando ocorre uma borda de descida no pulso de deslocamento, cada *flip-flop* recebe o valor armazenado previamente no *flip-flop* à esquerda, como você pode ver nas formas de onda da **Figura 4(b)**.

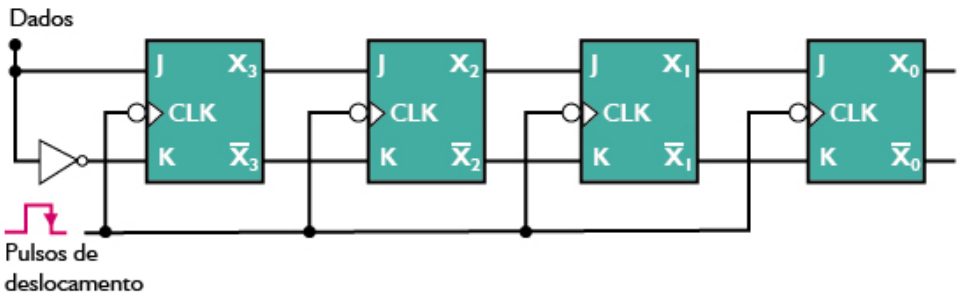
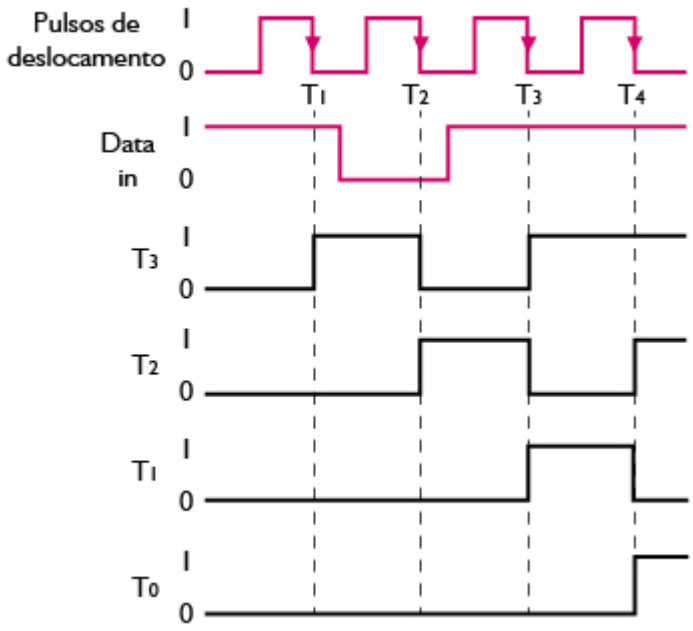


Figura 04 - Registrador de deslocamento de quatro bits.



Fonte: Tocci, Widmer, Moss (2007).

Na **Figura 4**, você vê o deslocamento de *bits* (transferência serial) dentro de um registrador de 4 *bits*. A **Figura 3**, por sua vez, mostrou a transferência paralela de 3 *bits* entre dois registradores.

Será que é possível fazer transferência entre dois registradores de 3 *bits* de forma serial?

A **Figura 5** responde a essa questão. Na ilustração, você pode ver dois registradores de 3 *bits* que usam *flip-flops* D e estão conectados de modo que o conteúdo do registrador X é transferido de forma serial para o registrador Y.

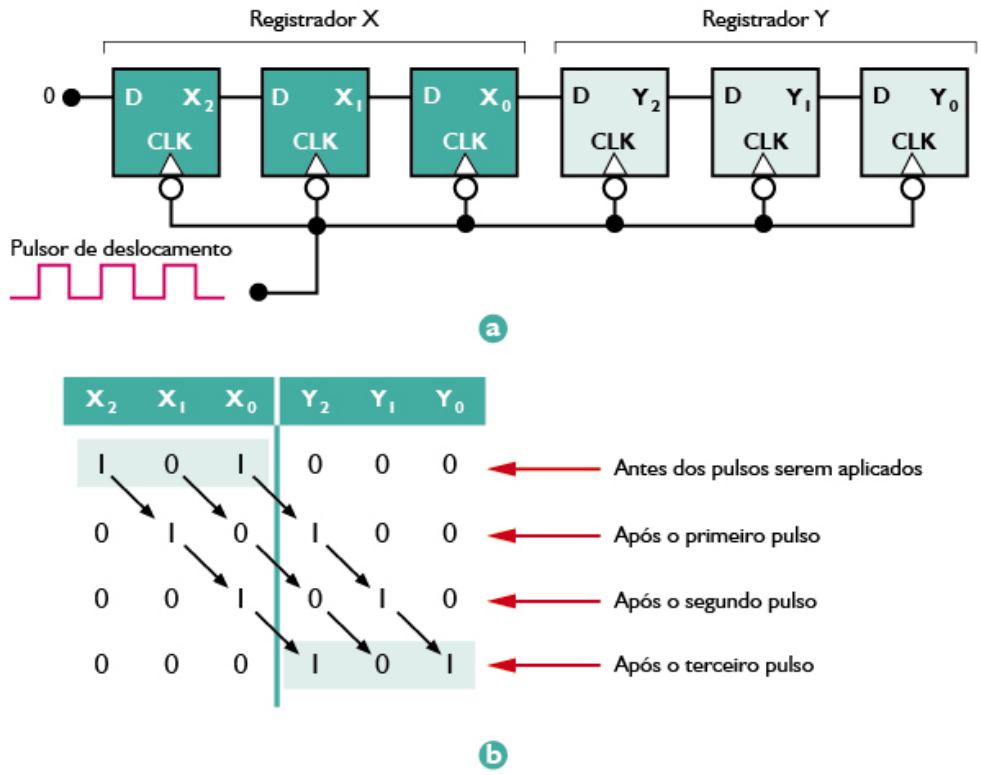
Quando os pulsos de deslocamento são aplicados, ocorre a seguinte transferência:

$$X_2 \rightarrow X_1 \rightarrow X_0 \rightarrow Y_2 \rightarrow Y_1 \rightarrow Y_0$$

Observe que a transferência completa dos 3 *bits* de X para Y só ocorre após 3 pulsos de deslocamento, ou seja, a cada pulso é transferido um *bit*.

Os *bits* dos *flip-flops* da **Figura 5** são deslocados da esquerda para a direita, mas poderia ser no sentido inverso. A escolha do sentido do deslocamento é uma questão da natureza da aplicação e decisão do projetista, mas não uma questão de vantagem em uma ou outra direção.

Figura 05 - Transferência serial de dados de um registrador X para um registrador Y.



Fonte: Tocci, Widmer, Moss (2007).

Atividade 01

---

1. Compare, agora, o circuito da **Figura 3** com da **Figura 5**. O que você percebe?

    a. Qual das transferências é mais rápida, ou seja, precisa de menos pulsos de *clock* para enviar todos os dados?

    b. Qual desses circuitos tem menos conexões?

Transferência Paralela e Serial

Na **transferência paralela (Figura 3)**, todos os dados são transferidos, simultaneamente, em um único pulso de transferência, não importando a quantidade de *bits* a serem transferidos (nesse caso, eram 3 *bits*).

Na **transferência serial (Figura 5)**, são necessários N pulsos de *clock* para conseguir a transferência completa de N *bits* (3 pulsos para 3 *bits*, 4 pulsos para 4 *bits* etc.).

Assim, transferências paralelas são mais rápidas que as seriais. Por outro lado, para fazer transferências paralelas, são necessárias mais conexões entre o registrador transmissor (X) e o registrador receptor (Y) que nas transferências seriais, onde apenas o último *flip-flop* do registrador X precisa ser conectado ao registrador Y.

Então, qual a melhor alternativa: transferência paralela ou serial? Qual delas escolher?

Essa escolha vai depender das necessidades da aplicação. Em muitos casos, usa-se uma combinação dos dois tipos de transferências, obtendo a vantagem da velocidade da paralela e a economia e simplicidade da serial.

Antes de estudarmos outros usos para os *flip-flops*, vamos ver se você compreendeu bem o uso deles em circuitos registradores. Responda a atividade a seguir

Atividade 02

---

1. Responda Verdadeiro ou Falso:

    ( ) A transferência assíncrona de dados usa a entrada de *clock* (CLK).

( ) A transferência paralela é o método mais rápido na transferência de dados de um registrador para outro.

( ) Na transferência paralela, os dados da origem são perdidos.

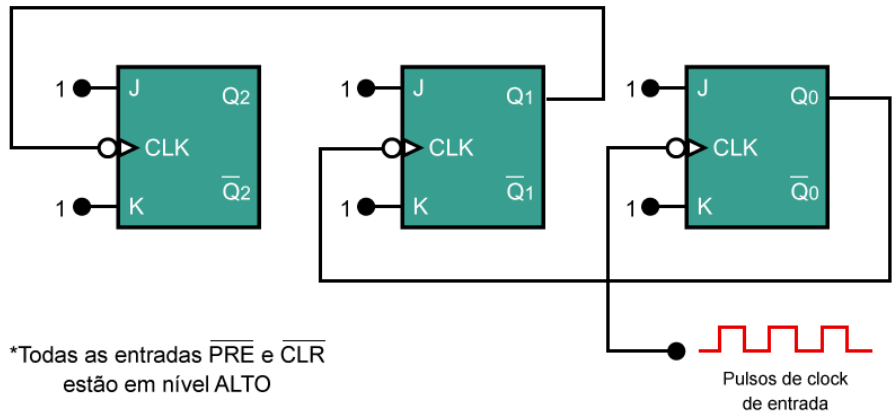
2. Qual é a maior vantagem da transferência serial sobre a paralela?
3. Em qual das formas de transferência de dados a fonte de dados (origem) não perde os dados?

Contadores

Além de serem usados para construir circuitos registradores, outro uso bastante comum para os *flip-flop*s são os contadores.

Na **Figura 6(a)**, cada *flip-flop* tem as entradas J e K em '1' para que comutem sempre que houver uma borda de descida do *clock* (CLK). O pulso de *clock* é aplicado somente na entrada CLK do 1º *flip-flop* (Q0). Os demais recebem, no CLK, a saída do *flip-flop* anterior, ou seja, a saída de Q0 é conectada no CLK de Q1, e a saída de Q1 é conectada no CLK de Q2. As formas de onda da **Figura 6**(b) mostram como os *flip-flop*s mudam de estado, conforme os pulsos são aplicados.

Figura 06 - *flip-flop*s JK conectados formando contador binário de 3 bits (módulo 8).



Fonte: Tocci, Widmer, Moss (2007).

Observando a **Figura 6(b)**, vemos que:

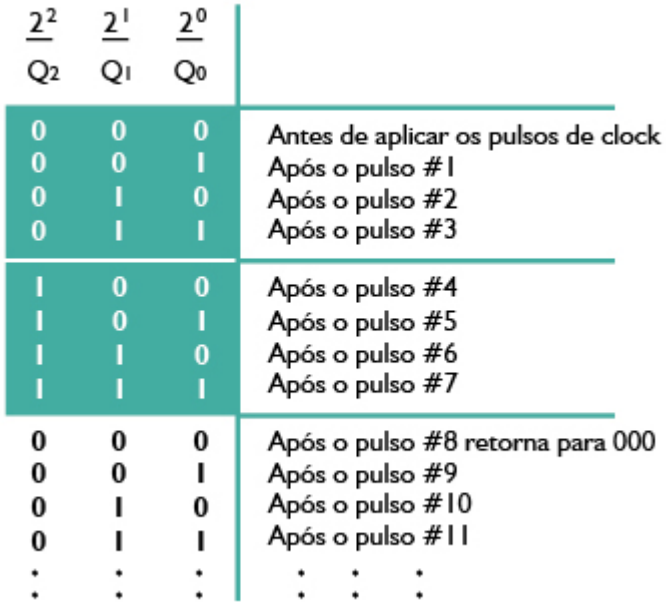
1. Q_0 comuta na borda de descida de cada pulso do *clock*. Assim, a forma de onda da saída Q_0 tem uma frequência que é, exatamente, a metade da frequência dos pulsos do *clock*.
2. Q_1 comuta cada vez que ocorre uma “borda de descida” na saída de Q_0, ou seja, quando a saída de Q_0 vai de '1' para '0'. Dessa maneira, a forma de onda de Q_1 tem uma frequência, exatamente, igual à metade da frequência da saída de Q_0, que representa um quarto da frequência do sinal do *clock*.
3. Q_2 comuta de estado cada vez que a saída Q_1 vai de '1' para '0', o que implica que a forma de onda Q_2 tem a metade da frequência de Q_1, que representa, por sua vez, um oitavo da frequência do *clock*.

Esse arranjo da **Figura 6** faz com que cada *flip-flop* divida a frequência do sinal de sua entrada por 2. Se fosse acrescentado um quarto *flip-flop*, ele teria uma frequência igual a 1/16 da frequência do *clock*. Se fosse acrescentado um quinto *flip-flop*, a frequência seria 1/32 da frequência do *clock* e assim por diante.

Em resumo, com N *flip-flop*s, a saída do último *flip-flop* seria 1 / 2N da frequência de entrada do 1º *flip-flop*. Esse tipo de circuito é conhecido como **divisor de frequência.**

Você deve ter percebido também que, além de funcionar como um divisor de frequência, o circuito da **figura 6** funciona como um **contador binário**. Se não ficou tão evidente, observe a tabela de estados dos *flip-flops*vista na **figura 7**.

Figura 07 - Tabela com os estados dos *flip-flop*s mostrando sequência de contagem binária.



2^2 Q_2	2^1 Q_1	2^0 Q_0	
0	0	0	Antes de aplicar os pulsos de clock
0	0	1	Após o pulso #1
0	1	0	Após o pulso #2
0	1	1	Após o pulso #3
1	0	0	Após o pulso #4
1	0	1	Após o pulso #5
1	1	0	Após o pulso #6
1	1	1	Após o pulso #7
0	0	0	Após o pulso #8 retorna para 000
0	0	1	Após o pulso #9
0	1	0	Após o pulso #10
0	1	1	Após o pulso #11
⋮	⋮	⋮	⋮ ⋮ ⋮

Fonte: Tocci, Widmer, Moss (2007).

Analisando a tabela da **figura 7**, vemos que os valores de $Q_2Q_1Q_0$ representam um número binário no qual Q_2 está na posição 2^2, Q_1 na 2^1 e Q_0 na 2^0. Os primeiros oito estados de $Q_2Q_1Q_0$ da tabela devem ser reconhecidos como uma contagem binária sequencial de 000 a 111.

Na primeira borda de descida do *clock*, os *flip-flop*s passam para o estado $001 (Q2 = 0, Q1 = 0$ e $Q0 = 1)$ que representa 001(binário) (equivalente ao decimal 1).

Na segunda borda de descida do *clock*, os *flip-flop*s passam para o estado $010_{(binário)}$ = $2_{(decimal)}$. No pulso de *clock* seguinte, vai para o estado $011_{(binário)} = 3_{(decimal)}$, e assim sucessivamente, até que ocorram sete pulsos de *clock*, quando chegaremos a $111_{(binário)} = 7_{(decimal)}$.

Na oitava borda de descida do *clock*, os *flip-flop*s retornam para estado inicial (000) e a sequência binária se repete para os pulsos de *clock* seguintes.

Esse contador tem $2^3 = 8$ estados diferentes (000 a 111). Dizemos, então, que é um contador de módulo 8, onde o valor do módulo indica o número de estados da sequência de contagem. Se fosse acrescentado um quarto *flip-flop*, a sequência iria

de 0000 a 1111, num total de 16 estados, e seria um contador de módulo 16. Assim, você pode concluir que se tivermos N *flip-flop*s, o contador será de módulo N e contará de 0 até $2^N - 1$.

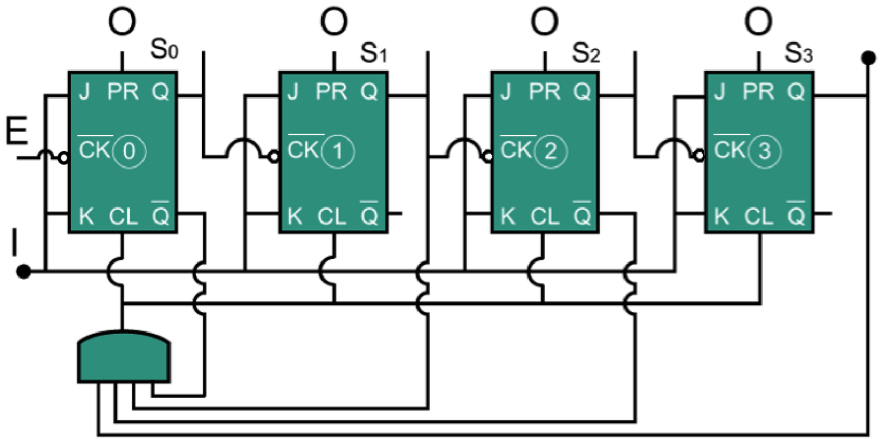
Contadores Assíncronos

Se você observar novamente a **Figura 7**, perceberá que o contador apresentado é um **contador assíncrono**, pois as entradas de controle (*clock*) dos diversos *flip-flops* que os compõem não trabalham na mesma frequência. O *clock* externo é entrada apenas do primeiro *flip-flop*. Os demais recebem como *clock* a saída do *flip-flop* anterior.

Vimos que esse contador assíncrono da **Figura 7** tem 3 *flip-flops* e pode contar até 7 (módulo 8) e que, se acrescentássemos um quarto *flip-flops*, esse contador poderia contar até 15 (módulo 16). Entretanto, no dia a dia, é bastante comum querermos contar somente até 9 (módulo 10), que é a base de numeração que costumamos usar. Para isso, precisaríamos de 10 pulsos (ou décadas), mas 10 não é potência inteira de 2. Então, como fazer? Quantos *flip-flops* usar?

Se usarmos somente 3 *flip-flops*, conseguiremos contar somente até 7 (módulo 8). Então, podemos usar 4 *flip-flops* e para que ele conte somente até 9 e não até 15, podemos usar o artifício mostrado na Figura 8, onde uma porta NAND é conectada nas entradas **CLEAR** dos *flip-flops*.

Figura 08 - Contador assíncrono de década



Fonte: <http://www.mspc.eng.br/eledig/eldg1140.shtml>. Acesso em: 10 maio 2012.

Na **Figura 8**, você verifica que as entradas da porta recebem os valores $S_3, \overline{S}_2$ (equivalente a $\overline{Q}$ do *flip-flop* 2), $S_1 e S_0$ (equivalente a $\overline{Q}$ do *flip-flop* 0).

Dessa forma, quando o valor nessas entradas for igual a 1010 (10 em binário), as entradas **CLEAR** serão '1', zerando os *flip-flops* e reiniciando a contagem (perceba que o CL é ativado em '1'). Observe que esse artifício pode ser adaptado para qualquer tamanho da sequência, desde que seja menor que 2^N, onde N é o número de *flip-flops*.

Contador assíncrono decrescente

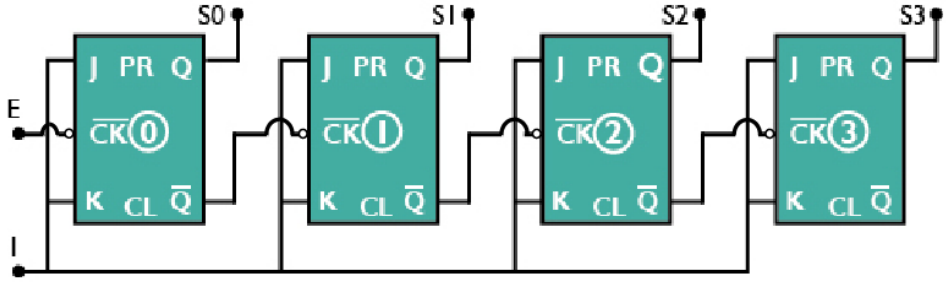
Os contadores que vimos até aqui contam de forma crescente. Algumas aplicações exigem que essa contagem seja feita de forma decrescente. Essa contagem decrescente tem saídas complementares em relação aos valores dos contadores crescentes, ou seja, $1111, 1110$, ..., na sequência decrescente para $0000, 0001$, ..., na contagem crescente.

Mas como fazer o contador contar de forma decrescente?

Uma das formas é usar a saída complementar $\overline{Q}$ ao invés da saída Q para os *flip-flops* das extremidades ($S_0 e S_3$ na **Figura 8**), mas conservando as conexões das saídas Q com as entradas dos *clocks* (CK) dos *flip-flops* seguintes.

Outra forma de se conseguir isso (**Figura 9**) é conectar as entradas dos *clocks* com as saídas $\overline{Q}$ ao invés das saídas Q, mas mantendo essas saídas Q como as saídas dos *flip-flops*.

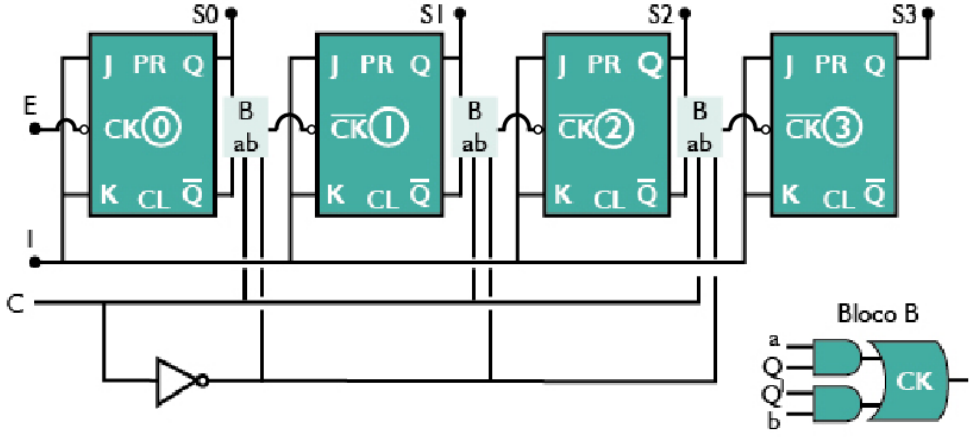
Figura 09 - Contador assíncrono decrescente



Fonte: <http://www.mspc.eng.br/eledig/eldg1140.shtml>. Acesso em: 10 maio 2012.

Existe, ainda, uma alternativa mais interessante onde poderíamos ter um contador crescente/decrescente. A escolha de uma ou de outra sequência seria feita por meio de chaves lógicas presentes nos blocos B adicionados ao contador assíncrono, conforme você vê na **Figura 10**. A contagem crescente ou decrescente depende do nível lógico da entrada de controle C.

Figura 10 - Contador assíncrono crescente-decrescente

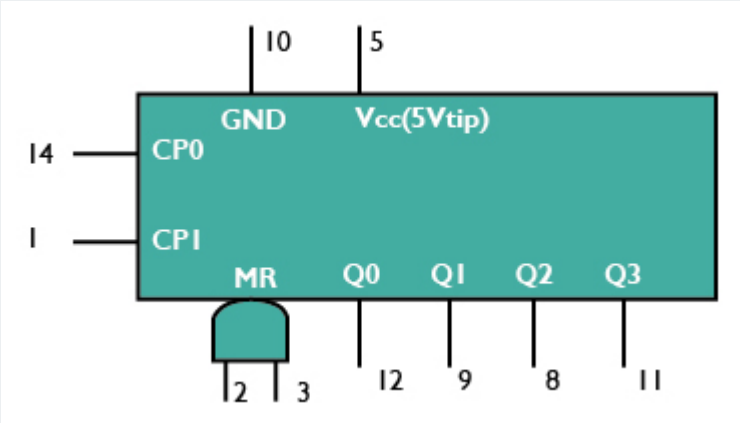


Fonte: <http://www.mspc.eng.br/eledig/eldg1140.shtml>. Acesso em: 10 maio 2012.

Exemplo de circuito integrado de contador assíncrono

A **Figura 11** mostra o diagrama lógico do circuito integrado (CI) 74HC / HCT93 da Philips. É um CI de 14 pinos (apenas 10 são usados) que funciona como um contador de 4 *bits*, pois possui 4 *flip-flops*, mas o *flip-flop* '0' (entrada CP0) é separado dos demais (entrada CP1).

Figura 11 - Diagrama lógico do CI 74HC/HCT93 da Philips



Fonte: <http://www.mspc.eng.br/eledig/eldg1140.shtml>. Acesso em: 10 maio 2012.

Para esse circuito funcionar como contador de 4 *bits*, a entrada CP_0 deve ser usada e a entrada CP_1 deve ser ligada externamente com a saída Q_0.

Se a saída CP_1 for usada, o circuito funcionará como um contador de 3 *bits*, por meio das saídas Q_1, Q_2 e Q_3. As entradas de controle dos pinos 2 e 3 funcionam como "*masterreset*" (MR) e a contagem é zerada se ambas forem colocadas em '1'

Atividade 03

1. Considere o circuito do contador da **Figura 6**, cuja tabela é vista na **Figura 7**, e responda:

    a. Se o contador começar em '000', qual será o valor da contagem após 13 pulsos de *clock*? E após 99 pulsos? E após 256 pulsos?

    b. Se o contador começar em '100', qual será o valor da contagem após 13 pulsos de *clock*? E após 99 pulsos? E após 256 pulsos?

Contadores Síncronos

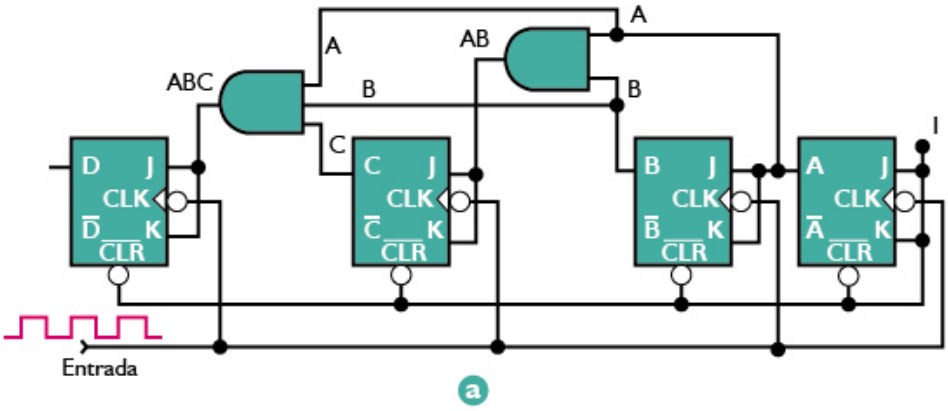
Nos contadores assíncronos, os *flip-flops* são ligados em cascata, tendo diferentes frequências. Cada *flip-flop* opera na metade da frequência do anterior. Os circuitos são simples, mas podem apresentar problemas com frequências muito

altas, pois, devido a variações de tempos de resposta, podem ocorrer atrasos de propagação dos *flip-flops*.

Esses problemas de atrasos podem ser reduzidos com o uso de **contadores síncronos** ou **paralelos** onde os *flip-flops* recebem, simultaneamente (em paralelo), o mesmo *clock* de entrada.

Por outro lado, como todos os *flip-flops* são acionados pelo mesmo *clock*, alguma lógica de controle precisa ser adicionada para controlar o momento em que um *flip-flop* deve comutar e o momento que deve permanecer inalterado, quando ocorrer um pulso de *clock*. A **Figura 12** mostra um contador síncrono de 4 *bits* (módulo 16) com essa lógica adicional.

Figura 12 - Contador síncrono módulo 16 (4 *bits*).



a

Contagem	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
.	.	.	.	.
.	.	.	.	.
.	.	etc	.	.

b

Fonte: Tocci, Widmer e Moss (2007).

Comparando esse contador síncrono da **Figura 12** com o contador assíncrono da **Figura 8**, podemos identificar as seguintes diferenças:

- No contador síncrono (**Figura 12(a)**), as entradas de *clock* (CLK) estão conectadas juntas e, dessa forma, todos os *flip-flops* recebem o sinal de *clock*, simultaneamente.

- Apenas o *flip-flop* A (que representa o *bit* menos significativo) tem suas entradas J e K sempre em '1'. As entradas J e K dos outros *flip-flops* são acionadas por uma combinação lógica das saídas dos *flip-flops*.

- O contador síncrono requer um circuito maior do que o contador assíncrono, devido à lógica de controle adicional.

Mas, como essa lógica de controle foi definida?

Se você observar a **Figura 12(b)**, que mostra a tabela do contador síncrono, poderá verificar que:

- A sequência de contagem mostra que o *flip-flop* A tem de mudar de estado em toda borda de descida do *clock*. Por isso, suas entradas J e K estão sempre em '1' para que ele comute em cada borda de descida do *clock* de entrada.

- Na sequência de contagem, o *flip-flops* B tem de mudar de estado nas bordas de descida que ocorrerem quando A = 1. Por exemplo, quando a contagem for 0001, a próxima borda de descida deverá comutar B para '1' para que o próximo valor seja 0010; quando for 0011, deverá comutar B para '0' para que o próximo valor seja 0100; e assim por diante. Isso vai ser conseguido conectando a saída de A nas entradas J e K de B, e assim ter em B: J = K = 1 apenas quando A = 1.

- O *flip-flop* C só deve comutar quando A = B = 1. Então, é necessário conectar o sinal lógico AB nas entradas J e K de C para que ele comute apenas quando A = B = 1.

- Por analogia, o *flip-flops* D só deve comutar quando A = B = C = 1. Então, é necessário conectar o sinal lógico ABC nas entradas J e K de D para que ele comute apenas quando A = B = C = 1.

Os contadores síncronos também podem ser usados como contadores de década, bem como fazerem contagens decrescentes, assim como podemos fazer com os contadores assíncronos.

Agora que você entendeu como funcionam os contadores síncronos e assíncronos, responda a atividade a seguir:

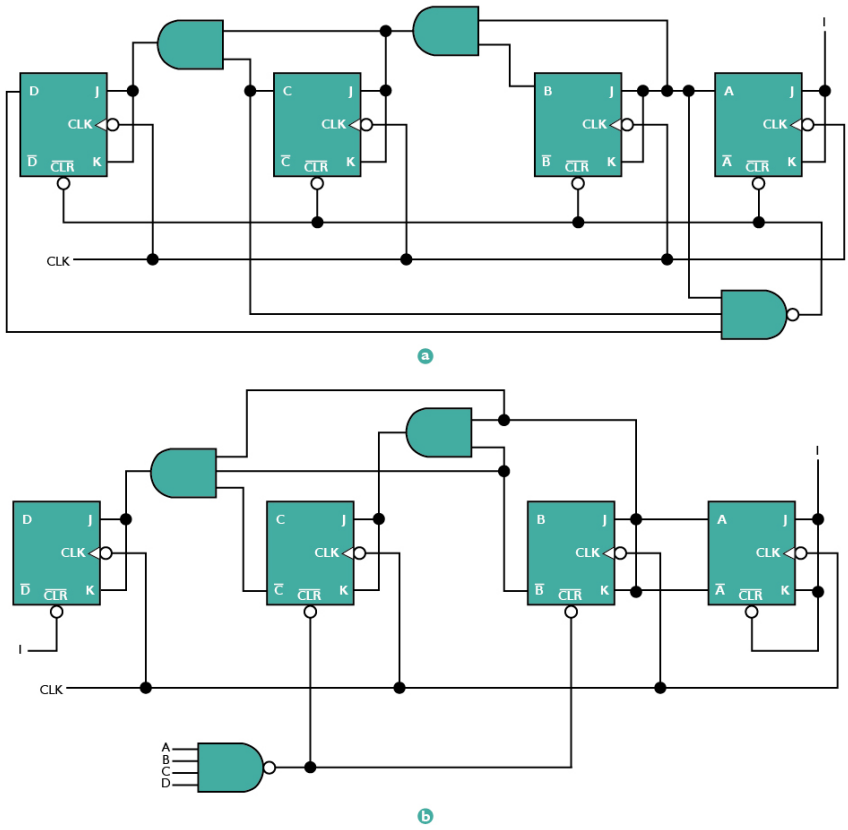
Atividade 04

1. Quais são as vantagens e desvantagens dos contadores assíncronos e síncronos?

2. Quantos *flip-flops* são necessários para construir um contador paralelo de módulo 64?

3. Analise os contadores síncronos da **Figura 13** e determine o diagrama de tempo para cada um dos contadores, além de responder qual o módulo de cada contador.

Figura 13 - Circuitos para a questão 3 da Atividade 4.



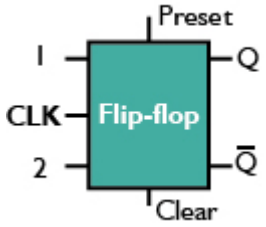
Fonte: Tocci, Widmer, Moss (2007)

Revisando

Estamos chegando ao final da nossa aula. Vamos, então, lembrar o que vimos até aqui.

1. O bloco básico de um circuito sequencial é o *flip-flop*, cujas portas são (**Figura 14**):
    - Duas entradas principais: 1 e 2 (ou uma entrada principal, no caso dos *flip-flops* T e D).
    - Uma entrada de controle (*clock*): **CLK**.
    - Duas saídas complementares: Q e $\overline{Q}$.
    - Uma entrada (opcional) de *set* ou *preset*.
    - Uma entrada (opcional) de *reset* ou *clear*.

Figura 14 - Modelo genérico de um *flip-flop*.



2. Dependendo do tipo de ligação das portas 1 e 2, temos um novo tipo de *flip-flop*, onde essas portas recebem letras de identificação distintas: **S** e **R; J** e **K; T; D**.
3. Algumas das principais aplicações para os *flip-flops* são: armazenamento, deslocamento e transferência de dados, divisão de frequência e contagem.
4. Os circuitos mais comuns que usam *flip-flops* são os registradores e os contadores.
5. Esses registradores são usados para armazenar dados, podendo ser **registradores de deslocamento**, que fazem transferência serial de dados, ou podem ser **registradores síncronos**, que fazem transferência paralela de dados.

6. Já os **contadores** binários podem ser do tipo **síncrono** ou **assíncrono**, mas ambos os tipos podem contar de forma crescente e/ou decrescente.

Você conseguiu compreender o funcionamento dos circuitos sequenciais?

Se tiver ainda alguma dúvida, releia o que for necessário nesta aula e na aula anterior e refaça as atividades. O importante é não restar nenhuma dúvida.

Roteiro Prático da Aula 09

Apresentação

Neste roteiro, você irá construir circuitos de flip-flops D e JK. Os flip-flops são circuitos primários para a construção de circuitos de armazenamento. Serão necessários os conceitos da linguagem VHDL e do software Quartus. Você vai aprender a utilizar o conceito de “event” da linguagem VHDL, importante para a construção de circuitos que utilizam de “clock”.

Objetivos

Ao final das atividades previstas para este roteiro, você será capaz de:

1. Construir flip-flops D e JK;
2. Aprender a utilizar a função “event” do VHDL;
3. Fixar os conceitos de flip-flops com a simulação dos mesmos.

Construindo um Flip-Flop JK

Como visto nas aulas anteriores, o circuito flip-flop (FF) serve como base para a construção de circuitos de armazenamento. O Flip-Flop JK é um dos exemplos dessa classe de circuitos de armazenamento, podendo ser de borda de subida ou borda de

descida. Aqui vamos implementar o FF JK com borda de subida, que funciona de acordo com a tabela 01.

J	K	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	$\overline{Q}$	Q

Tabela 1 - Tabela do Flip-Flop JK

Considerando Q' o valor anterior de Q. Para ocorrer essa atribuição dos valores de Q e $\overline{Q}$ é necessário que ocorra uma borda de subida do clock (CLK) do circuito, ou melhor, o CLK passe do valor '0' para '1'. A linguagem de descrição de hardware VHDL tem uma função específica para sentir essa mudança no clock, chamada de "event".

A função "event" é aplicada a um sinal e retorna verdadeiro quando ocorre qualquer mudança no sinal. Veja nas duas expressões abaixo como sentir quando ocorre uma borda de subida ou uma borda de descida no sinal CLK.

- Expressão 01:

$$((CLK'event)and(CLK = `1'))$$

- Expressão 02:

$$((CLK'event)and(CLK = `0'))$$

A expressão 01, acima, tem duas condições, uma que retorna verdadeiro quando ocorre uma mudança no sinal CLK, realizada pela função "event", e outra que verifica se o sinal CLK é o valor '1' naquele momento. Essa expressão realiza verificação de uma borda de subida no sinal CLK. Verifique que a expressão 02 implementa uma borda de descida no sinal CLK.

Agora que você já sabe como implementar uma borda de subida ou descida analise o código 01, dado abaixo, que implementa um circuito do FF JK.

```
    library ieee;
use ieee.std_logic_1164.all;

--Entidade--
entity flipflopJK is
port(J, K, clk: in std_logic;
Q, notQ: inout std_logic);
end flipflopJK;

--Arquitetura--
architecture ffJK of flipflopJK is
begin
process(J, K, clk)
begin
if(clk'event and clk='1') then
if(J='0' and K='0') then
Q <= Q;
notQ <= notQ;
elsif(J='1' and K='0') then
Q <= '1';
notQ <= '0';
elsif(J='0' and K='1') then
Q <= '0';
notQ <= '1';
elsif(J='1' and K='1') then
Q <= not Q;
notQ <= not notQ;
end if;
end if;
end process;
end ffJK;


```

Código 1

Atividade 05

---

a. Crie um projeto no Quartus II de nome “flipflopJK”;

a. Utilize o código 01 para construir o flip- flop JK desse projeto;

Obs.: Lembre-se que o nome do arquivo VHDL deve ser igual ao do projeto para compilar normalmente.

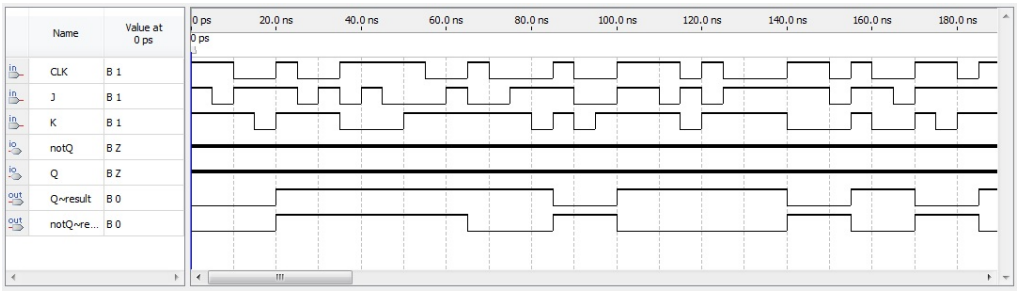
b. Crie um arquivo VHDL para o flipflopJK;

c. Compile.

Simulando o FF JK

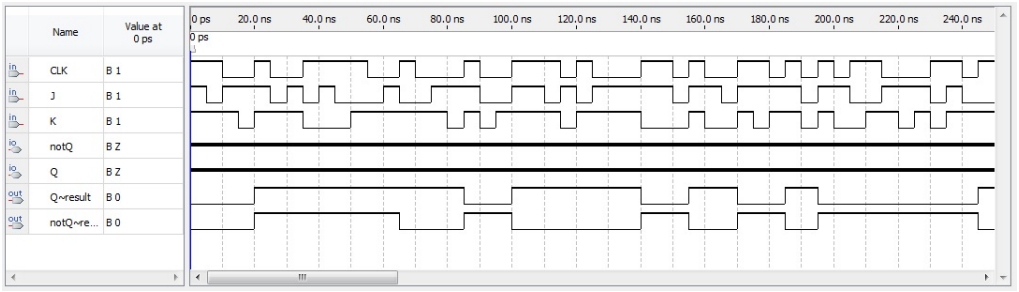
Agora vamos simular o projeto do FF JK. Você já conhece os procedimentos para simular um projeto no Quartus II 13.0, então o nosso foco aqui não é esse. Utilize vários valores de J, K e CLK para realizar a simulação. Na figura 15, dada abaixo, verifique que existe um nó diferente, definido no código VHDL como “inout”.

Figura 15 - Preparando a simulação do FF JK



No nosso caso a simulação pode ser realizada normalmente e então esses nós vão adotar ‘0’ como valores iniciais. A figura 16 mostra como ficou a simulação.

Figura 16 - Simulando o FF JK



Atividade 06

a. Crie o arquivo de simulação do projeto;

b. Simule com vários valores diferentes e verifique os quatro casos abaixo. Para cada caso mostre a forma de onda das saídas Q e $\overline{Q}$ ao seu tutor.

a. J=0, K=1 e clk com borda de subida;

b. J=0, K=1 e clk com borda de descida;

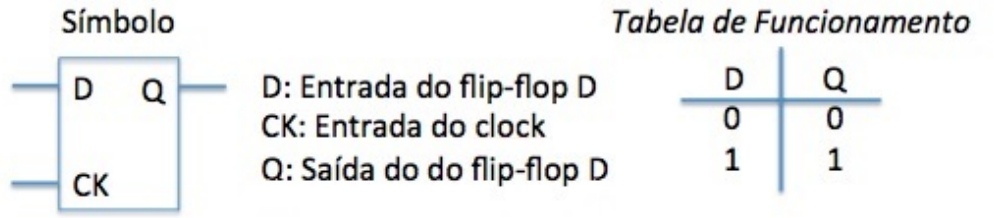
c. J=0, K=0 e clk com borda de subida;

d. J=1, K=1 e clk com borda de subida;

Construindo um FF D

Agora que você já viu como construir um flip-flop JK, utilize a mesma metodologia de implementação para construir o FF D. Você teve a oportunidade de aprender sobre circuitos do tipo flip-flop D na aula 07 (Circuitos Sequenciais - Parte I). No flip-flop D tem-se que o valor atual da saída irá sempre assumir o valor de entrada quando houver uma mudança no clock. Na figura 17, abaixo, temos o símbolo e a tabela de função do flip-flop D.

Figura 17 - Símbolo e Tabela do FF D



É possível implementar um FF-D com os conhecimentos anteriormente aprendidos ao construir um circuito FF-JK. Que tal implementar um FF-D?

Atividade 07

a. Crie um projeto no Quartus II de nome “flipflopD”;

b. Crie um arquivo VHDL para o flipflopD;

1. Construa o código VHDL para o flip-flop D tomando como base a figura 03;

OBS.: Dessa vez você vai precisar de apenas um bloco de “if-then”;

c. Compile;

d. Simule;

a. Utilizando os valores abaixo mostre como seria os valores de Q.

$$D: \ 0\,0\,1\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1$$

$$CLK: \ 0\,1\,0\,1\,0\,1\,0\,0\,0\,1\,1\,0\,0\,1\,1\,0\,1\,0$$

Leitura Complementar

[1] WAGNER, Flávio R.; REIS, André I.; RIBAS, Renato P. **Fundamentos de Circuitos Digitais**. Bookman, 2008.

[2] PEDRONI, Volnei. **Eletrônica Digital Moderna e VHDL**. Campus, 2010.

Resumo

Nessa aula, você estudou os circuitos lógicos sequenciais, cujo bloco básico é o flip-flop. Viu que eles podem ser usados em aplicações como: armazenamento, deslocamento e transferência de dados, divisão de frequência e contagem. Além disso, você conheceu os circuitos mais comuns que usam flip-flops: os registradores e os contadores. Aprendeu que esses registradores podem fazer o armazenamento, o deslocamento e a transferência de dados e que essa transferência pode ser paralela ou serial. Aprendeu, ainda, que os contadores podem fazer contagem binária crescente e/ou decrescente e, também, a divisão de frequência do clock.

No roteiro prático você reforçou seus conhecimentos sobre o software Quartus II, construiu os circuitos do flip-flop JK e D e também aprendeu como utilizar a função “event” da linguagem VHDL.

Autoavaliação

1. Considerando as transferências de dados serial e paralela, responda:
    a. Qual delas é a mais rápida?
    b. Qual é a mais simples por necessitar menos conexões?
    c. Em qual delas a fonte de dados (origem) não perde os dados?
2. Comparando os contadores síncronos e os assíncronos, quais são as vantagens e desvantagens de cada um deles?

3. Quantos *flip-flops* são necessários para construir um contador paralelo de módulo 32?

Referências

COSTA, Cesar da. **Projetos de Circuitos Digitais com FPGA**. Editora Érica, 2009.

MSPC – Informações Técnicas. Eletrônica digital XI-10: *flip-flops*. Disponível em: <http://www.mspc.eng.br/eledig/eldg1110.shtml>. Acesso em: 27 jul. 2012.

PEDRONI, Volnei. **Eletrônica Digital moderna e VHDL**. Rio de Janeiro: Elsevier, 2010.

_____. **Eletrônica digital XI-30:** *flip-flops*. Disponível em: <http://www.mspc.eng.br/eledig/eldg1130.shtml>. Acesso em: 27 jul. 2012.

_____. **Eletrônica digital XI-40:** Contadores assíncronos. Disponível em: <http://www.mspc.eng.br/eledig/eldg1140.shtml>. Acesso em: 27 jul. 2012.

_____. **Eletrônica digital XI-50**: Contadores síncronos. Disponível em: <http://www.mspc.eng.br/eledig/eldg1150.shtml>. Acesso em: 27 jul. 2012.

TOCCI, Ronald; WIDNER, Neal S.; MOSS, Gregory L. **Sistemas digitais**: princípios e aplicações. 10. ed. São Paulo: Prentice Hall, 2007.

WIKIPÉDIA. **Flip-flop**. Disponível em: <http://pt.wikipedia.org/wiki/flip-flop>. Acesso em: 27 jul. 2012.