Sistemas Digitais

Aula 08 - Circuitos Sequenciais (Parte I)

Apresentação

Na aula anterior, você conheceu os circuitos lógicos combinacionais. Como vimos, esses circuitos têm uma “amnésia” danada, pois seus resultados são somente relacionados aos valores que estão na entrada naquele instante. Eles esquecem tudo o que tinha no instante anterior.

Nesta aula, vamos estudar circuitos sem “amnésia”, pois eles sabem o que tinha no instante anterior e conseguem memorizar os valores que existiam nesse instante. Esses circuitos são os **circuitos sequenciais**.

Descobriremos que esses circuitos são formados por elementos chamados de ***latches*** e/ou por outros elementos chamados de ***flip-flop*s**.

Veremos também que esses *flip-flops* têm entradas síncronas, dependentes do *clock*, e entradas assíncronas, que operam independentemente das entradas síncronas e do *clock*.

Objetivos

Ao final desta aula, você será capaz de:

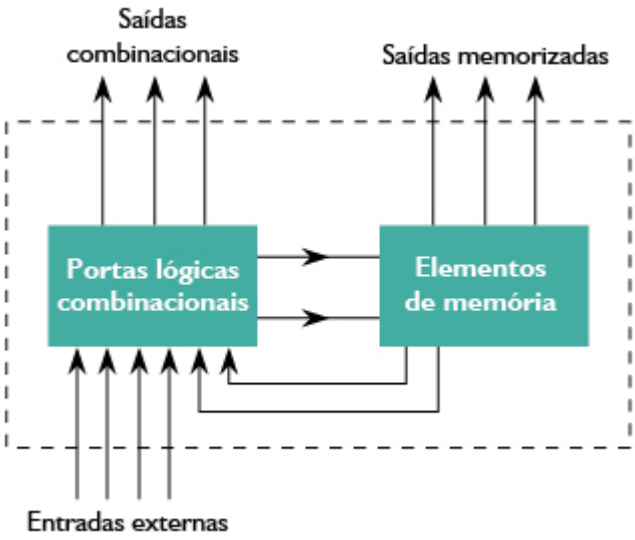
- Descrever a diferença entre sistemas síncronos e assíncronos.
- Compreender o funcionamento e diferenciar cada um dos tipos de *latches* e *flip-flops*.
- Entender o uso de *flip-flops* em circuitos de sincronização.
- Identificar a diferença entre entradas síncronas e entradas assíncronas.

Circuitos Digitais Sequenciais

Já sabemos a diferença entre circuitos analógicos e circuitos digitais. Já estudamos os circuitos combinacionais, compostos de portas lógicas. Sua saída depende apenas de valores que são colocados nas entradas, ou seja, dos níveis lógicos das entradas. Assim, um circuito combinacional não tem memória, é um "esquecido", suas saídas dependem apenas das entradas atuais. Por outro lado, os circuitos que têm memória são **chamados de circuitos sequenciais. São estes circuitos que estudaremos nesta aula.**

Na **Figura 1**, podemos ver um diagrama geral de um sistema digital, composto por uma parte combinacional e outra parte sequencial (com memória).

Figura 01 - Diagrama geral de um sistema digital



Fonte: Tocci, Widmer, Moss (2007).

Importante!

Para compreender melhor esta aula, é importante que você tenha entendido bem o conteúdo sobre portas lógicas e circuitos combinacionais visto nas últimas aulas. Se tiver dúvida, volte ao assunto e estude um pouco mais.

A saída de um circuito combinacional, em qualquer instante de tempo, depende apenas de uma combinação das entradas naquele instante de tempo. Quaisquer condições anteriores de entrada não têm efeito sobre a saída atual.

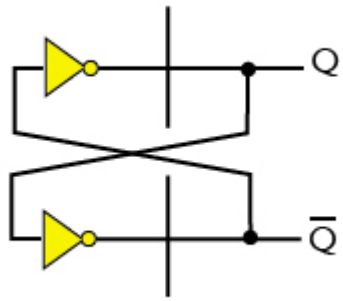
Por outro lado, nos circuitos sequenciais, a saída depende tanto da combinação das entradas atuais como das condições anteriores de entrada. Como é possível lembrar quais os valores anteriores das entradas?

É necessário o uso de elementos de memória que armazenem essas informações. Dessa forma, o circuito pode armazenar o estado atual para utilizá-lo no estado seguinte. Em outras palavras, circuitos sequenciais precisam de elementos de memória. Vamos, então, ver alguns exemplos de elementos de memória e de circuitos sequenciais.

Latches

A forma mais básica para implementar um circuito de memória é usando o **latch** que, em português, significa trinco ou ferrolho. O *latch* é composto por duas portas lógicas inversoras e possui duas saídas: a variável lógica Q e o seu complemento Q (Q barrado), como vemos na **Figura 2**.

Figura 02 - Representação de um latch



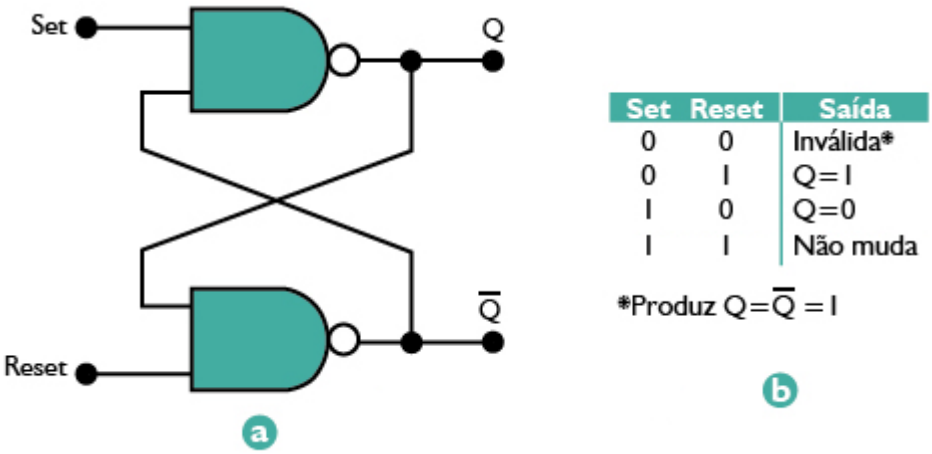
Note que, se você coloca o nível lógico alto '1' em Q, o complemento dele ($\overline{Q}$) vai para o nível lógico baixo '0'. Esse estado ($Q = 1$ e $\overline{Q} = 0$) permanece até que você mude o nível lógico de Q para '0', quando $\overline{Q}$ mudará para '1', que era o valor anterior de Q. Assim, temos um elemento de memória, certo?

Mas, observe que o *latch* só armazena um único bit. Se os dados tiverem mais de um bit, será necessário um *latch* para cada bit. Por exemplo, para uma palavra de 16 bits, precisaríamos usar 16 *latches*.

Latches SR

Além das portas inversoras (NOT), podemos também construir um *latch* com outros tipos de portas lógicas (OR e AND) e, como adicional, conseguimos disponibilizar entradas para o *latch*. Um *latch* construído dessa forma é chamado LATCH-SR. Veja na **Figura 3(a)** um *latch-SR* construído com portas NAND.

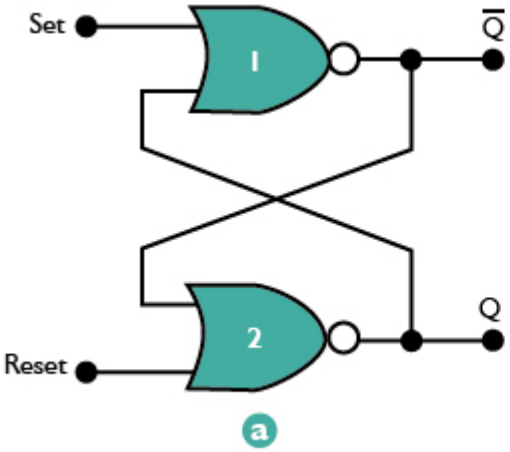
Figura 03 - (a) Latch com portas NAND (b) tabela verdade



Fonte: Tocci, Widmer, Moss (2007).

Agora, observe a **Figura 4(a)** e veja um *latch-SR* construído com portas NOR.

Figura 04 - (a) Latch com portas NOR (b) tabela-verdade



Set	Reset	Saída
0	0	Não muda
0	1	Q=0
1	0	Q=1
1	1	Inválida*

*Produz $Q=\overline{Q}=1$

b

Fonte: Tocci, Widmer, Moss (2007).

Um *latch-SR* é composto por 2 portas lógicas (NAND ou NOR) interligadas de modo cruzado, de forma que a saída da 1ª porta é conectada a uma das entradas da 2ª porta e vice-versa.

Agora, vamos observar as tabelas verdade das **Figuras 3(b)** e **4(b)** para entendermos a diferença entre esses dois tipos de *latches SR*.

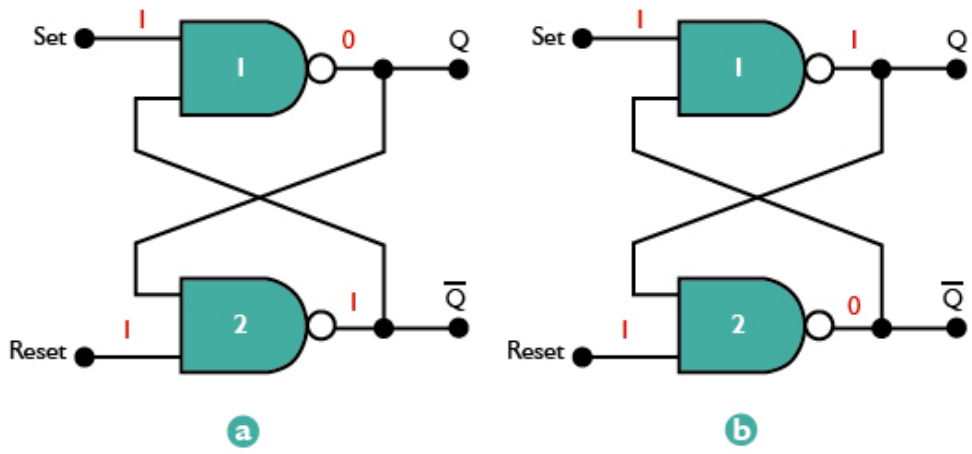
Para o latch com portas NAND temos:

Condição	Situação	Explicação
SET=1 e RESET=1	estado de repouso	As saídas Q e Q permanecem nos mesmos estados que estavam antes essa condição de entrada.
SET=1 e RESET=0	resetar o latch	Faz a saída Q = 0, mantendo esse valor, mesmo que a entrada RESET retorne para ‘1’.
SET=0 e RESET=1	setar o latch	Faz a saída Q = 1 e mantém esse valor mesmo que a entrada SET retorne para '1'.

Condição	Situação	Explicação
SET=0 e RESET=0	estado inválido	Essa condição tenta, ao mesmo tempo, setar e resetar o latch, além de produzir Q=Q=1. Se as entradas SET e RESET retornarem a '1' simultaneamente, o resultado é imprevisível. NUNCA UTILIZAR ESTA CONDIÇÃO

Você deve estar se perguntando: quando colocamos as entradas SET e RESET em '1', não ocorre nenhuma mudança nas saídas Q e $\overline{Q}$? Veja, então, a **Figura 5** e verifique que existem duas possibilidades de valores para as saídas, mas em ambos os casos ocorre a permanência do valor já existente, ou seja, as saídas continuam em estado de repouso.

Figura 05 - Estados do latch com portas NAND quando SET = RESET = 1



Fonte: Tocci, Widmer, Moss (2007).

E para o *latch* com portas NOR, como será?

Se observarmos o funcionamento do *latch* com portas NOR, teremos:

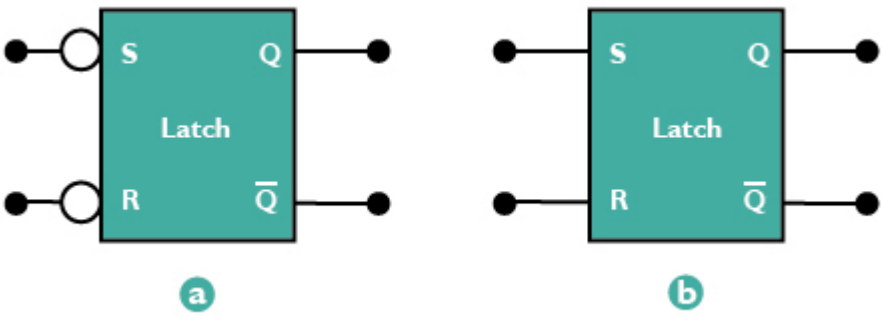
1. **SET=RESET=0:** estado de repouso. As saídas Q **e** $\overline{Q}$ **permanecem nos mesmos estados** que estavam antes dessa condição de entrada.
2. **SET=0 e RESET=1:** resetar o *latch*. Faz a saída $Q = 0$ e mantém esse valor mesmo que a entrada RESET retorne para '0'
3. **SET=1 e RESET=0:** setar o *latch*. Faz a saída $Q = 1$, mantendo esse valor, mesmo que a entrada SET retorne para '0'.
4. **SET=RESET=1:** estado *inválido*. Essa condição tenta, ao mesmo tempo, setar e resetar o latch, além de produzir $Q = \overline{Q} = 0$.

   Se as entradas SET e RESET retornarem a $'0'$ simultaneamente, o resultado é imprevisível.

   ESSA CONDIÇÃO NÃO DEVE SER USADA.

Comparando os dois tipos de *latches SR*, você percebe que o *latch* NOR funciona de forma semelhante ao latch NAND, exceto pelo fato de as entradas SET e RESET serem ativadas no nível lógico '1' para os *latches* do tipo NOR e no nível lógico '0' para os *latches* do tipo NAND. A **Figura 6** mostra uma representação simplificada baseada nesse comportamento. O *latch* NAND está representado na **Figura 6(a)** e o *latch* NOR na **Figura 6(b)**.

Figura 06 - Símbolo simplificado do latch SR usando portas (a) NAND e (b) NOR



Fonte: Tocci, Widmer, Moss (2007).

Atividade 01

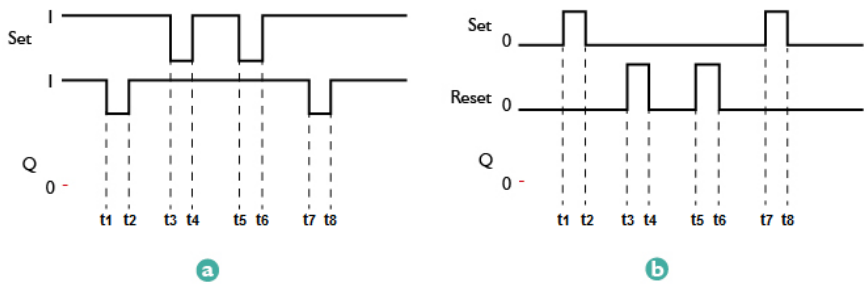
1. Faça uma pesquisa na internet e descubra mais sobre a história dos *latches*.

2. Você compreendeu bem a diferença entre o *latch* SR com portas NAND e o *latch* SR com portas NOR? Que diferença básica você consegue ver entre os dois?

3. Agora que você já sabe a diferença entre os *latches* SR, mostre como fica a forma de onda da saída Q para o latch NAND, usando como referência a **Figura 7(a)**, e para o *latch* NOR, usando como referência a **Figura 7(b)**. Considere, em ambos os casos, que o valor inicial de Q é o nível lógico $'0'$ ($Q = 0$).

   Dica: procure avaliar o valor esperado para cada um dos tempos indicados (T1, T2, ..., T8), baseando-se nas respectivas tabelas verdade vistas nas **Figuras 3 e 4.**

Figura 07 - Formas de onda dos *latches* SR com portas **(a)** NAND e **(b)** NOR

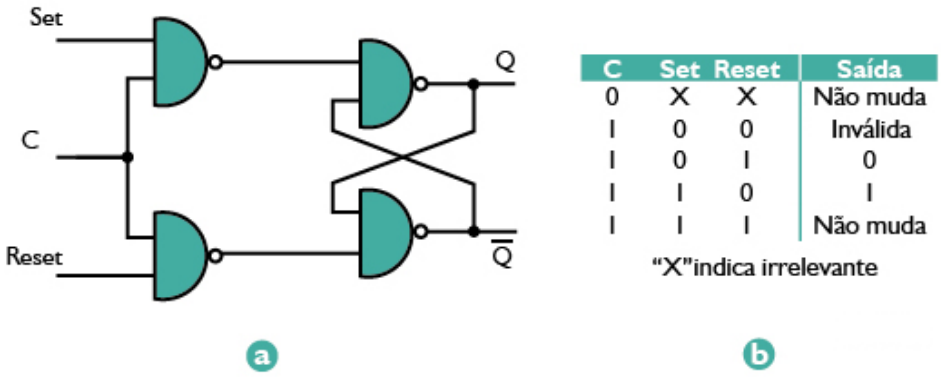


Fonte: Tocci, Widmer, Moss (2007).

Bem, apesar das diferenças entre os *latches* NAND e NOR, você pode perceber que, em ambos, existe o problema de haver uma combinação de entradas que leva a um estado inválido. Uma alternativa para driblar essa desvantagem dos *latches* SR pode ser a inclusão de uma terceira entrada, como podemos ver no latch NAND da **Figura 8**.

Perceba que a entrada **C** funciona como um controlador do latch, restringindo as entradas que possam afetar o estado do latch. Por exemplo, na **Figura 8(b)**, podemos ver que a situação em que SET=RESET=0 implicaria em uma saída instável, mas isso pode ser evitado, colocando a entrada de controle em '0'.

Figura 08 - Latch SR com entrada de controle



Fonte: Tocci, Widmer, Moss (2007).

Até agora, você conheceu os *latches* e descobriu como diferenciar alguns de seus tipos. Mas, e qual é a diferença entre os *latches* e os *flip-flop*s? Para descobrir isso, vamos conhecer melhor os *flip-flop*s.

Sistemas Síncronos e Assíncronos

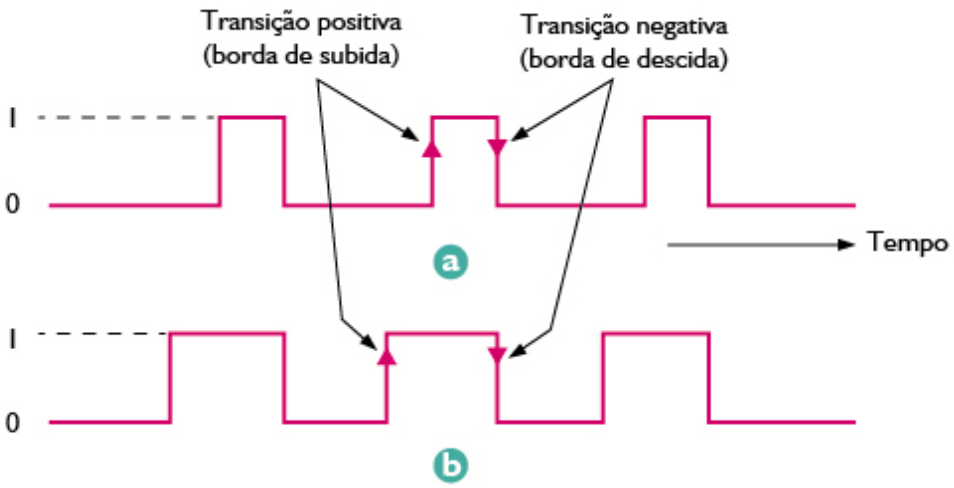
Bom, já conhecemos os *latches*. Agora, vamos ver o que os diferencia dos *flip-flop*s. Para isso, vamos começar relembrando os tipos de operação dos sistemas digitais. Os sistemas digitais podem operar tanto no modo **assíncrono** quanto no modo **síncrono.**

Nos sistemas **assíncronos**, as saídas podem mudar de estado a qualquer momento. Basta que uma ou mais entradas mude de estado.

Já nos sistemas **síncronos**, as saídas mudam de estado somente no instante em que uma entrada, denominada de *clock* (em português, relógio), muda de valor. Em outras palavras, existe um sincronismo entre as variações nas saídas e a variação da entrada de *clock*. E também, as variações das outras entradas somente serão consideradas no momento em que o sinal do *clock* sofre essa variação. Essa variação pode ser uma mudança de '0' para '1' ou então de '1' para '0'.

A **Figura 9** mostra exemplos de sinais de *clock* com suas transições (ou bordas). O sinal do *clock* é, em geral, distribuído para todas as partes do sistema e todas (ou a maioria) as saídas mudam de estado apenas quando ocorre uma transição no sinal do *clock*.

Figura 09 - Exemplos de sinais de *clock*



Fonte: Tocci, Widmer, Moss (2007).

Quando o *clock* faz uma transição de '0' para '1', dizemos que houve uma transição positiva (borda de subida), quando a transição é de '1' para '0', chamamos de transição negativa (borda de descida). Resumindo, se um *flip-flop* é de borda de subida, só podemos analisar a sua tabela verdade somente quando o *clock* mudar de '0' para '1'. Já se o *flip-flop* é de borda de descida, só podemos analisar a sua tabela verdade quando o *clock* muda de '1' para '0'. Tenha isso em mente, OK?

Atividade 02

---

1. Você lembra que, no latch SR, colocamos uma entrada adicional **C** para fazer o controle das saídas? Se não lembra, volte na **Figura 8** e reveja. Se essa entrada **C** for um sinal de *clock*, o que teremos?

Se você respondeu “um circuito síncrono”, você acertou. Algumas denominações que você também encontrará para esse tipo de circuito são: *flip-flop* SR com *clock*, *flip-flop* SR disparado por borda.

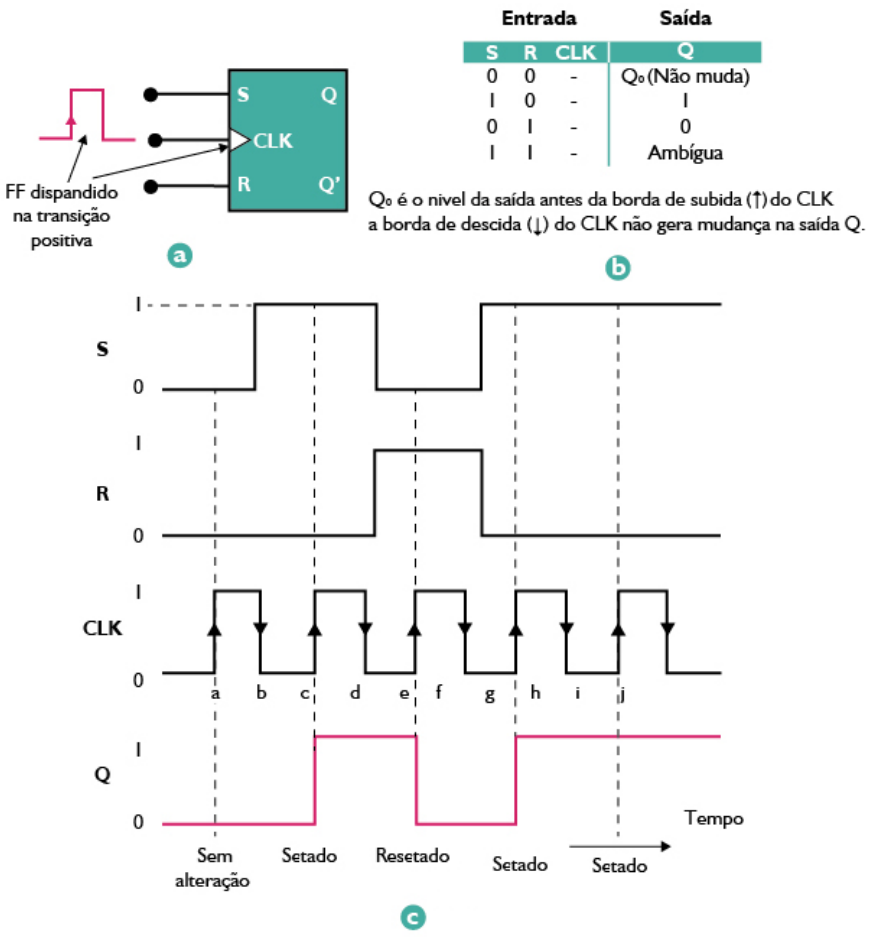
Mas, você deve estar se perguntando: que diferença existe entre o latch e um *flip-flop*? Por que chamá-lo de *flip-flop* e não de latch?

Podemos dizer que o latch é um tipo de *flip-flop*. De uma forma geral, podemos representar o *flip-flop* como um bloco no qual temos duas saídas: Q e $\overline{Q}$, entradas para as variáveis (como vimos nos *latches*) e mais uma entrada de controle, o *clock*.

Os *flip-flop*s são comumente usados em sistemas lógicos sequenciais. Costuma-se dizer que *flip-flop* **é o bloco básico dos circuitos sequenciais**.

Na **Figura 10**, você pode ver um *flip-flop* SR com *clock* disparado pela borda de subida. Perceba que na **Figura 10(c)**, a saída Q não é analisada na borda de descida (ver linhas tracejadas) .

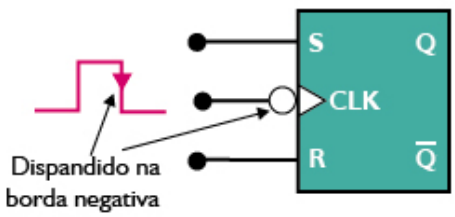
Figura 10 - **(a)** Diagrama de *flip-flop* SR com *clock* que responde apenas à borda de subida **(b)** Tabela verdade **(c)** Formas de onda típicas



Fonte: Tocci, Widmer, Moss (2007).

Já na **Figura 11** vemos um *flip-flop* SR com *clock* disparado apenas nas bordas de descida. A análise do *flip-flop* é semelhante ao da **Figura 10(c)**, exceto que verificaríamos somente na borda de descida.

Figura 11 - Diagrama e tabela verdade de *flip-flop* SR com *clock* de transição negativa



Entradas			Saída
S	R	CLK	Q
0	0	↓	Q_0 (Não muda)
0	1	↓	0
1	0	↓	1
1	1	↓	Ambígua (inválida)

Fonte: Tocci, Widmer, Moss (2007)

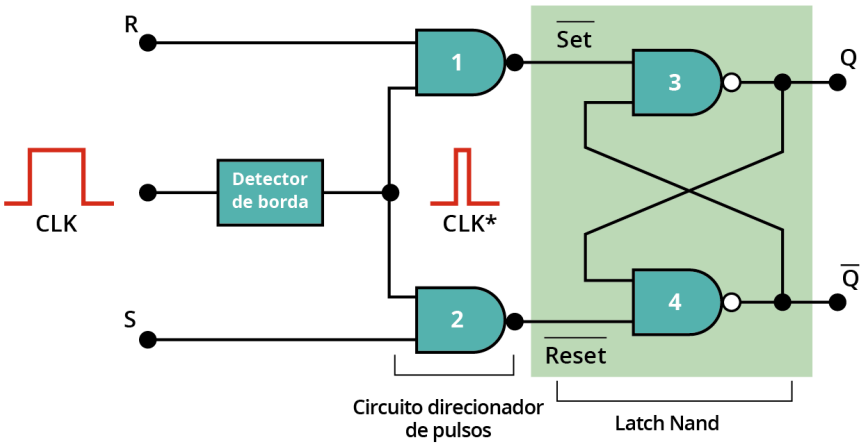
Borda de Subida ou Descida?

Para descobrir se um *flip-flop* é de borda de descida, basta ver na simbologia do *flip-flop* se a entrada do *clock* é negada (com uma bola vazia). De maneira análoga, se o *flip-flop* não possui *clock* barrado, ele é de borda de subida. Esteja muito atento!

A **Figura 12** mostra uma versão simplificada dos componentes de um *flip-flop* SR disparado por borda. Como você pode perceber, um desses "componentes" é um latch NAND.

Observe que as entradas set (S) e reset (R) não têm efeito no *flip-flop*, exceto nos instantes de ocorrência de variação do *clock*, pois estão conectadas em portas NAND juntamente com o sinal de *clock*. Assim, se for um *flip-flop* como o da **Figura 10**, o acionamento ocorrerá na borda de subida (transição positiva) do *clock* e, se for um *flip-flop* como o da **Figura 11**, o acionamento será na borda de descida (transição negativa) do *clock*.

Figura 12 - Versão simplificada do circuito interno de um *flip-flop* SR disparado por borda

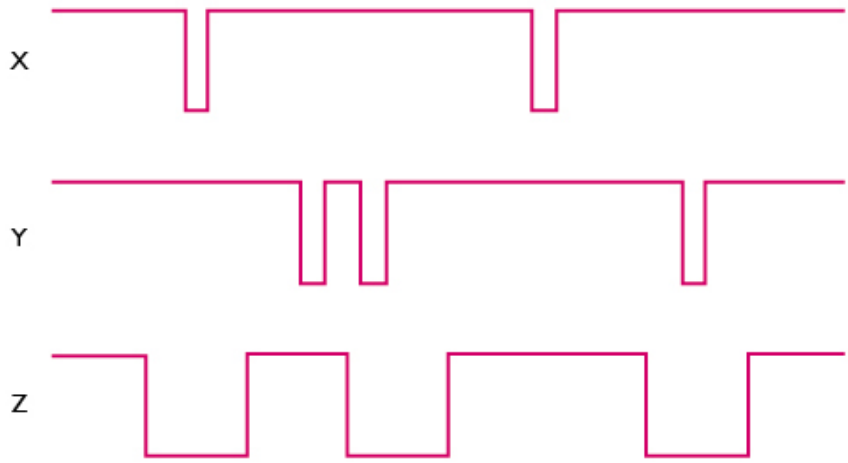


Fonte: Tocci, Widmer, Moss (2007).

Atividade 03

1. Considerando que as formas de onda X e Y da **Figura 13** são aplicadas respectivamente nas entradas SET e RESET de um latch NAND (**Figura 3**), considerando também que, inicialmente, a saída $Q = 0$, determine quais serão as formas de onda das saídas Q e Q'.

Figura 13 - Formas de onda para serem usadas nas questões 1 a 3 da Atividade 3

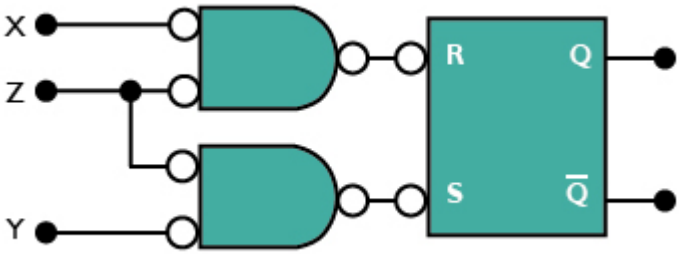


Fonte: Tocci, Widmer, Moss (2007)

2. Inverta a forma de onda X e a forma de onda Y da **Figura 13** e aplique, respectivamente, nas entradas SET e RESET de um latch NOR como o da **Figura 4**. Com essas entradas, determine a forma de onda das saídas Q e $\overline{Q}$, considerando que inicialmente a saída é $Q = 0$.

3. Aplique as formas de onda da **Figura 13** no circuito da **Figura 14**, considerando inicialmente a saída $Q = 0$, determine a forma de onda da saída Q.

Figura 14 - Circuito para a questão 3 da Atividade 3



Fonte: Tocci, Widmer, Moss (2007)

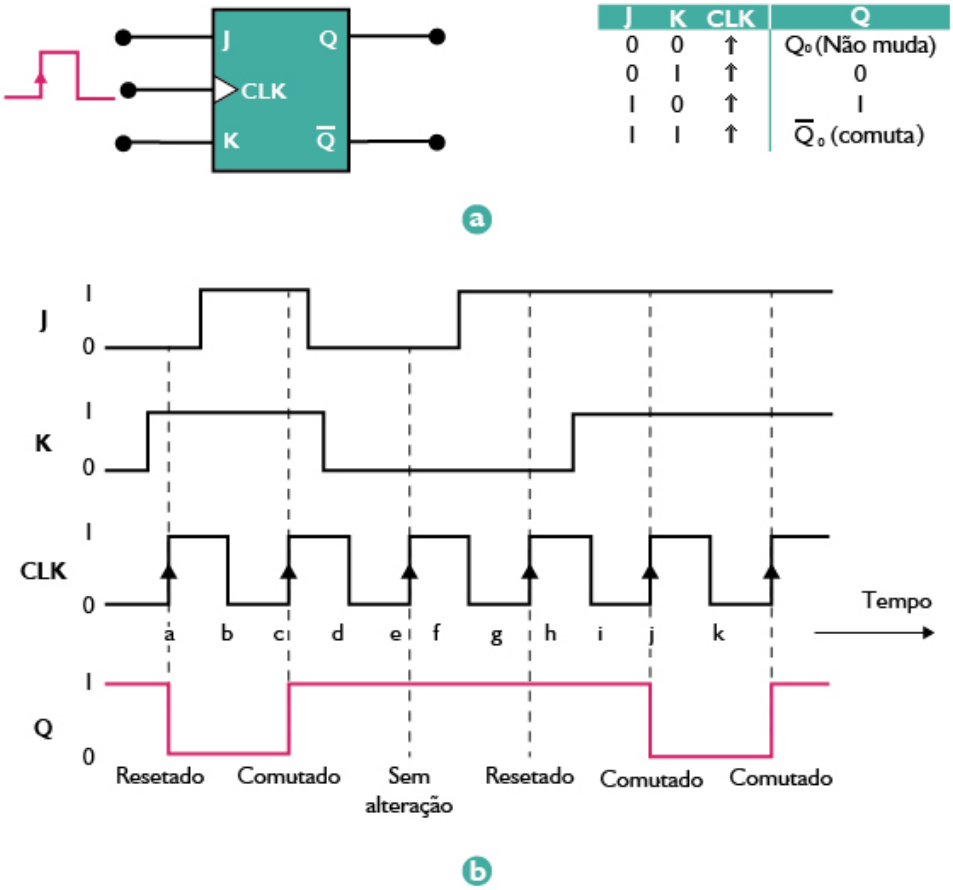
Flip-flop JK

Vimos que o sinal de *clock* pode controlar as saídas de um *flip-flop* SR, mas não resolve o problema da existência de um estado indefinido na saída, quando S=R=1.

Uma solução para isso é utilizar outro tipo de *flip-flop*, o JK, pois ele aprimora o funcionamento do *flip-flop* SR, interpretando a condição S=R=1 como um comando de inversão e não como um estado indefinido.

Esse modo é chamado de **comutação**, pois o valor do sinal de saída mudará de um estado lógico para outro (de '0' para '1' ou de '1' para '0') em cada borda de subida do *clock*, como você pode ver na **Figura 15**.

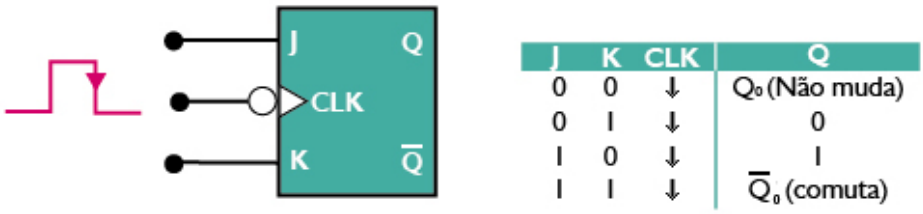
Figura 15 - **(a)** *flip-flop* JK com *clock* que responde apenas à borda de subida **(b)** Formas de onda típicas



Fonte: Tocci, Widmer, Moss (2007)

Já na **Figura 16**, você observa o caso de um *flip-flop* JK disparado apenas nas bordas de descida do *clock*.

Figura 16 - *flip-flop* JK disparado apenas na borda de descida do *clock*



Fonte: Tocci, Widmer, Moss (2007)

Observando as tabelas verdade nas **Figuras 15** e **16**, você pode verificar que, em relação às combinações de valores de J e K:

1. **J = K = 0** o *flip-flop* mantém o valor da saída Q;
2. **J = 0** e **K = 1** é um comando para desativar (reset) a saída Q do *flip-flop*;
3. **J = 1** e **K = 0** é um comando para ativar (set) a saída Q do *flip-flop*;
4. **J = K = 1** é um comando para inverter (comutar) o *flip-flop*, ou seja, trocar o sinal de saída Q pelo seu complemento $(\overline{Q})$.

Assim, você pode concluir que o *flip-flop* JK pode fazer tudo que um *flip-flop* SR faz, além de operar no modo de comutação, sem ter estados ambíguos ou inválidos.

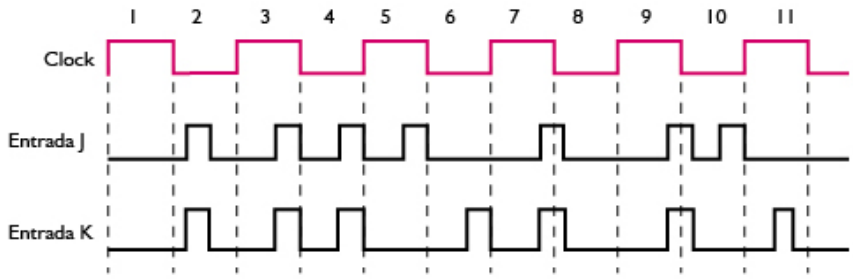
Atividade 04

Agora que já fomos apresentados aos *flip-flop*s SR e JK, vamos praticar um pouco, respondendo a essas questões:

1. Um *flip-flop* JK pode ser usado como um *flip-flop* SR, mas um *flip-flop* SR não pode ser usado como um *flip-flop* JK. Verdadeiro ou Falso?
2. Um *flip-flop* JK tem alguma condição de entrada ambígua ou inválida?
3. Qual a condição de entrada de J e K que faz a saída Q=1 no instante da transição do *clock*?
4. As formas de onda da **Figura 17** são aplicadas em dois *flip-flop*s diferentes:
    a. JK disparado na borda de subida do *clock* (como na Figura 15).
    b. JK disparado na borda de descida do *clock* (como na Figura 16).

    Determine a forma de onda da saída Q para cada um desses dois *flip-flop*s, considerando que, inicialmente, em ambos, Q=0.

Figura 17 - Formas de onda para a questão 4 da Atividade 4



Fonte: Tocci, Widmer, Moss (2007)

Flip-flop T e *Flip-flop* D

Bem, você viu que, nos *flip-flops* JK, quando J=K=1, ocorre uma comutação, ou seja, a saída muda para o sinal lógico oposto (de '1' para '0' ou de '0' para '1'). E quando J=K=0, a saída é mantida com o mesmo valor.

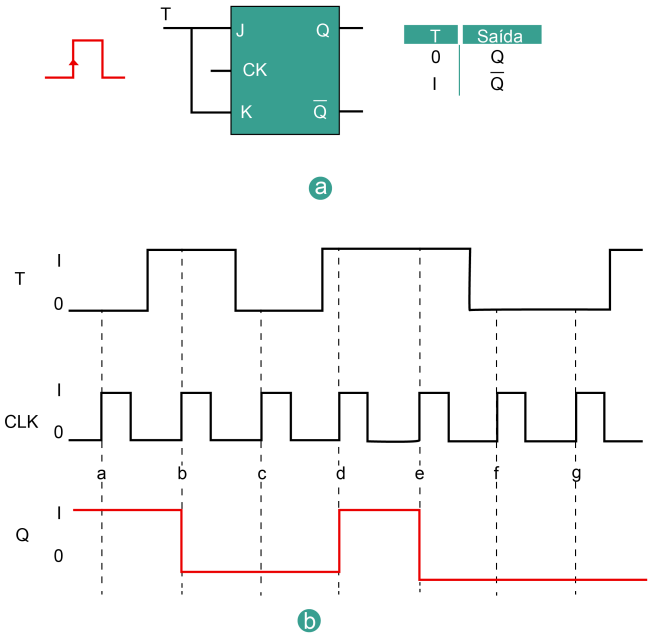
Também viu que, quando os valores de J e K são diferentes, o *flip-flop* irá ser ativado (**set**) ou desativado (**reset**).

Mas, e se não precisarmos ativar ou desativar o *flip-flop*, se precisarmos apenas manter o valor da saída ou invertê-lo (comutar)?

Nesse caso, teríamos sempre J=K (igual a '0' para manter o valor ou igual a 1 para comutar). Então, se sempre vamos usar o mesmo valor para J e K, não seria melhor ter uma única entrada ao invés de duas?

Isso é o que faz o *flip-flop* T (**Figura 18**). Um *flip-flop* T é um *flip-flop* JK com as entradas interligadas, ou seja, os valores de J e K somente podem ser o mesmo (J=K='0' ou J=K='1').

Figura 18 - *flip-flop* T: (a) diagrama e tabela verdade (b) Formas de onda

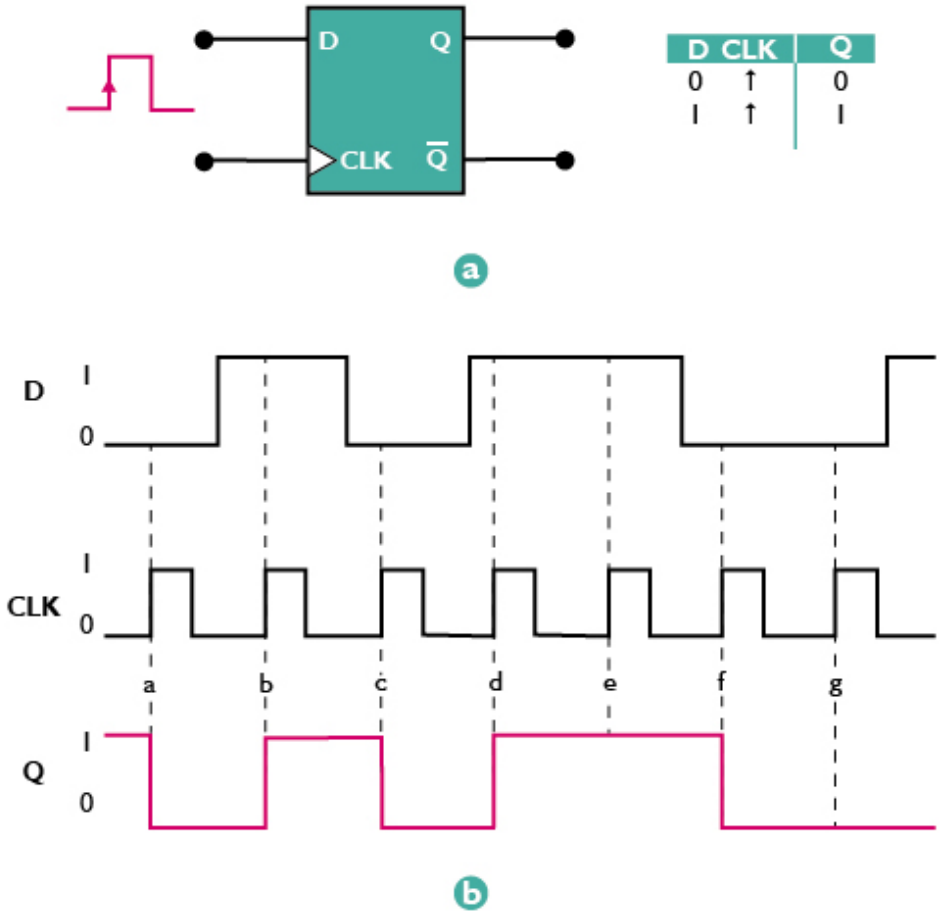


Fonte: IMD

Flip-flop D

Assim como o *flip-flop* T e ao contrário dos *flip-flop*s SR e JK, o *flip-flop* D tem apenas uma entrada de controle síncrona, a entrada D, que representa a palavra data (dado). A **Figura 19** mostra um *flip-flop* D com *clock* disparado na borda de subida do *clock*. Se você observar a tabela verdade na **Figura 19(a)**, verá que a saída Q irá para o mesmo estado lógico presente na entrada D quando ocorrer uma borda de subida do *clock* (CLK).

Figura 19 - **(a)** *flip-flop* D disparado apenas na borda de subida do *clock* **(b)** Formas de onda

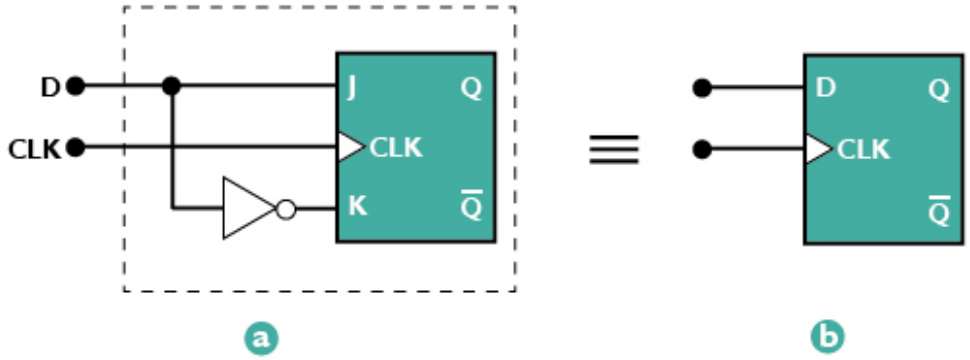


Fonte: Tocci, Widmer, Moss (2007)

Se você observar as formas de onda mostradas na **Figura 19(b)**, perceberá que o nível lógico presente em D será armazenado no *flip-flop* somente no instante em que ocorrer a borda de subida do *clock*. Mas como é que isso é possível?

Um *flip-flop* D, com *clock* disparado por borda, pode ser implementado quando acrescentado um único inversor a um *flip-flop* JK com *clock* disparado por borda, como mostra a **Figura 20**. Se você fizer um teste com os possíveis valores para D ('0' e '1'), verá que a saída Q assumirá o mesmo valor que estiver na entrada D quando ocorre uma borda de subida do *clock* (CLK).

Figura 20 - Implementação de *flip-flop* D com *clock* disparado por borda a partir de *flip-flop* JK

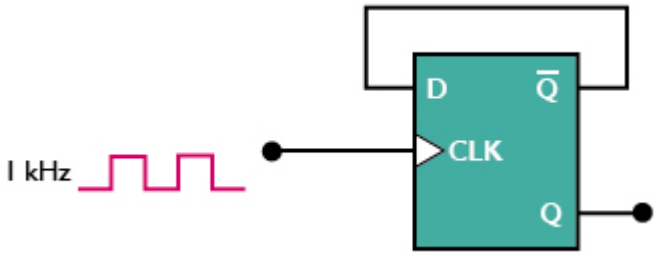


Fonte: Tocci, Widmer, Moss (2007)

Atividade 05

1. Um *flip-flop* D disparado por borda pode ser configurado para operar como *flip-flop* T, conforme mostra a **Figura 21**. Considerando que inicialmente Q=0, determine a forma de onda da saída Q.

Figura 21 - Circuito para a questão 1 da Atividade 5



Fonte: Tocci, Widmer, Moss (2007)

Entradas Síncronas e Assíncronas

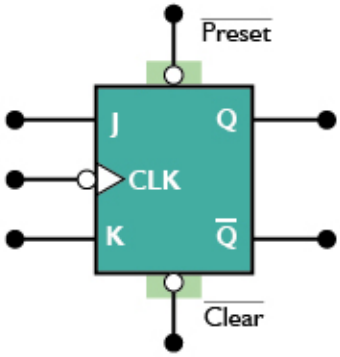
Nos *flip-flop*s que estudamos até agora, as entradas S, R, J, K e D são denominadas entradas de controle ou entradas síncronas, pois seu efeito na saída do *flip-flop* é sincronizado com entrada *clock* (CLK).

A maioria dos *flip-flop*s com *clock* também tem uma ou mais entradas assíncronas que operam independentemente das entradas síncronas e da entrada de *clock*. Essas **entradas assíncronas** podem ser usadas para colocar o *flip-flop* em

'0' ou '1' em qualquer instante, independentemente das condições das outras entradas, inclusive do *clock*.

A **Figura 22** mostra um *flip-flop* JK com duas entradas assíncronas PRESET e CLEAR. Elas são ativas em nível lógico baixo ('0'), conforme está indicado pelo uso dos pequenos círculos ('o') no símbolo do *flip-flop*.

Figura 22 - *flip-flop* JK com *clock* e entradas assíncronas



$\overline{\text{Preset}}$	$\overline{\text{Clear}}$	Reposta do FF
1	1	Operação com clock*
0	1	Q=1 (independente do CLK)
1	0	Q=0 (independente do CLK)
0	0	Não usada

*Q responderá a JK e CLK

Fonte: Tocci, Widmer, Moss (2007)

Analisando a tabela verdade que está na **Figura 22**, podemos ver os seguintes casos:

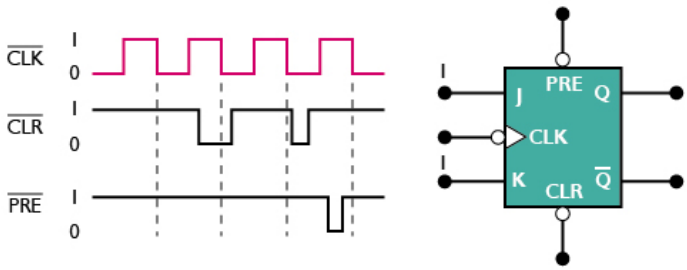
1. **PRESET = CLEAR = 1**. As entradas assíncronas estão desativadas e o *flip-flop* fica disponível para responder às entradas J, K e CLK, ou seja, as operações síncronas podem ser realizadas.
2. **PRESET = 0** e **CLEAR = 1**. A entrada PRESET está ativada e a saída Q é colocada imediatamente em '1', independente dos valores que tenham as entradas síncronas J, K e CLK. Enquanto PRESET estiver em '0', a entrada CLK não afeta o *flip-flop*.
3. **PRESET = 1** e **CLEAR = 0**. A entrada CLEAR está ativada, e a saída Q é colocada imediatamente em '0', independente dos valores que tenham as entradas síncronas J, K e CLK. Enquanto CLEAR estiver em '0', a entrada CLK não pode afetar o *flip-flop*.
4. **PRESET = CLEAR = 0.** Essa condição não deve ser usada, pois resulta em resposta ambígua ou inválida.

Em resumo, podemos concluir que as entradas assíncronas servem para “forçar” um estado na saída do *flip-flop*, independente das entradas síncronas. Muitos *flip-flop*s que estão disponíveis em circuitos integrados (chips) têm essas duas entradas assíncronas, alguns têm apenas a entrada CLEAR.

Atividade 03

1. Determine a forma de onda da saída Q do *flip-flop* da Figura 23. Considere inicialmente $Q = 0$ e lembre-se que entradas assíncronas (**preset** — PRE e **clear** — CLR) tem prioridade em relação a todas as outras entradas.

Figura 23 - Circuito e formas de onda para a questão 1 da Atividade 6



Fonte: Tocci, Widmer, Moss (2007)

flip-flops disparados por borda (com *clock*) são dispositivos versáteis que podem ser usados em uma ampla variedade de aplicações, incluindo:

- contagem;
- armazenamento binário de dados;
- transferência de dados de um local para outro.

Quase todas essas aplicações lógicas usam *flip-flops* com *clock* como bloco básico e muitas dessas aplicações são circuitos sequenciais. Você lembra que os circuitos sequenciais são aqueles em que as saídas seguem uma sequência predeterminada de estados, com um novo estado ocorrendo a cada pulso de *clock*, não é mesmo?

Então, na próxima aula, vamos ver algumas dessas aplicações com *flip-flop*s.

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

Nesta aula, você estudou os circuitos sequenciais e viu o funcionamento de *latches* e *flip-flop*s. Você aprendeu a diferenciar os *latches* dos *flip-flop*s e descobriu que o *flip-flop* é o bloco básico dos circuitos lógicos sequenciais e que ele pode ser utilizado em uma ampla variedade de aplicações como: contagem, armazenamento binário de dados, deslocamento e transferência de dados. Além de entender o funcionamento de cada um dos tipos de *latches* e *flip-flop*s, você aprendeu também a diferenciar sistemas síncronos de sistemas assíncronos.

Autoavaliação

1. Se tivermos a entrada SET=0 em um latch com portas NAND, as saídas serão:

   ( ) Q=0 e $\overline{Q}$=1.

   ( ) Q=1 e $\overline{Q}$=0.

   ( ) Q=1 e $\overline{Q}$=1.

   ( ) Q=0 e $\overline{Q}$=0.

2. Uma diferença principal entre um *flip-flop* D e um JK e RS é o fato que:

    ( )FF D tem uma entrada de controle e nenhuma entrada de *clock*.

    ( )FF D tem duas entradas de controle e nenhuma entrada de *clock*.

    ( )FF D tem apenas uma entrada de controle e uma entrada de *clock*.

    ( )FF D tem apenas uma entrada de controle e duas entradas de *clock*.

3. Qual a diferença entre a operação de uma entrada síncrona e a de uma entrada assíncrona?

Referências

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: <https://pt.wikipedia.org/wiki/Flip-flop>. Acesso em: 26 jun. 2012.