

SISTEMAS DIGITAIS

UNIDADE 4 – CONTADORES BINÁRIOS

Fernando Cortez Sica

Introdução

Olá, prezado estudante! Seja bem-vindo a esta unidade de **Sistemas Digitais**.

A partir de seus conhecimentos sobre *latches* e *flip-flops*, chegou a hora de dialogarmos sobre outro representante da lógica sequencial: os contadores. Mas o que vêm a ser os contadores? Como o próprio nome diz, são circuitos responsáveis por estabelecer uma sequência de contagem.

Será que as contagens devem ser sempre lineares, ou seja, com os números consecutivos? Na verdade, não. Por esse motivo, abordaremos os contadores síncronos e assíncronos. Os assíncronos permitem que sejam produzidas apenas contagens lineares, já com os síncronos, podemos definir sequências não lineares.

Mas qual é a utilidade/aplicação das contagens produzidas pelos contadores além do óbvio? Bem, com a contagem, podemos visar, por exemplo, a divisão de frequência e o controle inerente a uma máquina de estados. E será que com os componentes digitais, é possível montar um circuito contador? Sim, porém, para isso, vamos descrever dois novos tipos de *flip-flops*: os *flip-flops* tipo “JK” e tipo “T”. Tenha em mente que eles são a base dos circuitos contadores, tanto assíncronos quanto síncronos.

Assim como os demais circuitos combinacionais e sequenciais, os contadores também podem ser implementados utilizando Linguagens de Descrição de Hardware (HDL, ou *Hardware Description Language*) ou circuitos integrados que implementam os *flip-flops*, ou que implementam os próprios contadores.

Para que possamos conversar sobre o assunto, vamos primeiro abordar os novos tipos de *flip-flops* (tipo “JK” e tipo “T”), para que possamos aplicá-los aos contadores assíncronos. Após vermos os conceitos envolvidos e a forma de implementação dos contadores assíncronos, falaremos sobre como limitar a sua contagem. Em seguida, dialogaremos sobre os contadores síncronos, sobre como podemos implementá-los, bem como sobre suas vantagens sobre os assíncronos. Por fim, para que possamos conversar sobre o emprego de contadores nas máquinas de estado, conceituaremos tais máquinas de estados e onde poderemos utilizá-las.

Preparado para iniciar este estudo? Então, vamos lá!

4.1 Contadores binários assíncronos

Como mencionamos, para que possamos conversar sobre contadores, teremos que falar sobre dois novos tipos de *flip-flops*. Relembrando: *flip-flops* são componentes da lógica digital sequencial capazes de armazenar informações. Para que ocorra a carga do bit, os *flip-flops* são sensíveis às transições do sinal do *clock* (tanto de subida quanto de descida).

4.1.1 Flip-flop tipo “JK” e tipo “T”

Um *flip-flop* tipo “JK” lembra um pouco o *flip-flop* tipo “RS”, ou seja, os seus terminais desempenham as funcionalidades de “SET” e “RESET”. Porém não há a preocupação, como nos “RS”, de evitar que os dois terminais sejam ativados simultaneamente. A figura a seguir mostra a tabela-verdade e o diagrama esquemático do *flip-flop* tipo “JK”.

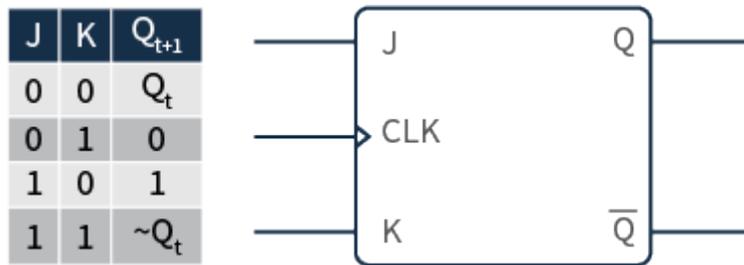


Figura 1 - Tabela-verdade e simbologia do flip-flop tipo “JK”. A saída “ Q_{t+1} ” reflete o momento futuro do valor armazenado em função do valor corrente no tempo “ t ” (valor “ Q_t ”).

Fonte: Elaborada pelo autor, 2019.

Como você pode observar na tabela-verdade apresentada na figura, quando os terminais “J” e “K” recebem, simultaneamente, o valor lógico “0”, o valor armazenado no *flip-flop* “JK” permanece inalterado mesmo no pulso do *clock*. Por sua vez, quando ativados isoladamente, os terminais “J” e “K” fazem com que o valor armazenado se torne “1” e “0”, respectivamente. A diferença do “JK” para o “RS” encontra-se na última linha. Enquanto que no “RS” deve-se evitar essa combinação com ambos os terminais “R” e “S” no nível lógico “1”, no *flip-flop* “JK” essa combinação faz com que o valor armazenado seja complementado, ou seja, caso esteja armazenado o valor “1”, o próximo valor será “0” e vice-versa (IDOETA; CAPUANO, 2012).

VOCÊ O CONHECE?



O *flip-flop* “JK” foi assim batizado em homenagem ao criador do circuito integrado: Jack Kilby, um engenheiro da Texas Instruments. Para conhecer sua história, você poderá acessar o artigo escrito por Dingman (2013), que está disponível em: <https://pcworld.com.br/a-lenda-de-jack-kilby-e-os-55-anos-do-circuito-integrado/>.

Uma variação do *flip-flop* “JK” consiste no *flip-flop* “T”. O tipo “T” (*Toggle* – alternância) pode ser implementado a partir de um “JK” com seus terminais interconectados. Assim, o *flip-flop* assume apenas duas possibilidades: ou mantém o valor armazenado ou o inverte. A figura a seguir mostra a implementação do *flip-flop* “T”, bem como sua tabela-verdade e um exemplo de uso. Confira!

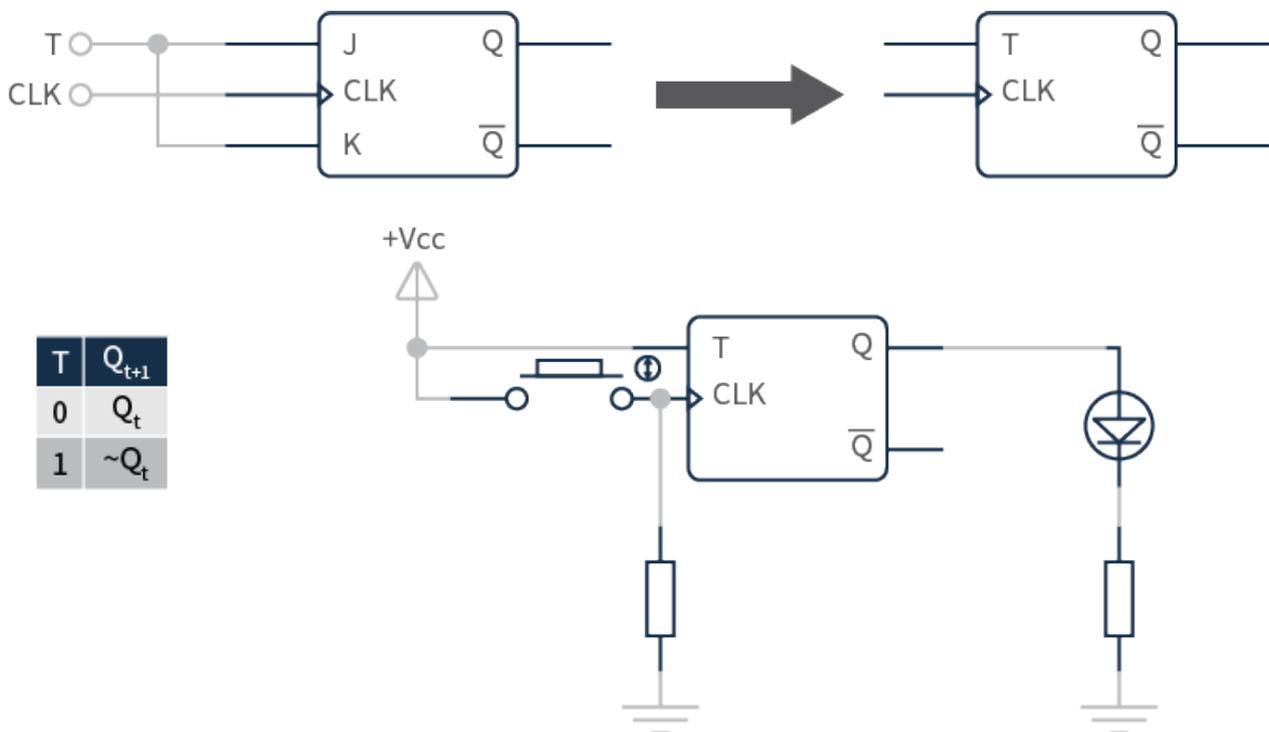


Figura 2 - Implementação de um flip-flop tipo “T” a partir de um tipo “JK” (parte superior), tabela-verdade e exemplo de aplicação do flip-flop tipo “T”.

Fonte: Elaborada pelo autor, 2019.

Na parte superior da figura, temos a forma de implementação de um *flip-flop* tipo “T” a partir de um “JK”. Podemos notar que os terminais “J” e “K” estão conectados entre si, fazendo com que o tipo “T” tenha apenas duas possibilidades, expressas na tabela-verdade inserida na figura: ou mantém-se o valor armazenado (quando “T” for associado ao valor lógico “0”) ou o valor armazenado é invertido (quando “T” receber o valor lógico “1”).

Para uma melhor abstração, temos, ainda na figura, uma aplicação envolvendo o *flip-flop* tipo “T”. Trata-se de uma chave “liga-desliga”. A cada toque no botão, um pulso é enviado ao *clock* do *flip-flop*, fazendo com que haja a alternância do estado do circuito controlado pelo *flip-flop* (no caso do exemplo, um LED). Assim, podemos ligar e desligar o circuito (por exemplo, um aparelho) ao pressionarmos um botão.

Mas como utilizar efetivamente o *flip-flop* “JK” ou “T” nos circuitos contadores? Conversaremos, agora, sobre esse assunto.

4.1.2 Contadores assíncronos

Como conversamos há pouco, os *flip-flops* tipo “JK” e tipo “T” têm a particularidade de permitir uma alternância do valor armazenado. Essa alternância será diretamente aproveitada nos contadores. Para começarmos a falar sobre a relação entre esses *flip-flops* e o contador assíncrono, podemos adiantar que uma contagem linear consiste na produção de valores consecutivos. Valores consecutivos se alternam entre positivos e negativos.

Em representação binária, qual é o último bit (o menos significativo – último bit à direita) quando o número é par? E quando o número é ímpar? Já podemos começar a notar essa alternância: o último bit varia entre “0” e “1” para representar os valores pares e ímpares, respectivamente. Já começou a imaginar como fica a implementação?

Podemos iniciar nossa implementação envolvendo um *flip-flop* “JK” ou “T” no bit menos significativo do contador. A cada pulso de *clock* (que poderá ser, por exemplo, a cada pessoa que passe em uma catraca), o valor é alternado, expressando os pares e ímpares. Mas como ficariam os demais bits da palavra sob contagem? Para

melhor abstrairmos, vamos analisar a tabela-verdade expressa na figura a seguir. Na mesma figura, perceba, encontra-se também a implementação de um contador para uma palavra de três bits, ou seja, realiza a contagem do valor $000_{(2)}$ ($0_{(10)}$) até $111_{(2)}$ ($7_{(10)}$).

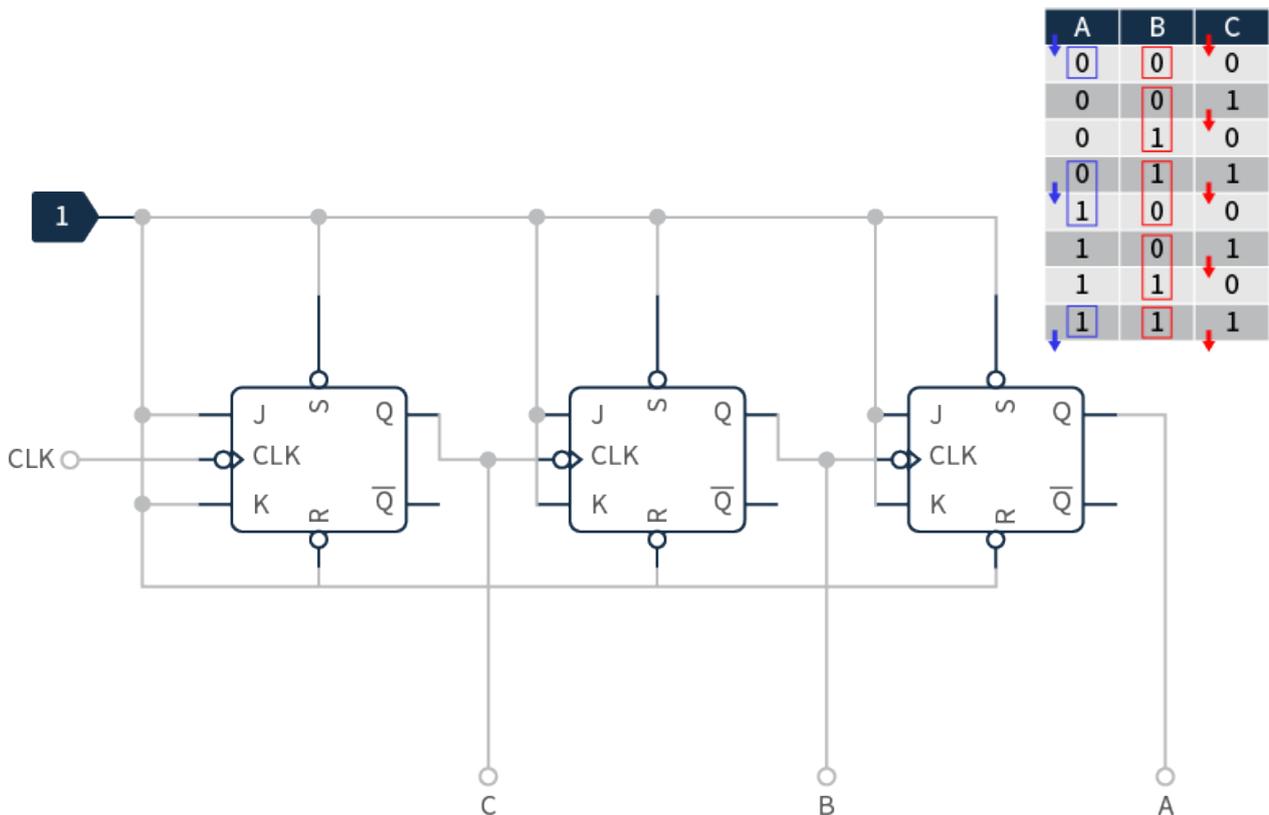


Figura 3 - Tabela de transições dos valores em um contador binário assíncrono e a respectiva implementação utilizando flip-flops “JK”.

Fonte: Elaborado pelo autor, 2019.

Na figura, percebemos que a mudança de um bit da palavra binária se dá apenas quando houver a transição de descida do bit de ordem imediatamente inferior. Por exemplo, o bit “B” apenas varia quando o bit “C” passa do nível lógico “1” para o nível lógico “0”. O mesmo se aplica ao bit “A”, que depende da transição de descida do bit “B”. Esse é o motivo de utilizarmos *flip-flops* com a lógica negativa no sinal do *clock* (o símbolo de inversão junto à entrada do *clock* sinaliza o uso da lógica negativa). Assim, os *flip-flops* apenas são ativados na transição de descida do sinal do *clock*. Quando ativado, o *flip-flop* “JK” (poderia ter sido usado o tipo “T”) alterna seu valor armazenado devido ao fato de que seus terminais “J” e “K” estão conectados ao nível lógico “1”. No nível lógico “1”, estão também conectados todos os terminais “S” (set) e “R” (reset) dos *flip-flops*. Tais terminais, quando ativados, fazem com que o valor armazenado assuma o valor “1” ou “0”, respectivamente, independentemente do sinal do *clock* ou dos terminais “J” e “K”. Os terminais “S” e “R”, nesse caso, também operam na lógica negativa, ou seja, são ativados quando alimentados com o nível lógico “0”.

VOCÊ QUER VER?



Comercialmente, existem diversas implementações de contadores binários por meio de circuitos integrados. Dentro das implementações, podemos encontrar aqueles que fazem tanto a contagem crescente quanto a decrescente, em que a seleção da funcionalidade é feita por meio de um pino de controle. Para saber um pouco sobre uma implementação comercial de contador, você poderá assistir ao vídeo do canal Eletrônica Fácil (2015), disponível em: <https://www.youtube.com/watch?v=v2AW8mwxLSg>.

Você deve estar se perguntando: “Mas e para realizar contagens decrescentes?”. Bem, para que a contagem seja realizada decrescentemente, basta realizar uma das seguintes opções:

Coletar os valores das saídas negadas	Os valores decrescentes correspondem aos bits complementados dos valores crescentes.
Usar a lógica positiva no sinal do <i>clock</i>	Pode-se, para realizar a contagem decrescente, utilizar a lógica positiva junto ao sinal do <i>clock</i> . Dessa forma, as alterações dos valores armazenados nos <i>flip-flops</i> ocorrerão na transição de subida do sinal de <i>clock</i> . Nesse caso, deve-se manter as saídas da palavra sob contagem atreladas aos pinos “Q” dos <i>flip-flops</i> .

Mas e se uma aplicação, como os segundos de um relógio, requerer uma contagem que não chega até a capacidade máxima de contagem do dispositivo? Como proceder para limitar uma contagem? Veremos isso a seguir.

VAMOS PRATICAR?



Um gerador de onda triangular pode ser obtido por intermédio da ligação de um conversor D/A (digital analógico) na saída de um contador binário crescente-decrescente. Implemente um contador binário que alterne, automaticamente, o seu modo de contagem, ou seja, em um certo momento, ele se comporta como um contador crescente (variando os valores, por exemplo, de 0 a 7) e, ao atingir o topo da contagem, ele passa a ser decrescente (variando os seus valores de 7 a 0). Essa alternância deve ser automática quando a contagem atingir o seu limite.

4.1.3 Limitação de contagem nos contadores assíncronos

Vamos supor que precisamos implementar o campo das unidades e das dezenas de segundos de um relógio. Sabemos que as unidades dos segundos variam de 0 até 9 e, por sua vez, o campo das dezenas dos segundos varia de 0 até 5. Inicialmente, podemos observar que, para expressar valores de 0 até 9, precisaremos de 4 bits.

Porém, com os 4 bits, poderemos ter uma contagem de 0 até 15 ($2^4 - 1$). Assim, não podemos deixar com que a contagem chegue até o valor máximo, ou seja, devemos reiniciá-la antes. Mas como fazer isso? Nesse ponto, surge uma das aplicações dos sinais de “SET” e “RESET” dos *flip-flops*. Quando a contagem alcançar o primeiro valor indesejável (no caso, o valor 10), deve-se ativar os terminais “RESET” de todos os *flip-flops*, fazendo com que o valor armazenado volte a valer 0.

A figura a seguir mostra como limitar as contagens e, ainda, como estabelecer a relação entre a contagem das unidades e as dezenas dos segundos de um relógio.

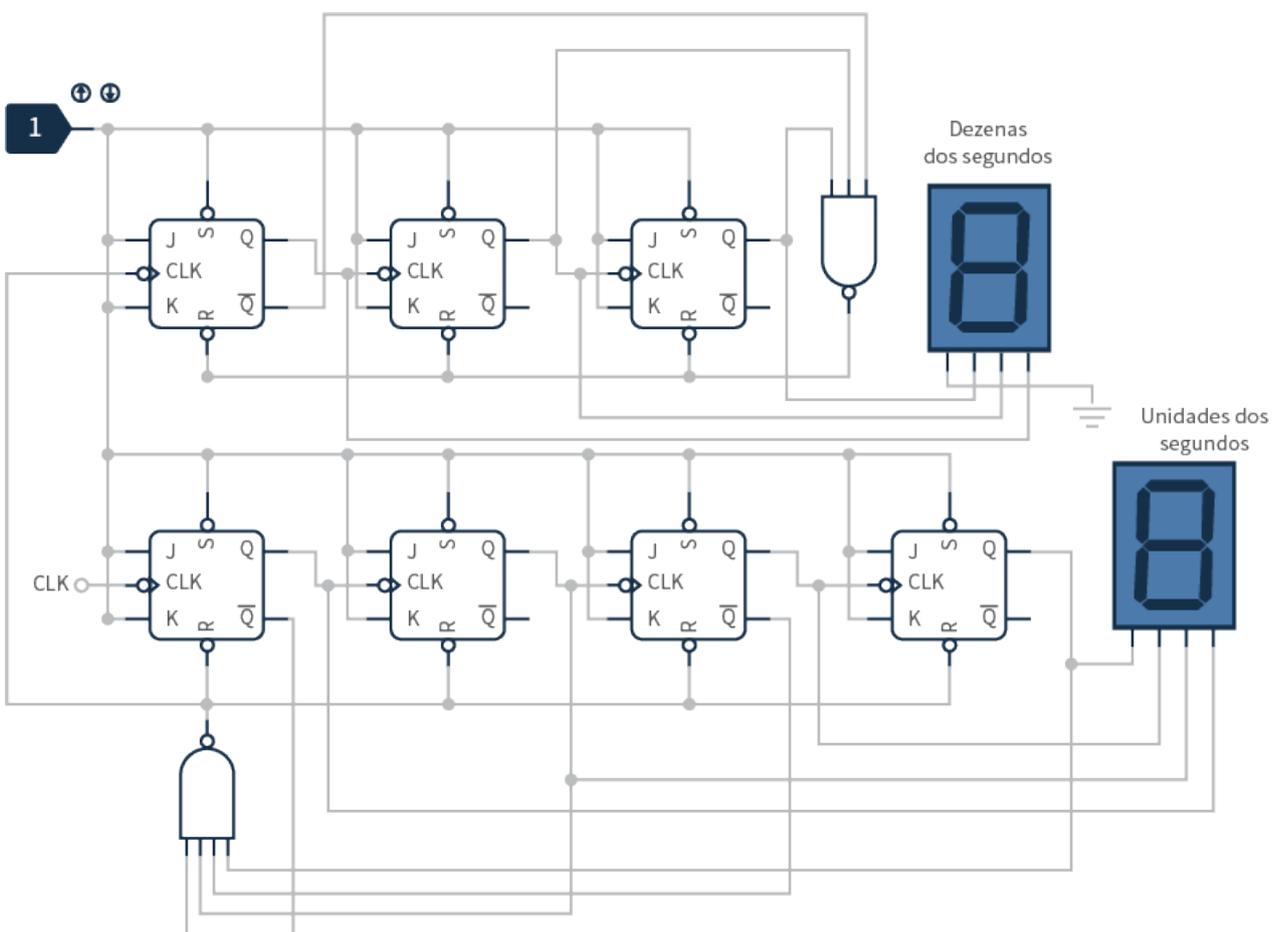


Figura 4 - Exemplificação da limitação de contagem dos contadores assíncronos na geração das dezenas e das unidades dos segundos em um relógio.

Fonte: Elaborada pelo autor, 2019.

Na figura, perceba que temos dois circuitos contadores assíncronos: um relativo à contagem de 0 a 9 (unidade dos segundos) e o outro responsável pela contagem de 0 a 5 (dezenas dos segundos). O *clock* externo, atribuído ao contador das unidades de segundos deve ser de 1 pulso por segundo (1 Hz). Por sua vez, o *clock* do contador das dezenas de segundos é o próprio sinal de “reset” do contador das unidades. A cada reciclagem da contagem das unidades, incrementa-se a dezena dos segundos.

Podemos notar, ainda em relação à figura anterior, que a limitação da contagem é realizada por intermédio de uma porta “NAND” (devido à lógica negativa do sinal de “reset” dos *flip-flops*, quando atingir um valor inválido, por exemplo, $6_{(10)}$ ($110_{(2)}$) no contador das dezenas. A porta “NAND” resultará no nível lógico “0”, ativando os terminais de “reset” e, conseqüentemente, zerando o valor armazenado no contador.

O processo de limitação da contagem pode ser aplicado para se obter divisão de frequência. No caso das unidades dos segundos, para que o sinal de “reset” seja emitido, são necessários 10 pulsos de *clock*. Dessa forma, temos um divisor de frequência por 10, também denominado como “divisor de década”.

Os contadores assíncronos são assim chamados porque cada *flip-flop* atua em uma frequência distinta. O primeiro *flip-flop* atua na frequência “F”, o segundo já na frequência “F/2”, o terceiro em “F/4”, e assim por diante. Uma limitação inerente aos contadores assíncronos consiste no tempo para que, dado um pulso de *clock*, tenhamos o valor de forma estável. Para a estabilidade do valor, os pulsos de *clock* devem ser propagados de *flip-flop* em *flip-flop*, causando lentidão na operação. Além disso, caso façamos a coleta do valor antes que todos os sinais de *clock* tenham sido propagados, podemos coletar um valor errôneo, representando, assim, um ruído do valor.

Para resolvermos isso, podemos implementar um contador denominado “**contador síncrono**”, o qual abordaremos a seguir. Continue acompanhando!

VAMOS PRATICAR?



Mencionamos que podemos limitar a contagem de um contador binário. Existe um divisor de frequência denominado “divisor de década”, que limita a contagem no valor 10. Implemente um contador com essa particularidade.

4.2 Contadores binários síncronos

Como mencionado, um contador assíncrono apresenta, como ponto negativo, a possibilidade de um valor errôneo de contagem durante a transição do sinal do *clock* entre os seus *flip-flops*. Para resolver esse problema, podemos fazer uso dos contadores binários síncronos. Saiba mais sobre eles a seguir!

4.2.1 Tabelas de transições de valores dos *flip-flops* do tipo “JK”

Os contadores síncronos são assim denominados como tal pelo fato de que todos os *flip-flops* integrantes recebem o sinal de *clock* ao mesmo tempo (VAHID; LASCHUK, 2008). Dessa forma, as suas ativações ocorrem simultaneamente. Mas, devido à ativação ao mesmo tempo, como definir o que cada *flip-flop* deve fazer? Para isso, cada entrada “J” e “K” deve ser associada a um circuito combinacional que fornece um valor de acordo com o estado atual de contagem. Isso possibilita, também, a geração de contagens não lineares, ou seja, os valores de contagem não necessitam ser sequenciais.

VOCÊ QUER LER?



Como qualquer outro sistema lógico digital (tanto combinacional quanto sequencial), os contadores síncronos também poderão ser implementados utilizando uma HDL. Para conhecer um pouco mais sobre esse universo da descrição de hardware usando Verilog, assim como verificar como implementar um contador usando essa técnica, você poderá acessar o artigo de Alexandro, Delgado e Ogg (2011), disponível em: <https://www.cin.ufpe.br/~voo/sd/Aula6.Verilog.pdf>.

Mas em que embasar a geração dos valores dos circuitos combinacionais cujas saídas serão atribuídas aos terminais “J” e “K” dos *flip-flops*? Para isso, teremos que olhar as transições de cada bit da palavra referente à contagem entre cada linha subsequente. Antes de exemplificarmos esse processo, apresentaremos, na figura a seguir, as possibilidades de transições e os valores associados às entradas “J” e “K”.

0 → 1	1 → 0
J = 1	J = X
K = X	K = 1
0 → 0	1 → 1
J = 0	J = X
K = X	K = 0

Figura 5 - Valores das entradas “J” e “K” associados às transições relativas à contagem demandada. As transições devem ser observadas em cada bit, na sequência de linha por linha.

Fonte: Elaborada pelo autor, 2019.

Veremos, agora, como efetivamente implementar os contadores síncronos.

4.2.2 Implementação dos contadores binários síncronos

Para que possamos conversar sobre como podemos implementar um contador binário síncrono, vamos supor que queremos construir um contador cuja contagem é: $0 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 5$. Como o valor máximo da contagem é 5, necessitamos de 3 bits; conseqüentemente, 3 *flip-flops*. A figura a seguir ilustra a tabela-verdade, contemplando as transições dos bits da palavra sob contagem, assim como os valores associados aos terminais “J” e “K” dos *flip-flops* constituintes do contador.

	A	B	C	Ja	Ka	Jb	Kb	Jc	Kc
0	0	0	0	0	X	1	X	1	X
3	0	1	1	0	X	X	0	X	1
2	0	1	0	1	X	X	1	0	X
4	1	0	0	X	0	0	X	1	X
5	1	0	1	X	1	0	X	X	1

Figura 6 - Tabela-verdade referente à contagem “0 → 3 → 2 → 4 → 5”, com os valores de “Ja”, “Ka”, “Jb” e “Kb”, associados às transições dos bits “A” e “B”, em destaque.

Fonte: Elaborada pelo autor, 2019.

Na figura, temos a formação dos valores relativos aos terminais “Ja”, “Ka”, “Jb”, “Kb”, “Jc” e “Kc”, relativos às transições dos bits “A”, “B” e “C” que compõem a palavra sob contagem. Por exemplo, na transição do valor da contagem 0 para 3, temos no bit “A” a transição “0 → 0”. Assim, temos os valores “Ja” e “Ka”, relativos a essa transição, valendo “0” e “X”, respectivamente. Por sua vez, por exemplo, na transição dos valores de 2 para 4, temos, em “B”, a transição do valor lógico “1” para o valor lógico “0”. Dessa forma, “Jb” e “Kb”, para a referida transição, devem corresponder a “X” e “1”, respectivamente.

Extraindo todas as expressões e realizando o processo de simplificação, temos as seguintes expressões booleanas:

- $J_a = \sim A \cdot B \cdot \sim C$
- $K_a = A \cdot \sim B \cdot C$
- $J_b = \sim A \cdot \sim C$
- $K_b = \sim A \cdot \sim C$
- $J_c = \sim B \cdot \sim C$
- $K_c = A \oplus B$

Essas expressões poderão ser observadas na figura a seguir, que ilustra a implementação do contador síncrono, que realiza a sequência “0 → 3 → 2 → 4 → 5”.

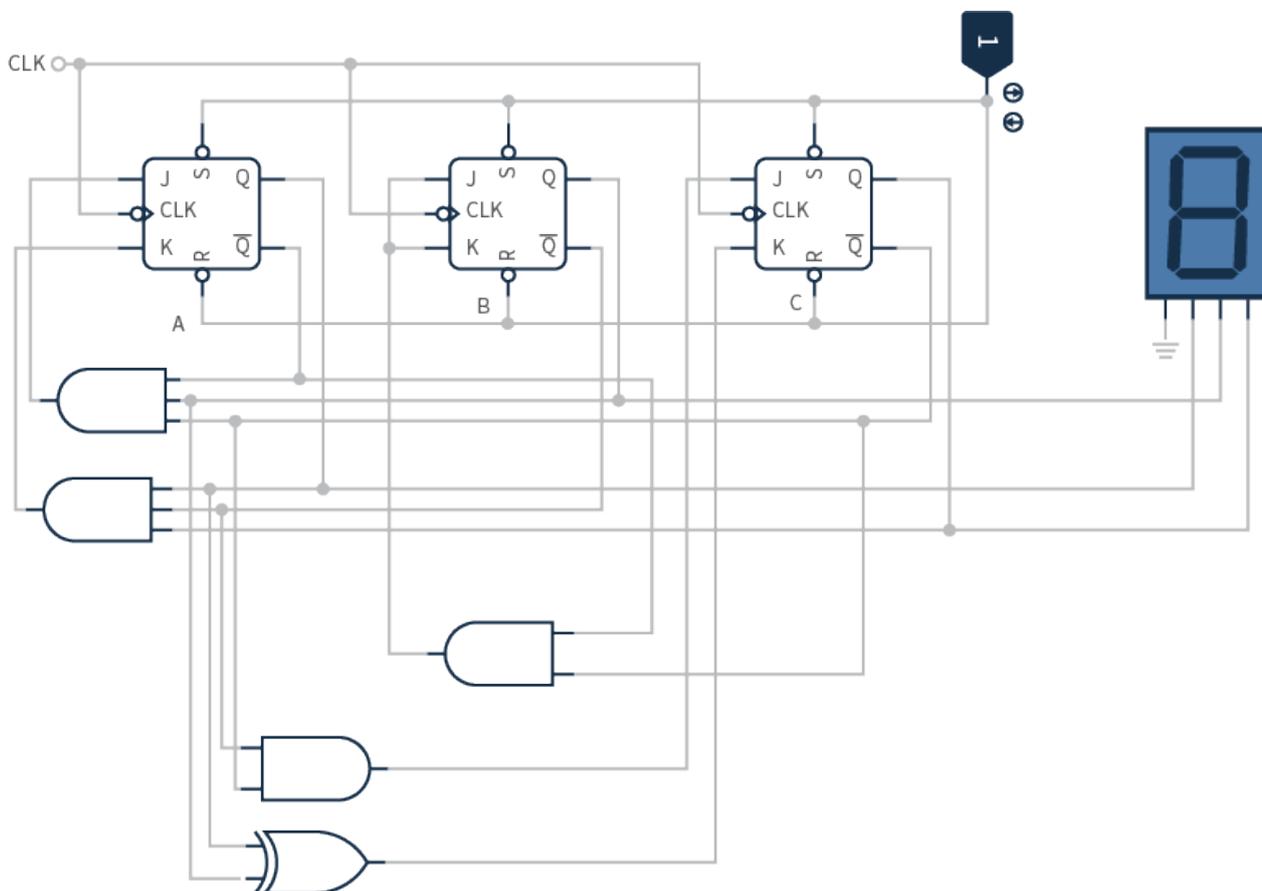


Figura 7 - Implementação do contador síncrono para a exibição, no display, da sequência “0 → 3 → 2 → 4 → 5”.
 Nota-se que, pelo fato do sinal do clock chegar simultaneamente a todos os flip-flops, a ordem dos bits de saída é irrelevante.

Fonte: Elaborada pelo autor. 2019.

Existem alguns números que não foram contemplados na sequência. No caso do exemplo anterior, temos os valores 1, 6 e 7. Para evitar transtornos em possíveis desvios de contagem (causados, por exemplo, por distúrbios externos ou fadiga dos próprios componentes utilizados), sugerimos adotar uma das três possibilidades relacionadas a seguir.

Voltar imediatamente a um valor da contagem de forma assíncrona. Nesse caso, aciona-se imediatamente os terminais “PRESET” (ou “set”) e “CLEAR” (ou “reset”) dos flip-flops.

Criar transições entre cada elemento fora da contagem para um valor da contagem. Por exemplo, no nosso caso: “1 → 0”, “6 → 0” e “7 → 0”. Nessa possibilidade, a volta à contagem determinada é realizada de forma síncrona (espera-se o pulso do clock para voltar à sequência correta).

Criar uma sequência à parte envolvendo os elementos fora da contagem para que, no final, convirja para algum valor da contagem estabelecida. No nosso caso, teríamos, por exemplo: “7 → 6 → 1 → 0”. A volta à sequência correta, nesse caso, é também realizada de forma síncrona.

Existem outras aplicações para utilizarmos os contadores síncronos? Veremos, a seguir, uma aplicação direta. Trata-se da manipulação de **máquinas de estado**.

VAMOS PRATICAR?



Sabemos que um contador síncrono permite realizar uma contagem com uma sequência não linear. Implemente, no caso, um contador para realizar a seguinte sequência: $0 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 0 \rightarrow 3 \dots$

4.3 Máquinas de estados

De acordo com Tocci, Widmer e Moss (2018), máquinas de estados são circuitos com a capacidade de determinar um próximo estado em função do estado corrente e das variáveis de entrada. Mas o que são estados? Estados correspondem às funcionalidades implementadas. Se pensarmos no mundo da programação, podemos, inclusive, fazer uma analogia com uma função. Assim, temos funções em execução.

Mas como passar de uma função para outra, ou seja, como passar de um estado para o outro? Saiba que a passagem de estados é decorrente da ação de eventos. Um evento pode ser, por exemplo, a variação do valor de uma variável ou resposta de um sensor. Com essa breve conceituação, vamos ver na prática o que é uma máquina de estados e como podemos implementá-la. Continue acompanhando!

4.3.1 Implementação de Máquinas de estados

Para que possamos dialogar sobre como implementar uma máquina de estados, vamos imaginar uma máquina automática para a venda de refrigerantes. Como estados, podemos mencionar:

Estado Inicial	Representa o estado de “standby”, ou seja, a máquina fica aguardando uma ação. A única ação plausível de acontecimento para sair do estado de “standby” poderá ser de inserção de dinheiro.
Inserindo Dinheiro	Esse estado representa a ação de contabilização da quantia depositadas na máquina de refrigerantes. A cada inserção, ocorre a somatória do valor inserido para a totalização do saldo. Para sair desse estado, o usuário poderá pressionar o botão de cancelamento da compra ou o botão para a seleção do produto.
Liberando Produto	Denota o procedimento para a liberação do produto selecionado, atuando sobre as partes mecânicas da máquina. Enquanto o produto ainda estiver em liberação, a máquina permanecerá nesse estado. Para simplificar nosso exemplo, não está sendo testado se o saldo inserido é suficiente para a compra do produto. A máquina finalizará esse estado somente quando a liberação do produto for concluída.
Devolvendo Troco	O estado “Devolvendo Troco” envolve as ações para a devolução do saldo ainda disponível após a compra ou após o usuário ter solicitado o cancelamento da compra. A máquina permanecerá nesse estado enquanto ainda houver saldo a ser liberado. Após a liberação total da quantia devida ao usuário, a máquina retornará ao estado de “standby”.

Como podemos notar, os estados correspondem às ações que poderão ser realizadas pelo sistema.



Um projetista recebeu a incumbência de implementar um circuito para detectar uma sequência de dados recebidos por uma rede de computadores. Essa sequência é relacionada ao protocolo utilizado, de forma a identificar sinais de controle. Verificando a sequência, ele logo percebeu que a melhor forma de se implementar é utilizando uma máquina de estados. Porém, ele sabe que existem dois modelos para se implementar: a máquina de Moore e a máquina de Mealy. Pesquisando, ele tomou conhecimento que, nas máquinas de Moore, as saídas são decorrentes apenas do estado atual, e, nas máquinas de Mealy, as saídas são decorrentes tanto do estado atual quanto dos valores assumidas pelas entradas.

Em relação ao recebimento de informações pela rede, ele ficou com receio de que as variações das informações das entradas pudessem afetar o valor da saída de forma assíncrona, ou seja, caso as entradas fossem modificadas, poderiam ser provocados ruídos nos valores obtidos na saída do circuito. Porém, por outro lado, as respostas geradas por uma máquina de Moore são mais lentas em relação à de Mealy, que proporciona, também, uma implementação menos complexa e, conseqüentemente, mais barata em relação à de Moore. Porém, após refletir sobre a questão, o projetista resolveu adotar o modelo de Moore em função da questão relacionada ao ruído que poderia aparecer na máquina de Mealy para esse tipo de aplicação.

A passagem de um estado para o outro ocorre na ocorrência de eventos. Como eventos, ainda utilizando a máquina automática para venda de refrigerantes, podemos citar algumas ações. Confira!

Dinheiro Inserido	Corresponde à ação de inserção de dinheiro e a respectiva passagem da cédula ou moeda no sensor que identifica o valor depositado.
Botão Selecciona Produto Pressionado	Ação decorrente do pressionamento, pelo usuário, do botão para a seleção de um produto.
Botão Cancelamento Pressionado	Denota a ação de pressionamento, pelo usuário, do botão para o cancelamento da operação de compra de refrigerantes.
Produto em Liberação	Evento que indica que o produto ainda não completou o seu curso para a liberação. Indicativo fornecido, por exemplo, do sensor de passagem do produto.
Produto Liberado	Evento que indica que o produto já foi liberado.
Ainda tem Saldo a Devolver	Evento associado ao saldo restante a ser devolvido ao usuário.
Saldo Zerado	Evento disparado quando o saldo estiver zerado.

Nada	Por se tratar de uma máquina síncrona, ou seja, sincronizada pelos pulsos de <i>clock</i> , o evento “nada” é associado quando nenhuma ação externa foi realizada momento do pulso do <i>clock</i> .
------	--

Na máquina de estados, os estados são representados por nós (elipses ou circunferências) e os eventos são representados por arcos ou *links* (setas), interligando os estados. Juntando os estados e os eventos mencionados, podemos estabelecer a máquina completa, conforme ilustra a figura a seguir:

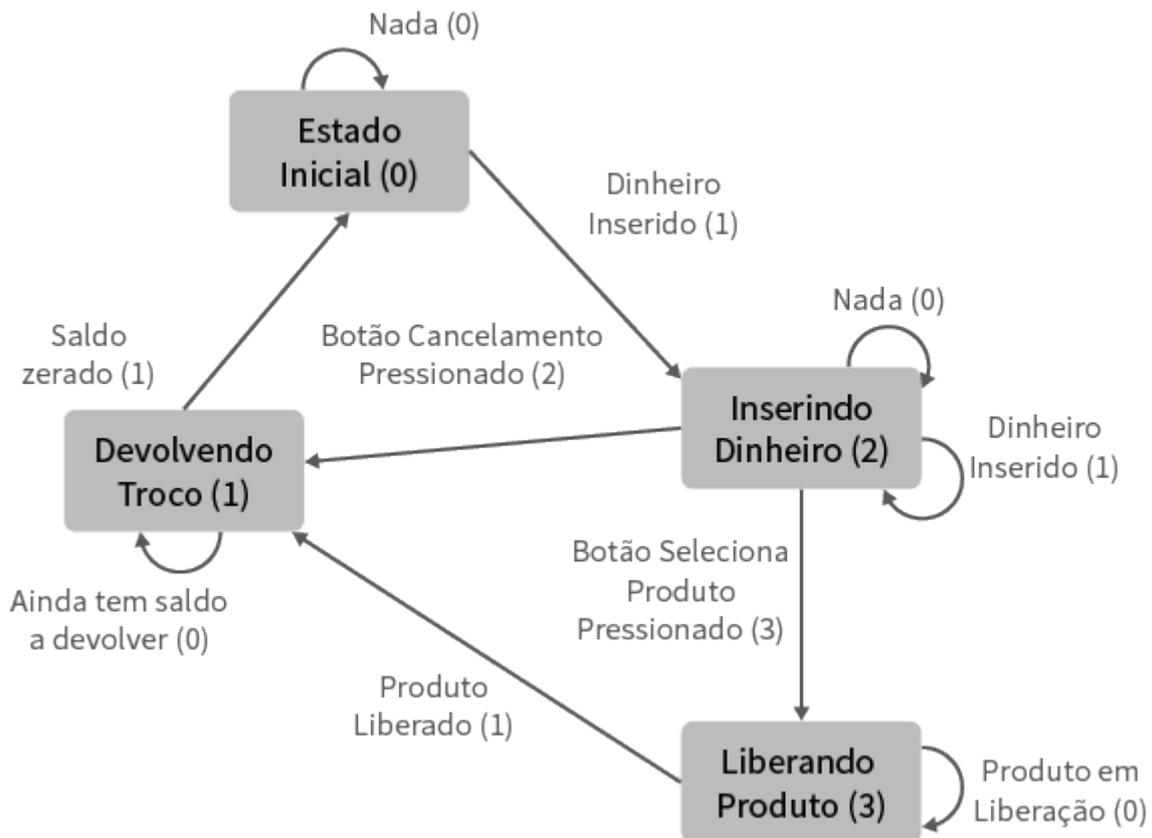


Figura 8 - Exemplo de uma máquina de estados para a venda automática de refrigerantes. Os “nós” representam os estados e os arcos representam os eventos.

Fonte: Elaborada pelo autor, 2019.

Como ilustrado na figura, podemos notar que os eventos interligam os estados. Os eventos apenas atuam sobre o estado corrente da máquina de estados caso estiverem associados ao próprio estado corrente. Por exemplo, caso o usuário pressione o botão de seleção de produto enquanto a máquina estiver no estado inicial, nada acontecerá, pois o referido evento não está associado ao estado inicial.

Ainda em relação à figura anterior, podemos notar que tanto os estados quanto os eventos encontram-se numerados. A numeração dos eventos é reiniciada dentro de cada estado. A numeração se faz presente pelo fato de que, com ela, montaremos a tabela de transições.

Você percebe alguma aproximação com algum tópico já conversado neste capítulo? Se olhar atentamente, na máquina de estado, você verá uma sequência de valores, por exemplo, do estado “0” para o estado “3”. Dessa forma, podemos já identificar a sequência “0 → 3”. Ficou mais claro que estamos tratando de um circuito contador binário síncrono, certo? Mas como podemos montar a tabela completa contemplando os estados e os eventos?

Nesse caso, poderemos juntar a identificação dos estados com a identificação dos eventos, formando, portanto, uma palavra que corresponderá aos bits de entrada da tabela-verdade do contador síncrono. Mas quais serão os bits de saída? Para a questão da máquina de refrigerantes, desejamos obter, como saída, o próprio estado corrente.

Assim, construiremos uma tabela-verdade tendo, como entradas, os bits “E₁” e “E₀” para denotar os estados, e também os bits “V₁” e “V₀” para representar os eventos. Tratando de um contador binário síncrono, as saídas da tabela-verdade corresponderão aos bits dos terminais “J” e “K” dos *flip-flops* “JK” associados aos bits dos estados. Podemos, então, notar que o estado atual influenciará na obtenção do novo estado.

VOCÊ SABIA?



Você sabia que máquinas de estado não são implementadas somente a nível de hardware? Os sistemas computacionais baseados em software também podem ser modelados utilizando-se as máquinas de estado. Para saber mais, acesso o artigo escrito por Bertoleti (2015), disponível em: <https://www.embarcados.com.br/maquina-de-estado/>.

A figura a seguir contempla a tabela-verdade usada para a obtenção das transições relativas à máquina automática para venda de refrigerantes, realçando as transições presentes na máquina de estado.

E1	E0	V1	V0	JE1	KE1	JE0	KE0
0	0	0	0	0	X	0	X
0	0	0	1	1	X	0	X
0	1	0	0	0	X	X	0
0	1	0	1	0	X	X	1
1	0	0	0	X	0	0	X
1	0	0	1	X	0	0	X
1	0	1	0	X	1	1	X
1	0	1	1	X	0	1	X
1	1	0	0	X	0	X	0
1	1	0	1	X	1	X	0

Figura 9 - Tabela-verdade relativa às transições entre os estados para a obtenção dos valores dos terminais dos flip-flops representativos do estado corrente. As cores dos valores de cada “J” e “K” correspondem às cores das transições.

Fonte: Elaborada pelo autor, 2019.

Podemos notar, pela figura, que os valores relativos aos terminais “J” e “K” são obtidos observando-se as transições ocasionadas pelo estado atual, em conjunto com os eventos acontecidos. Por exemplo, a primeira linha da tabela-verdade refere-se ao “estado 0/evento 0”. Para essa combinação de estado/evento, tem-se a

transição apontando para a própria linha. Assim, tanto “E1” quanto “E2” possuem a transição “0 → 0”, resultando no valor lógico “0” para “JE1” e “JE0” e no valor lógico “X” para “KE1” e “KE0”.

Por sua vez, a segunda linha (estado 0 / evento 1) tem sua transição para a quinta linha (estado 2). Assim, “E1” tem uma transição “0 → 1” e, “E0”, uma transição “0 → 0”, resultando, portanto, nos valores “1”, “X”, “0”, “X” para os terminais “JE1”, “KE1”, “JE0” e “KE0”, respectivamente.

Realizando a extração das expressões a partir da tabela-verdade contida na figura, temos:

- $JE1 = \sim E0 . \sim V1 . V0$
- $KE1 = E0 . \sim V1 . V0 + E1 . \sim E0 . \sim V0$
- $JE0 = E1 . \sim E0 . V1$
- $KE0 = \sim E1 . \sim V1 . V0$

Notamos que necessitamos extrair as expressões apenas dos *flip-flops* relativos à geração do estado corrente (bits “E1” e “E0”), devido ao fato de que os bits “V1” e “V0” são variáveis de entrada, provenientes dos sensores da máquina automática de venda de refrigerantes. Assim, os sensores são multiplexados, de forma que apenas aqueles relacionados ao estado atual possam ter seus valores manipulados pelo contador binário síncrono.

VAMOS PRATICAR?



Mencionamos que uma máquina de estado tem por objetivo representar uma sequência de estados. Por sua vez, um contador também visa estabelecer uma sequência de estados que são os próprios valores sob contagem. Assim, usando máquinas de estados, implemente um contador Gray de 2 bits. Lembrando que a contagem Gray segue a seguinte sequência: 00, 01, 11 e 10.

Síntese

Chegamos ao fim desta unidade. Neste nosso encontro, pudemos dialogar sobre circuitos com a propriedade de realizar contagens, os chamados “**circuitos contadores binários**”. Conceituamos e analisamos as diferenças dos contadores binários assíncronos e os contadores binários síncronos. Vimos, ainda, uma aplicação direta do contador síncrono, que consiste na máquina de estados.

Dessa forma, você, com os conhecimentos aqui adquiridos, poderá implementar sistemas lógicos digitais para a manipulação de palavras binárias, seja como elemento principal de processamento, seja como circuitos auxiliares ou de interfaceamento lógico. Os pontos aqui abordados também servirão de base para um entendimento de como funciona um sistema lógico digital, ou, ainda, servir como bagagem para a tomada de decisões nos momentos de sua implementação, quer usando circuitos integrados ou quer usando HDL.

Nesta unidade, você teve a oportunidade de:

- ter contato com a teoria relacionada aos contadores binários;
- saber diferenciar e aplicar soluções baseadas em contadores binários assíncronos e contadores binários síncronos;
- desenvolver máquinas de estado;
- implementar sistemas lógicos digitais que envolvam contadores binários.

Bibliografia

ALEXANDRO, D.; DELGADO, R.; OGG, V. **Sistemas Digitais: Linguagem Verilog**. Recife: Universidade Federal de Pernambuco, Centro de Informática, 2011. Disponível em: <https://www.cin.ufpe.br/~voo/sd/Aula6.Verilog.pdf>. Acesso em: 23/09/2019.

BERTOLETI, P. **Máquina de Estado**, 07 ago. 2015. Disponível em: <https://www.embarcados.com.br/maquina-de-estado/>. Acesso em: 23/09/2019.

DINGMAN, H. **A lenda de Jack Kilby e os 55 anos do circuito integrado**, 13 set. 2013. Disponível em: <https://pcworld.com.br/a-lenda-de-jack-kilby-e-os-55-anos-do-circuito-integrado/>. Acesso em: 23/09/2019.

ELETRÔNICA FÁCIL. **Eletrônica Digital 33** - Contador crescente e decrescente usando o CI 4510, 12 maio 2015. Disponível em: <https://www.youtube.com/watch?v=v2AW8mwxLSg>. Acesso em: 23/09/2019.

IDOETA, I. V.; CAPUANO, F. G. **Elementos de Eletrônica Digital**. 41. ed. São Paulo: Érica, 2012.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas Digitais: Princípios e Aplicações**. 12. ed. São Paulo: Pearson Education do Brasil, 2018.

VAHID, F.; LASCHUK, A. **Sistemas Digitais: Projeto, otimização e HDLs**. Porto Alegre: Bookman, 2008.