6. CIRCUITOS LÓGICOS SEQUENCIAIS

Os circuitos até aqui estudados e denominados por circuitos combinatórios caracterizam-se pelo valor lógico presente na saída ser unicamente determinado pelos valores lógicos presentes nas entradas em cada momento. Iniciaremos agora o estudo de circuitos denominados *circuitos sequenciais*, que como o nome indica, são circuitos que em função da ***sequência*** de valores que as variáveis de entrada foram tomando ao longo do tempo adquirem diferentes comportamentos (diferentes valores nas saídas) para uma mesma combinação das variáveis de entrada, ou seja, o circuito reage às entradas em função de acontecimentos passados.

Tal comportamento implica a existência de memória pois o sistema guarda informação de acontecimentos passados.

6.1 Modelo Geral de um Circuito Sequencial

Os circuitos sequenciais são constituídos por circuitos combinatórios e elementos de memória, cujo modelo geral é apresentado na Figura 6-1.

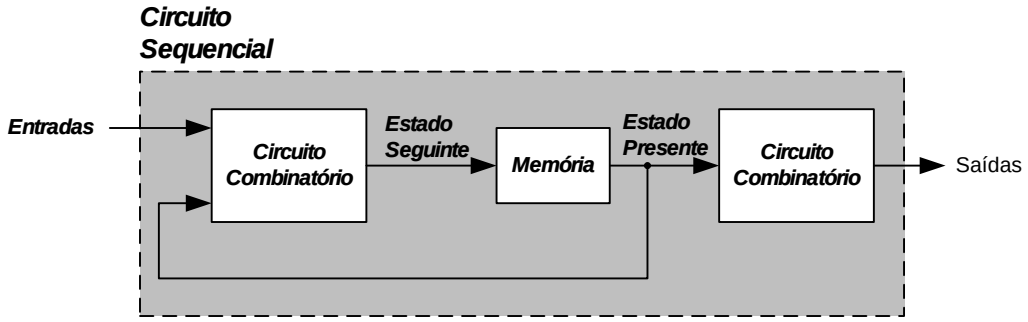


Figura 6-1

Este modelo é conhecido por modelo *Moore*. A informação binária guardada no elemento de memória estabelece o estado presente do sistema. Na Figura 6-1, os circuitos combinatórios desempenham duas funções: o primeiro gera o estado seguinte e o segundo as saídas do sistema. Na Figura 6-2 é apresentado um outro diagrama de blocos do modelo geral, conhecido como modelo *Mealy*, em que a função geradora de saídas está dividida em saídas que dependem exclusivamente do estado presente e saídas que dependem do estado corrente e das variáveis de entrada.

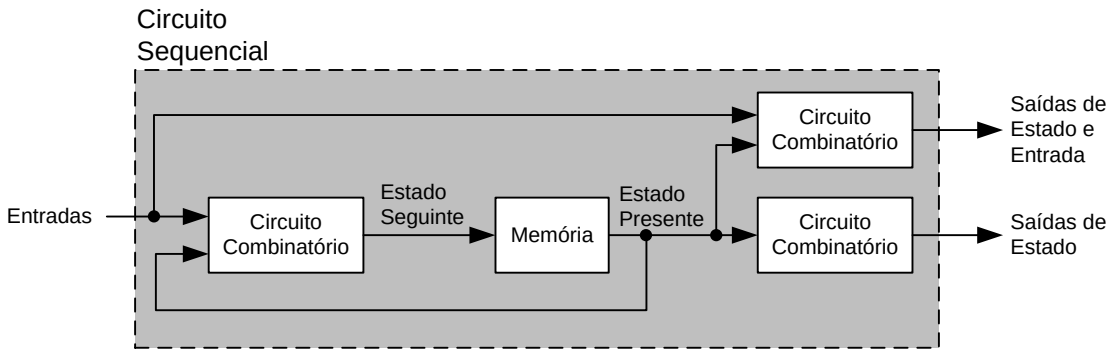


Figura 6-2

Como se pode observar na Figura 6-2, as saídas do sistema e o estado seguinte dependem do estado presente e do valor lógico das entradas.
Os circuitos sequenciais são classificados em dois tipos: os circuitos sequenciais *assíncronos* e os sequenciais *síncronos*. Esta classificação é atribuída, função do instante temporal em que as entradas são observadas e o momento em que o estado do circuito se altera. Nos *circuitos sequenciais assíncronos*, o estado muda no momento em que as entradas mudam, nos *circuitos sequenciais síncronos* a observação das entradas e subsequente evolução de estado, realiza-se em sincronismo com a transição de um sinal de entrada específico denominado por *clock* (relógio). A designação de *clock* advém do facto de este sinal ser normalmente periódico. Como podemos concluir existe um novo factor que é o tempo, sendo por isso necessário analisar o comportamento dos componentes que constituem os sistemas digitais no domínio do tempo.

6.2 Comportamento temporal dos circuitos sequenciais

A comutação electrónica implica o movimento de cargas e como em qualquer outro fenómeno físico tem implícita a inércia do repouso e do movimento das cargas. Por esta razão o instante de tempo em que o valor lógico à entrada de uma porta lógica se modifica, não é o mesmo em que a saída dessa mesma porta se altera. Diz-se que existe um tempo de propagação da entrada para a saída. Por outro lado, se uma entrada fica activa durante um tempo muito pequeno, a energia transmitida à porta por este sinal pode não ser suficiente para que o acontecimento se reflicta na saída. Por esta razão, no projecto de circuitos sequenciais os tempos de propagação têm que ser levados em linha de conta pelo projectista do sistema, uma vez que a geração de estado seguinte é feita com base no estado presente.
Se admitirmos por exemplo que o circuito combinatório que gera cada um dos bits do vector de estado seguinte, não é simétrico relativamente ao número de portas que o constitui, é natural que os vários bits do vector estado seguinte, tenham tempos de propagação diferentes, o que vai implicar que o vector de estado seguinte possa percorrer várias combinações antes de atingir a combinação estabelecida pelo projectista. Se o elemento de memória é assíncrono este comportamento é desastroso levando a que a sequência estabelecida pelo projectista não seja cumprida. Se o elemento de memória é síncrono, este problema não é relevante, sendo necessário apenas garantir que o período do sinal de *clock* do elemento de memória, é maior que o tempo de propagação do circuito combinatório gerador do estado seguinte.

6.3 Dispositivos de memória

Nos sistemas electrónicos, tal como noutro tipo de sistemas, podem constituir elementos de memória, os elementos capazes de conservar energia de forma intrínseca ou através de realimentação positiva. Diz-se que existe realimentação positiva ou retro-alimentação positiva num circuito electrónico quando ao apresentarmos a informação da saída na entrada do sistema este mantém a tendência de subida ou de descida que a tensão na saída apresenta. O elemento de memória que constitui a memória DRAM (*Dynamic Random Accesses Memory*) dos nossos computadores pessoais, é o condensador. No entanto estes elementos não têm uma utilização genérica, por manterem a energia armazenada durante um reduzido intervalo de tempo exigindo circuitos complexos que realizem o refrescamento (adição de energia). Os circuitos sequenciais que

vamos estudar, são sintetizados a partir de células unitárias de memória, referidos como circuitos biestáveis ou mais comummente por *flip-flop*.
Um *flip-flop* permite memorizar um bit de informação, dado que pode tomar de forma permanente um de dois estados possíveis, entendendo-se como estado o valor lógico presente na sua saída.
Um *flip-flop* pode ser sintetizado como mostra a Figura 6-3.

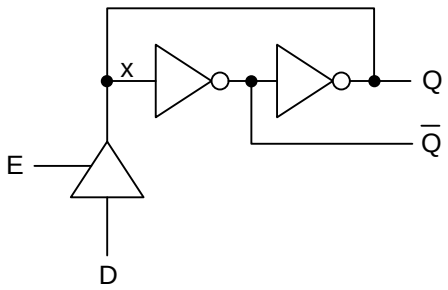


Figura 6-3

Como se pode observar na Figura 6-3, a implementação de memória é conseguida pela realimentação positiva do valor lógico da saída **Q** na entrada **x**.
Enquanto a entrada **E** (*Enable*) estiver com o valor lógico 1, o circuito é transparente de **D** para **Q**, ou seja, a saída **Q** toma o valor lógico que a entrada **D** (**D**ata) tomar. Logo que **E** muda de 1 para 0, devido à realimentação, o valor lógico presente na entrada **D** é *latched* (trincado) na saída **Q** e assim permanece enquanto a entrada **E** se mantiver a 0. Isto é, **Q** mantêm de memória o valor que estava presente em **D** no momento em que a entrada **E** tomou o valor zero, ficando o *flip-flop* insensível a entrada **D** enquanto **E** se mantiver a zero. Este *flip-flop* tem um comportamento assíncrono e denomina-se por *flip-flop* tipo ***D-latch*** ou ***Transparent latch***.
Embora esta implementação seja a mais comum na produção de memórias SRAM utilizadas nos sistemas baseados em microprocessadores, ela tem como inconveniente o aumento de carga momentâneo, quando se modifica o valor lógico da entrada D enquanto E se encontra a 1, pois para alterar o valor de **x** necessita forçar o valor existente na saída **Q**.

6.3.1 Flip-flop S-R

Uma vez que o problema do circuito anterior é a inserção de um novo valor lógico no *loop* Q x, podemos ter como solução, substituir as portas NOT, por portas lógicas com duas entradas que permitam através de uma das entradas forçar um valor lógico na saída ou conferir-lhe o comportamento de um NOT.
O circuito da Figura 6-4 apresenta duas soluções possíveis utilizando portas NAND ou NOR. Este *flip-flop* é denominado de S-R *latch* o qual apresenta o seguinte comportamento:

- É sensível a duas entradas: S (*set*) para impor o estado 1, R (reset) para impor o estado 0;
- Mantém o estado (memoriza) enquanto as entradas S e R se mantêm desactivas;
- A saída Q toma o valor lógico 1 quando activamos S e não activamos R;

- A saída Q toma o valor lógico 0 quando activamos R e não activamos S;
- O comportamento do *flip-flop*, quando se activa S e R em simultâneo, não está definido por ser ambíguo.

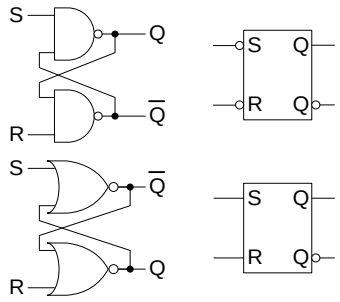


Figura 6-4

6.3.2 Flip-flop Latch

Baseado no *flip-flop* SR podemos sintetizar um *flip-flop* tipo *D-latch* como mostra a Figura 6-5. O descodificador, assegura por um lado, que não é activado S e R em simultâneo e quando inibido pela entrada E desactiva simultaneamente S e R, trincando (*latch*) o valor lógico que se encontrava na saída nesse momento. Enquanto a entrada E se mantiver desactivada, o *flip-flop*, fica insensível ao valor lógico presente na entrada D e mantém na saída o valor memorizado. Enquanto a entrada E se mantiver activa o sinal de entrada D transparece para a saída Q.

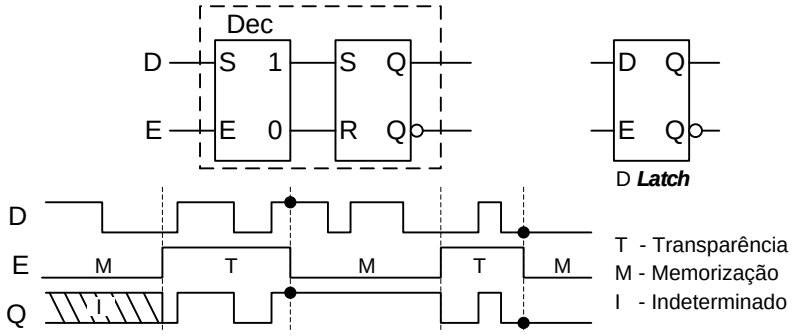


Figura 6-5

6.3.3 Flip-flop edge-triggered

O projecto de sistemas sequenciais complexos baseado em elementos de memória assíncronos, torna-se complexo, pois o seu comportamento fica dependente do tempo de propagação das portas lógicas, produzindo efeitos difíceis de controlar ao nível da realimentação do estado como já anteriormente foi referido. Por esta razão, o projecto de sistemas sequenciais é normalmente implementado recorrendo a elementos de memória síncronos. Não obstando o facto de ser mais difícil a utilização de circuitos assíncronos, estes tornam-se por vezes indispensáveis na construção de sistemas sequenciais.

Nos circuitos sequenciais ditos síncronos, é necessário que os *flip-flops* que o constituem reajam às entradas somente quando a entrada de controlo de sincronismo (*clock*) transita de 0 para 1

(transição ascendente) e não durante todo o tempo que este permanece a 1 (reacção a nível). Este comportamento é denominado por *edge-triggered* (borda de disparo) e caracteriza-se pelo seguinte: a observação das entradas, e implicitamente a mudança de estado, efectuam-se num instante discreto de tempo em *sincronismo* com o momento da transição ascendente do sinal de *clock*.

A Figura 6-6 mostra uma implementação do *flip-flop* D (***D**elay*)- *edge-triggered* e o diagrama temporal correspondente ao seu comportamento. O símbolo triangular identifica a entrada de sincronismo e por esta razão não é identificada por uma letra. O circuito tem o seguinte comportamento:

- Enquanto CLK se mantem a zero, a entrada D transparece para x, a saída Q permanece no estado anterior, e que no início é indeterminado;
- No instante em que CLK transita de zero para um, Q toma o valor de x (valor de D no instante imediatamente anterior à transição) e o primeiro *flip-flop* fica insensível a D memorizando o valor que D tinha nesse instante;
- Quando CLK transita de novo para zero, o segundo *flip-flop* deixa de ser sensível a x, memorizando o valor que este tinha e o primeiro *flip-flop* volta a deixar transparecer D para x. Desta forma Q continua a apresentar o valor que D tinha no momento da transição de um para zero do sinal CLK e assim se manterá até que ocorra nova transição ascendente no sinal CLK.

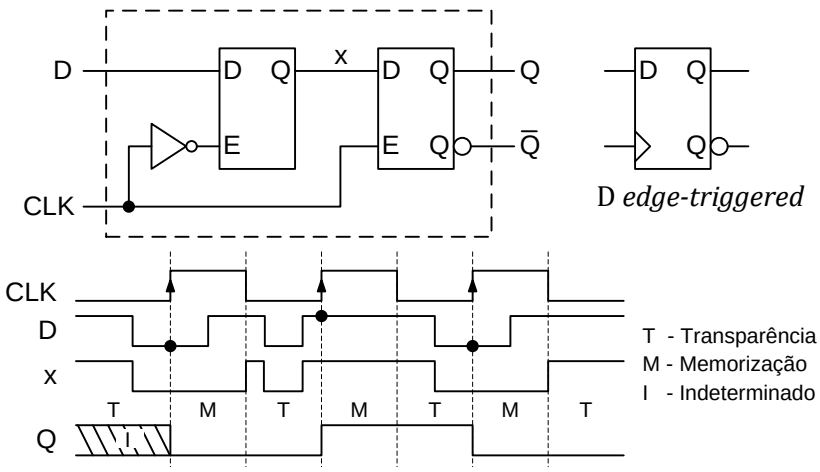


Figura 6-6

Assim sendo, podemos dizer que um flip-flop D *edge-triggered* a cada transição ascendente do sinal de sincronismo (clock) coloca na saída Q o valor lógico presente na entrada D, ou seja, a entrada D aparece na saída atrasada (*Delayed*) pelo sinal de *clock*.

Estes dois tipos de *flip-flops* (D *latch* e D *edge-triggered*) constituem o elemento base das estruturas de memória e dos sistemas sequenciais. Entre estas estruturas destaca-se o *registo* e a memória SRAM.

6.4 Registo

O registo é um elemento no qual se pode armazenar (registar) uma palavra constituída por n bits. O registo é caracterizado por ter uma entrada de dados de n bits, uma saída de dados de n bits, e uma entrada de controlo que quando activada *regista* os n bits presentes na entrada de dados e exibe-os na saída, só voltando a alterar o valor registado quando se activar novamente a entrada de controlo. Na Figura 6-7 estão representados dois registos de 4 bits, um com natureza *latch* e outro *edge-triggered*, e os respectivos diagramas temporais, evidenciando as siglas que normalmente estão disponíveis na especificação funcional (data *sheet*) deste tipo de componentes.

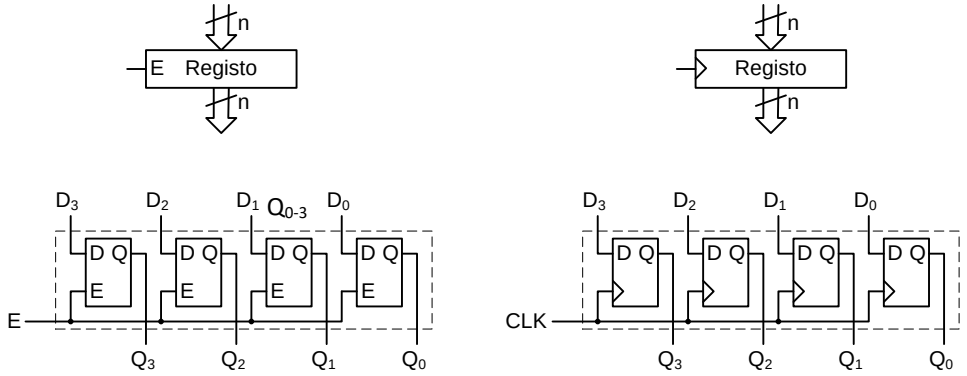


Figura 6-7

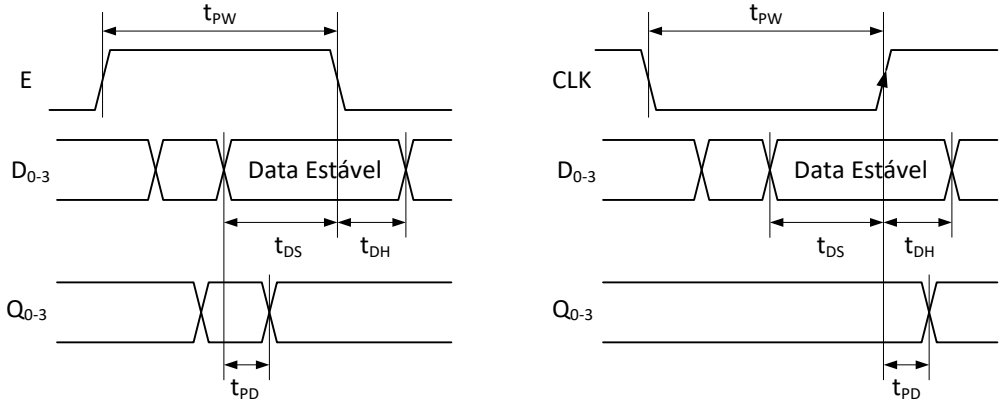


Figura 6-8

t_{PW} (*Pulse Width time*) duração mínima do sinal de memorização nesse patamar.

t_{DS} (*Data Set-up time*) intervalo de tempo mínimo a respeitar entre o estabelecimento de informação estável na entrada de dados e o momento da memorização.

t_{DH} (*Data Hold time*) intervalo mínimo de tempo durante o qual ainda se torna necessário manter a informação estável na entrada de dados após o sinal de memorização ter terminado.

t_{PD} (*Propagation Delay time*) tempo máximo de propagação entre um acontecimento na entrada de dados e o seu reflexo na saída do circuito.

6.5 Síntese de circuitos sequenciais

Dado o modelo geral do circuito sequencial anteriormente descrito, a síntese deste tipo de circuito, consiste em determinar as expressões booleanas do circuito combinatório que gera o estado seguinte e os valores lógicos de saída, função do estado presente e do valor lógico presente nas entradas, conferindo ao sistema o comportamento que foi determinado pelo projectista.

O primeiro desafio que se nos coloca é descrever o comportamento do sistema sequencial, que por ser dinâmico, não é passível de uma descrição facilmente assimilável se for feita através de uma tabela de verdade. O método que iremos utilizar, foi sugerido por C. R. Clare, e consiste em descrever o comportamento através de um fluxograma a que denominou ASM (*Algorithm State Machine*). Neste diagrama pode-se observar de uma forma simples qual o comportamento do sistema em função do estado presente e do valor das entradas. Outra vantagem que o método apresenta e que parte de uma premissa realista, é que em cada estado não são tomadas em consideração todas as variáveis de entradas do circuito para decidir qual o próximo estado, tornando mais simples a determinação da função geradora de estado seguinte. Quanto às variáveis de saída só são evocadas aquelas que nesse estado ficam activas.

6.5.1 ASM (*Algorithm Stat Machine*)

O ASM consiste em três símbolos como mostra a Figura 6-9.

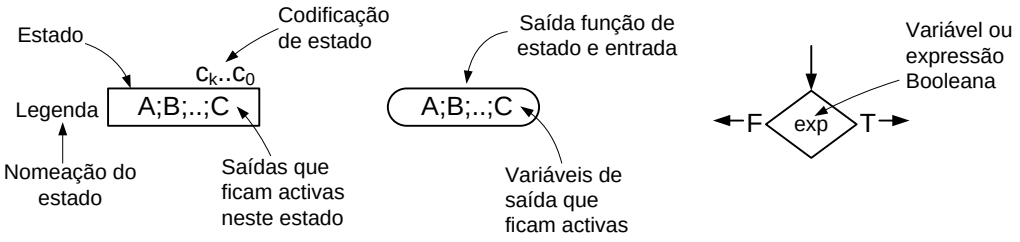


Figura 6-9

O rectângulo representa o estado corrente do circuito sequencial, a oval representa a activação de saídas em função do estado presente e de variáveis de entrada e o losango representa a decisão binária em que uma variável ou uma expressão booleana é avaliada, sendo assim determinados dois fluxos em função do resultado da expressão. No símbolo de decisão é conveniente que se utilize a letra F (*False*) e T (*True*), em vez do dígito 0 e 1, pois esta última forma promove confusão entre o valor lógico da variável, que pode ser activa em zero, e a acção desempenhada pela variável. Daí que também a escolha do nome a atribuir à variável deva ser bastante criteriosa para permitir uma melhor leitura do ASM.

Caso o ASM represente o comportamento de um circuito sequencial síncrono, ou seja, utilizando *flip-flops edge-triggered*, a interpretação do ASM, necessita das seguintes considerações:

- A entrada *clock*, por ser um sinal periódico, é omissa na representação;
- A avaliação da variável ou da expressão de controlo de fluxo é efectuada na transição ascendente de *clock*. Este facto implica, que o circuito só pode controlar acontecimentos com duração superior a um período do *clock*;
- O tempo mínimo que a máquina permanece num estado é um período de *clock*, pelo que uma variável de saída função de estado tem a duração mínima de um período de *clock*.

Como já anteriormente foi referido, recorreremos preferencialmente a implementações síncronas, por apresentarem um comportamento mais estável. No entanto, sempre que a situação o justifique poderemos recorrer a implementações assíncronas.

Exemplos de aplicação:

Exemplo 1:
Tomemos como exemplo o controlo do enchimento de um reservatório de água com o diagrama de blocos da Figura 6-10.

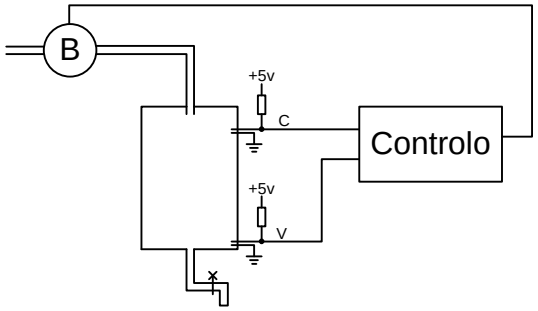


Figura 6-10

O reservatório é constituído pelos seguintes elementos:

- Bomba de água **B**, para enchimento;
- Sensor de reservatório cheio **C**;
- Sensor de reservatório vazio **V**;

Considerando o facto de que a água é condutora de corrente eléctrica os sensores C e V são constituídos por dois eléctrodos que conduzem corrente eléctrica quando mergulhados na água.
Para que a bomba **B** não esteja intermitentemente a arrancar e a parar, pretende-se o seguinte comportamento: A bomba B só entra em funcionamento quando o reservatório está completamente vazio (V=+5v). A bomba, após entrada em funcionamento, só pára quando o reservatório fica completamente cheio (C=0V). Na Figura 6-11 está representado o ASM que descreve o comportamento do sistema de controlo de enchimento do reservatório, segundo o critério estabelecido. Considera-se que o caudal de enchimento produzido pela bomba B é superior ao caudal da torneira de esvaziamento.

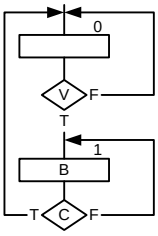


Figura 6-11

Quanto à implementação, uma vez que temos apenas dois estados, será necessário apenas um *flip-flop*. Quanto ao tipo de implementação, poderemos optar por uma implementação síncrona ou

assíncrona. Na Figura 6-12 podemos observar as duas implementações, uma assíncrona recorrendo a um *flip-flop* SR e outra síncrona recorrendo a *flip-flop* D *edge-trigger*. A negação da variável C na entrada R deve-se ao comportamento do sensor C que fica a 0v quando mergulhado na água indicando nesta situação que o reservatório está cheio.

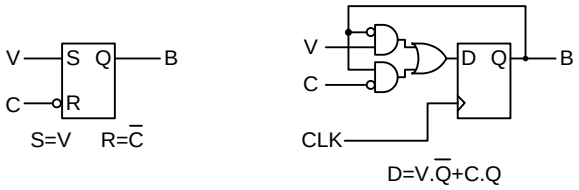
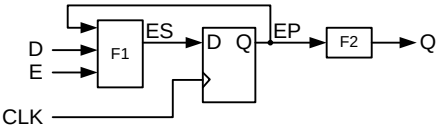


Figura 6-12

Exemplo 2:

Tomemos como exemplo a síntese de um *flip-flop* com o comportamento do *flip-flop* D *edge-triggered* ao qual adicionamos uma variável E (*Enable*). O novo *flip-flop* tem o comportamento do *flip-flop* D *edge-triggered* enquanto a entrada E estiver activa. Quando a entrada E está desactiva o *flip-flop* mantém o estado. Uma solução óbvia seria a intercepção do sinal de *clock* pelo sinal E. Esta solução tem dois defeitos: primeiro não confere o comportamento pretendido, uma vez que a activação de E com o sinal de *clock* ao valor lógico 1, levava a que o *flip-flop* mudasse de estado; segundo, soluções que envolvam intercepção da entrada *clock*, são desaconselhadas por poderem produzir transições indesejáveis e meta-estados, ou seja, instabilidade durante a transição de estado por o impulso de *clock* pode ser inferiores a t_{PW}.

Na Figura 6-13 está representado o diagrama blocos de uma solução envolvendo o controlo da entrada D, podendo-se nele identificar as entradas dos circuitos combinatórios geradores de estado seguinte e de saída.



ES – Estado Seguinte
EP – Estado Presente
CLK – ***clock***

F1 – Circuito gerador de estado seguinte

F2 – Circuito gerador de saída

Figura 6-13

Na Figura 6-14 está representado o ASM que descreve o comportamento do circuito sequencial e a sua implementação utilizando um *flip-flop* do tipo D *edge-triggered*. Como se pode observar, a máquina permanece no estado 0 ou 1 se a entrada E tiver valor lógico 0.

Quanto à implementação, e usando como a célula de memória o *flip-flop* D *edge-triggered* cujo comportamento anteriormente descrito determina que o valor da saída é igual ao da entrada, podemos ler no ASM que o estado seguinte ES é igual a um, se a máquina estiver no estado presente '0' e E e D forem 1, ou estando no estado presente '1' se E for 0 ou D for 1. Podemos então escrever a seguinte expressão $ES = \overline{EP}.E.D + EP(\overline{E} + D) = \overline{E}.EP + E.D$.

Quanto à saída Q, dada a codificação de estados do ASM, o seu valor é igual ao do estado presente EP. Dadas as expressões obtidas, podemos observar na Figura 6-14 uma possível implementação.

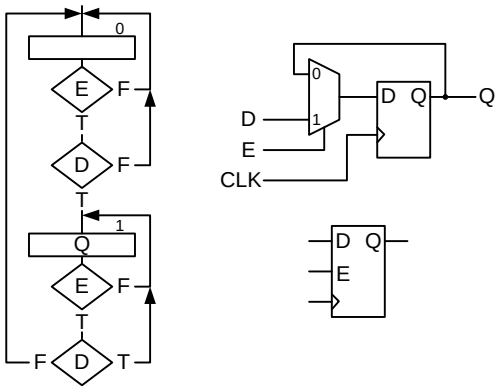


Figura 6-14

Exemplo 3:
Tomemos como exemplo a síntese de uma máquina de estados que obedece ao ASM da Figura 6-15.

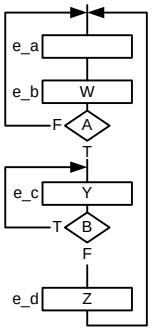


Figura 6-15

A máquina tem quatro estados (e_a, e_b, e_c, e_d), duas entradas A e B, e três saídas respectivamente W, Y e Z. A permanência da máquina no estado e_a, e_b e e_d é de apenas um período de clock, levando a que as variáveis de saída Z e W fiquem activas apenas durante um período de *clock*. Quanto ao estado e_c pode aqui permanecer durante um ou mais períodos de *clock* dependendo do valor lógico de B o que levará a que variável de saída Y fique activa durante um ou mais períodos de *clock*.

6.5.2 Memória de estado

Quanto à implementação, como já foi referido anteriormente, existem diferentes métodos dependendo da tecnologia que estejamos a utilizar. Uma vez que se trata de uma máquina com mais de dois estados a sua implementação implica que a memória de estado seja composta por mais que um *flip-flop*. Comecemos por analisar o número de *flip-flops* necessários à implementação da memória de estado.

One Hot

Uma vez que a máquina tem quatro estados podemos utilizar quatro *flip-flops* atribuindo a cada estado um *flip-flop*. Este método que é denominado por *one hot*, embora utilize um número elevado de *flip-flops* tem como vantagem a utilização de menos lógica na entrada de cada *flip-flop*, diminuindo desta forma a lógica da função geradora de estado seguinte. As saídas que ficam activas num determinado estado passam a depender apenas da saída de um único *flip-flop* diminuindo assim a lógica e como consequência tornando a máquina mais rápida. Este método é utilizado normalmente em implementações assíncronas ou quando dispomos de muitos *flip-flops* e pouca lógica associada às entradas de cada *flip-flop*, que é o caso da FPGA (*Field Programming Gate Array*). Na Figura 6-16 é apresentada a implementação do ASM da figura 6-14 utilizando o método *one hot*.

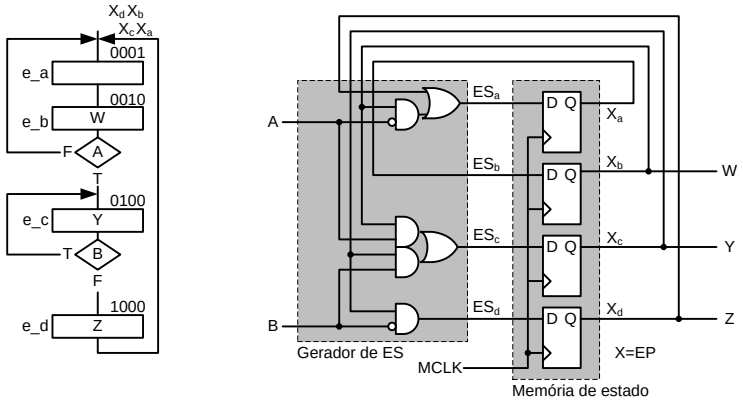


Figura 6-16

A expressão de cada uma das funções, que constituem o vector estado seguinte ES_{a-d} é obtida da seguinte forma:

A partir do ASM que pretendemos implementar e conhecendo o comportamento de cada um dos *flip-flops* (transições de estado) que constituem a memória de estado, estabelecemos funções geradoras de estado seguinte, que determinem, em cada estado presente, valores lógicos às entradas dos *flip-flop* que levem a que numa próxima transição ascendente do sinal de *clock*, a máquina de estados atinja o estado pretendido.

Como podemos observar na Figura 6-16 o *flip-flop* X_a só toma o valor lógico 1, ou seja, só alcança o estado e_a, se a máquina se encontrar no estado e_d ($X_d = 1$) ou estando no estado e_b a variável A for falsa, pelo que a entrada D do flip-flop X_a fica a "1" se Xd for "1" ou se Xb for "1" e simultaneamente A for "0". O *flip-flop* X_b só toma o valor lógico 1, se X_a tiver o valor lógico 1, ou seja, se a máquina estiver no estado e_a. A geração de estado seguinte dos *flip-flops* X_c e X_d seguem exactamente a mesma lógica, ou seja, estabelece-se na entrada D do *flip-flop* uma expressão que seja verdadeira quando estiverem criadas as condições para transitar para esse estado, tomando em consideração não só as entradas mas também o estado presente.

Highly-Encoded state assignments

Outro método, denominado por *Highly-Encoded state assignments*, e que iremos preferencialmente utilizar ao longo dos vários exercícios, consiste em numerar os vários estados em binário, no sentido de diminuir o número de *flip-flops* a utilizar. Este método tem como consequência o aumento da lógica associada à geração de estado seguinte e às saídas. Este método é mais adequado à implementação com PAL ou com lógica discreta. Fazendo uso deste método, podemos ver na Figura 6-17 a implementação do ASM da Figura 6-15, utilizando para memória de estado dois *flip-flops* tipo D *edge-triggered* e para gerador de estado seguinte dois multiplexes*.*

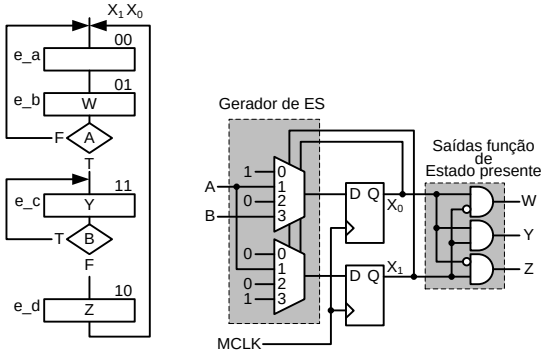


Figura 6-17

Na Figura 6-18, utilizando o mesmo método, é apresentada uma implementação que recorre a lógica discreta para gerar estado seguinte.

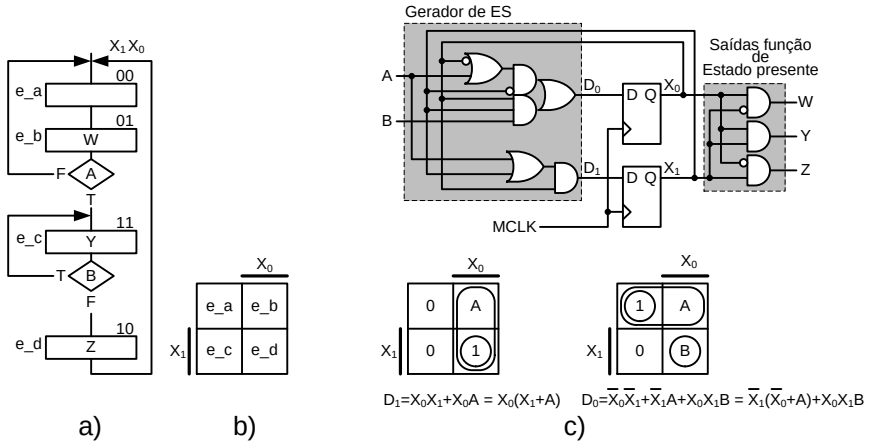


Figura 6-18

A expressão de cada uma das funções, que constituem o vector estado seguinte é obtida da seguinte forma:

- Para evitar a utilização de mapa de *Karnaugh* com muitos termos produto, vamos utilizar variáveis inseridas no mapa, como vimos no capítulo 2.
- A partir do ASM Figura 6-18 a) preenche-se o mapa de atribuição de estados Figura 6-18 b), representando cada intercepção o estado presente.

- Se atendermos à atribuição de estado adoptada, e reportando-nos ao estado e_a da máquina, verificamos que, neste estado, o *flip-flop* X_0 passa do estado 0 a 1 e X_1 mantêm-se no estado 0. Como o estado e_a da máquina corresponde ao quadrante superior esquerdo do mapa de atribuição de estados, (quadrado correspondente à configuração $\overline{X}_1.\overline{X}_0$), há que preencher este quadrado com 1 e 0, respectivamente, nos mapas D_0 e D_1. Reportando-nos ao estado e_b (quadrante superior direito) constatamos que X_0 passa de 1 a 0 se A=0 e passa de 1 a 1 se A=1 logo D_0 tem uma relação de identidade com A (D_0=A), quanto a X_1 permanece em 0 se A=0 e passa de 0 a 1 se A=1, logo (D_1=A). Com esta informação preenchem-se os quadrados correspondentes nos mapas D_0 e D_1. Nos restantes estados adoptaremos o mesmo critério.

Exemplo 4:
Tomemos como exemplo a síntese de um circuito para controlo da abertura e fecho de uma porta através de um sensor de presença P como mostra a Figura 6-19. O movimento da porta é realizado por um motor cujo movimento é controlado por dois sinais: ON para ligar o motor, e SF que determina o sentido de rotação do motor, e que quando activo estabelece o sentido de fecho. Para determinar se a porta se encontra aberta ou fechada existem dois sensores A e F que ficam activos quando a porta se encontra completamente aberta ou completamente fechada.

A abertura e fecho da porta devem obedecer ao seguinte critério:

- Quando a porta se encontra fechada, inicia a abertura se o sensor P ficar activo;
- Iniciada a abertura, esta só termina quando a porta se encontrar totalmente aberta;
- A porta depois de totalmente aberta inicia de imediato o fecho caso o sensor P não esteja activo, caso contrário permanece aberta enquanto o sensor P se mantiver activo.
- Iniciado o fecho, este só termina quando a porta se encontrar totalmente fechada.
- Durante o fecho, se o sensor P ficar activo, então o sistema reinicia a abertura de imediato.

Na Figura 6-19 está representado o diagrama de blocos do sistema de controlo da porta, bem como o ASM que descreve o seu comportamento segundo o critério estabelecido.

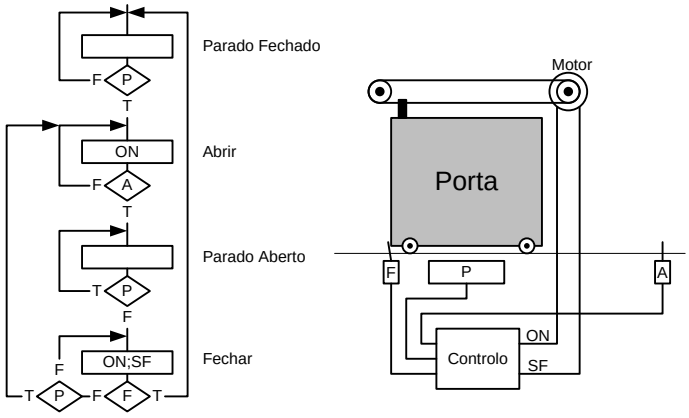


Figura 6-19

Se considerarmos que quando o sistema de controlo é ligado/iniciado não conhecemos a posição da porta e que esta pode estar entreaberta ou completamente aberta, a porta só se sincronizava com o sistema de controlo após a passagem de uma pessoa. Para resolver este problema podemos tomar como estado inicial o estado e3 garantindo assim o encerramento da porta. No entanto, esta nova solução apresenta um inconveniente, é que se o portão já se encontrar fechado quando o sistema se inicia, o motor é forçado por tentar fechar a porta quando esta já está completamente fechada. Para resolver este problema temos duas soluções: introduzir um estado inicial onde se verifica o estado de F e P, e assim transitar para o estado e3 porque a porta não está fechada e não existe pessoa, ou para o estado de repouso e0 porque a porta já se encontra fechada. Outra solução passa por adoptar uma técnica denominada saída função de estado e entrada para a activação do sinal ON.

6.5.3 Saída função de estado e entrada

No modelo até agora utilizado e conhecido por modelo de *Moore*, as variáveis de saída dependem exclusivamente do estado corrente. Um outro modelo conhecido como *Mealy* estabelece que as saídas podem também depender do estado corrente e do valor lógico presente nas entradas do sistema. Este modelo permite diminuir o número de estados, no entanto, como veremos adiante, nem sempre é possível utilizar, pois depende do comportamento pretendido e da relação com as variáveis de entrada.

Na Figura 6-23 Figura 6-24podemos ver o ASM do controlo de abertura e fecho da porta utilizando este modelo.

Implementação:

Utilizando a forma de implementação ***highly-encoded*** obtemos o circuito da Figura 6-20.

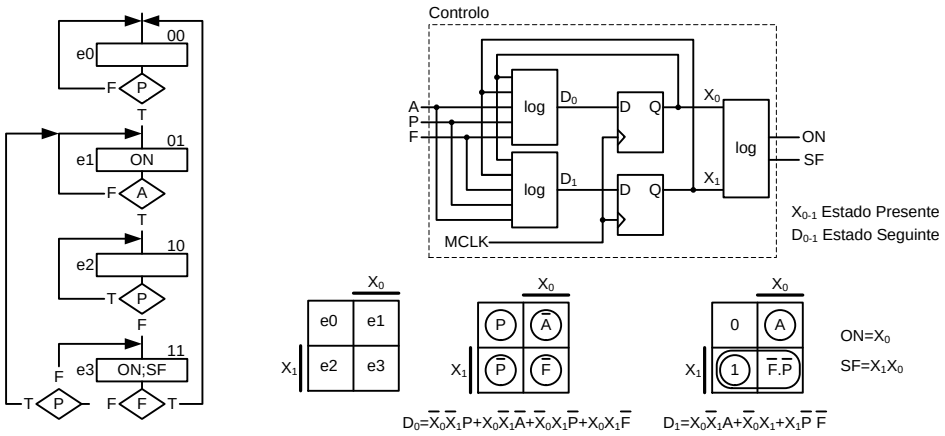


Figura 6-20

Como se pode observar na Figura 6-20, se a implementação se fizer utilizando portas lógicas de duas entradas, seriam necessárias 13 ANDs, 5 ORs e 3 NOTs, o que na família 7400 corresponderia a utilizar 6 circuitos integrados.

No sentido de diminuir a lógica das funções geradoras de estado seguinte, iremos introduzir um novo tipo de *flip-flop*, cujo comportamento iremos estudar e sintetizar a partir do *flip-flop* tipo D e que é disponibilizado pelos fabricantes de componentes lógicos.

6.5.4 *Flip-flop* JK

O *flip-flop* JK deve o seu nome a Jack Kilby. O *flip-flop* JK, para além do sinal de *clock*, apresenta mais duas entradas, o J e o K e que lhe conferem o comportamento descrito no ASM da Figura 6-21. Como se pode observar também na Figura 6-21, o *flip-flop* JK quando está no estado 0, só está sensível à entrada J e caso esta esteja activa no momento da transição de *clock*, faz *set* ao *flip-flop*. Quando está no estado 1, só está sensível à entrada K e caso esta esteja activa no momento da transição de *clock*, faz reset ao *flip-flop*. Também na Figura 6-21 podemos observar como este *flip-flop* é sintetizado a partir de um *flip-flop* D por comparação entre o ASM do D e do JK, podemos determinar a seguinte expressão $D = \overline{Q}.J + Q.\overline{K}$.

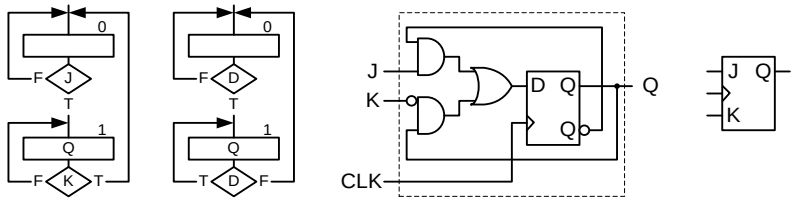


Figura 6-21

A utilização do *flip-flop* JK na implementação do ASM da Figura 6-19 produziria a solução apresentada na Figura 6-22, que como se pode observar só utiliza 6 ANDs, 3 ORs e um NOT. No entanto, relembramos que as vantagens que se obtêm com a utilização do *flip-flop* JK, é dependente da tecnologia que estivermos a utilizar para a implementação da função geradora de estado seguinte.

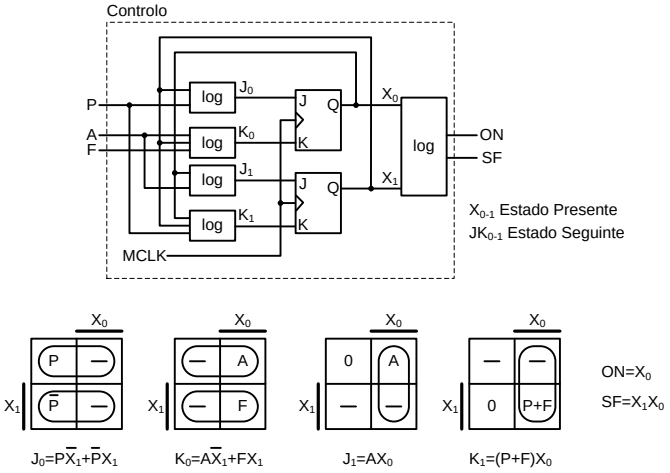


Figura 6-22

6.5.5 Saída função de estado e entrada

No modelo até agora utilizado e conhecido por modelo de *Moore*, as variáveis de saída dependem exclusivamente do estado corrente. Um outro modelo conhecido como *Mealy* estabelece que as saídas podem também depender do estado corrente e do valor lógico presente nas entradas do sistema. Este modelo permite diminuir o número de estados, no entanto, como veremos adiante, nem sempre é possível utilizar, pois depende do comportamento pretendido e da relação com as variáveis de entrada. Na Figura 6-23 Figura 6-24podemos ver o ASM do controlo de abertura e fecho da porta utilizando este modelo.

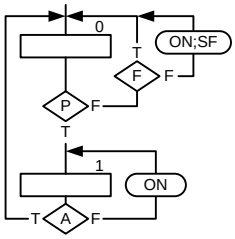


Figura 6-23

Observemos o comportamento da máquina de estados preconizado:
Admitamos que a porta se encontra fechada, então a máquina permanece no estado 0 aguardando que P seja verdadeiro, logo que P seja verdade transita para o estado 1, activa ON enquanto A falso e ai permanece até que a porta fique completamente aberta. Logo que a porta fique completamente aberta transita para o estado 0, desligando o motor. Se quando atinge o estado 0 e P permanecer activo, a máquina fica a transitar entre o estado 0 e o estado 1 mantendo o motor desligado.

Agora no estado 0, logo que P seja falso, fixa-se em 0, e enquanto não existir P e F for falso, mantém o portão a abrir.
Dado o ASM obtido, o *flip-flop* mais adequado é o JK, pois P faz *set* e A *reset* e cuja implementação podemos observar na Figura 6-24.

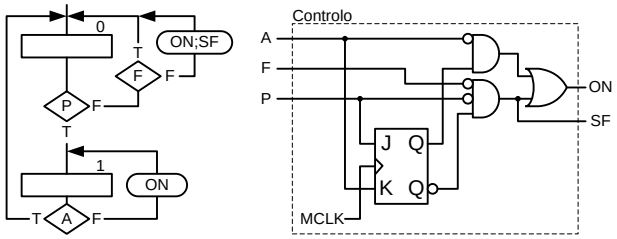


Figura 6-24

6.5.6 Interligação entre Máquinas de estado

O recurso a saídas função de estado e de entrada para diminuir o número de estados, como se utilizou no exemplo 4, nem sempre é possível, pois depende do comportamento pretendido e da relação com as variáveis de entrada, devendo-se evitar a activação de uma variável de saída função de estado e entrada na transição de estado, principalmente quando se trate de variáveis de interligação entre máquinas.
Em seguida é apresentado um exemplo que torna este facto relevante.
Imaginemos que se pretende controlar a abertura e fecho desta mesma porta não com um sensor de presença mas sim com um único botão. Sempre que se prime o botão alteramos o estado de movimento da porta, ou seja, se está em andamento pára, se está parado inicia o movimento em sentido contrário àquele em que se encontrava quando parou. O movimento da porta mantém o critério anterior, ou seja, também cessa por detecção dos sensores de aberto e fechado. Ora o teste continuado do botão por parte do controlo, levaria a que o movimento da porta ficasse a transitar entre parar e inversão de marcha enquanto o botão permanecesse premido como se pode ver na Figura 6-25.

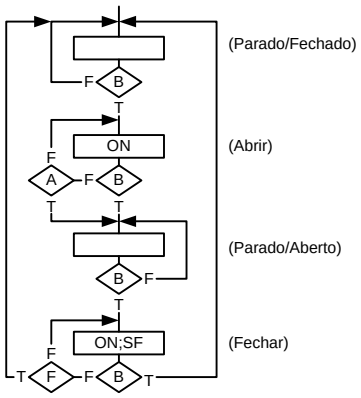


Figura 6-25

Para tal é necessário detectar as actuações do botão, ou seja, produzir uma única acção por cada vez que se pressiona o botão. Como se pode observar na Figura 6-25 o botão é testado em todos os

estados o que equivalia a duplicar o número de estados, para realizar a detecção *edge trigger* do botão. De forma a simplificar a implementação do controlo, poderemos pensar numa implementação recorrendo a duas máquinas de estado interligadas como mostra a Figura 6-26. Uma das máquinas realiza a detecção *edge trigger* da actuação do botão (DET) e outra realiza o controlo do movimento da porta (CMP).

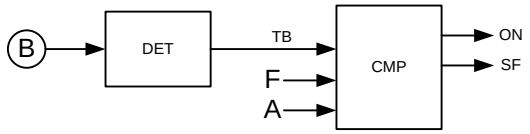


Figura 6-26

Na interligação proposta no diagrama de blocos da Figura 6-26, só existe informação na direcção DET CMP, o que implica que o sinal TB tem que ter uma duração tal que possa ser observado pelo CMP, mas não muito longa para não ser observada por mais que uma vez. Assim sendo o DET deverá ter o ASM descrito na Figura 6-27. A máquina tem três estados e a saída TB, função de estado, é activa durante um só estado (estado 01). Esta solução implica que as máquinas DET e CMP têm que estar síncronas, ou seja, terem o mesmo sinal de *clock*. Para que a observação de TB por parte do CMP não se faça no mesmo instante em que é alterado pelo DET, o sinal de *clock* das duas máquinas é desfasado de 180º de forma a garantir que o sinal TB está estável no momento em que é testado pelo CMP. Esta deverá ser por isso, a forma preferencial de interligar duas ou mais máquinas de estado, ou seja, partilhar o mesmo sinal de *clock* por forma a garantir o sincronismo entre elas.

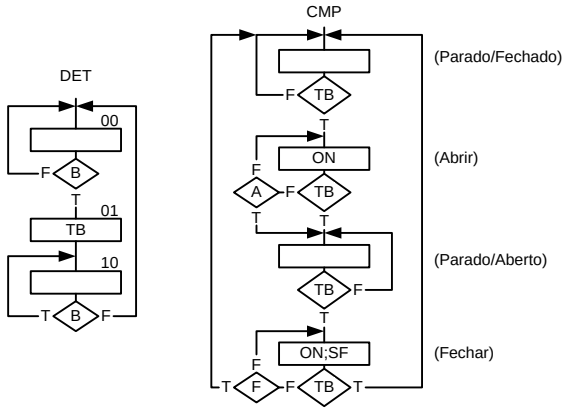
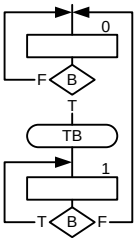


Figura 6-27

Poderemos simplificar esta máquina fazendo com que a saída TB, seja uma saída função de estado e entrada como mostra a Figura **6-28**, passando a máquina a ter apenas dois estados, ou seja, utilizando um único *flip-flop*.



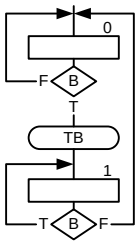


Figura 6-28

Esta implementação apresenta o seguinte problema: se o momento em que se actua o botão for posterior à transição descendente de MCLK e anterior ao momento da transição ascendente de MCLK, como se mostra no diagrama temporal da Figura 6-29, o tempo de activação da saída TB não é observada pelo CMP. Por esta razão, a utilização de variáveis de saída função de estado e entrada, tem que ser muito criteriosa.

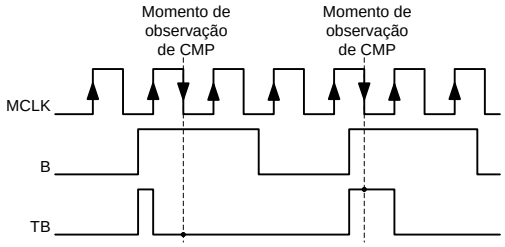


Figura 6-29

Se admitirmos que as duas máquinas não têm o mesmo *clock*, ou seja, não estão síncronas nem ao mesmo ritmo, a solução passará por garantir que o CMP informa o DET que já consumiu aquele acontecimento, obtendo-se assim o diagrama de blocos da Figura 6-30.

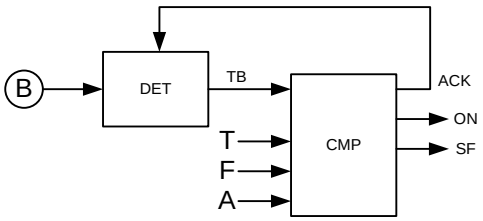


Figura 6-30

Esta solução implica intercalar entre cada um dos estados do ASM do módulo CMP um estado de activação do sinal ACK (Acknowledge) e alterar a máquina do DET como mostra a Figura 6-31 a). A solução apresentada na Figura 6-31 a) diz-se que é semiligada, ou seja, CMP não tem confirmação de que o sinal ACK foi recebido pelo DET. Na Figura 6-31 b) é apresentada uma solução dita totalmente interligada, ou seja, o CMP espera pela activação do sinal TB, e quando tal acontece, em resposta, activa o sinal ACK esperando que por essa razão o DET desactive TB. Após a desactivação de ACK por parte do CMP, o DET prosseguirá aguardando nova transição ascendente de B como mostra o diagrama temporal da Figura 6-31 c).

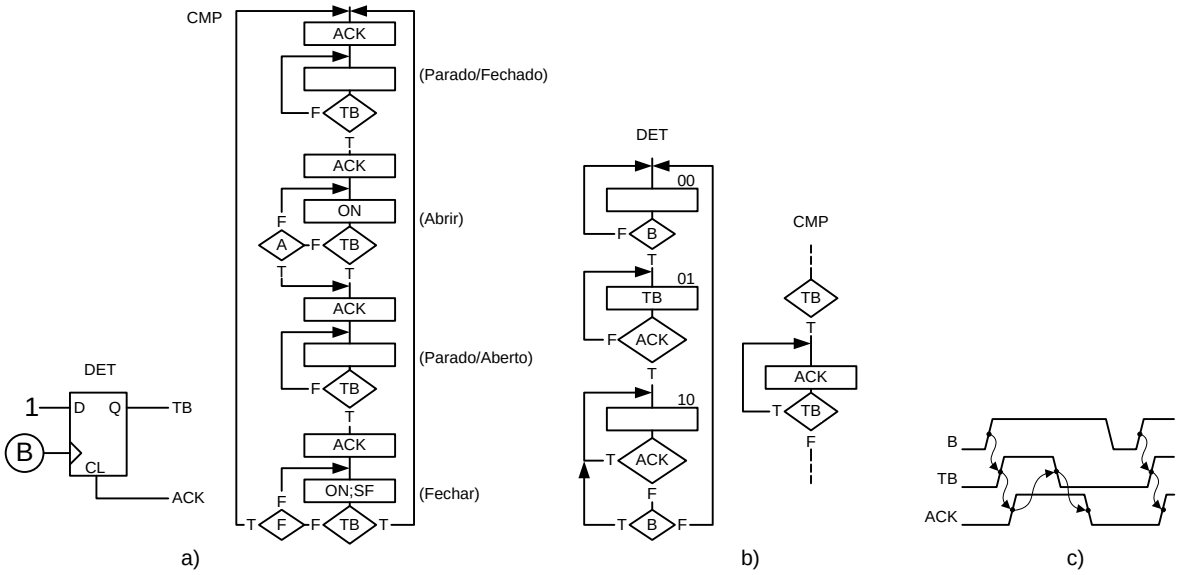


Figura 6-31

É de notar que a variável B, que fica activa quando se prime o botão B, tem que ser obtida através de um circuito de *debounce* do comutador/interruptor que lhe está associado.

6.5.7 Bounce

Quando se gera um sinal digital, por actuação de um interruptor ou comutador mecânico, surge um fenómeno físico denominado *bounce*. Este fenómeno, que é produzido no instante da abertura ou do fecho dos contactos devido à vibração destes e à disrupção do ar quando os contactos estão muito próximos, leva a que corrente eléctrica se estabeleça e se quebre várias vezes durante uma fracção de tempo muito pequena. Esta variação que pode durar alguns milissegundos, é problemática se o objectivo for excitar entradas com tempos de reacção muito pequenos, por exemplo o sinal de *clock* de um elemento *edge-triggered*. Por exemplo na tecnologia HCT, os tempos de reacção t_{PW} (***P**ulse **W**idth **t**ime*) são na ordem dos três nano segundos. Este fenómeno, levaria a que produzíssemos não uma, mas várias transições por cada actuação do interruptor ou comutador. O circuito para eliminar este fenómeno, é denominado por circuito de *debounce*. Este circuito apresenta diferentes arquitecturas dependendo do elemento que o produziu. Na Figura 6-32 estão representados os esquemas para cada uma das três situações.

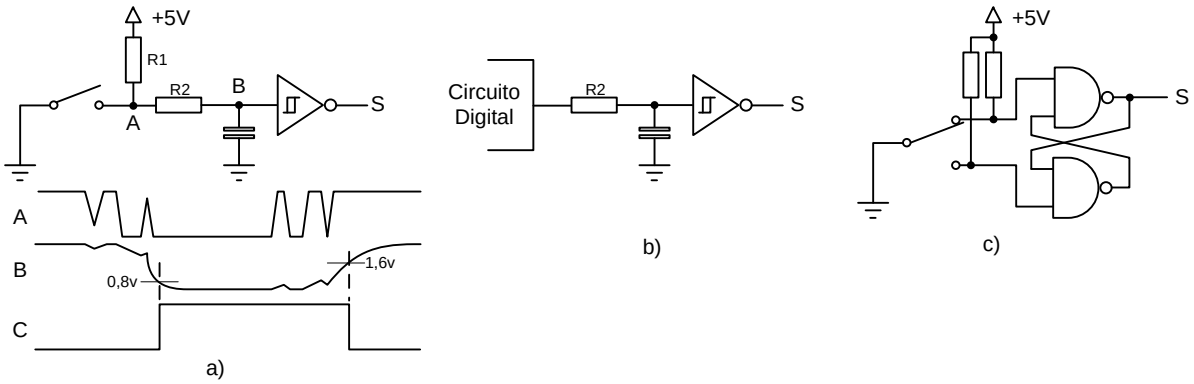


Figura 6-32

Quando se trata do *bounce* aos terminais de um interruptor SPSD (*Single Pole Single Throw*), ou à saída de um circuito digital, é necessário utilizar um filtro passa baixo Figura 6-32 a) e b).
Como é mostrado no gráfico temporal da Figura 6-32, a saída deste filtro, produz um sinal, que apresenta um *ripple* residual e um tempo de subida e descida incompatível com os tempos de transição especificados para os circuitos digitais comuns. Por esta razão é necessário utilizar portas lógicas com característica de entrada *Schmitt-trigger*. Estas portas que apresentam à entrada histerese, eliminam este efeito, porque o limiar de tensão na entrada a que se dá a comutação da saída, desloca-se imediatamente após a comutação da saída, filtrando assim pequenas variações do sinal em torno de um ponto. Quanto ao tempo de subida e descida serem diferentes, deve-se ao facto de a carga ser realizada por R1+R2 e a descarga ser realizada através de R2.
Quando se trata de um comutador SPDT (*Single Pole Double Throw*) não é necessário o filtro passa baixo, sendo o *bounce* retirado com utilização de um *flip-flop* SR como mostra a Figura 6-32 c).

6.5.8 Temporização

Admitamos que se pretende alterar o comportamento do sistema de controlo da porta, para que inicie o movimento de fecho, 5 segundos após a porta ter ficado completamente aberta. Para implementar esta funcionalidade põem-se duas soluções: ou introduzimos estados até perfazer 5 segundos, sabendo que cada estado consome um período de *clock*; ou utilizamos um elemento temporizador externo ao controlo. Como já foi referido anteriormente, o tempo de reacção das máquinas de estados síncronas às entradas depende da frequência do *clock*, pelo que é natural que este tenha uma frequência elevada a fim de não perder acontecimentos nas entradas. Assim sendo, a primeira solução implicaria um grande número de estados. Por esta razão iremos utilizar a segunda solução. Admita que o elemento temporizador tem uma entrada S (*Start*) que enquanto activa, mantém o temporizador parado num estado inicial, e quando posta a zero inicia a contagem de tempo. O temporizador põe disponível uma saída T para indicar que o tempo já decorreu. A saída T permanece activa até que o contador seja novamente colocado no estado inicial.
O CMP passaria a ter o diagrama de blocos e o ASM mostrados na Figura 6-33.

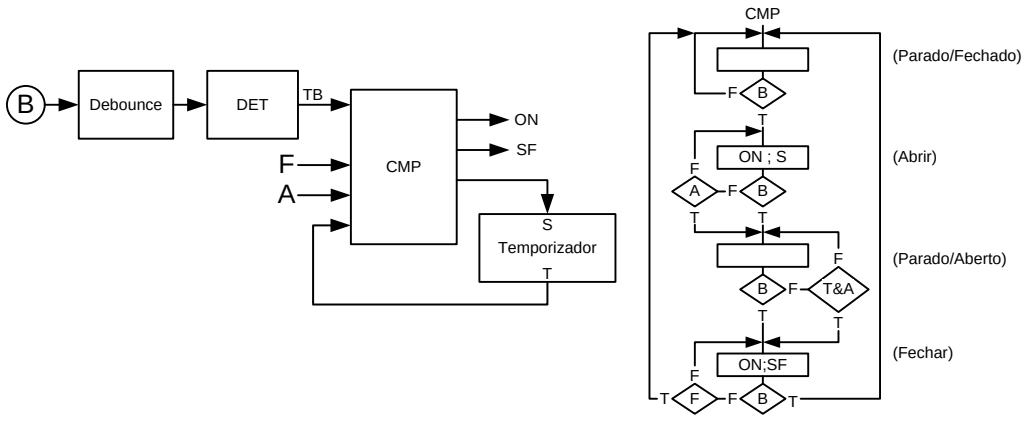


Figura 6-33

Para síntese do elemento temporizador, podemos recorrer a uma de duas soluções: ou utilizamos o tempo de carga e descarga de um condensador numa malha RC; ou utilizamos um sinal periódico C com período inferior a 5 segundos e contamos períodos desse sinal até perfazer os 5 segundos. Para a contagem dos períodos de C podemos utilizar uma de duas soluções: ou sintetizamos um ***contador*** que realize a contagem binária do número de transições ascendentes de C; ou sintetizamos um registo de deslocamento (***Shift Register***) constituído por uma cadeia de n *flip-flops* que, a cada transição ascendente do sinal C desloca de um bit o valor nele registado, injectando no *flip-flop* inicial da cadeia o valor lógico 1. A chegada do valor lógico 1 ao último *flip-flop* da cadeia indica que já decorreram os n períodos do sinal C. Os contadores binários e os *shift registers,* como veremos adiante no capítulo 8, constituem elementos fundamentais para a síntese de sistemas lógicos digitais.