8. MÓDULOS FUNCIONAIS SEQUENCIAIS

8.1 Contadores

O contador consiste num registo *edge-triggered* que, a cada transição ascendente da entrada de sincronismo (*Clock*), ao valor binário nele registado é acrescentado ou diminuído o valor 1, dependendo de se tratar de um contador crescente ou decrescente. O contador, como veremos adiante, constitui um importante módulo para a síntese de sistemas lógicos sequenciais. Estão disponíveis no mercado um grande número de contadores que apresentam diferentes características. A Figura 8-1 apresenta o esquema bloco de um contador, evidenciando várias entradas e saídas de controlo que os fabricantes adicionam no sentido de flexibilizar a utilização destes módulos.

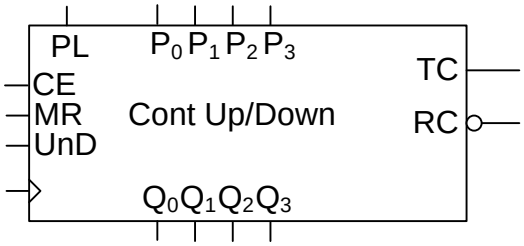


Figura 8-1

As entradas mostradas na Figura 8-1, desencadeiam uma série de acções, cujo comportamento passaremos a descrever.

UP/DOWN: Entrada UnD que determina se o sentido da contagem é crescente ou decrescente. A nomenclatura UnD pretende estabelecer que quando um sinal realiza duas acções (uma com zero e outra com um), o identificador associada à letra n (**n**o), neste caso o D (*Down*), se realiza quando o sinal toma o valor lógico zero.

ENABLE*:* Entrada CE (*Count Enable*) que, só quando activa, permite a contagem.

LOAD: Designada por PL (***P**arallel **L**oad*) quando activa, regista no contador o valor binário presente nas entradas $P_0 \ldots P_n$, sendo tomado este valor como estado presente do contador. Assim sendo, a evolução da contagem (crescente ou decrescente) faz-se a partir do valor assim estabelecido.

RESET: Entrada MR (*Master Reset*) que quando activada coloca o estado do contador a zero. Esta acção tem prioridade sobre a acção LOAD.

Para além do valor corrente de contagem que codifica o estado presente do contador, os contadores podem apresentar saídas cujo comportamento passamos a descrever. É de notar que alguns destes sinais só fazem sentido quando o módulo é disponibilizado num IC, e que pressupõe a sua concatenação com outros módulos idênticos no sentido de estender o módulo da contagem.

TC: *Terminal Count*, saída que indica enquanto activa, que o contador se encontra num dos estados extremos de contagem: n-1 na contagem crescente e 0 na contagem decrescente.

$\overline{RC}$: *Ripple Clock*, esta saída fica activa quando a contagem atinge um dos extremos, TC está activo e o sinal de *clock* está a zero.

8.1.1 Tipos de contador

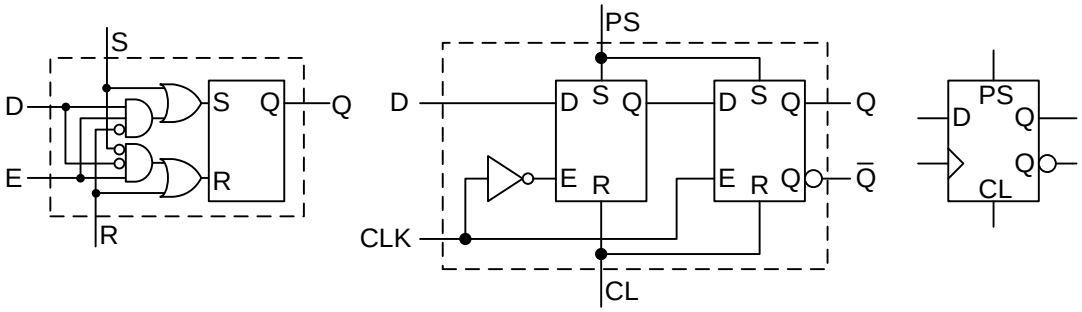
Os contadores podem ter dois tipos de comportamento **síncrono** ou **assíncrono**. Ou seja, se os vários *flip-flops* que o constituem evoluem de forma sincronizada (todos em simultâneo), diz-se que o contador é síncrono, se os vários *flip-flops* evoluem de forma dessincronizada (evoluem em tempos diferenciados) diz-se que o contador é assíncrono.

8.1.2 Entradas síncronas e assíncronas

As acções de RESET e LOAD podem ser síncronas ou assíncronas, ou seja, se acção for síncrona só se realiza quando a entrada está activa e existe uma transição ascendente do sinal de sincronismo. Se a entrada é dita assíncrona, a acção por ela desenvolvida é realizada durante todo o tempo que esta se mantiver activa. O comportamento assíncrono implica que, por exemplo no caso da acção LOAD, as saídas fiquem transparentes de $P_0 \ldots P_n$ para $Q_0 \ldots Q_n$, ou seja, variações nos valores de $P_0 \ldots P_n$ são imediatamente reflectidas nas saídas $Q_0 \ldots Q_n$.

8.1.3 Entradas assíncronas dos *Flip-Flops*

Para que possamos realizar acções assíncronas sobre o contador, é necessário que os *flip-flops* que o constituem disponham de entradas que afectem a saída de forma assíncrona. Estas entradas são o CL (***CL**ear*) ou AR (***A**synchronous **R**eset*) e o PS (***P**roprietary **S**et*) ou AS (***A**synchronous **S**et*) como é mostrado na Figura 8-2 c). Este tipo de entradas estão normalmente presentes nos vários tipos de *flip-flops* disponíveis no mercado. A denominação ***P**roprietary **S**et* advém do facto de prevalecer a acção *Set*.



a) b) c)

Figura 8-2

a) *Flip-flop* tipo D *latch* com entradas *Set* e *Reset*, sintetizado a partir de um *flip-flop* SR
b) *Flip-flop* tipo D *edge-triggered* com entradas PS e CL, sintetizado a partir de dois *flip-flop* D *latch*.
c) Diagrama esquemático de um *flip-flop* tipo D *edge-triggered* com entradas PS e CL.

8.2 Sintetização de Contadores

Para a realização do contador binário, vamos introduzir um novo *flip-flop* denominado por tipo T, cujo comportamento se adequa ao objectivo pretendido.

8.2.1 *Flip-flop* tipo T (*Toggle*)

O *flip-flop* tipo T tem uma única entrada T, e tem o comportamento descrito no ASM da Figura 8-3, ou seja, quando a entrada T está a zero e existe uma transição ascendente na entrada de sincronismo (*clock*) o valor lógico da saída não se altera (permanece no estado). Quando a entrada T está a 1 e existe uma transição ascendente na entrada de sincronismo o *flip-flop* inverte o valor lógico presente na saída (transita de estado). Por comparação com o ASM do *flip-flop* D podemos concluir que $D = \overline{Q}T + Q\overline{T} = Q \oplus T$, obtendo-se a implementação da figura. Como se pode observar o *flip-flop* tipo T é um contador de um bit enquanto T igual a 1, o mesmo é dizer que é um divisor de frequência por 2 do sinal de *clock*.

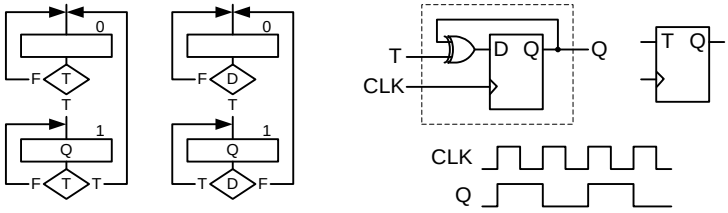


Figura 8-3

8.2.2 Contador Assíncrono

Na Figura 8-4 está representado um contador binário assíncrono de três bits, modo crescente.

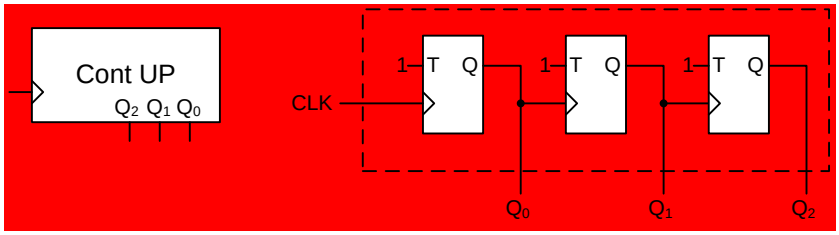


Figura 8-4

O contador diz-se assíncrono porque os vários *flip-flops* que o constituem não transitam em simultâneo. Como se pode observar, quando o contador passa de 011 (3) para 100 (4), a transição do *flip-flop* do extremo direito Q_2 só transita quando Q_1 transita e este por sua vez só transita quando Q_0 transita. Ou seja, quando o contador passa da configuração 3 para 4, primeiro passa pela configuração 010 (2) depois 000 (0) e finalmente 100 (4). É necessário considerar este facto quando se pretende gerar um sinal que indica que o contador atingiu uma certa configuração. Se por exemplo pretendêssemos activar um sinal de saída, aquando da passagem por 4, era necessário incluir o sinal de *clock* na conjunção, como mostra a Figura 8-5, caso contrário, esta saída também ficaria activa quando o contador passasse da configuração 7 para a configuração 0. Neste caso o tempo de activação da saída não seria um período do sinal CLK, mas um intervalo de tempo correspondente ao tempo de propagação do flip-flop Q_2, que dependendo da função por ele desempenhada poderia ser inconveniente.

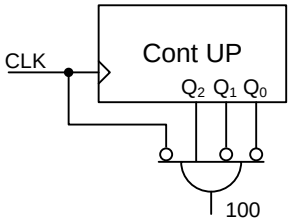


Figura 8-5

Este tipo de contador tem como vantagem apresentar uma estrutura simples e de baixo consumo dado que os vários *flip-flops* não transitam em simultâneo. Tem como desvantagem a limitação da frequência de *clock,* pois o período do sinal de *clock* não pode ultrapassar o tempo de propagação do conjunto dos *flip-flops*, razão pela qual tenha caído em desuso.

8.2.3 Contador Síncrono

O contador diz-se síncrono quando os vários *flip-flops* que o constituem transitam todos em simultâneo, ou seja, em sincronismo. A implementação deste tipo de contador, é baseada no controlo das entradas de um registo *edge-trigger*.

8.2.4 Contador crescente (UP)

A Figura 8-6 mostra o diagrama de blocos de um contador síncrono crescente baseado num registador de quatro bits, com entrada de controlo de contagem CE (*Count Enable*).

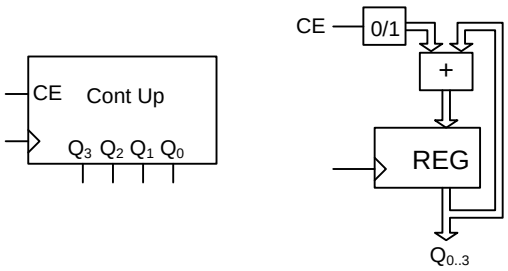


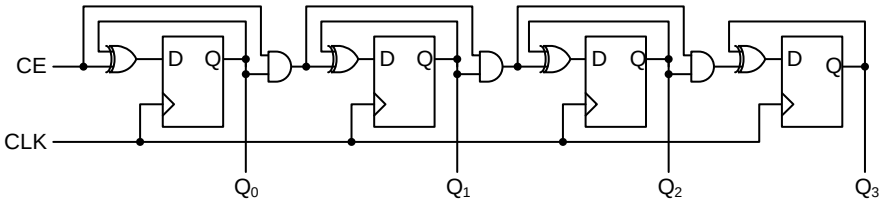
Figura 8-6

Para contar de forma crescente (UP) adicionamos o valor binário 1 ao valor presente no registador. Como o contador tem controlo de contagem CE, será esta entrada que quando activa adiciona o valor 1 ao valor registado e quando desactiva adiciona o valor 0, obtendo-se desta forma as seguintes expressões:

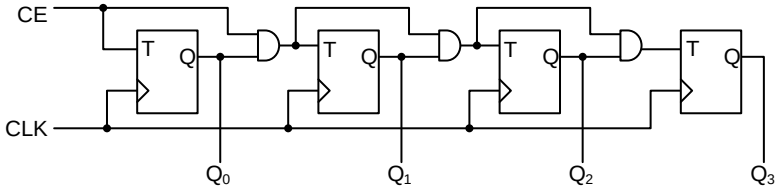
$$D_0 = Q_0 \oplus CE$$
$$Cy_1 = Q_0 . CE$$
$$D_1 = Cy_1 \oplus Q_1 = CE . Q_0 \oplus Q_1$$
$$Cy_2 = Cy_1 . Q_1 = (CE . Q_0) . Q_1$$
$$D_2 = Cy_2 \oplus Q_2 = (CE . Q_0) . Q_1 \oplus Q_2$$
$$Cy_3 = Cy_2 . Q_2 = ((CE . Q_0) . Q_1) . Q_2$$
$$D_3 = Cy_3 \oplus Q_3 = \Big(((CE . Q_0) . Q_1) . Q_2\Big) \oplus Q_3 = (CE . Q_0 . Q_1 . Q_2) \oplus Q_3$$

	Cy_3	Cy_2	Cy_1	
	Q_3	Q_2	Q_1	Q_0
+	0	0	0	CE
	D_3	D_2	D_1	D_0

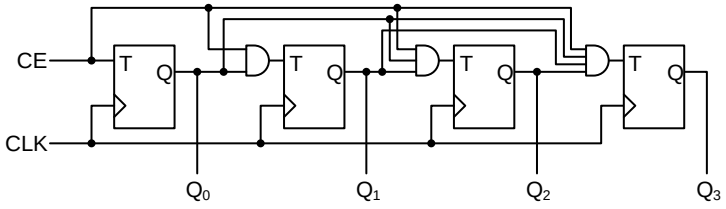
Na Figura 8-7 a) podemos observar a sintetização do contador UP com controlo de contagem, baseada em *flip-flops* do tipo D e que correspondem à implementação preconizada, ou seja, baseada num registo *edge-trigger* de quatro bits.
Na Figura 8-7 b) está sintetizado o mesmo contador recorrendo a *flip-flops* do tipo T e que apresenta como vantagem a utilização de menos lógica de concatenação. Esta implementação é denominada por contador modo série, isto porque o termo produto que determina a evolução de estado de cada *flip-flop* é constituído pela associação de todos os termos produto anteriores, dada a associatividade do produto lógico. O objectivo desta implementação é diminuir o número de entradas das portas AND utilizadas na concatenação. Esta implementação apresenta como desvantagem a limitação do período de CLK ao tempo de propagação de uma porta lógica AND multiplicada pelo número de *flip-flops* que constituem a cadeia.
A Figura 8-7 c) apresenta a mesma solução em modo paralelo. Esta implementação apresenta como vantagem, suportar uma maior frequência de *clock* relativamente à implementação série, isto porque a evolução de estado dos vários *flip-flops* da cadeia fica determinada após o tempo de propagação de uma porta AND.



a) Implementação com *flip-flop* tipo D



b) Implementação com *flip-flop* tipo T modo série



c) Implementação com *flip-flop* tipo T modo paralelo

Figura 8-7

8.2.5 Contador decrescente (*Down*)

Para contar *down*, subtraímos 1 ao valor registado. Considerando que o contador tem controlo de contagem CE será esta entrada que quando activa subtrai o valor 1 e quando desactiva subtrai o valor 0, obtendo-se assim o diagrama de blocos da Figura 8-8.

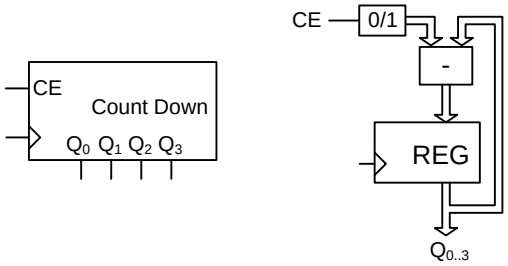


Figura 8-8

$$D_0 = Q_0 \oplus CE$$
$$Bw_1 = CE.\overline{Q}_0$$
$$D_1 = Bw_1 \oplus Q_1 = CE.\overline{Q}_0 \oplus Q_1$$
$$Bw_2 = Bw_1.\overline{Q}_1$$
$$D_2 = Bw_2 \oplus Q_2 = ((CE.\overline{Q}_0).\overline{Q}_1) \oplus Q_2$$
$$Bw_3 = Bw_2.\overline{Q}_2$$
$$D_3 = Bw_3 \oplus Q_3 = (((CE.\overline{Q}_0).\overline{Q}_1).\overline{Q}_2) \oplus Q_3$$

$Bw_3Bw_2Bw_1$

	Q_3	Q_2	Q_1	Q_0
-	0	0	0	CE
	D_3	D_2	D_1	D_0

Na Figura 8-9 podemos observar duas implementações do contador *Down*, uma recorrendo a *flip-flops* tipo D e que correspondem à implementação baseada num registo *edge-trigger* de quatro bits e outra recorrendo a *flip-flops* tipo T que utiliza menos lógica de concatenação.

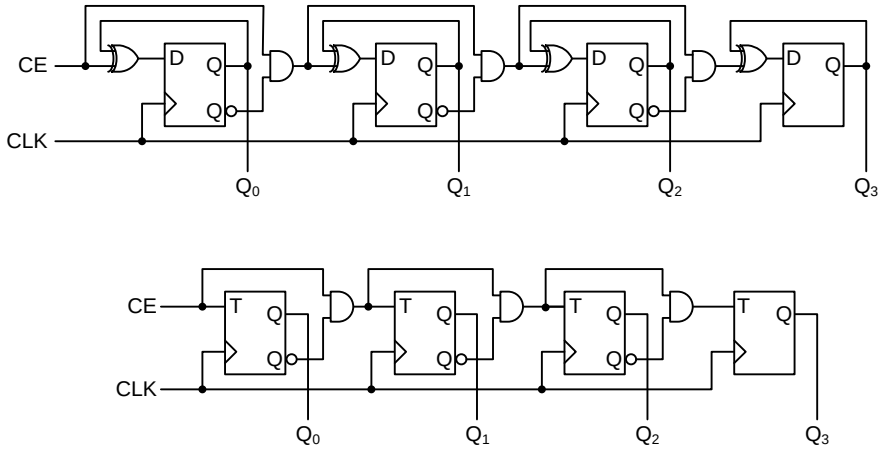


Figura 8-9

Como se pode observar na Figura 8-9, o contador tem uma estrutura muito semelhante ao do contador crescente. Comparando as duas estruturas, conclui-se que o contador realiza contagem crescente ou decrescente (UP ou DOWN) dependendo da saída que tomar para a adição (Q *ou* $\overline{Q}$).

8.2.6 Contador *UP/DOWN*

Pelo que foi observado anteriormente, podemos concluir que para sintetizarmos um contador binário que realize contagem crescente ou decrescente basta introduzir um multiplexer que, em função de um bit de controlo UnD, escolha a saída Q *ou* $\overline{Q}$.

Na Figura 8-10 está sintetizado o contador com as funcionalidades anteriormente estabelecidas e as quais são controladas pelos sinais UnD e CE.

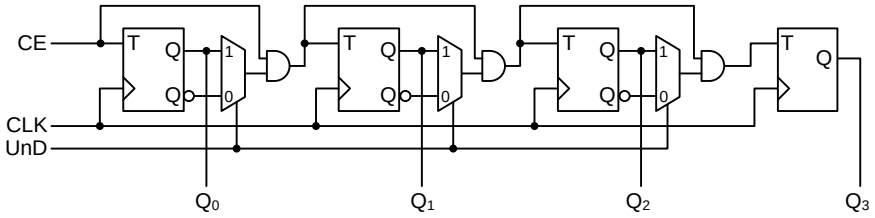


Figura 8-10

8.2.7 MR (*Master Reset*) e PL (*Parallel Load*)

O estabelecimento de um valor inicial de contagem, é uma funcionalidade de grande importância nos módulos de contagem. Com este objectivo existem dois sinais de entrada: o MR (*Master Reset*) que estabelece o valor zero em todos os *flip-flops* e o sinal PL (*Parallel Load*) que permite estabelecer um qualquer valor nos *flip-flops* que constituem o contador. Com a funcionalidade desempenhada pelo sinal PL, estes módulos passam a desempenhar duas funções: a de registo e a de contagem. Qualquer um dos dois sinais MR e PL inibem a contagem enquanto estão activos. Ambas as acções podem ter natureza assíncrona ou síncrona. Na Figura 8-11 está sintetizado um contador crescente de três bits com entrada MR e PL assíncronos. Como se pode observar na Figura 8-11 a entrada MR é prioritária relativamente à entrada PL.

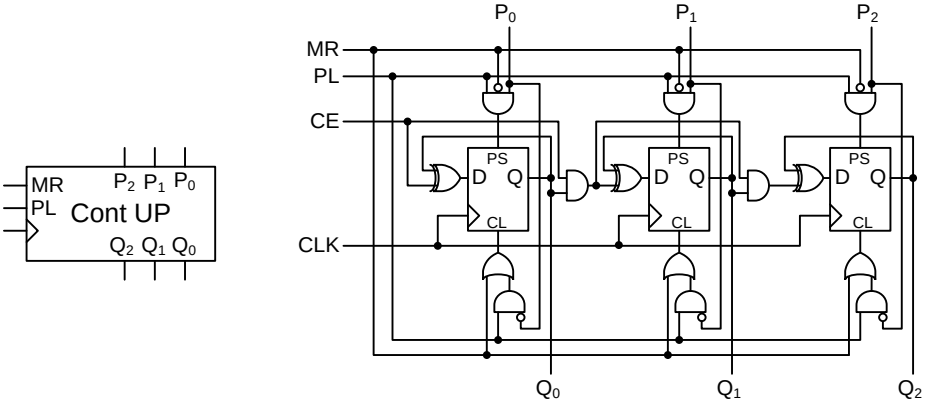


Figura 8-11

Na Figura 8-12 está sintetizado um módulo contador/registador de 3 bits com modo de contagem UP/Down e em que as acções de PL e MR têm natureza síncrona.

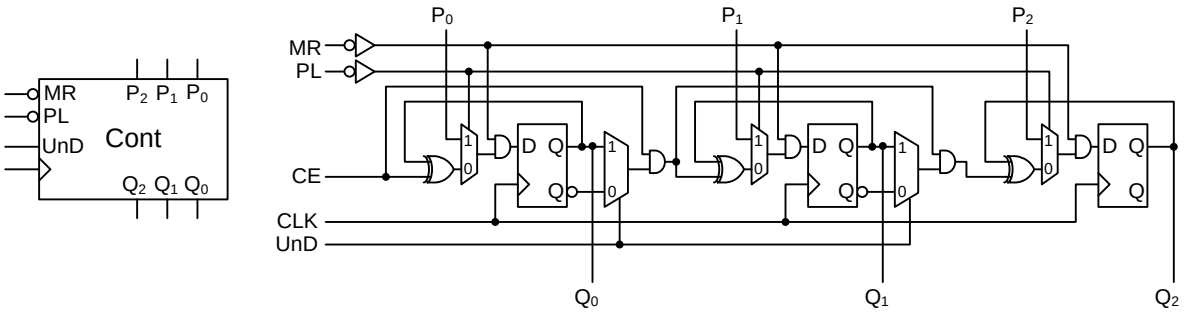


Figura 8-12

É de notar que na maioria dos dispositivos contadores disponíveis no mercado a acção MR tem normalmente natureza assíncrona, isto para permitir colocar a zero as saídas do módulo sem que exista CLK, no momento em que este é alimentado (*Power On Reset*) como mostra a Figura 8-13. Esta é a razão também pela qual estas entradas são normalmente activadas com zero. Quando é estabelecida a alimentação o condensador C está descarregado ficando a realizar MR. Esta saída é desactivada após o tempo de carga do condensador através da resistência R. O díodo D permite descarregar rapidamente o condensador C logo que a alimentação é desligada, permitindo assim, que se realize MR mesmo que exista uma falha de alimentação durante um curto espaço de tempo.

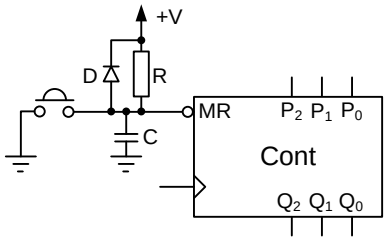


Figura 8-13

8.2.8 Limite de contagem

Os módulos contadores disponíveis no mercado disponibilizam saídas que indicam que o contador atingiu uma configuração extrema, ou seja, todos os flip-flops com o valor lógico 1 se a contagem é crescente ou zero se a contagem é decrescente. Na Figura 8-14 está sintetizado um contador Up/Down que põe disponível uma saída TC (*Terminal Count*). Para permitir a sintetização de contadores de n bits por concatenação assíncrona de módulos idênticos, é disponibilizada uma saída denominada RC (*ripple clock*).

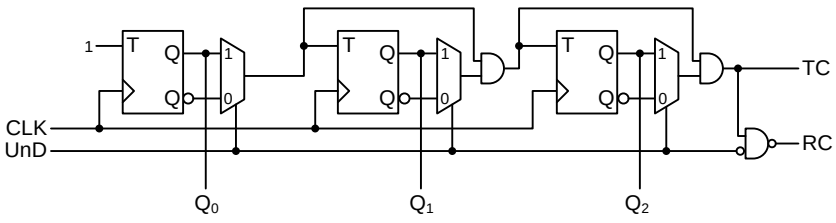


Figura 8-14

8.2.9 Exemplos de aplicação

Exemplo 1:

O exemplo que se segue corresponde à descrição em CUPL de um contador de 4 bits módulo 16, com entrada de CE (*Count Enable*) e saídas Max (*Maxim*) e TC (*Terminal Count*) registado. A implementação do contador utiliza como célula de memória *flip-flops* do tipo T.

A titulo de exemplo, é utilizado um pino de I/O como entrada e é reutilizado o *flip-flop* da macro-célula que lhe está associado para registar a passagem do contador por Max.

É de notar que devido à estrutura da PAL, o contador tem implementação paralela, ou seja, a entrada T de cada célula toma simultaneamente o estado de todas as células que lhe estão a montante. Dado que a saída TC é registada, é disponibilizado um sinal de entrada TCR para realizar a acção de *reset* do registo de TC.

```
Name     Cont16 ;
PartNo   00 ;
Date     27-03-2010 ;
Revision 02 ;
Designer JParaiso ;
Company  isel ;
Assembly None ;
Location  ;
Device   v750c ;

/*************** INPUT PINS *********************************************/
PIN    1 = CLK;        /* clock de contagem                             */
PIN    2 = CE;         /* entrada de Count Enable                       */
PIN    15= TCR;        /* Terminal Count Reset (utiliza um pino de I/O) */

/***************** OUTPUT PINS *****************************************************************/
PIN 14 = Max;                /* saída que indica que o contador atingiu o valor maximo        */
PINNODE  [25..28] = [Q0..3]; /* 4 flip-flops do registo                                       */
PIN 36 = TC;                 /* registo da passagem por Max. Utiliza a macro celula do PIN 15 */

Max=[Q0..3]:'h'F;

TC.AR=TCR;
TC.SP='b'0;
TC.CK= !CLK;
TC.D= !Max&TC # Max;

/* A solucao que a seguir se descreve para activação e registo do sinal TC e que corresponde ao
controlo do sinal de clock por um termo produto, nao deve ser utilizada, pois o sinal Max pode
surgir na passagem de configurações que correspondam à mudança de todos os bits, por exemplo do
valor 7 para o valor 8.

TC.CK= Max;
TC.D= 'b'1;
*/

[Q0..3].AR='b'0;
[Q0..3].SP='b'0;
[Q0..3].CK= CLK;

Q0.t=CE;
Q1.t=Q0&CE;
Q2.t=Q0&Q1&CE;
Q3.t=Q0&Q1&Q2&CE;
```

Exemplo 2:
Realizar um divisor de frequência por 6. A Figura 8-15 mostra duas soluções, sendo que b) apresenta um *duty cycle* de 50%. A entrada PL em ambos os contadores tem natureza síncrona. Um esquema idêntico ao da figura a) pode ser utilizado como temporizador.

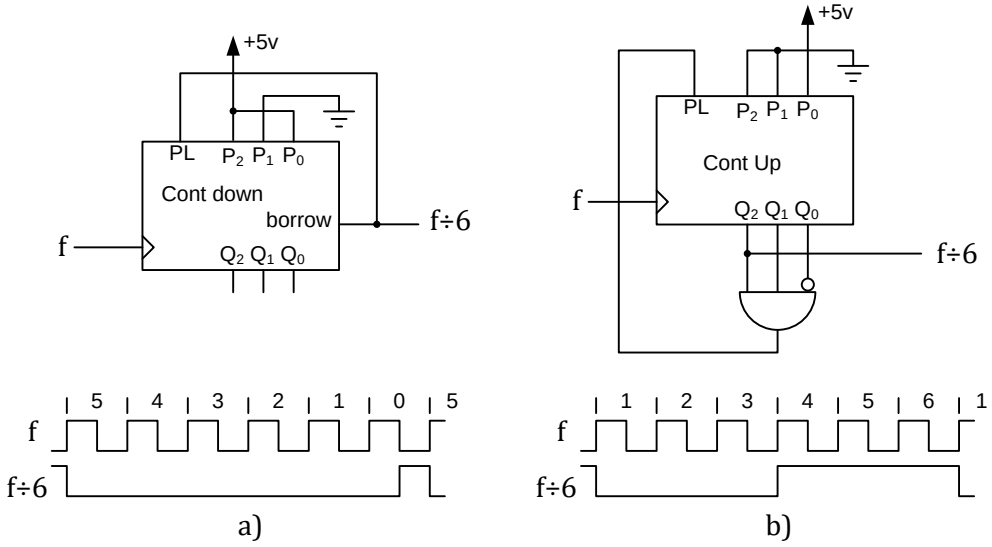


Figura 8-15

Exemplo 3:
O exemplo que se segue, mostra a utilização do dispositivo contador em duas funcionalidades: uma como elemento temporizador e outra como contador de acontecimentos.

Pretende-se implementar um sistema de controlo de abertura de uma fechadura electromecânica accionada por um botão **B** segundo o esquema seguinte:

- Ao premir o botão **B** acende-se uma lâmpada **L** durante 6 segundos;
- Durante o intervalo de tempo em que a lâmpada **L** está acesa, é necessário actuar o botão **B** um número **n** de vezes. O valor de **n** é estabelecido através de um *Thumb Wheel Switch*, com **n** compreendido entre 1 e 8;
- Findo os 5 segundos a fechadura **F** é accionada se o número de actuações de **B** coincidir com o valor **n** pré estabelecido. Como a fechadura é electromecânica é necessário que esta seja activada pelo menos durante 2 segundos.

Uma solução possível passa por utilizar dois contadores, um para contabilizar os tempos 2 e 6 segundos e outro para contabilizar o número de actuações do botão **B**. Na Figura 8-16 é mostrado o diagrama de blocos da solução preconizada. Como a frequência utilizada para contabilizar os tempos de espera é de 2Hz, os tempos de 6 e 2 segundo têm uma precisão de meio segundo.
Em ambos os contadores, as entradas PL têm natureza assíncrona. Os contadores podem ser assíncronos, pois a informação que é utilizada pelo ASM é a saída *borrow* que, como vimos anteriormente, é interceptada com o sinal de *clock* e só fica activa quando o *clock* está a zero. O

objectivo de manter activo o sinal L5 ainda no estado 1, é garantir *hold time* à acção de *load* do contador de tempo, pois de outra forma, a passagem do estado 0 para o estado 1 implicava retirar PL e simultaneamente alterar o valor na entrada P_3.

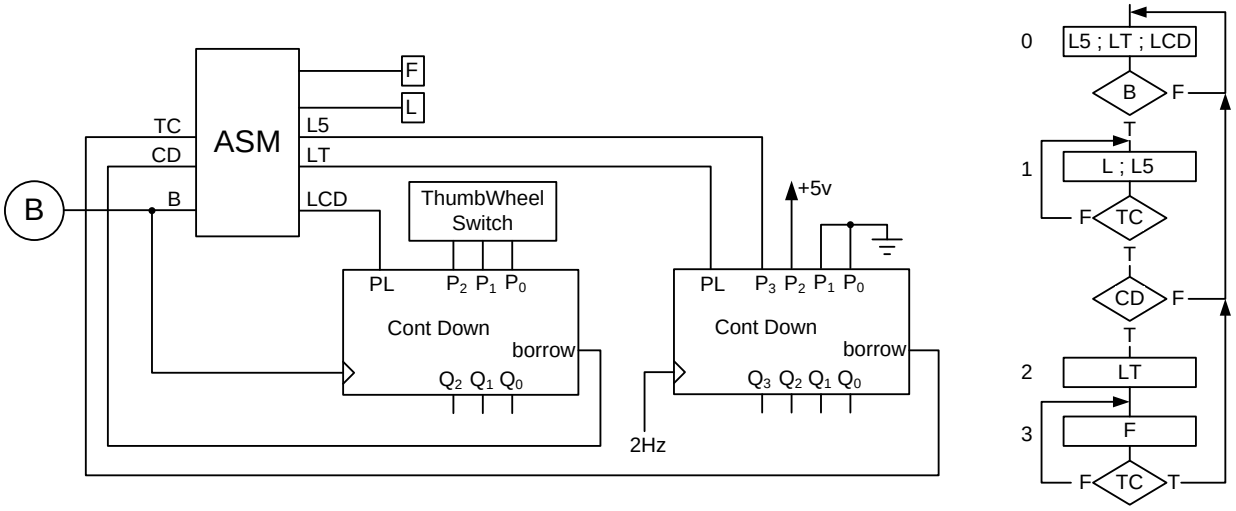


Figura 8-16

8.3 *Shift-Register*

O *shift-register* ou registador de deslocamento, tal como o nome indica, é um registo de n bits que a cada transição ascendente de *clock* desloca de um bit a informação nele registado. Ou seja, o bit de peso k toma o valor lógico do bit de peso k-1. Na Figura 8-17 está representado o esquema bloco de um *shift-register* com as entradas e saídas que lhe são mais características.

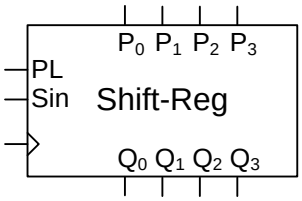


Figura 8-17

O *shift-register* para além da entrada de dados $P_{0\text{-}3}$ tem mais duas entradas. Uma de controlo com característica síncrona denominada PL (*Parallel Load*), que determina duas acções: quando activa determina que a acção a realizar no momento da transição ascendente de *clock* é de carregamento das entradas $P_{0\text{-}3}$ em $Q_{0\text{-}3}$; quando desactiva determina deslocamento (*shift*) dos bits $Q_{0\text{-}3}$ no sentido dos bits de menor peso para os de maior peso, ou seja, Q_1 toma o valor de Q_0, Q_2 toma o valor de Q_1 e assim sucessivamente.

A outra entrada é de dados, denominada por Sin (***S**erial **in**put*), e que determina o valor lógico a ser inserido no *flip-flop* Q_0 (que corresponde ao *flip-flop* inicial da cadeia).

Na Figura 8-18 está sintetizado um *shift-register* de 4 bits utilizando *flip-flop*s tipo D *edge-triggered*.

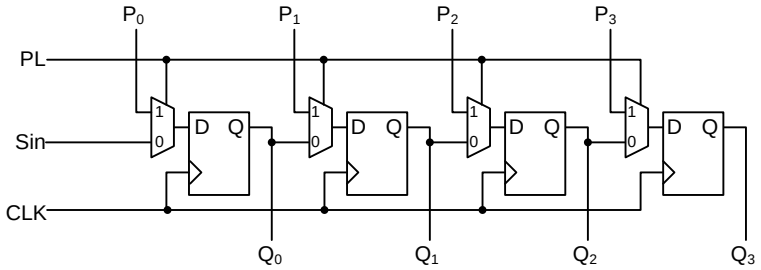


Figura 8-18

Dada que no esquema da Figura 8-18 a informação se desloca da esquerda para a direita, é usual referir esta funcionalidade como *shift-right*. O bit Q3 é considerado o bit de saída do *shift-right* e é normalmente denominado por Serial Output (Sout).

8.3.1 Exemplos de aplicação

8.3.1.1 Temporizador

Uma possível utilização do *shift register* é a implementação de temporizadores. Na Figura 8-19 está representado um temporizador de 5 segundos, configurável entre 1 e 6 segundos através de um *jumper*.

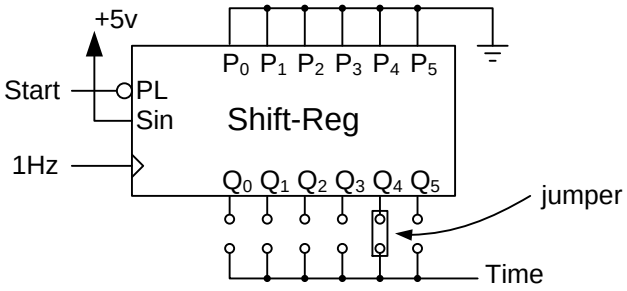


Figura 8-19

8.3.1.2 Transmissor série

Talvez a aplicação mais importante do *shift-register* seja a comunicação em série de informação. A Figura 8-20 mostra como dois sistemas podem trocar informação através de duas linhas, uma de dados e outra de sincronismo, convertendo informação paralela em informação série e vice-versa.
Diz-se que a informação é transmitida em paralelo quando vários bits são disponibilizados em simultâneo através de várias linhas de dados. Diz-se que a informação é transmitida em série quando os vários bits são disponibilizados numa sequência temporal através de uma única linha de dados.

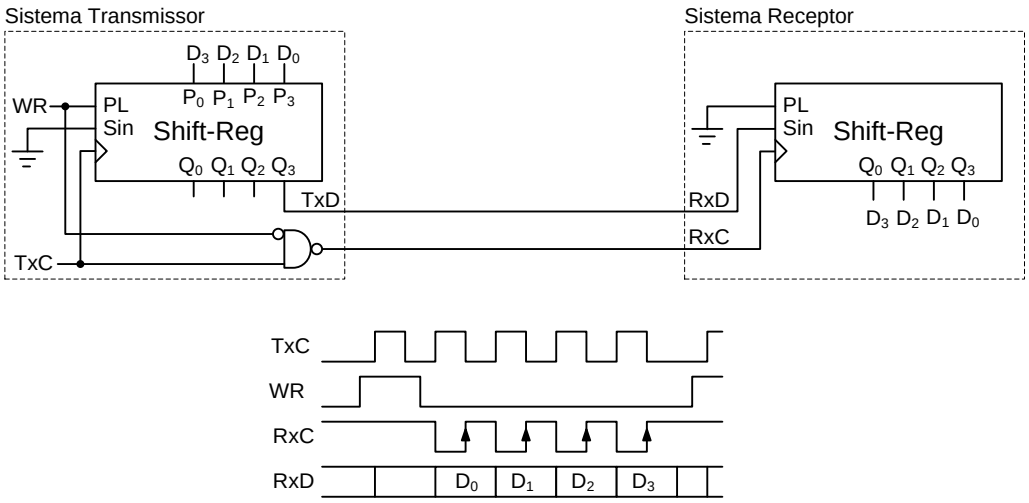


Figura 8-20

8.3.1.3 Somador Série

Outro exemplo de utilização do *shift-register* é o somador série de dois operandos A e B de n bits mais C_{in}. Esta solução embora mais lenta que a solução combinatória apresenta como vantagem a utilização de menos hardware. Na **Figura 8-21** podemos ver uma possível solução. A cada *clock* um bit do operando A é somado com um bit do operando B e registado o resultado em A. Caso seja produzido arrasto este é registado no *flip-flop* D, para posteriormente ser adicionado conjuntamente com os bits de peso seguinte do operando A e B. O resultado da adição estará presente em R e C_{out} ao fim de quatro períodos de *clock*.

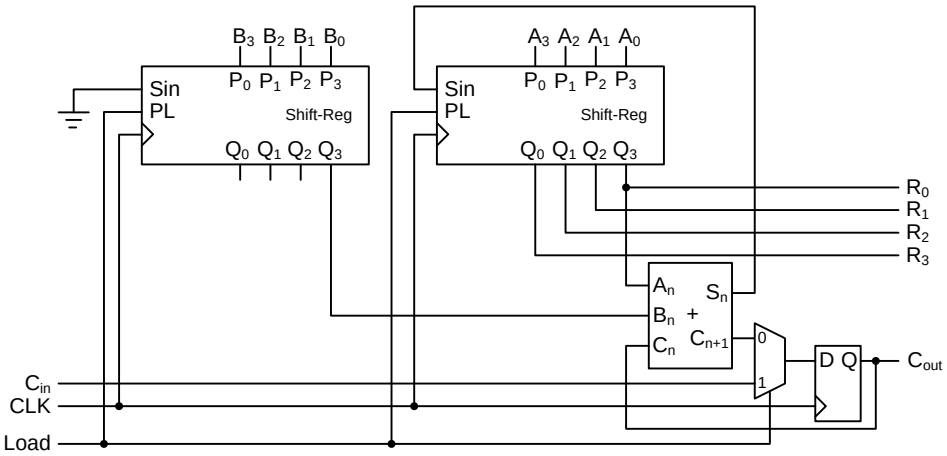
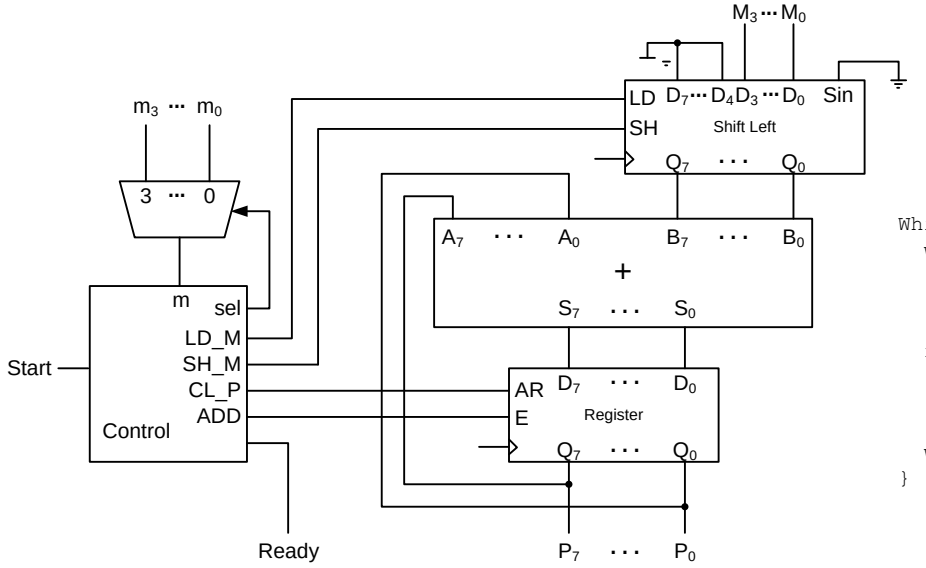


Figura 8-21

8.3.1.4 *Multiplicador add-shift*

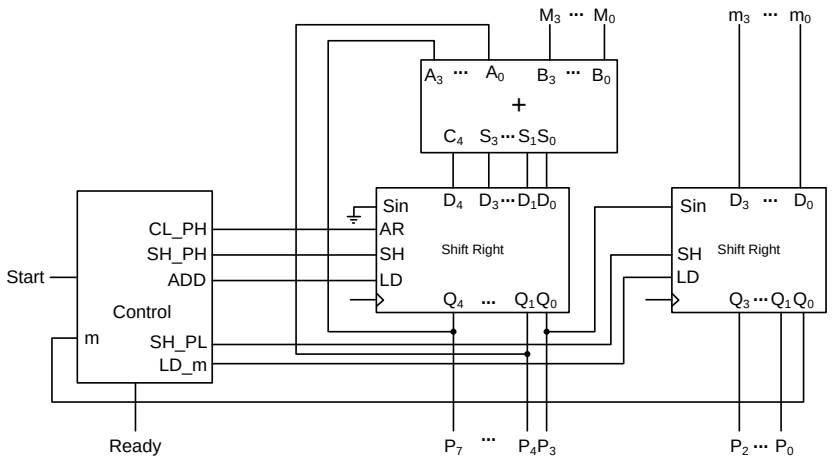
Outro exemplo de grande importância na utilização do *shift-register* é o multiplicador utilizando o algoritmo *add-shift* (ver Cap 4-16). Na Figura 8-22 podemos observar o diagrama de blocos de um multiplicador de dois operandos **M** e **m** ambos de 4 bits. A estrutura é baseada num *shift left register* de 8 bits e num somador-acumulador igualmente com 8 bits.




```
While (true) {
  while (!start) {
    LD_M;
    CL_P;
  }
  for (sel=0;sel<=3;++sel){
    if (m) ADD; //P=P+M
    SH_M;//M=M*2
  }
  while (start) ready;
}
```

Figura 8-22

Na Figura 8-23 é apresentada uma arquitectura optimizada do multiplicador e que consiste em deslocar o produto **P** para a direita em vez do operando **M** para a esquerda. Esta estrutura tem como vantagens: um somador de 4 bits em vez 8, menos um *clock* nas iterações com adição e a utilização dos 3 bits de menor peso do produto **P** para armazenar temporariamente o valor de **m**.
O teste aos bits de **m**, para determinar se existe adição, é realizado no bit de saída (*Serial out*) do *shift-register* do operando **m/P** evitando-se assim um multiplex. É de notar, que no *shift-register* da parte alta de **P** (P_3 a P_7), caso se realize soma, existe um deslocamento implícito de **P** para a direita por troca de ligações, ou seja, (P_3 a P_7) recebe a soma de (P_4 a P_7) com (M_0 a M_3).




```
While (true) {
  while (!start) {
    LD_m;
    CL_PH;
  }
  for (i=0;i<=3;++i){
    if (m)
      SH_PL; ADD;  //P=(P/2)+M
    else
      SH_PL; SH_PH; //P=P/2
  }
  while (start) ready;
}
```

Figura 8-23