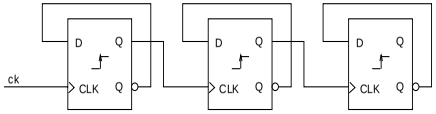
Capítulo 5

Análise e Síntese de Circuitos Sequenciais Síncronos

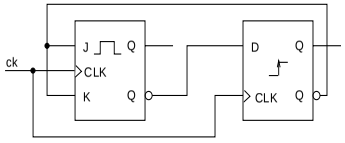
Neste capítulo apresentaremos uma forma de análise e de síntese de circuitos sequenciais (ou máquinas sequenciais, ou máquinas de estados finitos) síncronos.

Por **máquinas sequenciais síncronas** entendem-se as máquinas sequenciais onde os elementos de memória (*flip-flops* e *latches*), num número finito, são claramente identificáveis e utilizam um sinal de relógio comum, que sincroniza os eventos a que está sujeita a máquina.

Exemplo 5.1 *O seguinte circuito é assíncrono, porque o sinal de relógio dos flip-flops ligados em cascata não é o mesmo. Mais precisamente, o sinal de entrada de um é o sinal de saída do anterior.*



O circuito abaixo é, por sua vez, uma máquina síncrona, mesmo que as mudanças de estado ocorrem em diferentes bordas do sinal de relógio (comum) – no primeiro, na borda de descida e no segundo na borda de subida.



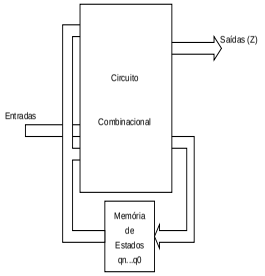
5.1 Modelos de Máquinas de Estados

Uma máquina de estado é composta de três blocos funcionais:

Lógica do próximo estado: conjunto de portas lógicas (circuito combinacional) que geram, a partir das entradas e/ou estado corrente, os **sinais de excitação** necessárias para mudar a máquina para o próximo estado;

Memória de estados: é um conjunto de n *flip-flops* que armazena o estado corrente da máquina, um dos 2^n distintos estados que a máquina consegue armazenar.

Lógica de Saídas: conjunto de portas lógicas (circuito combinacional) que determina a saída do circuito em função do estado corrente e, às vezes, também das entradas.

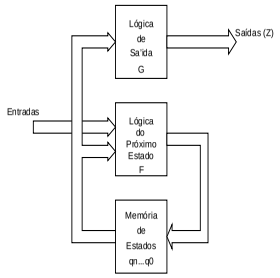


Há essencialmente dois modelos de máquinas de estados: modelo de Moore e modelo de Mealy.

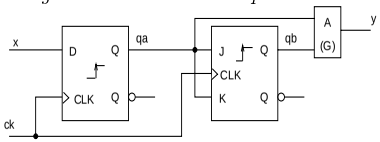
Um circuito sequencial cujas saídas só dependem do estado corrente da máquina, isto é

$$Z(t) = G(q(t)),$$

é conhecido como **máquina de Moore**.



Exemplo 5.2 *O seguinte circuito é uma máquina de Moore*

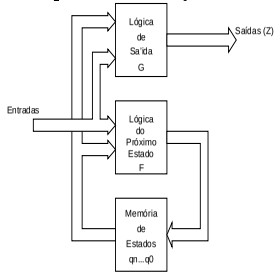


pois a saída $y = G(q_b)$ *é uma função do estado corrente* q_b.

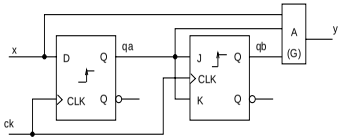
Quando as saídas dependem também das entradas, ou seja,

$$Z(t) = G(q(t), entradas),$$

dizemos que ela é uma **máquina de Mealy**.



Exemplo 5.3 *O seguinte circuito é uma máquina de Mealy*



pois a saída $y = G(q_b, x)$ *é uma função do estado corrente* q_b *e da entrada* x.

A única diferença entre os dois modelos reside na forma como as saídas são geradas. A mudança de estados em ambos os modelos pode ser expressa através da expressão

$$q^* = F(q, entradas).$$

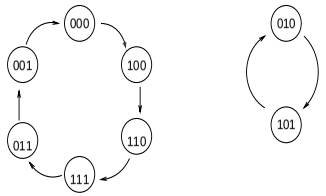
Na prática, muitas máquinas podem ser classificadas como máquinas de Mealy, porque elas possuem uma ou mais saídas do tipo *Mealy*. Entretanto, muitas destas máquinas também possuem uma ou mais saídas do tipo *Moore* que dependem somente dos estados correntes.

Nos projetos de máquinas de alto desempenho, é comum assegurar que as saídas sejam disponíveis o mais cedo possível e que elas sejam mantidas durante um período do relógio. Uma forma de conseguir isso é codificar nas variáveis de estado as saídas (*output-coded state assignment*), resultando em máquinas de Moore com as saídas Z(t) nulas.

5.2 Diagrama de Estados

Vimos no capítulo anterior que com uso de tabelas de estados, de tabelas de transições e/ou de tabelas de saídas, podemos descrever as relações entre entradas, estados correntes, saídas e próximos estados. Uma outra ferramenta gráfica bastante utilizada para representar estas relações são os diagramas de estado. Um **diagrama de estados**, também conhecido como **diagrama de transições**, é um grafo orientado, onde cada estado da máquina corresponde a um nó. De cada nó emenam p arcos orientados, correspondendo às transições de estados causadas pela ocorrência da entrada. Cada arco orientado é rotulado com a entrada que determina aquela transição e com a saída gerada.

Exemplo 5.4 *Da tabela do exemplo 5.5 consegue-se construir facilmente um diagrama de estados*



Vale, porém, ressaltar que, embora seja mais intuitiva uma representação por diagrama de estados, ela é mais sujeita a erros:

- uma tabela de estados lista de forma exaustiva todas as possíveis combinações de estados e entradas. Nenhuma ambiguidade pode ocorrer.
- um diagram de estados contém um conjunto de arcos e a cada arco é associada uma expressão de transição de estados em função das entradas e do estado corrente. Muitas vezes, é difícil garantir que uma dada expressão (um arco) cobre todas as possíveis combinações de entrada.

O diagrama de estado de uma máquina de Moore pode ser mais simples do que o de uma máquina de Mealy. Com um certo abuso de notação, podemos colocar o valor da saída correspondente a cada estado dentro do nó de estado, uma vez que ela é apenas expressa em função dos estados. Note, entretanto, que a tal saída é preservada na transição de um estado para o outro (ao longo do arco no diagrama de estado).

Ressaltamos ainda que através de manipulações apropriadas, pode-se mapear um modelo de máquina no outro. Por exemplo, podemos colocar na saída de uma máquina de *Mealy* um conjunto de *flip-flops* para armazenar as saídas e considerar o conteúdo destes *flip-flops* como varíaveis de estado.

5.3 Um Procedimento para Análise das Máquinas de Estado Síncronas

Para analisar um circuito sequencial síncrono **podemos** utilizar o seguinte procedimento:

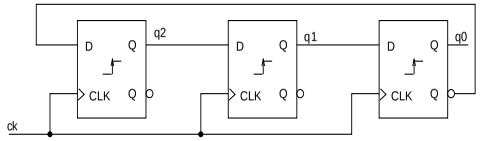
1. Determinar as **equações de excitação** de cada *flip-flop*.

2. Substituir as equações de excitação pelas equações características de cada *flip-flop* para obter **equações de transição**. Como já vimos no capítulo anterior, as equações características dos principais *latches* e *flip-flops* são

Tipos	Equações Características
Latch SR	$q^* = s + r' \cdot q$
Latch D	$q^* = d$
Flip-flop D	$q^* = d$
Flip-flop D com *enable* e	$q^* = e \cdot d + e'q$
Flip-flop SR sensível a pulso	$q^* = s + r'q$
Flip-flop JK sensível a pulso	$q^* = j \cdot q' + k' \cdot q$
Flip-flop JK sensível a borda	$q^* = j \cdot q' + k' \cdot q$
Flip-flop T	$q^* = q'$
Flip-flop T com *enable* e	$q^* = e \cdot q' + e'q$

3. Usar as equações de transição para construir a **tabela de transição**.
4. Determinar as **equações de saída** em função dos estados correntes e entradas.
5. Adicionar as saídas na tabela de transição para cada estado (máquina de Moore) ou para cada combinação de entradas/estado (máquina de Mealy) para estender a tabela de transições para **tabela de transições e saídas**.
6. Atribuir a cada combinação de variáveis de estado um estado para obter **tabela de estados/saídas**.
7. (Opcional) Desenhar o diagrama de estados correspondente à tabela de estados/saídas.

Exemplo 5.5 *Vamos analisar o seguinte circuito*



1. *Equações de excitação:* $d_2 = q_0'$, $d_1 = q_2$ *e* $d_0 = q_1$.

2. *Equações de transição: $q_2^* = q_0\prime$, $q_1^* = q_2$ e $q_0^* = q_1$, sendo a equação caracterísitca dos flip-flops D $q^* = d$.*

3. *A partir destas equações constrói-se a tabela de transições*

$q_2q_1q_0$	
000	*100*
001	*000*
010	*101*
011	*001*
100	*110*
101	*010*
110	*111*
111	*011*
	$q_2^*q_1^*q_0^*$

4. *Equações de saída: Como não há saída, pode-se omitir este passo.*

5. *Atualização da tabela de transições/saídas: Como não há saída, não há atualização.*

6. *Podemos, por exemplo, atribuir a cada estado uma identificação S_i*

Estado	
S_0	S_4
S_1	S_0
S_2	S_5
S_3	S_1
S_4	S_6
S_5	S_2
S_6	S_7
S_7	S_3
	Próximo estado

7. *(Opcional) A partir da tabela de estados é fácil desenhar o diagrama de estados correspondente.*

5.4 Um Procedimento para Síntese de Máquinas de Estado Síncronas

Algumas vezes, podemos implementar um dispositivo digital com uso se *flip-flops* de forma bastante intuitiva. Quando se trata de um sistema com um número grande de estados e a relação entre estes estados não ser trivialmente perceptível, deve-se adotar uma metodologia de projeto para evitar erros.

Um procedimento para síntese de um circuito sequencial é exatamento o inverso do procedimento de análise que vimos, começando com a especificação dos estados relevantes e as transições entre estes estados:

1. Construir a **tabela de estado/saídas** a partir da especificação funcional do problema.

2. (Opcional) **Minimizar** o número de estados na tabela de estado.

3. Definir as **variáveis de estado** e assinalar a cada estado da tabela de estados uma combinação das variáveis de estado.

4. Substituir os estados da tabela de estados/saídas pelas combinações das variáveis de estado para criar uma **tabela de transições**. Com isso, tem-se as mudanças que devem ocorrer em cada elemento de memória.

5. **Escolher os tipos de** *flip-flops* (p. ex., D ou JK) a serem utilizados na implementação.

6. Construir para cada *flip-flop* uma **tabela de excitação** em função dos estados correntes e das entradas, lembrando que

q^*	q	D	J	K	T
0	0	0	0	X	0
0	1	1	1	X	1
1	0	0	X	1	1
1	1	1	X	0	0

7. Derivar a partir de cada tabela de excitação a **equação (booleana) de excitação** requerida para o pino de entrada do *flip-flop* correspondente.

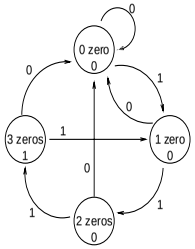
8. Derivar a partir da tabela de transições/saídas a **equação de saída**.

9. Esboçar o diagrama lógico do circuito a partir das equações de excitação e de saída, lembrando que todos os elementos de memória recebem o mesmo sinal de relógio.

Exemplo 5.6 *Projete um detector de sequência de 3 zeros consecutivos sem sobreposição.*

Primeiro, veremos a realização deste detector como uma máquina de Moore, ou seja, a saída y (indica se os 3 zeros foram detectados ou não) depende somente dos estados:

1. *Tabela de estado/saídas: Note que quando a sequência é detectada, a saída passa para 1; do contrário, ela permanece em 0.*



Estado	*x*		*Saída*
	0	*1*	
S1 (0 zero)	*S1*	*S2*	*0*
S2 (1 zero)	*S1*	*S3*	*0*
S3 (2 zeros)	*S1*	*S4*	*0*
S4 (3 zeros)	*S1*	*S2*	*1*
	Próximo estado		

2. *Definir as* **variáveis de estado***: São quatro estados (S1, S2, S3 e S4), portanto precisamos de 2 variáveis q_1q_0.*

3. *Tabela de transições/saídas: Atribuimos arbitrariamente os valores 00, 01, 11 e 10, que q_1q_0 podem assumir aos estados S1, S2, S3 e S4, respectivamente.*

q_1q_0	*x*		*z*
	0	*1*	
00	*00*	*01*	*0*
01	*00*	*11*	*0*
11	*00*	*10*	*0*
10	*00*	*01*	*1*
	$q_1^*q_0^*$		

4. *Escolher os tipos de flip-flops: dois flip-flops tipo D.*

5. *Tabelas de excitações:*

q_1q_0	*x*	
	0	*1*
00	*0*	*0*
01	*0*	*1*
11	*0*	*1*
10	*0*	*0*

q_1q_0	*x*	
	0	*1*
00	*0*	*1*
01	*0*	*1*
11	*0*	*0*
10	*0*	*1*

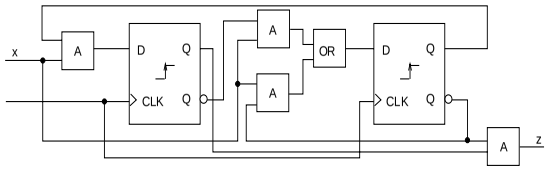
6. *Equações de excitação de cada flip-flop:*

   - $d_1 = x \cdot q_0$
   - $d_0 = x \cdot q_1\prime + x \cdot q_0\prime$

7. *Equação de saída:* $z = q_1 \cdot q_0\prime$

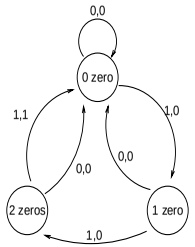
q_1q_0	*x*	
	0	*1*
00	*0*	*0*
01	*0*	*0*
11	*0*	*0*
10	*1*	*1*

8. *Esboçar o diagrama lógico do circuito a partir das equações de excitação e de saída, lembrando que todos os elementos de memória recebem o mesmo sinal de relógio.*



Segundo, veremos a realização deste detector como uma máquina de Mealy, ou seja, a saída y depende dos estados e da entrada:

1. *Tabela de estado/saídas: Note que, como antes, quando a sequência é detectada, a saída passa para 1; do contrário, ela permanece em 0. Observe ainda que, sendo a saída uma função das entradas, podemos economizar um estado: assim que um 1 for detectado com a máquina no estado de 2 zeros consecutivos, 1 é gerado na saída!*



Estado	*x*	
	0	*1*
S1 (0 zero)	*S1,0*	*S2,0*
S2 (1 zero)	*S1,0*	*S3,0*
S3 (2 zeros)	*S1,0*	*S1,1*
	Próximo estado,Saída	

2. *Definir as* **variáveis de estado***: São três estados (S1, S2 e S3), portanto precisamos de 2 variáveis q_1q_0.*

3. *Tabela de transições/saídas: Atribuimos arbitrariamente os valores 00, 01 e 11, que q_1q_0 podem assumir, aos estados S1, S2 e S3, respectivamente. A combinação binária 10 é irrelevante.*

q_1q_0	*x*	
	0	*1*
00	*00,0*	*01,0*
01	*00,0*	*11,0*
11	*00,0*	*00,1*
10	*X*	*X*
	$q_1^*q_0^*,z$	

4. *Escolher os tipos de flip-flops: dois flip-flops tipo D.*

5. *Tabelas de excitações:*

q_1q_0	*x*	
	0	*1*
00	*0*	*0*
01	*0*	*1*
11	*0*	*0*
10	*X*	*X*

q_1q_0	*x*	
	0	*1*
00	*0*	*1*
01	*0*	*1*
11	*0*	*0*
10	*X*	*X*

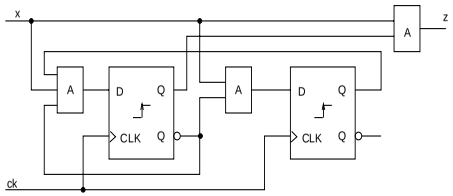
6. *Equações de excitação de cada flip-flop:*

   - $d_1 = x \cdot q_1\prime \cdot q_0$
   - $d_0 = x \cdot q_1\prime$

7. *Equação de saída:* $z = x \cdot q_1$

q_1q_0	*x*	
	0	*1*
00	*0*	*0*
01	*0*	*0*
11	*0*	*1*
10	*X*	*X*

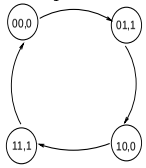
8. *Esboçar o diagrama lógico do circuito a partir das equações de excitação e de saída, lembrando que todos os elementos de memória recebem o mesmo sinal de relógio.*



5.5 Redução de Estados

Durante o projeto conceitual de um circuito, quando a sua descrição funcional é transformada num diagrama de estados ou numa tabela de estados, é comum a ocorrência de estados redundantes. Entende-se por **estados redundantes** ou **estados K-equivalentes** os estados que tem para toda possível sequência de K entradas a mesma sequência de saídas.

Exemplo 5.7 *Projeto um circuito que realiza o seguinte diagrama de estado:*



ou em termos de tabela de estado,

q_1q_0	*Após 1 pulso*
00	*01,0*
01	*10,1*
10	*11,0*
11	*00,1*
	$q_1^*q_0^*,z$

Observe que, independentemente da entrada x, os estados 01,11 $\in$ $\mathcal{A}$ *produzem a mesma saída, z=1, enquanto os estados 00,10* $\in$ $\mathcal{B}$ *geram a saída z=0. Em outras palavras, se a saída for 1, podemos afirmar que o estado corrente do circuito é 00 ou 11 e se for 0, o estado corrente é 00 ou 10. Dizemos, então, que os estados 01,11 e 00,10 são equivalentes em relação à saída ou* **1-equivalente**.

Vamos ver agora o comportamento do circuito para uma sequência de duas entradas.

q_1q_0	*Após 2 pulsos*
00	*01*
01	*10*
10	*01*
11	*10*
	2 Saídas

Novamente os estados 01,11 $\in$ $\mathcal{A}$ *possuem a mesma sequência de saídas, 01, e os estados 00,10* $\in$ $\mathcal{B}$ *a mesma sequência 10. Dizemos, então, que 01,11 e 00,10 são* **2-equivalentes**.

E para uma sequência de três entradas.

q_1q_0	*Após 3 pulsos*
00	*010*
01	*101*
10	*010*
11	*101*
	3 Saídas

Pela tabela é fácil ver que 01,11 e 00,10 são **3-equivalentes**.

Um outra maneira de dizer que dois estados S_i e S_j são equivalentes é quando para qualquer entrada I_p,

1. suas saídas são equivalentes (1-equivalente), e
2. transições para os próximos estados são equivalentes (para uma mesma entrada resulta-se um estado que pertence à mesma partição à qual o estado corrente pertence).

A remoção de estados redundantes é importante por várias razões:

Custo: vimos que o número de elementos de memória está diretamente relacionado com o número de estados.

Complexidade: maior o número de estados, maior o número de portas lógicas associadas para controlar a mudança de estado de cada elemento de memória.

Análise: a maioria das rotinas de auxílio à depuraçao e ao diagnóstico pressume que não haja redundância no circuito em análise.

Basicamente, existem três técnicas para determinar os estados equivalentes:

1. por inspeção
2. por partição, e
3. por tabelas de implicação.

Detalharemos as duas primeiras técnicas nas próximas seções.

5.5.1 Por Inspeção

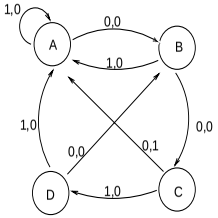
É a mais intuitiva e propensa a erros. Ela consiste simplesmente em analisar a tabela de estados para todas as possíveis combinações de entradas e eliminar as linhas dos estados redundantes. No exemplo 5.7 podemos, por exemplo, eliminar as linhas que correspondem, respectivamente, a 01 e 10.

5.5.2 Por Partição

Entende-se como **partição de equivalência** a definição de um conjunto de classes de equivalência, tal que estados em uma mesma classe tenham o mesmo comportamento para todas as possíveis combinações de entradas.

A ténica por partição envolve a determinação sucessiva de classes de equivalência de estados P_k, $K = 1, 2, 3, \cdots, l$, tal que os estados contidos em P_K são K-equivalentes. O procedimento encerra quando $P_l = P_{l+1}$ e dizemos que P_l é uma **partição de equivalência**.

Exemplo 5.8 *Seja o seguinte diagrama de estados que descreve a máquina de Mealy para um detector de 3 zeros consecutivos*



vamos reduzir os estados desta máquina com uso da técnica de partição. Transcrevendo o diagrama de estados na seguinte tabela de estados:

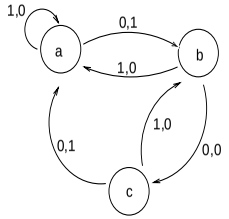
q_1q_0	x	
	0	*1*
A	*B,0*	*A,0*
B	*C,0*	*A,0*
C	*A,1*	*D,0*
D	*B,0*	*A,0*
	Próximo estado,z	

1. *Partição do tipo 1 (P_1): por* **saída** *de cada estado corrente conclui-se que podemos particionar os estados em dois conjuntos equivalentes A,B,D e C.*

2. *Partição do tipo 2 (P_2): numa partição do tipo P_k deve-se garantir que cada conjunto de equivalência só contém os estados cujos próximos estados, para uma dada entrada, pertencem aos mesmos conjuntos de equivalência da partição do tipo P_{k-1}. Os estados A e D satisfazem esta condição. Note, porém, que para a entrada x=1, o próximo estado de B é C que não pertence ao mesmo conjunto da partição P_1, então o conjunto A,B,D é dividido em dois conjuntos A,D e B.*

3. *Partição do tipo 3 (P_3): é fácil ver que para todas as posíveis entradas, $x = 0$ ou $x = 1$, os próximos estados dos estados de cada conjunto da partição P_2 caem no(s) mesmo(s) conjunto(s). Portanto, $P_3 = P_2$. Consequentemente, P_k=P_2, $\forall K > 2$ e P_2 é a partição de equivalência da máquina.*

Assim, pode-se reduzir a máquina numa máquina de três estados segundo o esquema

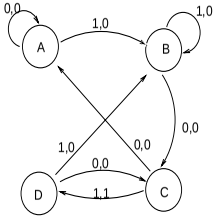


E, a partir deste diagrama "otimizado", pode-se utilizar o procedimento de síntese para implementar um circuito que o realiza.

Podemos ainda sintetizar, em forma tabular, os três passos de partição acima

Est.	*Entradas*	*Saída*	*Próx, est*	*Part. 1*	*Part. 2*	*Part. 3*
A	*0,1*	*0,0*	*A,B*	*1*	*1*	*1*
B	*0,1*	*0,0*	*A,C*	*1*	*3*	*3*
C	*0,1*	*1,0*	*A,D*	*2*	*2*	*2*
D	*0,1*	*0,0*	*B,A*	*1*	*1*	*1*

Exemplo 5.9 *Seja o seguinte diagrama de estados que descreve a máquina de Mealy para um detector da sequência 101*

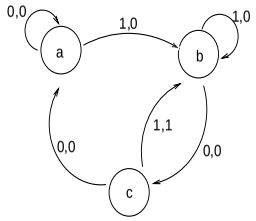


que corresponde à seguinte tabela de estados

q_1q_0	x	
	0	*1*
A	*A,0*	*B,0*
B	*C,0*	*B,0*
C	*A,0*	*D,1*
D	*C,0*	*B,0*
	Próximo estado,z	

1. *Partição do tipo 1 (P_1): A,B,D e C.*
2. *Partição do tipo 2 (P_2): A, B,D e C.*
3. *Partição do tipo 3 (P_3): $P_3 = P_2$. Portanto, P_2 é a partição de equivalência.*

Assim, pode-se reduzir a máquina numa máquina de três estados segundo o esquema



E, a partir deste diagrama "otimizado", pode-se utilizar o procedimento de síntese para implementar um circuito que o realiza.