6. SÍNTESE: MdE SÍNCRONA COMPLETAMENTE ESPECIFICADA

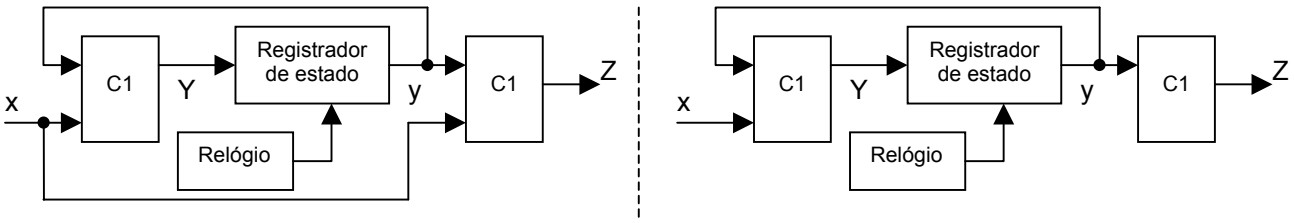
A realização de circuitos seqüenciais síncronos consiste em um conjunto de células binárias de memória (Flip-Flops) para armazenar o estado (estado presente) e os circuitos combinacionais para realizar as funções de saída e de transição (próximo estado).

Uma forma-padrão para todos os circuitos seqüenciais é a **implementação canônica** (também chamada de **implementação de Huffman-Moore**), que consiste diretamente na descrição de estados de um sistema:

- 'r' equações de próximo estado $\rightarrow Y_i = \delta_i(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_r)$
- 'm' equações de saída $\rightarrow Z_j = \omega_j(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_r)$ {modelo Mealy}
  $Z_j = \omega_j(y_1, y_2, \ldots, y_r)$ {modelo Moore}

Esta implementação, como pode ser vista nas figuras dadas abaixo, consiste em:

- um **registrador de estado** para armazenar o **estado atual** (y); e
- um conjunto de **circuitos combinacionais** para implementar as equações lógicas de **saída** (Z) e de **transição** (Y) (ou **estado futuro**).



(a) Máquina de Mealy (b) Máquina de Moore

É bom lembrar que qualquer sistema seqüencial pode ser implementado utilizando-se de uma máquina de Mealy ou de uma máquina de Moore. Estes sistemas diferem levemente como ilustra a figura acima. A máquina de Mealy difere da máquina de Moore apenas na função de saída!

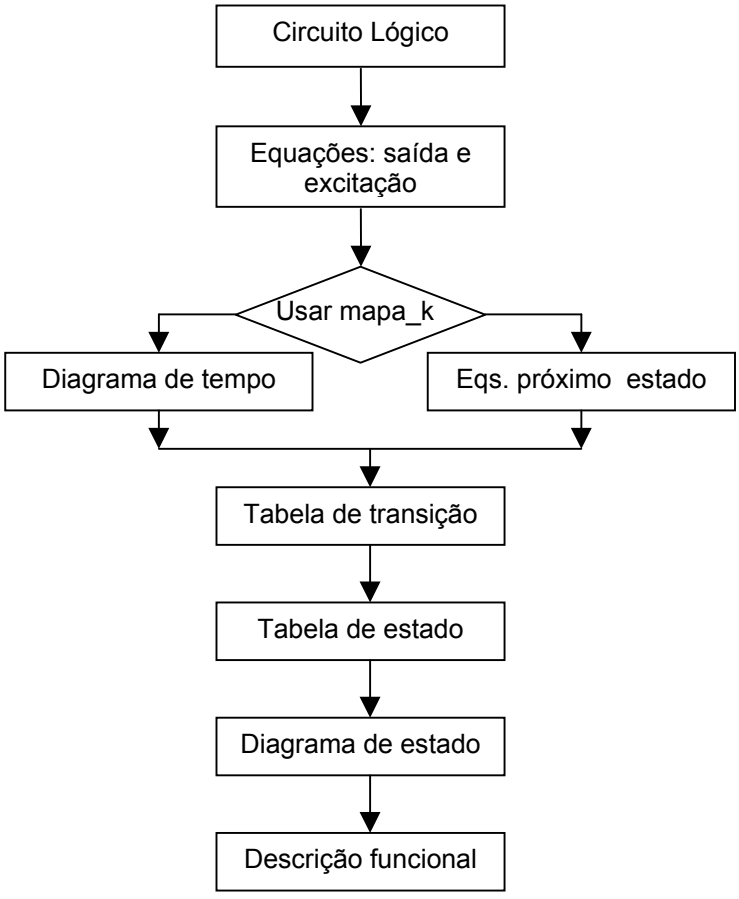
6.1. O Procedimento da Síntese

De uma maneira geral, o procedimento da síntese para um circuito lógico seqüencial envolve seis passos. Ele se apresenta como sendo o caminho inverso ao procedimento da análise. O procedimento da síntese deve seguir os seguintes passos:

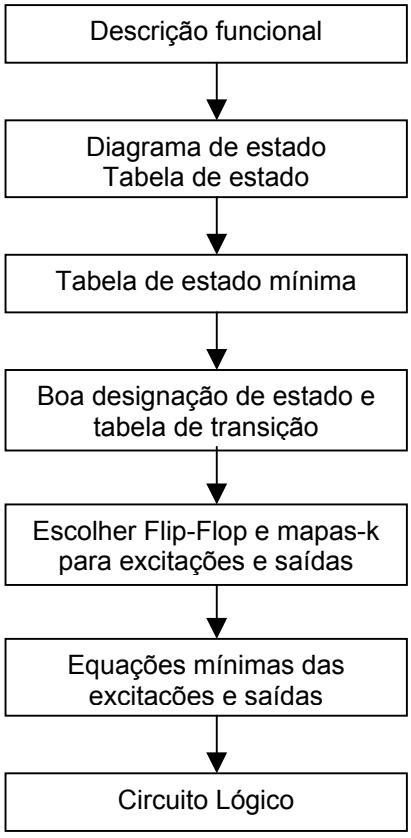
1. Da descrição por palavras do comportamento funcional do sistema digital a ser projetado, obter o **diagrama de estado** e em seguida a **tabela de estado**;

2. Usar as técnicas de **redução de estado** para chegar a uma máquina de estado mínima equivalente;
3. Estabelecer uma **boa designação** de estado e gerar as tabelas de transição e, a partir daí, obter as tabelas de próximo estado para cada um dos elementos de memória (Flip-Flops);
4. Escolher o tipo de Flip-Flop a serem usado e construir os mapas_k para cada uma das equações de excitação dos Flip-Flops;
5. A partir dos mapas_k das excitações dos Flip-Flops e dos mapas_k das saídas achar as equações lógicas mínimas do circuito;
6. Desenhar o diagrama lógico do circuito a partir das equações lógicas correspondentes.

Para melhor esclarecer os passos do procedimento da síntese, bem como o da análise, de um circuito lógico seqüencial síncrono, segue os seguintes fluxogramas:



(a) Procedimento da **Análise**



(b) Procedimento da **Síntese**

No que se refere ao procedimento da síntese, verifica-se que:

- O primeiro passo daquela seqüência é de natureza totalmente intuitiva, requerendo assim algumas tentativas e uma certa experiência para executá-lo corretamente;
- Os passos 5 e 6 são bastantes conhecidos das técnicas combinacionais;
- Os passos 2, 3 e 4 serão vistos com bastante detalhes mais adiante.

Vale ressaltar ainda, que o passo 2 possui uma grande diferenciação, conforme a síntese seja orientada para circuitos lógicos seqüenciais **completamente** ou **incompletamente** especificados.

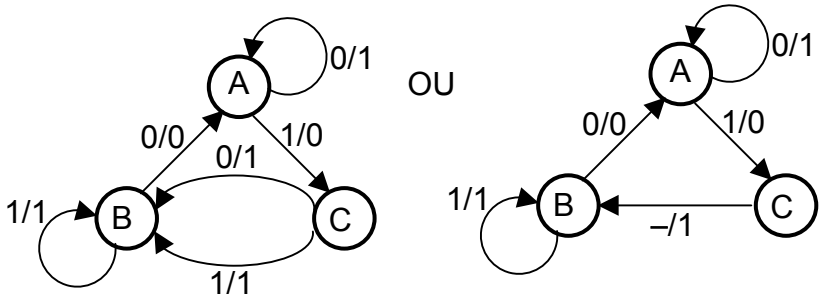
6.2. Especificação dos Circuitos Lógicos Seqüenciais

Considerar-se-á alguns exemplos, tanto para máquinas de estado completamente especificadas, como para as não completamente especificadas.

6.2.1. Circuitos Seqüenciais Completamente Especificados

Circuitos seqüenciais completamente especificados, são aqueles nos quais todos os pares, ***estado seguinte* / *saída***, são inteiramente definidos.

- Exemplo de uma MdE completamente especificada.

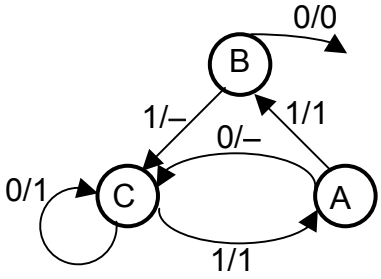


\ x y \	0	1
A	A/1	C/0
B	A/0	B/1
C	B/1	B/1

6.2.2. Circuitos Seqüenciais Incompletamente Especificadas

Circuitos seqüenciais não completamente especificados, são aqueles nos quais pelo menos um dos pares, ***estado seguinte* / *saída***, possui um dos elementos ou ambos não especificado.

- Exemplo de uma MdE não completamente especificada.



\ x y \	0	1
A	C/–	B/1
B	–/0	C/–
C	C/1	A/1

6.3. Redução de Estado em MdE Completamente Especificada

Quando se pretende minimizar o número de estados em um circuito seqüencial completamente especificado, o conceito de ***equivalência de estado*** é empregado.

DEFINIÇÃO:

Dois estados q_i e q_j, de um circuito seqüencial completamente especificado, são ditos serem ***equivalentes*** ($\equiv$) se, para toda e qualquer seqüência de entrada possível, é produzida a mesma seqüência de saída, independentemente de ser q_i ou q_j o estado de origem.

Da definição acima tem-se o seguinte teorema:

TEOREMA:

Sejam q_l e q_k os estados destinos de uma máquina M completamente especificada ao ser aplicada uma seqüência de entrada 'x' quando seus estados de origem eram q_i e q_j respectivamente. Logo, q_i e q_j são ditos eqüivalentes, se e somente se, para qualquer seqüência de entrada 'x' possível, tem-se:

1. A seqüência de saída produzida a partir do estado q_i for igual a seqüência de saída produzida a partir do estado q_j. Pode-se escrever matematicamente como:
   $\omega(x,q_i) = \omega(x,q_j) \quad \rightarrow$ modelo de Mealy
   $\omega(q_i) = \omega(q_j) \quad \rightarrow$ modelo de Moore

2. Os estados de destino q_l e q_k serem equivalentes, ou seja:
   $q_l = \delta(x,q_i) \equiv \delta(x,q_j) = q_k$ para qualquer um dos modelos.

Necessidades:

1. Se $\delta(x,q_i) \neq \delta(x,q_j)$, então $q_i \neq q_j$.
2. Se $q_l \neq q_k$, então existe pelo menos uma seqüência de entrada tal que as seqüências de saída a partir de q_l e q_k e, consequentemente a partir de q_i e q_j, serão diferentes.

Dois estados equivalentes ($\equiv$) respeitam sempre as seguintes propriedades:

- ***Simetria***: $A \equiv A$
- ***Reflexibilidade***: $A \equiv B \Rightarrow B \equiv A$
- ***Transitividade***: $A \equiv B$ e $B \equiv C \Rightarrow A \equiv C$

Basicamente existem três métodos de determinação de estados equivalentes:

1º) INSPEÇÃO;
2º) PARTIÇÕES SUCESSIVAS;
3º) TABELA DE IMPLICAÇÃO.

6.3.1. Simplificação de estados através da Inspeção

A Inspeção não é bem um método para minimizar o número de estados de um circuito seqüencial, ele simplesmente se baseia na verificação da tabela de estado. Se por observação, notar-se que duas linhas de uma dada tabela de estado são exatamente iguais, conclui-se facilmente que estes dois estados são equivalentes, logo uma das linhas (estado) pode ser removida.

Exemplo: Por ***inspeção***, ver-se que as linhas do estado 'B' e do estado 'D' são idênticas, logo o estado 'B' é equivalente ao estado 'D' ($B \equiv D$). A tabela de estado simplificada é obtida eliminando uma das linhas (no caso a linha D!).

y \ x	0	1
A	B/0	C/1
B	C/0	A/1
C	B/1	B/0
D	C/0	A/1

$\Rightarrow$

y \ x	0	1
A	B/0	C/1
B	C/0	A/1
C	B/1	B/0

6.3.2. Simplificação de estados através das Partições Sucessivas

Inicialmente é necessário apresentar o conceito matemático do que vem a ser uma ***partição***.

DEFINIÇÃO:

Seja $P_k = \{A_1, A_2, \ldots, A_q\}$ uma coleção de subconjuntos de $\boldsymbol{S}$. Um conjunto P_k é chamado uma partição de $\boldsymbol{S}$ se e somente se:

1. $\bigcup_{i=1}^{q} A_i = S$
2. $A_i \bigcap A_j = \Phi \qquad \forall\ i \neq j$

onde cada subconjunto A_i é chamado bloco da partição P_k.

O método da partição consiste na determinação de P_k, k=1, 2, … e sendo P_k composta por blocos, cada um deles contendo um ou mais estados de **S**. Os estados contidos em cada bloco de P_k devem ser 'k' equivalentes, ou seja, equivalentes para qualquer seqüência de comprimento 'k'.

O processo para determinação das partições sucessivas envolve os seguintes passos:

1. A primeira partição P_1 é formada, colocando-se o máximo de estados em um mesmo bloco de P_1, desde que suas saídas sejam iguais, para cada uma das entradas possíveis. Assim os estados pertencentes a um dos blocos de P_1, são '1' equivalentes, ou seja, equivalentes para qualquer seqüência de comprimento '1'.
2. As partições sucessivas, P_k para k=2, 3, …, l, são determinadas mantendo-se um ou mais estados no mesmo bloco de P_k, se e somente se seus próximos estados estão contidos em um mesmo bloco de P_{k-1}. O processo é interativo.

3. Quando se obtiver uma partição $P_{k+1}=P_k$, o proceso pára, isto porque, se as partições se repetirem, os estados de cada bloco da partição P_k, que são 'k' equivalentes, serão também 'k+1' equivalentes, 'k+2' equivalentes, etc,. Então a partição P_k é dita uma ***partição equivalente*** e os estados contidos em um determinado bloco de P_k são equivalentes. Outra possibilidade de encerrar o processo, é quando todos os blocos da partição P_k só contiver um único estado, concluindo-se portanto que a tabela de estado já é mínima!

Exemplo: Reduzir os estados da tabela dada abaixo:

y \ x	0	1
A	E/0	D/0
B	A/1	F/0
C	C/0	A/1
D	B/0	A/0
E	D/1	C/0
F	C/0	D/1
G	H/1	G/1
H	C/1	B/1

Para formar a partição P_1 é necessário agrupar em blocos os estados que possuem as mesmas saídas para cada um dos valores das entradas. Observando a tabela ao lado verifica-se que:

- A e D estão em um mesmo bloco (saídas =00);
- B e E, pois suas saídas são idênticas (=10);
- da mesma forma C e F (com saídas =01);
- G e H também estão em um só bloco (saídas=11).

Conclui-se portanto que P_1 é escrito como:

$$P_1 = \{(A, D), (B, E), (C, F), (G, H)\}$$

Para se obter P_2 a partir de P_1, depois P_3 a partir de P_2 e assim sucessivamente e interativamente, procede-se como se segue:

- Obtenção de P_2: aplica-se aos estados de cada bloco de P_1 as entradas possíveis e verifica os próximos estados que podem pertencer ou não aos blocos de P_1. Caso pertençam, aqueles estados se mantêm em P_2, caso contrário, aqueles estados se dividem em novos bloco em P_2, como se segue:

$$\text{(A, D)}\rightarrow\begin{cases}0\rightarrow(E,B)ok\\1\rightarrow(D,A)ok\end{cases}\qquad \text{(B, E)}\rightarrow\begin{cases}0\rightarrow(A,D)ok\\1\rightarrow(F,C)ok\end{cases}$$

$$\text{(C, F)}\rightarrow\begin{cases}0\rightarrow(C,C)ok\\1\rightarrow(A,D)ok\end{cases}\qquad \text{(G, H)}\rightarrow\begin{cases}0\rightarrow(H,C)\notin\\1\rightarrow(G,B)\notin\end{cases}\Rightarrow(G),(H)$$

logo P_2 é:

$$P_2 = \{(A, D), (B, E), (C, F), (G), (H)\} \neq P_1 \quad \Rightarrow \quad \text{segue}$$

- Obtenção de P_3:

$$\text{(A, D)}\rightarrow\begin{cases}0\rightarrow(E,B)ok\\1\rightarrow(D,A)ok\end{cases}\ \text{(B, E)}\rightarrow\begin{cases}0\rightarrow(A,D)ok\\1\rightarrow(F,C)ok\end{cases}\ \text{(C, F)}\rightarrow\begin{cases}0\rightarrow(C,C)ok\\1\rightarrow(A,D)ok\end{cases}$$

logo P_3 é:

$$P_3 = \{(A, D), (B, E), (C, F), (G), (H)\} = P_2 \quad \Rightarrow \quad \text{pára!}$$

Conclui-se que os blocos da partição P_2 agrega os estados que são equivalentes entre si, ou sejam:

$$A \equiv D, \qquad B \equiv E \qquad \text{e} \qquad C \equiv F;$$

fazendo:

$a \equiv A \equiv D$, $b \equiv B \equiv E$, $c \equiv C \equiv F$, $d \equiv G$ e $e \equiv H$,

chega-se finalmente a tabela de estado mínima!

y \ x	0	1
a	b/0	a/0
b	a/1	c/0
c	c/0	a/1
d	e/1	d/1
e	c/1	b/1

Para fixar melhor o método das Partições Sucessivas, visando a redução de estado em circuitos seqüenciais completamente especificados, será feita a redução da tabela de estado dada abaixo:

Tabela de Estado

y \ x	0	1
A	A/0	B/0
B	H/1	C/0
C	E/0	B/0
D	C/1	D/0
E	C/1	E/0
F	F/1	G/1
G	B/0	F/0
H	H/1	C/0

Para formar a partição P_1 é necessário agrupar em blocos os estados que possuem as mesmas saídas para cada um dos valores das entradas. Observando a tabela ao lado verifica-se que:

- A, C e G (os três com saídas =00);
- Φ (nenhum com saídas =01);
- B, D, E e H (os quatro com saídas =10);
- F (o único com saídas =11).

Conclui-se portanto que P_1 é escrito como:

P_1 = {(A, C, G), (B, D, E, H), (F)}

➢ Obtenção de P_2:

(A, C, G) $\rightarrow \begin{cases} 0 \rightarrow ((A),(E,B)) \notin \Rightarrow (A),(C,G) \\ 1 \rightarrow ((B,B),(F)) \notin \Rightarrow (A,C),(G) \end{cases} \Rightarrow (A),(C),(G)$

(B, D, E, H) $\rightarrow \begin{cases} 0 \rightarrow ((H),[C,C],(H)) \notin \Rightarrow (B,H),[D,E] \\ 1 \rightarrow ((C),[D,E],(C)) \notin \Rightarrow (B,H),[D,E] \end{cases} \Rightarrow (B,H),(D,E)$

logo P_2 é:

P_2 = {(A), (B, H), (C), (D, E), (F), (G)} $\neq P_1 \Rightarrow$ segue

➢ Obtenção de P_3:

(B, H) $\rightarrow \begin{cases} 0 \rightarrow (H,H)ok \\ 1 \rightarrow (C,C)ok \end{cases}$

(D, E) $\rightarrow \begin{cases} 0 \rightarrow (C,C)ok \\ 1 \rightarrow (D,E)ok \end{cases} \Rightarrow$ logo P_3=$P_2 \Rightarrow$ pára!

Tabela de Estado (reduzida)

y \ x	0	1
(A) a	a/0	b/0
(B,H) b	b/1	c/0
(C) c	d/0	b/0
(D,E) d	c/1	d/0
(F) e	e/1	f/1
(G) f	b/0	e/0

6.3.3. Simplificação de Estados através da Tabela de Implicação

Este método é um pouco mais trabalhoso do que o das Partições Sucessivas, embora seja mais geral e possa ser aplicado, com algumas modificações, às Máquinas de Estado Incompletamente Especificadas.

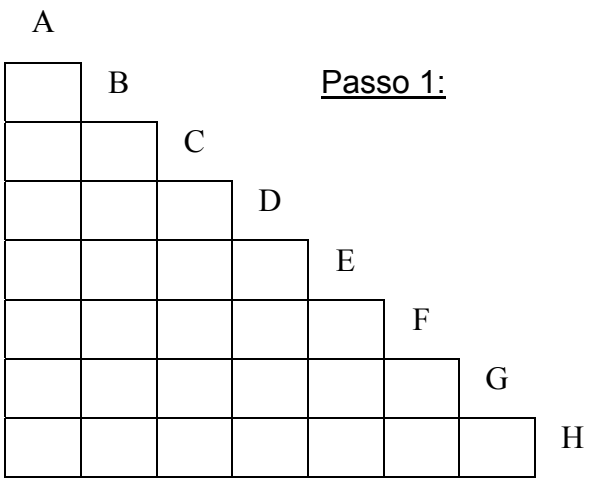
O algoritmo para o método da Tabela de Implicação deve seguir os passos seguintes:

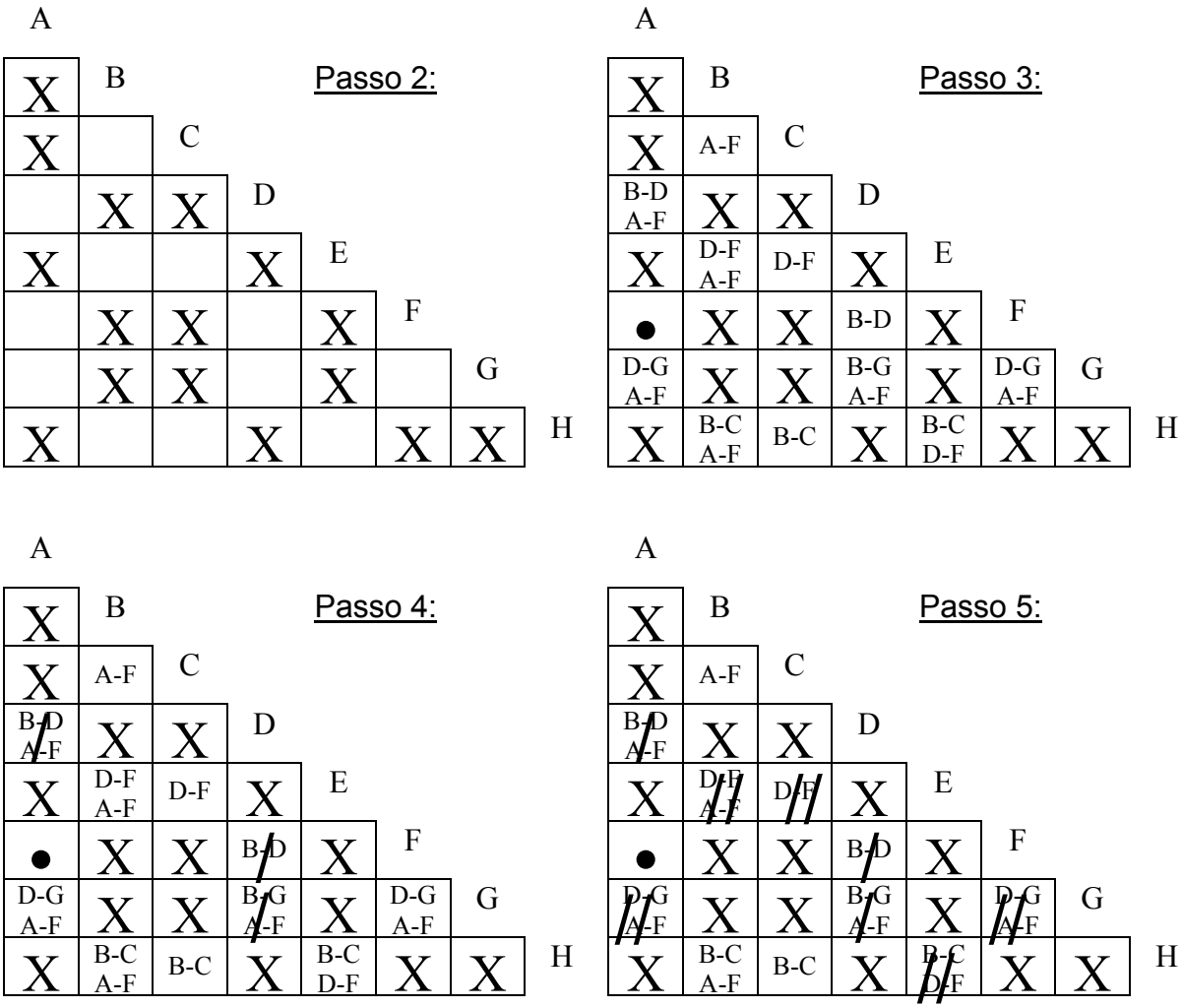
1. Formar uma tabela triangular, listando na sua diagonal todos os estados do circuito seqüencial a minimizar. A tabela resultante deve indicar todas as possíveis combinações entre estados e, cada célula na tabela corresponde a uma interseção de uma linha e uma coluna, representa as características de dois (2) estados cuja eqüivalência será testada.
2. Marcar com um 'X' as células cujos estados não são 1 equivalentes (são 1 distinguíveis, porque possuem saídas diferentes para uma mesma entrada).
3. Nas células vazias (em branco), escrever os pares de estados que são implicados (atingidos) pelos dois estados, cuja interseção define a célula. Evitar os pares de estados implicados quando estes são os mesmos estados da interseção. Marcar um ponto '•' na célula quando nenhum par de estados for estabelecido, significando que o par de estados em questão, são equivalentes, pois não existe restrição alguma que os impeça de o serem!
4. Verificar, a partir do resultado do passo anterior, os pares de estados implicados escritos nas células ainda não marcados, observando se não são ou se são todos equivalentes. Caso não sejam todos equivalente, a célula é também riscada.
5. O passo 4 é repetido toda vez que uma célula for riscada, até que nenhuma célula adicional seja riscada.
6. Terminado o passo 5, os estados que definem células que não foram riscadas são estados equivalentes.

O exemplo a seguir serve para ilustrar o algoritmo de redução de estado utilizando o método da Tabela de Implicação. Reduzir a tabela de estado:

Tabela de Estado

y \ x	0	1	2	3
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0





Após o passo 5, verifica-se que nenhuma célula poderá ser riscada novamente, portanto todos os pares de estados não riscados, bem como aqueles com a marca '•' nas células, correspondem aos pares de estados equivalentes entre si, que são:

$$A \equiv F,\ B \equiv C,\ B \equiv H \text{ e } C \equiv H;$$

como é satisfeita a propriedade da transitividade, tem-se: $B \equiv C \equiv H$. Para se construir a nova tabela de estado reduzida, faz-se: $a \equiv A \equiv F$, $b \equiv B \equiv C \equiv H$, $c \equiv D$, $d \equiv E$ e $e \equiv G$.

Tabela de Estado (reduzida)

y \ x	0	1	2	3
(A,F) a	*c/0*	*c/0*	*a/0*	*a/0*
(B,C,H) b	*b/1*	*c/0*	*d/1*	*a/0*
(D) c	*c/0*	*b/0*	*a/0*	*a/0*
(E) d	*b/1*	*a/0*	*d/1*	*a/0*
(G) e	*e/0*	*e/0*	*a/0*	*a/0*

Diagrama de Estado

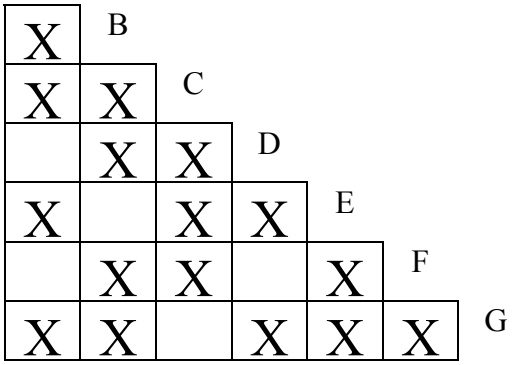
Fazer o diagrama de estado para a máquina reduzida.

Como último exemplo de redução de estados em um circuito seqüencial síncrono completamente especificado, é dada a seguir a tabela de estado, expressa através do modelo de "Moore". Ela será simplificada utilizando, simultaneamente, os dois métodos apresentados.

Tabela de Estado

y \ x	0	1	Z
A	*A*	*E*	1
B	*E*	*C*	2
C	*A*	*D*	3
D	*F*	*G*	1
E	*B*	*C*	2
F	*F*	*E*	1
G	*A*	*D*	3

A $P_1=\{(A,D,F),(B,E),(C,G)\}$



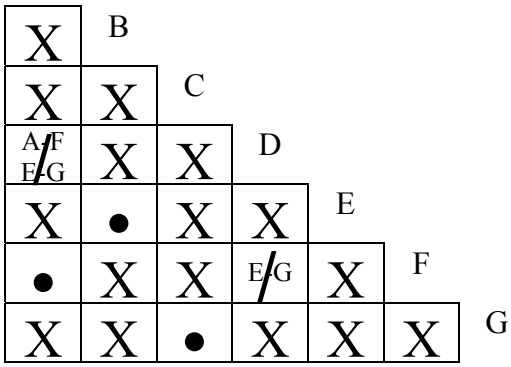
$$(B,\,E)\rightarrow\begin{cases}0\rightarrow(E,B)ok\\1\rightarrow(C,C)ok\end{cases}\qquad(C,G)\rightarrow\begin{cases}0\rightarrow(A,A)ok\\1\rightarrow(D,D)ok\end{cases}$$

$$(A,D,F)\rightarrow\begin{cases}0\rightarrow((A,F,F))ok\Rightarrow\Rightarrow(A,D,F)\\1\rightarrow((E),[G],(E))\not\in\Rightarrow(A,F),[D]\end{cases}$$

$$\Rightarrow(A,F),(D)$$

Facilmente vê-se que $P_3=P_2 \Rightarrow$ pára!

A $P_2=\{(A,F),(B,E),(C,G),(D)\}$



Portanto os estados equivalentes são:

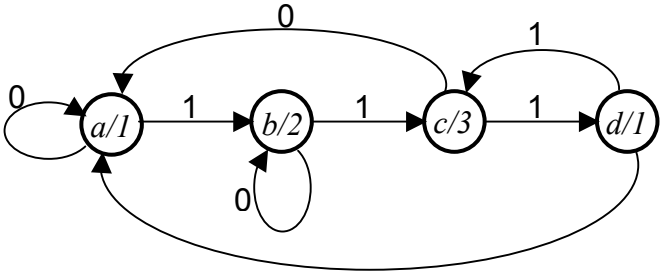
$$A\equiv F,\quad B\equiv E\quad \text{e}\quad C\equiv G;$$

Fazendo: $a\equiv A\equiv B$, $b\equiv B\equiv E$, $c\equiv C\equiv G$ e $e\equiv D$ tem-se a tabela de estado reduzida dada abaixo:

Tabela de Estado (reduzida)

y \ x	0	1	z
(*A,F*) *a*	*a*	*b*	1
(*B,E*) *b*	*b*	*c*	2
(*C,G*) *c*	*a*	*d*	3
(D) *d*	*a*	*c*	1

Diagrama de Estado



6.4. Regas para Escolher uma Boa Designação de Estados

Para ressaltar a importância da designação de estados em projetos de circuitos lógicos seqüenciais, visando a simplificação do bloco combinacional, será mostrado um exemplo que vai ilustrar bem a necessidade de estabelecer regras para escolher uma ***boa*** designação de estados.

Considerar como exemplo ilustrativo, a máquina de estado dada pela sua tabela de estado abaixo:

Tabela de Estado

y \ x	0	1
A	B/0	B/0
B	C/0	C/0
C	D/0	B/0
D	A/1	D/0
E	G/0	E/0
F	A/0	G/1
G	F/0	F/0

Supondo que os estados desta máquina tenham as seguintes designações, dadas por dois projetistas distintos, onde lê-se $y=(y_1y_2y_3)_2$:

Designação 1		Designação 2	
A=000	E=101	A=000	E=100
B=001	F=110	B=001	F=101
C=011	G=111	C=010	G=110
D=010		D=011	

Se esta máquina for realizada com Flip_Flops do tipo JK, ela terá as seguintes equações de excitação e de saída:

- Para a designação 1, tem-se $Z = \overline{x} \cdot \overline{y_1} \cdot y_2 \cdot \overline{y_3} + x \cdot y_1 \cdot \overline{y_3}$ e

$$\begin{cases} J_1 = x \cdot y_3 + x \cdot \overline{y_2} \\ K_1 = x + \overline{y_3} \end{cases} \qquad \begin{cases} J_2 = y_3 \\ K_2 = \overline{y_3} \end{cases} \qquad \begin{cases} J_3 = \overline{y_2} \\ K_3 = y_2 \end{cases}$$

- Para a designação 2, tem-se $Z = \overline{x} \cdot y_2 \cdot y_3 + x \cdot y_1 \cdot y_3$ e

$$\begin{cases} J_1 = x \cdot \overline{y_3} + x \cdot \overline{y_2} \\ K_1 = x + y_3 \end{cases} \qquad \begin{cases} J_2 = y_1 \cdot \overline{y_3} + \overline{y_1} \cdot y_2 \\ K_2 = x \cdot \overline{y_1} + \overline{x} \cdot y_1 + y_3 \end{cases} \qquad \begin{cases} J_3 = \overline{x} \cdot \overline{y_1} + y_2 \\ K_3 = 1 \end{cases}$$

Neste caso, para a 2ª designação, é necessário praticamente o ***dobro*** de portas lógicas das que seriam utilizadas para a 1ª designação!

É evidente, que pelo exemplo acima, a realização de uma máquina de estado pode ser mais ou menos complexa, segundo a designação dada aos estados.

Entretanto esta designação foi totalmente arbitrária!

Antes de anunciar as regras para se obter uma ***boa designação*** de estado, é feito a seguir, alguns comentários no que diz respeito ao número bastante elevado das diferentes possíveis designações de estado não equivalentes (N_{NE}).

Uma máquina com N_{FF} Flip_Flops pode, no máximo, possuir $2^{N_{FF}}$ estados. Geralmente, o número de estados N_E é superior a $2^{N_{FF}-1}$ e inferior ou igual a $2^{N_{FF}}$, a menos que exista redundância.

Se for desejado designar uma máquina com N_E estados, N_E dentre as $2^{N_{FF}}$ diferentes combinações, obtém-se o número N_{DP} para todas as diferentes designações possíveis, expresso por $A_{2^{N_{FF}}-1}^{N_E}$.

Resumindo:

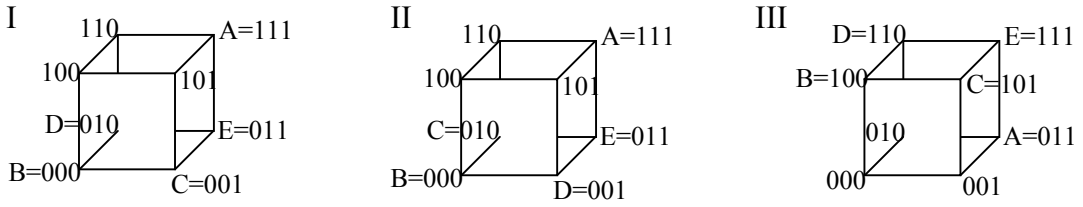
$$2^{N_{FF}-1} < N_E \leq 2^{N_{FF}}$$

e

$$N_{DP} = A_{2^{N_{FF}}-1}^{N_E} = \frac{2^{N_{FF}}!}{(2^{N_{FF}} - N_E)!}$$

A realização de uma máquina de estado tem sua complexidade diminuída ou aumentada segundo a posição relativa dos seus estados. Isto é, duas designações diferentes, cujos estados mantenham uma mesma posição relativa entre si, tem realizações igualmente complexas.

Os exemplos gráficos ilustram o que foi dito acima. Supondo $y=(y_1y_2y_3)$, tem-se:



Pode-se contatar facilmente que a designação II é obtida da designação I pela permuta de y_2 com y_3, ou seja: $II \leftarrow I(y_2 \leftrightarrow y_3)$. Da mesma forma, obtém-se a designação III substituindo na designação I y_1 pelo seu complemento, ou seja: $III \leftarrow I(y_1 \leftarrow \overline{y_1})$. Logo I, II e III são designações equivalentes!

Conclui-se daí, que qualquer sucessão de operações de *troca* de variáveis ou *inversão* das mesmas, acarretará em uma designação equivalente. Vê-se portanto que nem todas a s possíveis designação conduzem a graus de complexidade diferentes. Assim, o número de designações de realização ***não equivalente***, N_{NE}, é obtido dividindo N_{DP} por $2^{N_{FF}}$ (número possível de inversões de variáveis) e por $(N_{FF})!$ (número possível de trocas de variáveis).
Logo tem-se:

$$N_{NE} = \frac{N_{DP}}{(N_{FF}!) \times 2^{N_{FF}}} \rightarrow$$ (número de designações não equivalentes),

finalmente:

$$N_{NE} = \frac{\frac{2^{N_{FF}}!}{(2^{N_{FF}} - N_E)!}}{(N_{FF})! \times 2^{N_{FF}}} = \frac{(2^{N_{FF}} - 1)!}{(2^{N_{FF}} - N_E)! \times N_{FF}!}$$

A partir destas expressões, pode ser obtida a tabela abaixo mostra a explosão da quantidade de designações não equivalentes possíveis em função do número de estados da máquina de estado.

N_E	N_{FF}	N_{DP}	N_{NE}
2	1	2	1
3	2	24	3
4	2	24	3
5	3	6.720	140
6	3	20.160	420
7	3	40.320	840
8	3	40.320	840
9	4	$4{,}15*10^9 \cong 4.151.347.200$	10.810.800

Tabela

Infelizmente não existe até o presente nenhum método que permita encontrar a ***melhor*** designação!

Entretanto existem alguns procedimentos (regras) que devem ser respeitados, a fim de se encontrar uma ***boa designação***.

Alem do mais, como uma designação que é ***boa*** para um determinado tipo de flip_flop e, pode não o ser tão boa para outro tipo, serão dados apenas as ***regras*** que deverão ser obedecidas, sem muitas justificativas.

☞ Regras para um "boa designação de estados":

1. Estados que possuem o mesmo estado seguinte para uma mesma entrada, devem receber designações adjacentes. (↓R1)
2. Estados que possuem o mesmo estado seguinte para entradas diferentes, devem receber designações adjacentes. (×R2)
3. Estados que são os estados seguintes de um único estado presente, sob entradas logicamente adjacentes, devem receber designações adjacente. (→R3)
4. As saídas devem ter a máxima adjacência para os seus '1s' (ou '0s').
5. Devem ser satisfeitas o maior número possível de adjacências. Todos os conflitos existentes entre as regras, devem ser resolvidos à favor das regras 1, 2, 3 e 4, nesta ordem.

- Observação: Caso seja 'D' o tipo de flip_flop usado, recomenda-se escolher o estado que mais aparece na tabela de estado, para designá-lo com valor ***zero***.
- Para o exemplo anterior, tem-se:

  ↓ R1: $(D,F)^2$
  × R2: $(B,E)^2\ (C,G)^2\ (D,F)^2$
  →R3: $(B,E)\ (C,G)^2\ (D,F)^2$
  R4: (D,–) (F, –)

Designação_1

	y_2			
	A	B	C	D
y_1	–	E	G	F
		y_3		

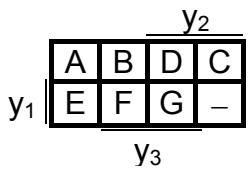
Designação_2

	y_2			
	A	B	D	C
y_1	E	F	–	G
		y_3		

Analisando os dois mapas_K correspondentes as duas designações dadas, verifica-se que para o conjunto das regras de adjacência, a designação_1 falha em apenas parte de R4, ou seja, apenas a exigência de adjacência (D,–) não foi obedecida. No que se refere a designação_2, esta obedeceu somente a regra R4 e apenas satisfez uma das três exigências de adjacência que compõe as regras R2 e R3, ou seja, a exigência de adjacência (C,G).

Uma das piores designação seria a designação_3 dada a seguir, pois ela não satisfaz regra alguma!

Designação_3



$\Rightarrow$

Designação_3

$y=(y_1y_2y_3)$	$y=(y_1y_2y_3)$
A= 0 0 0	E= 1 0 0
B= 0 0 1	F= 1 0 1
C= 0 1 0	G= 1 1 1
D= 0 1 1	

A máquina de estado dada pela sua tabela de estado abaixo, supostamente mínima, deve ser projetada usando flip_flop tipo JK. Para sua implementação, deve-se primeiramente estabelecer as regras par uma boa designação de estado para, em seguida, construir a tabela de transição, para só então, obter as equações de saída e de excitações dos flip_flops do tipo JK.

Tabela de Estado

y \ x	0	1
A	B/0	C/0
B	D/0	E/0
C	E/0	D/0
D	F/0	G/0
E	G/0	F/0
F	A/1	A/0
G	A/0	A/1

Regras:

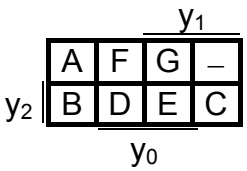
↓ R1: $(F,G)^2$

× R2: $(B,C)^2$ $(D,E)^2$ $(F,G)^2$

→R3: (B,C) $(D,E)^2$ $(F,G)^2$

R4: $(F,–)$ $(G,–)$

Escolha/designação

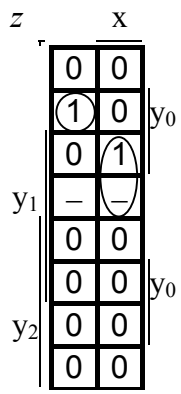


Valor/designação

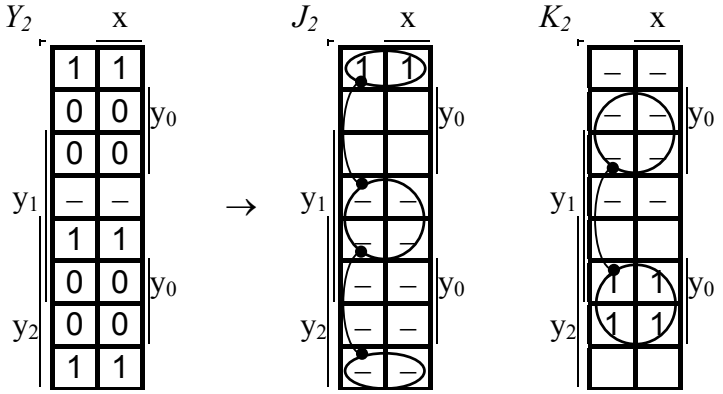
$y=(y_2y_1y_0)$	$y=(y_2y_1y_0)$
A= 0 0 0	E= 1 1 1
B= 1 0 0	F= 0 0 1
C= 1 1 0	G= 0 1 1
D= 1 0 1	–

Tabela de Transição

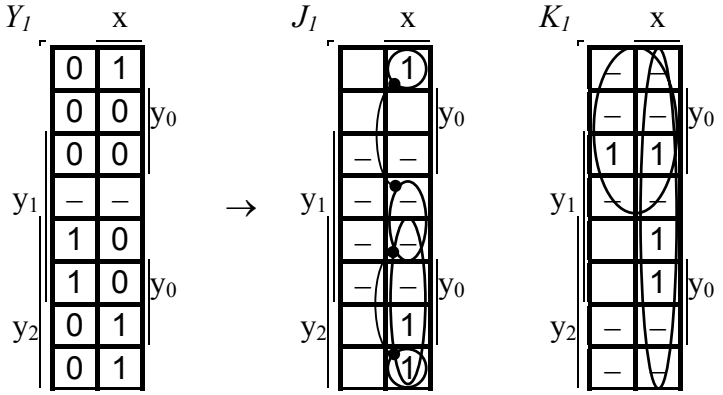
y \ x	$y_2y_1y_0$ \ x	0	1
A	000	100/0	110/0
F	001	000/1	000/0
G	011	000/0	000/1
–	010	–––/–	–––/–
C	110	111/0	101/0
E	111	011/0	001/0
D	101	001/0	011/0
B	100	101/0	111/0



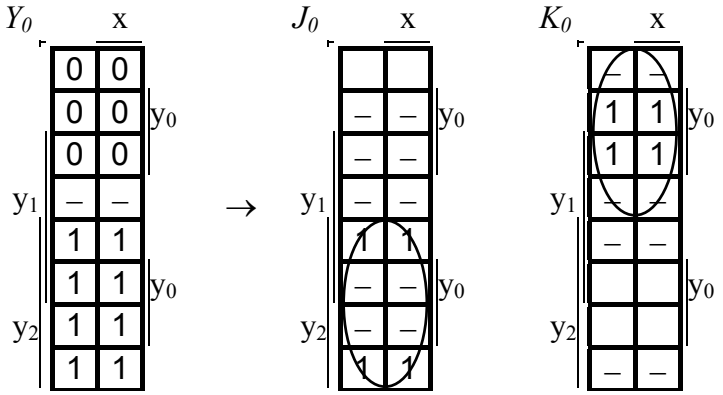
$$z = \bar{x} \cdot \overline{y_2} \cdot \overline{y_1} \cdot y_0 + x \cdot \overline{y_2} \cdot y_1$$



$$\begin{cases} J_2 = \overline{y_0} \\ K_2 = y_0 \end{cases}$$



$$\begin{cases} J_1 = x \cdot \overline{y_0} + x \cdot y_2 \\ K_1 = x + \overline{y_2} \end{cases}$$



$$\begin{cases} J_0 = y_2 \\ K_0 = \overline{y_2} \end{cases}$$

Para determinar uma boa designação de estado para a MdE dada abaixo pela sua tabela de estado e, implementá-la com flip_flop do tipo 'D', faz-se:

Tabela de Estado

y \ x	0	1
A	E/0	B/0
B	A/1	D/1
C	E/0	A/0
D	A/0	B/1
E	D/0	C/0

Regras:

↓ R1: (A,C) (A,D) (B,D)
× R2: (B,C) (B,E) (C,D)
→R3: (A,B) (A,D) (A,E) (B,E) (C,D)
R4: $(B,-)_3$ $(D,-)_3$ (B,D)

Escolha/designação

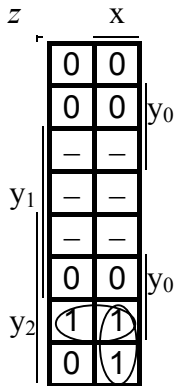
y_2 \ y_1			
	A	–	–	D
y_2	C	–	E	B
			y_0	

Valor/designação

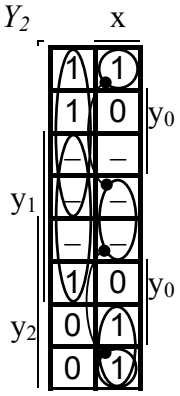
$y=(y_2y_1y_0)$	$y=(y_2y_1y_0)$
A= 0 0 0	E= 1 1 1
B= 1 0 1	
C= 0 0 1	
D= 1 0 0	

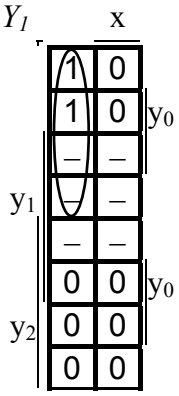
Tabela de Transição

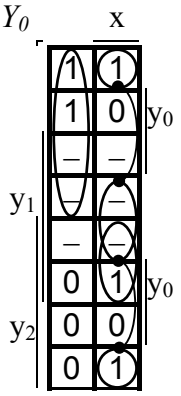
\ x y \	\ x $y_2y_1y_0$ \	0	1
A	000	111/0	101/0
C	001	111/0	000/0
–	011	–––/–	–––/–
–	010	–––/–	–––/–
–	110	–––/–	–––/–
E	111	100/0	001/0
B	101	000/1	100/1
D	100	000/0	101/1



$$z = y_2 \cdot \overline{y_1} \cdot y_0 + x \cdot y_2 \cdot \overline{y_1}$$







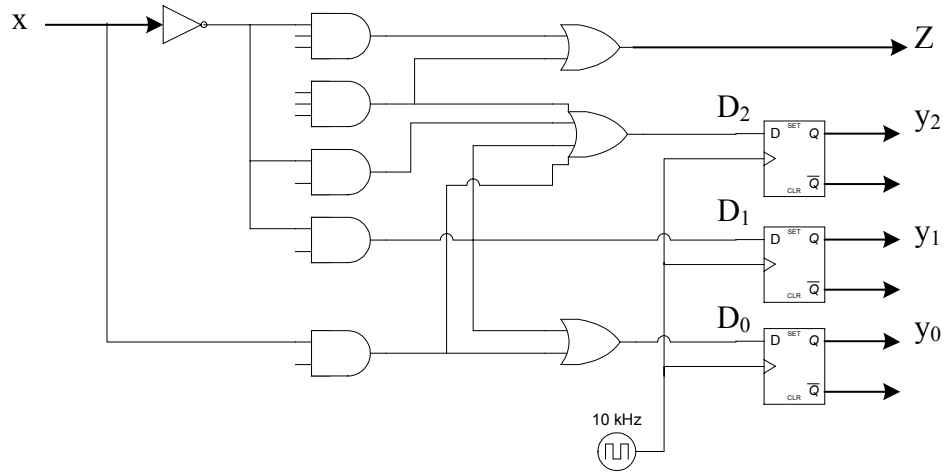
- Para Filp_Flop tipo 'D' sabe-se que $Y_i = D_i$, logo tem-se:

$$D_2 = Y_2 = x \cdot y_2 \cdot \overline{y_1} + \overline{x} \cdot \overline{y_2} + \overline{x} \cdot y_1 + x \cdot \overline{y_0}$$

$$D_1 = Y_1 = \overline{x} \cdot \overline{y_2}$$

$$D_0 = Y_0 = \overline{x} \cdot \overline{y_2} + x \cdot \overline{y_0}$$

Implementação AND-OR:



Implementação NAND-NAND: