

Sistemas Digitais 1

Tiago Alves

Faculdade UnB Gama
Universidade de Brasília



Aula 22

- Projeto de Máquinas de Estados
- Minimizando a Tabela de Estados
- State Assignment
- Projeto de Máquinas de Estados usando Flip-Flops D



Os passos para projetar uma máquina de estado apresentam o processo inverso da análise:

- 1 **Construir uma tabela de estado/saída** que corresponde à descrição do circuito desejado. Nesse ponto, usamos mnemônicos para o nome dos estados. Nesta etapa, também é comum **construir o diagrama de estados**.
- 2 **Minimizar** o número de estados na tabela.
- 3 **Escolher** um conjunto de **variáveis de estado** e assinalar códigos aos estados.
- 4 **Substituir os códigos dos estados** nas tabelas de estado/saída.
- 5 **Escolher um flip-flop** (no nosso caso, o flip-flop D).
- 6 **Construir as equações de excitação** para o flip-flop desejado.
- 7 **Derivar as equações de saída**.
- 8 **Desenhar** o esquemático do circuito.



A ideia é aprender a projetar um circuito a partir de uma *descrição* em linguagem natural (muitas vezes, vaga):

Exemplo

Projete uma máquina de estados com 2 entradas **A** e **B** e uma saída **Z**, onde a saída **Z** é 1 se:

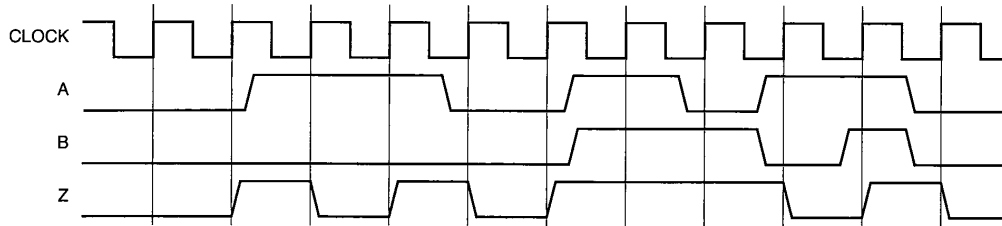
- **A** teve o mesmo valor nos dois últimos clocks; *ou*
- **B** era 1 desde a última vez que **Z** foi 1.

Caso contrário, **Z** é 0.



Uma das tarefas do projetista é converter essa descrição em uma especificação *não ambígua*.

Alguns projetos incluem um **diagrama de tempo** para ajudar nessa etapa:



Para começar o projeto, iniciamos com uma tabela de transição de estados vazia, onde colocamos o estado, as entradas e qual será o próximo estado dado uma combinação de estado atual / entrada.

No nosso caso, a saída é do tipo Moore, isto é, depende apenas do estado (porque depende do clock anterior da máquina). Nesse caso, a saída pode ocupar um espaço à parte na tabela - para máquinas do tipo Mealy, a saída deveria ser especificada para cada combinação estado atual / entrada.

Note que, neste ponto, chamamos nossos estados por **mnemônicos**!



Iniciamos com uma tabela vazia, com um único estado inicial:

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	?	?	?	?	0

A descrição não diz *o que* a máquina deve fazer quando é ligada, por isso, criamos o estado inicial **INIT**.

Exemplo

Projete uma máquina de estados com 2 entradas **A** e **B** e uma saída **Z**, onde a saída **Z** é 1 se:

- **A** teve o mesmo valor nos dois últimos clocks; *ou*
- **B** era 1 desde a última vez que **Z** foi 1.

Caso contrário, **Z** é 0.



Em seguida, temos que decidir o que a máquina deve fazer após o primeiro estado. No caso, devemos rastrear a condição do que houve com a entrada **A**, para sabermos se essa entrada foi igual por dois clocks seguidos.

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0					0
Último A = 1	A1					0

Exemplo

Projete uma máquina de estados com 2 entradas **A** e **B** e uma saída **Z**, onde a saída **Z** é 1 se:

- **A** teve o mesmo valor nos dois últimos clocks; ou
- **B** era 1 desde a última vez que **Z** foi 1.

Caso contrário, **Z** é 0.



Criamos dois novos estados e, agora, temos que decidir o que irá acontecer nesses estados. Mas temos que tomar cuidado: se, a todo momento, criarmos mais e mais estados, nunca terminaremos nossa máquina!

Portanto, a ideia é procurar novos estados que tenham o mesmo significado que os estados que já temos, e só criar novos estados quando tivermos um novo significado.

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último $A = 0$	A0					0
Último $A = 1$	A1					0



Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0	OK	OK	A1	A1	0
Último A = 1	A1					0
Dois As iguais	OK					1

Exemplo

Projete uma máquina de estados com 2 entradas **A** e **B** e uma saída **Z**, onde a saída **Z** é 1 se:

- **A** teve o mesmo valor nos dois últimos clocks; ou
- **B** era 1 desde a última vez que **Z** foi 1.

Caso contrário, **Z** é 0.



Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0	OK	OK	A1	A1	0
Último A = 1	A1	A0	A0	OK	OK	0
Dois As iguais	OK					1

Exemplo

Projete uma máquina de estados com 2 entradas **A** e **B** e uma saída **Z**, onde a saída **Z** é 1 se:

- **A** teve o mesmo valor nos dois últimos clocks; ou
- **B** era 1 desde a última vez que **Z** foi 1.

Caso contrário, **Z** é 0.



Projeto de Máquina de Estados

Quando entramos no estado OK, a descrição da máquina nos diz que podemos ficar aqui enquanto $B = 1$, sem nos importarmos com o A (ou seja, se $B = 1$, continuamos no estado OK).

Porém, se $B = 0$, temos que observar se o último A foi 1 ou 0. Porém, não temos essa informação na descrição do estado OK! Provavelmente, precisamos de um novo estado OK que inclua a informação do que acontece com a entrada A .

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último $A = 0$	A0	OK	OK	A1	A1	0
Último $A = 1$	A1	A0	A0	OK	OK	0
Dois As iguais	OK	?	OK	OK	?	1

Exemplo

Projete uma máquina de estados com 2 entradas A e B e uma saída Z , onde a saída Z é 1 se:

- A teve o mesmo valor nos dois últimos clocks; ou
- B era 1 desde a última vez que Z foi 1.

Caso contrário, Z é 0.

Criamos dois estados OK: OK0 e OK1, que guardam a informação do que houve com o último **A**.

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0	OK0	OK0	A1	A1	0
Último A = 1	A1	A0	A0	OK1	OK1	0
Ativo, último A = 0	OK0					1
Ativo, último A = 1	OK1					1

Exemplo

Projete uma máquina de estados com 2 entradas **A** e **B** e uma saída **Z**, onde a saída **Z** é 1 se:

- **A** teve o mesmo valor nos dois últimos clocks; ou
- **B** era 1 desde a última vez que **Z** foi 1.

Caso contrário, **Z** é 0.



Continuamos preenchendo a tabela e chegamos a:

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0	OK0	OK0	A1	A1	0
Último A = 1	A1	A0	A0	OK1	OK1	0
Ativo, último A = 0	OK0	OK0	OK0	OK1	A1	1
Ativo, último A = 1	OK1					1

Exemplo

Projete uma máquina de estados com 2 entradas A e B e uma saída Z, onde a saída Z é 1 se:

- A teve o mesmo valor nos dois últimos clocks; ou
- B era 1 desde a última vez que Z foi 1.

Caso contrário, Z é 0.



E:

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0	OK0	OK0	A1	A1	0
Último A = 1	A1	A0	A0	OK1	OK1	0
Ativo, último A = 0	OK0	OK0	OK0	OK1	A1	1
Ativo, último A = 1	OK1	A0	OK0	OK1	OK1	1

Exemplo

Projete uma máquina de estados com 2 entradas **A** e **B** e uma saída **Z**, onde a saída **Z** é 1 se:

- **A** teve o mesmo valor nos dois últimos clocks; ou
- **B** era 1 desde a última vez que **Z** foi 1.

Caso contrário, **Z** é 0.



Nesse ponto, fechamos a tabela: todas as condições de entrada estão definidas e não levam a nenhum estado que não aparece na tabela. Logo, nossa máquina de estados é *finita*.

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0	OK0	OK0	A1	A1	0
Último A = 1	A1	A0	A0	OK1	OK1	0
Ativo, último A = 0	OK0	OK0	OK0	OK1	A1	1
Ativo, último A = 1	OK1	A0	OK0	OK1	OK1	1

Exemplo

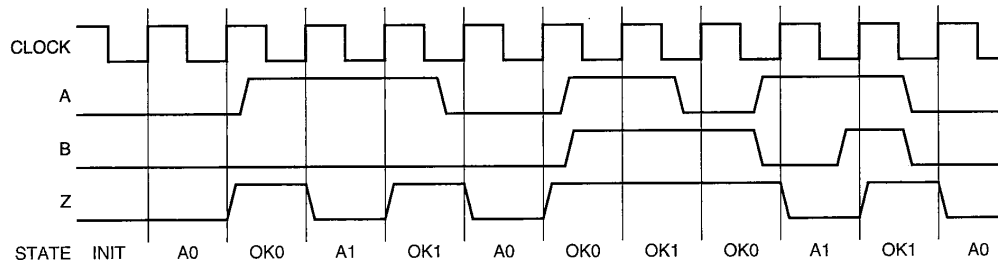
Projete uma máquina de estados com 2 entradas **A** e **B** e uma saída **Z**, onde a saída **Z** é 1 se:

- **A** teve o mesmo valor nos dois últimos clocks; ou
- **B** era 1 desde a última vez que **Z** foi 1.

Caso contrário, **Z** é 0.



Para garantir, olhamos novamente o diagrama de tempo, indicando em qual estado a máquina se encontra em cada ponto:



A tabela de estados que obtivemos é *mínima*, isto é, ela descreve o circuito usando o menor número de estados possível.

Porém, quando estamos construindo uma tabela de estados, pode ser que tenhamos criado *mais* estados do que o necessário. Como saber se foi este o caso?



Minimizando a Tabela de Estados

Veja o exemplo:

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0	OK00	OK00	A1	A1	0
Último A = 1	A1	A0	A0	OK11	OK11	0
Dois últimos A = 0	OK00	OK00	OK00	OKA1	A1	1
Dois últimos A = 1	OK11	A0	OKA0	OK11	OK11	1
Ativo, último A = 0	OKA0	OK00	OK00	OKA1	A1	1
Ativo, último A = 1	OKA1	A0	OKA0	OK11	OK11	1

Exemplo

Projete uma máquina de estados com 2 entradas A e B e uma saída Z, onde a saída Z é 1 se:

- A teve o mesmo valor nos dois últimos clocks; ou
- B era 1 desde a última vez que Z foi 1.

Caso contrário, Z é 0.

Minimizando a Tabela de Estados

A ideia básica é eliminar **estados equivalentes**. Olhe os estados OK00 e OKA0, por exemplo.

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT	A0	A0	A1	A1	0
Último A = 0	A0	OK00	OK00	A1	A1	0
Último A = 1	A1	A0	A0	OK11	OK11	0
Dois últimos A = 0	OK00	OK00	OK00	OKA1	A1	1
Dois últimos A = 1	OK11	A0	OKA0	OK11	OK11	1
Ativo, último A = 0	OKA0	OK00	OK00	OKA1	A1	1
Ativo, último A = 1	OKA1	A0	OKA0	OK11	OK11	1

Exemplo

Projete uma máquina de estados com 2 entradas A e B e uma saída Z, onde a saída Z é 1 se:

- A teve o mesmo valor nos dois últimos clocks; ou
- B era 1 desde a última vez que Z foi 1.

Caso contrário, Z é 0.

Os dois estados, OK00 e OKA0, tem exatamente o mesmo estado futuro e a mesma saída! Dizemos que dois estados são *equivalentes* se é impossível distinguir entre eles olhando apenas para o estado futuro e a saída do circuito (em caso de máquinas Mealy, a saída para cada combinação de estado / entrada). Isto é, se os estados tem a mesma linha na tabela.

Estados equivalentes podem ser combinados (eliminando-se um deles) para simplificar a sua máquina.

Significado	S	AB				Z
		00	01	11	10	
Estado Inicial	INIT					
Último $A = 0$	A0					
Último $A = 1$	A1					
Dois últimos $A = 0$	OK00	OK00	OK00	OKA1	A1	1
Dois últimos $A = 1$	OK11					
Ativo, último $A = 0$	OKA0	OK00	OK00	OKA1	A1	1
Ativo, último $A = 1$	OKA1					



O próximo passo é determinar quantas variáveis de estado precisamos. Como usamos variáveis binárias, temos que n variáveis de estado conseguem descrever $s = 2^n$ estados. Logo, para codificar s estados, precisamos de $n = \lceil \log_2 s \rceil$ (isto é, o menor inteiro que é maior ou igual a \log_2).

Porém, como assinalar códigos binários para cada estado?

Em geral, o número de maneiras possíveis de se fazer isso é muito grande! Existem $\binom{8}{5}$ formas possíveis de se assinalar cinco códigos entre oito estados possíveis, ou seja:

$$\frac{8!}{5! \cdot 3!} = 6720$$



A única forma de se garantir que encontramos a *melhor* maneira seria testar todas as alternativas, o que é definitivamente inviável. Por isso, usamos algumas ideias (e é aqui que a experiência do projetista conta):

- Escolha um estado inicial de forma que possamos forçá-lo facilmente (ex, **00...00** ou **11...11**).
- Procure minimizar o número de variáveis de estado que mudam a cada transição.
- Maximize o número de variáveis de estado que *não* mudam em um grupo de estados relacionados.
- Considere usar mais que o número mínimo de variáveis de estado.



Exemplos de formas possíveis:

Nome	Assignment			
	Mais Simples	Decomposto	One Hot	Quase One Hot
INIT	000	000	00001	0000
A0	001	100	00010	0001
A1	010	101	00100	0010
OK0	011	110	01000	0100
OK1	100	111	10000	1000



Uma vez assinalados os códigos dos estados, podemos montar nossas tabelas de transição com os valores do estado em cada momento.

Porém, se não tivermos utilizado todos os estados possíveis, essa tabela vai ser incompleta (isto é, se tivermos escolhido os estados decompostos **000,100,101,110** e **111**, o que acontece com a máquina se estivermos, por algum defeito, nos estados **001, 010** ou **011?**).

Existem duas abordagens principais: *custo mínimo* e *risco mínimo*.



Na abordagem de *risco mínimo*, preenchemos os estados não usados para levar a máquina a um novo estado de recuperação (por exemplo, o estado inicial). Assim, se por algum erro a máquina cair num desses estados, sabemos que ela irá voltar ao estado inicial e voltar a funcionar normalmente depois disso (isto é, ela se recupera).

Na abordagem de *custo mínimo*, assumimos que a máquina nunca entra em um desses estados. Assim, podemos usar essas entradas como *don't cares* para minimizar nossa lógica de próximo estado. Em geral, isso simplifica bastante a lógica do circuito, mas o comportamento da máquina pode ser bem estranho caso a máquina caia em um dos estados não usados (pior ainda, ela pode ficar num desses estados para sempre e jamais se recuperar).



O próximo passo é escrever a tabela com os códigos dos estados:

Estado	$Q_2 Q_1 Q_0$	AB				Z
		00	01	11	10	
INIT	000	100	100	101	101	0
A0	100	110	110	101	101	0
A1	101	100	100	111	111	0
OK0	110	110	110	111	101	1
OK1	111	100	110	111	111	1

$Q_2 * Q_1 * Q_0$

A partir desta tabela, derivamos as funções booleanas de lógica de próximo estado F e de lógica de saída G .



Para isso, precisamos da *tabela de excitação* dos flip-flops que serão utilizados. A vantagem de se utilizar flip-flops D nessa etapa é que a equação característica deste flip-flop é:

$$Q^* = D$$

Assim, a tabela de transição de estados é a mesma da tabela de excitação. Note que, se usássemos outro tipo de flip-flop (JK, por exemplo), teríamos que definir o que colocar nas entradas **J** e **K** em cada estado para que o flip-flop transite para o estado desejado.



Mapas de Karnaugh

Equações para Q_2^* :

$Q_2 = 0$

$Q_1 Q_0$ \ AB		AB				Q_2^*
		00	01	11	10	
$Q_1 Q_0$	00	1	1	1	1	
	01	x	x	x	x	
	11	x	x	x	x	
	10	x	x	x	x	

$Q_2 = 1$

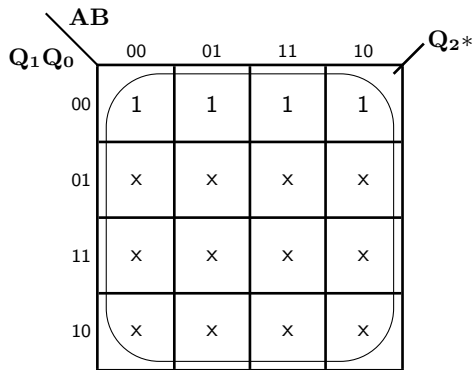
$Q_1 Q_0$ \ AB		AB				Q_2^*
		00	01	11	10	
$Q_1 Q_0$	00	1	1	1	1	
	01	1	1	1	1	
	11	1	1	1	1	
	10	1	1	1	1	

$Q_2^* = ?$

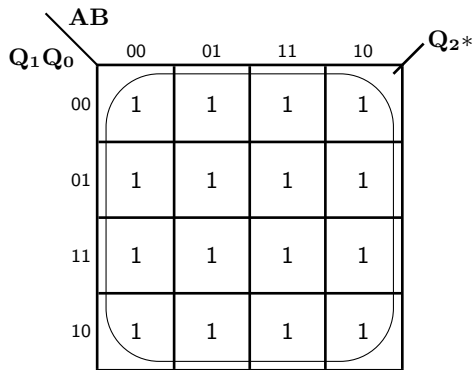


Mapas de Karnaugh

Equações para Q_2^* :



$Q_2 = 0$



$Q_2 = 1$

$Q_2^* = 1$



Mapas de Karnaugh

Equações para $Q1^*$:

		AB				$Q1^*$
		00	01	11	10	
$Q1 Q0$	00	0	0	0	0	
	01	x	x	x	x	
	11	x	x	x	x	
	10	x	x	x	x	

$Q2 = 0$

		AB				$Q1^*$
		00	01	11	10	
$Q1 Q0$	00	1	1	0	0	
	01	0	0	1	1	
	11	0	1	1	1	
	10	1	1	1	0	

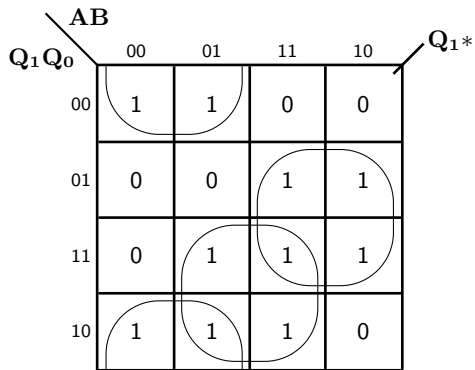
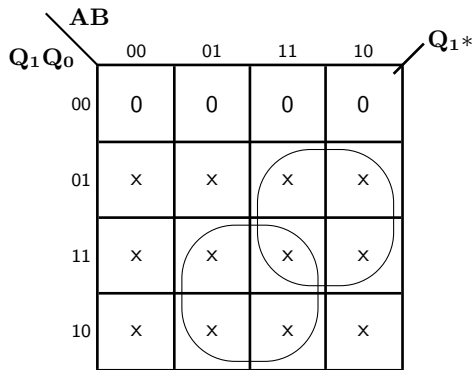
$Q2 = 1$

$Q1^* = ?$



Mapas de Karnaugh

Equações para $Q1^*$:



$$Q1^* = Q2 \cdot \overline{Q_0} \cdot \overline{A} + Q_0 \cdot A + Q_1 \cdot B$$



Mapas de Karnaugh

Equações para Q_0^* :

		AB				Q_0^*
		00	01	11	10	
$Q_1 Q_0$	00	0	0	1	1	
	01	x	x	x	x	
	11	x	x	x	x	
	10	x	x	x	x	

$Q_2 = 0$

		AB				Q_0^*
		00	01	11	10	
$Q_1 Q_0$	00	0	0	1	1	
	01	0	0	1	1	
	11	0	0	1	1	
	10	0	0	1	1	

$Q_2 = 1$

$Q_0^* = ?$



Mapas de Karnaugh

Equações para Q_0^* :

$Q_2 = 0$

$\begin{matrix} \text{AB} \\ \swarrow \searrow \end{matrix}$		$Q_1 Q_0$				Q_0^*
		00	01	11	10	
00	0	0	1	1		
01	x	x	x	x		
11	x	x	x	x		
10	x	x	x	x		

$Q_2 = 1$

$\begin{matrix} \text{AB} \\ \swarrow \searrow \end{matrix}$		$Q_1 Q_0$				Q_0^*
		00	01	11	10	
00	0	0	1	1		
01	0	0	1	1		
11	0	0	1	1		
10	0	0	1	1		

$$Q_0^* = A$$



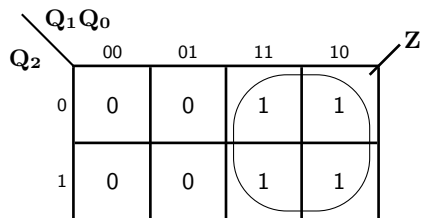
Equações para Z :

$Q_1 Q_0$		00	01	11	10
Q_2	0	0	0	1	1
	1	0	0	1	1

$$Z = ?$$



Equações para Z :



A Karnaugh map for the output Z. The vertical axis is labeled Q₂ with values 0 and 1. The horizontal axis is labeled Q₁ Q₀ with values 00, 01, 11, and 10. The output Z is indicated by an arrow pointing to the right. The map contains 1s in the cells where Q₁ is 1 (columns 11 and 10) and 0s elsewhere. A circle is drawn around the four 1s, indicating the simplified expression Z = Q₁.

	00	01	11	10
0	0	0	1	1
1	0	0	1	1

$$Z = Q_1$$



E chegamos à lógica de próximo estado **F**:

$$Q_2^* = 1$$

$$Q_1^* = Q_2 \cdot \overline{Q_0} \cdot \overline{A} + Q_0 \cdot A + Q_1 \cdot B$$

$$Q_0^* = A$$

E à lógica de saída **G**:

$$Z = Q_1$$



Exercício

Projete uma máquina de estados com duas entradas, X e Y , e uma saída Z . A saída deve ser 1 se o número de entradas X ou Y iguais a 1 desde que a máquina foi ligada for um múltiplo de 4, e a saída deverá ser 0 caso contrário.



Exercício

Projete uma máquina de estados com uma entrada **X** e duas saídas **UNLK** e **HINT**. A saída **UNLK** deverá ser **1** se e *somente* se a entrada **X** for **0** e a sequência de **X** nos sete últimos períodos de clock foi **0110111**. A saída **HINT** deverá ser **1** se a posição atual de **X** leva a máquina mais próximo do estado “destrancada” ($\text{UNLK} = 1$).

