

Projeto de Contadores e Multiplicadores

Prof. Denis Fantinato
Profa. Letícia Rittner

O que será visto nesta aula

- Projeto de Contadores Síncronos.
- Multiplicadores

Projeto de Contadores Síncronos

- Circuitos de contadores síncronos podem ser projetados para gerar qualquer sequência de contagem.
- Isso pode ser feito usando-se somente as entradas síncronas dos FFs e circuitos lógicos combinacionais.
- Uma vantagem de não se usar as entradas assíncronas dos FFs, como CLEAR, é que não são gerados glitches na saída do contador (devidos aos estados temporários).
- Para se projetar contadores síncronos deve-se identificar as entradas de controle de cada FF em cada estado do contador.
- Uma tabela de estado ATUAL/PRÓXIMO é uma ferramenta útil nesta análise.

Projeto de Contadores Síncronos

- Considere o contador síncrono abaixo:

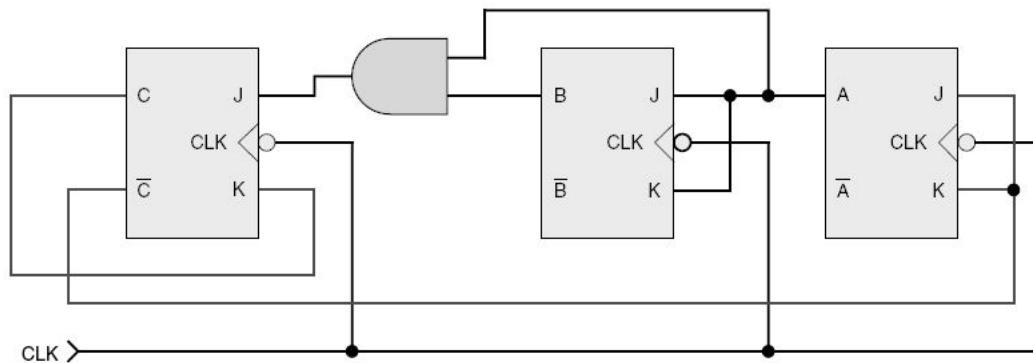


FIGURA 7.23 Contador síncrono com diferentes entradas de controle.

- As expressões lógicas das entradas dos FFs são:

$$\begin{aligned}J_A &= K_A = \bar{C} \\J_B &= K_B = A\end{aligned}$$

$$\begin{aligned}J_C &= A \cdot B \\K_C &= C\end{aligned}$$

Projeto de Contadores Síncronos

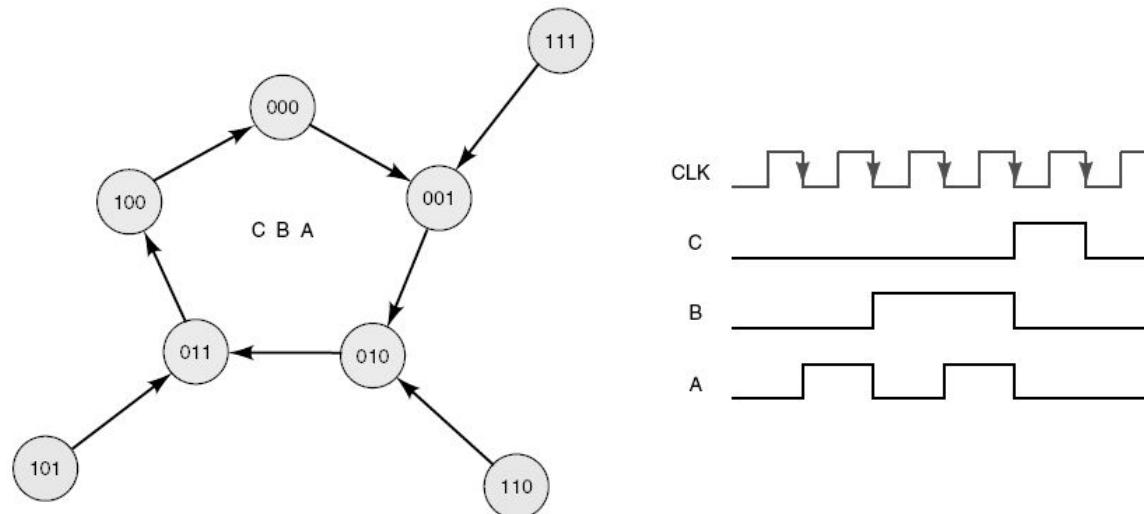
- Segue a tabela de estado ATUAL/PRÓXIMO estado desse contador:

Estado ATUAL			PRÓXIMO estado		
C	B	A	C	B	A
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	0	0	1

- Observa-se que esse é um contador **autocorretor**, pois os estados retornam para um estado da sequência normal.

Projeto de Contadores Síncronos

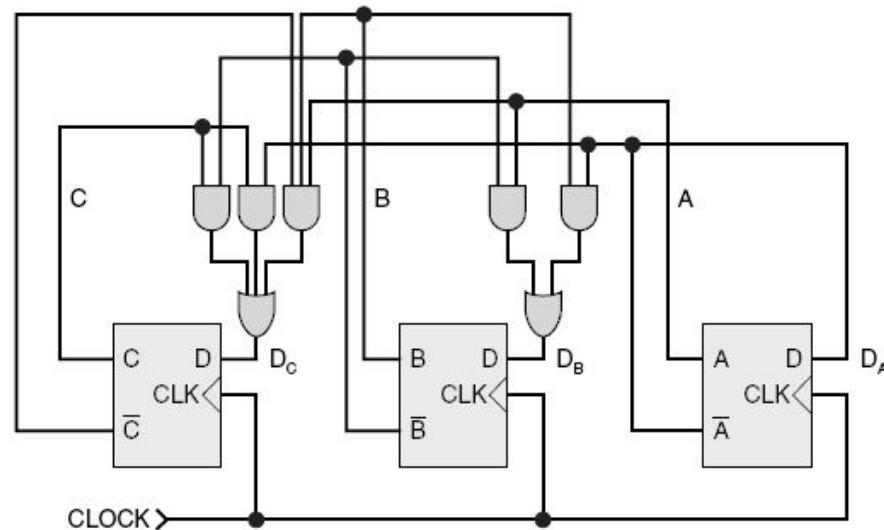
- Segue o diagrama de estados desse contador:



- O mesmo circuito pode ser projetado com flip-flops do tipo D.

Projeto de Contadores Síncronos

- O circuito de controle com FF do tipo D, de modo geral, é mais complexo do que de um equivalente com FFs J-K. Porém o número de entradas a controlar é menor.
- A maioria dos dispositivos lógicos programáveis utiliza FFs D como dispositivos de memória.
- Contador equivalente com FFs D:



Projeto de Contadores Síncronos

- Contador equivalente com FFs D:

Estado ATUAL			Entradas de controle			PRÓXIMO estado		
C	B	A	D_C	D_B	D_A	C	B	A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Projeto de Contadores Síncronos

- Contador equivalente com FFs D:

Estado ATUAL			Entradas de controle			PRÓXIMO estado		
C	B	A	D _C	D _B	D _A	C	B	A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Expressões lógicas para as entradas:

$$\begin{aligned} DC &= C\bar{B} + C\bar{A} + \bar{C}\bar{B}\bar{A} \\ &= C \oplus (AB) \end{aligned}$$

$$\begin{aligned} DB &= \bar{B}A + B\bar{A} \\ &= B \oplus A \end{aligned}$$

$$DA = \bar{A}$$

Projeto de Contadores Síncronos

- Veremos como projetar contadores com FFs J-K.
- Nos contadores síncronos, todos os FFs são disparados ao mesmo tempo.
- A ideia básica é fazer com que as entradas de todos os FFs estejam com os valores que levem para o próximo estado no momento em que houver a transição ativa de clock.

TABELA 7.3 Tabela de transição J-K.

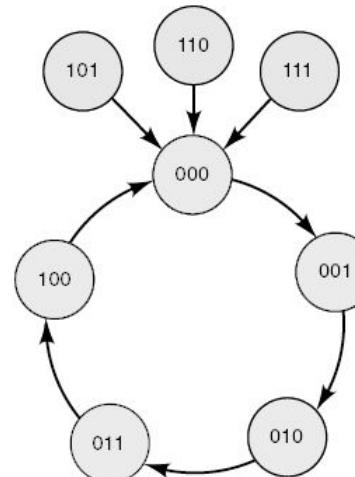
Transição na saída do FF	ATUAL estado Q_n	PRÓXIMO estado Q_{n+1}	J	K
$0 \rightarrow 0$	0	0	0	x
$0 \rightarrow 1$	0	1	1	x
$1 \rightarrow 0$	1	0	x	1
$1 \rightarrow 1$	1	1	x	0

Projeto de Contadores Síncronos

- Considere o projeto de um contador autocorretor de módulo 5.
 - Etapas do projeto:

C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
0	0	0
0	0	1
etc.		

1. Determine o número de FFs necessários e a sequência de contagem desejada.
2. Desenhe o diagrama de transição de estados **mostrando todos os estados.**



Projeto de Contadores Síncronos

3. Construa uma tabela de transição de estados listando todos os estados atuais e os próximos.
4. Acrescente a essa tabela uma coluna para cada entrada J e K.

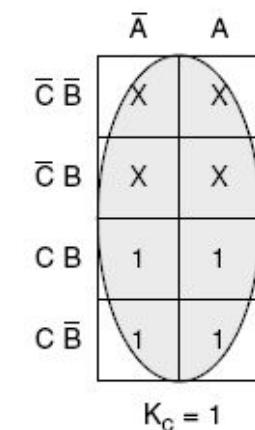
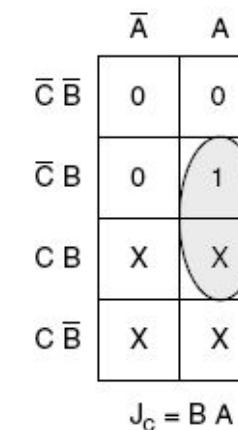
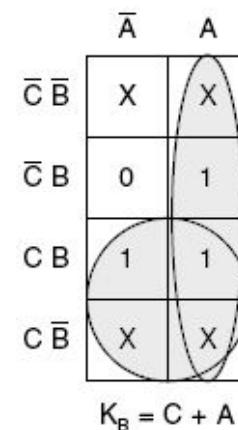
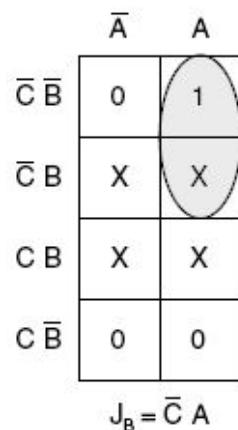
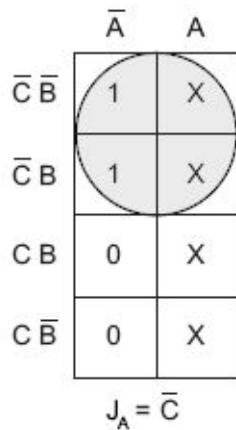
TABELA 7.6 Tabela de excitação do circuito.

Linha	Estado ATUAL			PRÓXIMO estado			J_C	K_C	J_B	K_B	J_A	K_A
	C	B	A	C	B	A						
1	0	0	0	0	0	1	0	x	0	x	1	x
2	0	0	1	0	1	0	0	x	1	x	x	1
3	0	1	0	0	1	1	0	x	x	0	1	x
4	0	1	1	1	0	0	1	x	x	1	x	1
5	1	0	0	0	0	0	x	1	0	x	0	x
6	1	0	1	0	0	0	x	1	0	x	x	1
7	1	1	0	0	0	0	x	1	x	1	0	x
8	1	1	1	0	0	0	x	1	x	1	x	1

Projeto de Contadores Síncronos

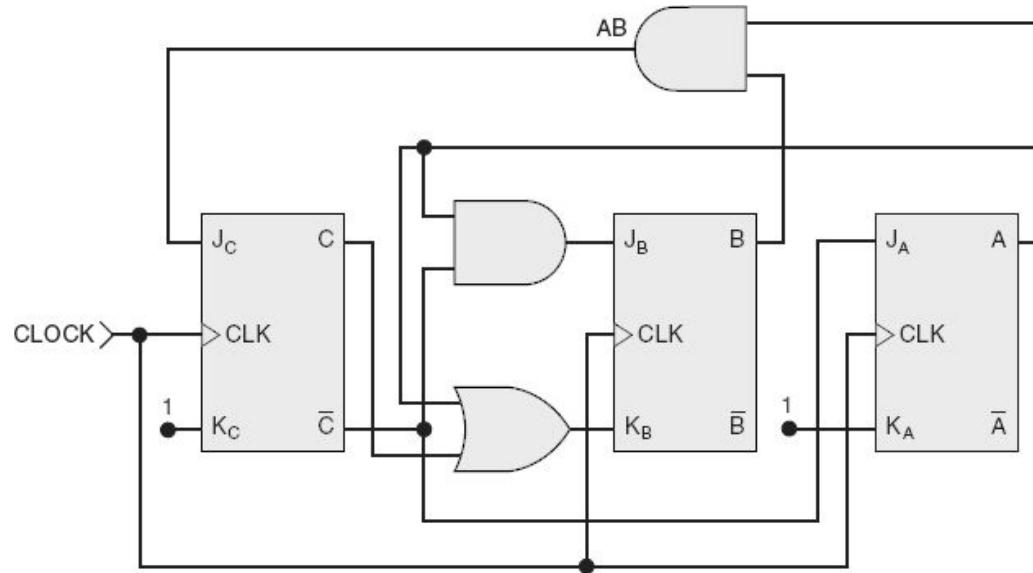
5. Projete os circuitos lógicos necessários para gerar os níveis requeridos em cada entrada J e K.

ATUAL			J _A
C	B	A	
0	0	0	1
0	0	1	x
0	1	0	1
0	1	1	x
1	0	0	0
1	0	1	x
1	1	0	0
1	1	1	x



Projeto de Contadores Síncronos

6. Implemente as expressões finais:



Projeto de Contadores Síncronos

- Projetar contadores síncronos com FFs D é mais fácil do que com FFs J-K.
- Considere o projeto do contador de módulo 5:

Estado ATUAL			PRÓXIMO estado			Entradas de controle		
C	B	A	C	B	A	D _C	D _B	D _A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

Projeto de Contadores Síncronos

- Considere o projeto do contador de módulo 5:

	\bar{A}	A
$\bar{C} \bar{B}$	0	0
$\bar{C} B$	0	(1)
C B	0	0
C \bar{B}	0	0

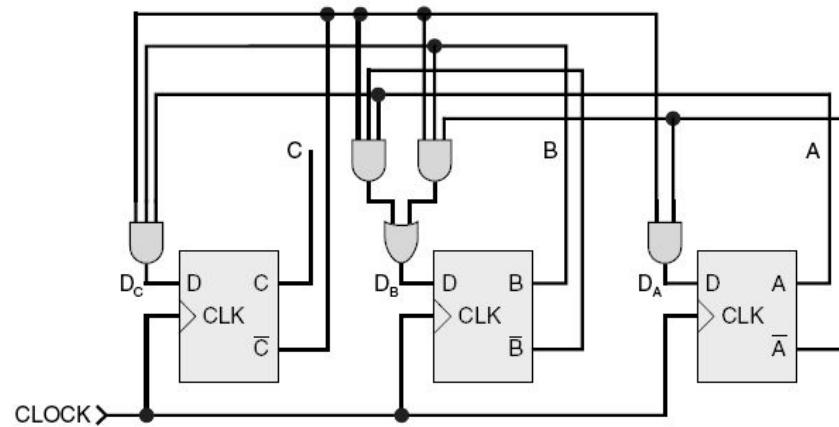
$$D_c = \bar{C} B A$$

	\bar{A}	A
$\bar{C} \bar{B}$	0	(1)
$\bar{C} B$	(1)	0
C B	0	0
C \bar{B}	0	0

$$D_B = \bar{C} \bar{B} A + \bar{C} B \bar{A}$$

	\bar{A}	A
$\bar{C} \bar{B}$	(1)	0
$\bar{C} B$	1	0
C B	0	0
C \bar{B}	0	0

$$D_A = \bar{C} \bar{A}$$



Resumo

- Flip-Flop D

Tabela Verdade

CLK	D	Q^*
	X	Q
↑	0	0
↑	1	1

Tabela Caraterística

Q	D	Q^*
0	0	0
0	1	1
1	0	0
1	1	1

Tabela Excitação

Q	Q^*	D
0	0	0
0	1	1
1	0	0
1	1	1

Resumo

- Flip-Flop JK

Tabela Verdade

CLK	J	K	Q^*
	X	X	Q
↑	0	0	Q
↑	0	1	0
↑	1	0	1
↑	1	1	Q'

Tabela Caraterística

Q	J	K	Q^*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Tabela Excitação

Q	Q^*	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Multiplicador

- Exemplo - Multiplicação manual:
 $(10111 \times 10011)_2$
- Note que a soma dos produtos parciais de n dígitos requer a soma de até n dígitos (com *vai-um*) em cada coluna
- Note também que uma multiplicação de $n \times m$ dígitos gera um resultado de até $m + n$ dígitos

Como implementar este algoritmo em
hardware digital?

23	Multiplicando	10111
<u>19</u>	Multiplicador	<u>10011</u>
{ Produtos Parciais 10111 10111 00000 00000 10111 ————— 110110101		
437	Produto	110110101

Multiplicador

- Em vez de usar um somador que some **n números** simultaneamente, é mais barato ter um circuito que some **dois números** apenas
- Em vez de fazer um **shift para a esquerda do multiplicando** a ser somado ao produto parcial, é melhor fazer um **shift para a direita do produto parcial** (assim, é possível usar um somador com n posições apenas, em vez de um de $2n$ posições)
- Quando o *bit* correspondente do multiplicador for 0, não há necessidade de somar todos os 0's ao produto parcial, já que eles não alteram o resultado

Multiplicador

23

10111

19

10011

10111

10111

00000

00000

10111

110110101

Multiplicador - Algoritmo Hardware

23

10111

Multiplicando

Sempre que o *bit* do multiplicador for 1, é adicionado o multiplicando, seguido de um *shift >* do produto parcial

19

10011

Multiplicador

Se o *bit* do multiplicador for 0, apenas o *shift >* é efetuado

00000

Produto parcial **inicial**

10111

Como o *bit* multiplicador é 1, some o multiplicando

10111

Produto parcial depois da soma e antes do *shift*

010111

Produto parcial depois do *shift*

10111

Como o *bit* multiplicador é 1, some o multiplicando

1000101

Produto parcial depois da soma e antes do *shift*

Overflow (vai-um)

1000101

Produto parcial depois do *shift*

01000101

Produto parcial depois do *shift* (multiplicador = 0)

001000101

Produto parcial depois do *shift* (multiplicador = 0)

10111

Como o *bit* multiplicador é 1, some o multiplicando

110110101

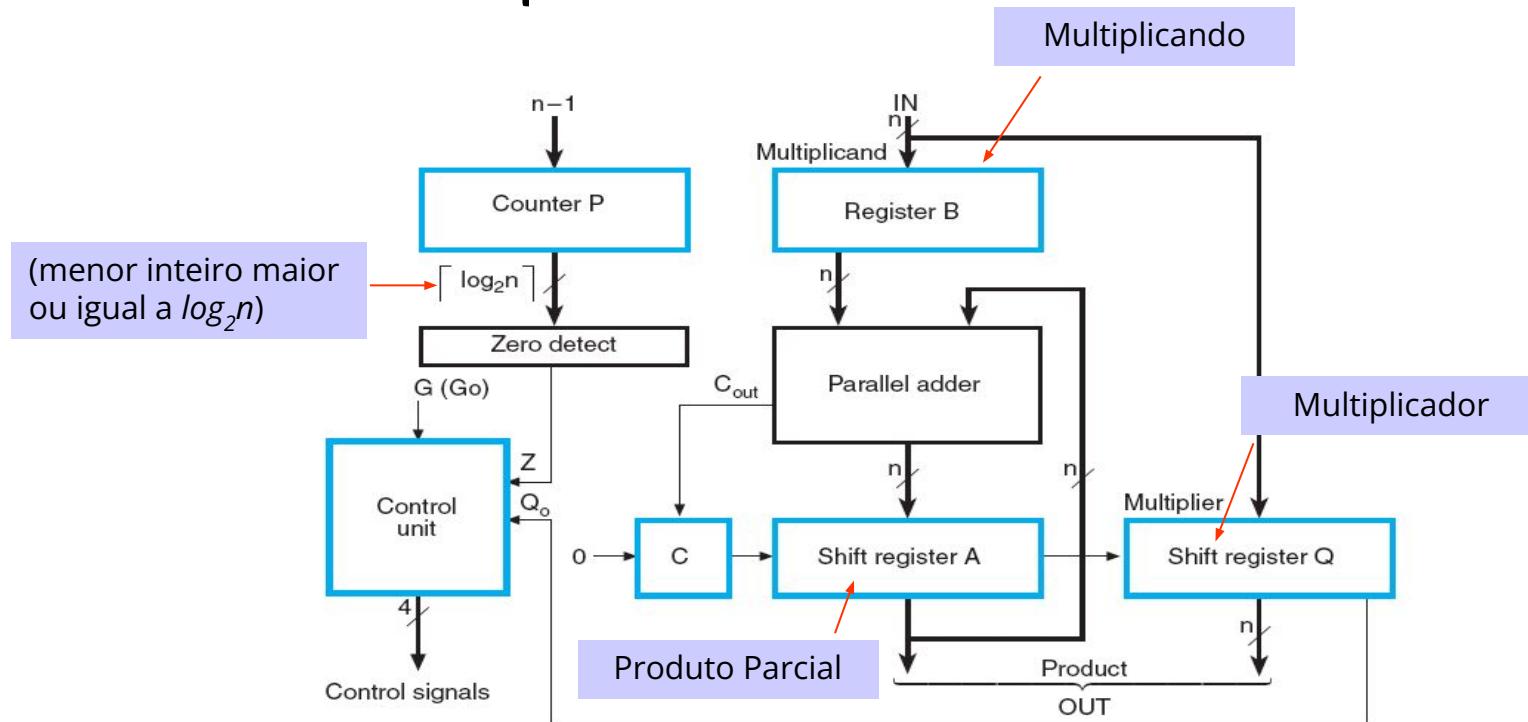
Produto parcial depois da soma e antes do *shift*

437

0110110101

Produto depois do *shift* final

Blocos do Multiplicador



Operações no Multiplicador

1. O **multiplicando** (1º operando) é carregado no **Registrador B**
2. O **multiplicador** (2º operando) é carregado no **Registrador Q**
3. Os **Registradores C||A** são inicializados em 0 quando **G** recebe 1
4. Os **produtos parciais** são formados nos **Registradores C||A||Q**
5. Cada *bit* do multiplicador, começando com o LSB, é processado (se o bit for 1, usar o somador para somar B ao produto parcial; se o bit for 0, não fazer nada)

Operações no Multiplicador

6. O conteúdo dos **registradores C | A | Q** são **deslocados para a direita**
 - Os bits LSB do produto parcial vão preenchendo as posições vacantes MSB de Q na medida em que o multiplicador vai sendo deslocado para fora de Q
 - Se ocorrer **overflow** na adição, o *vai-um* é recuperado do **FF C** durante o deslocamento para a direita
7. Os passos 5 e 6 são repetidos até que a condição Counter P = 0 seja atingida e sinalizada pelo detector de zero (Zero detect)
 - O Contador P é inicializado no passo 4 em n-1, n = nº. de bits no multiplicador, e seu conteúdo é testado antes de fazer um decremento

Controle do Multiplicador

- O controle do Multiplicador pode ser feito usando três estados, de acordo com um modelo combinado Mealy - Moore:
 - **Estado IDLE** – no qual:
 - as saídas da multiplicação anterior são mantidas até que Q seja carregado com o novo multiplicando
 - a entrada G é usada como condição para iniciar a multiplicação, e
 - C, A e P são inicializados

Operações no Multiplicador

- **Estado MUL0** – no qual:
 - a adição condicional é realizada baseada no valor de Q_0
- **Estado MUL1** – no qual:
 - é realizado um deslocamento para a direita para obter o produto parcial e posicionar o próximo *bit* do multiplicador em Q_0
 - é verificada a contagem final até 0, realizada no contador P, para avaliar o término ou a continuação da multiplicação

Máquina de Estados

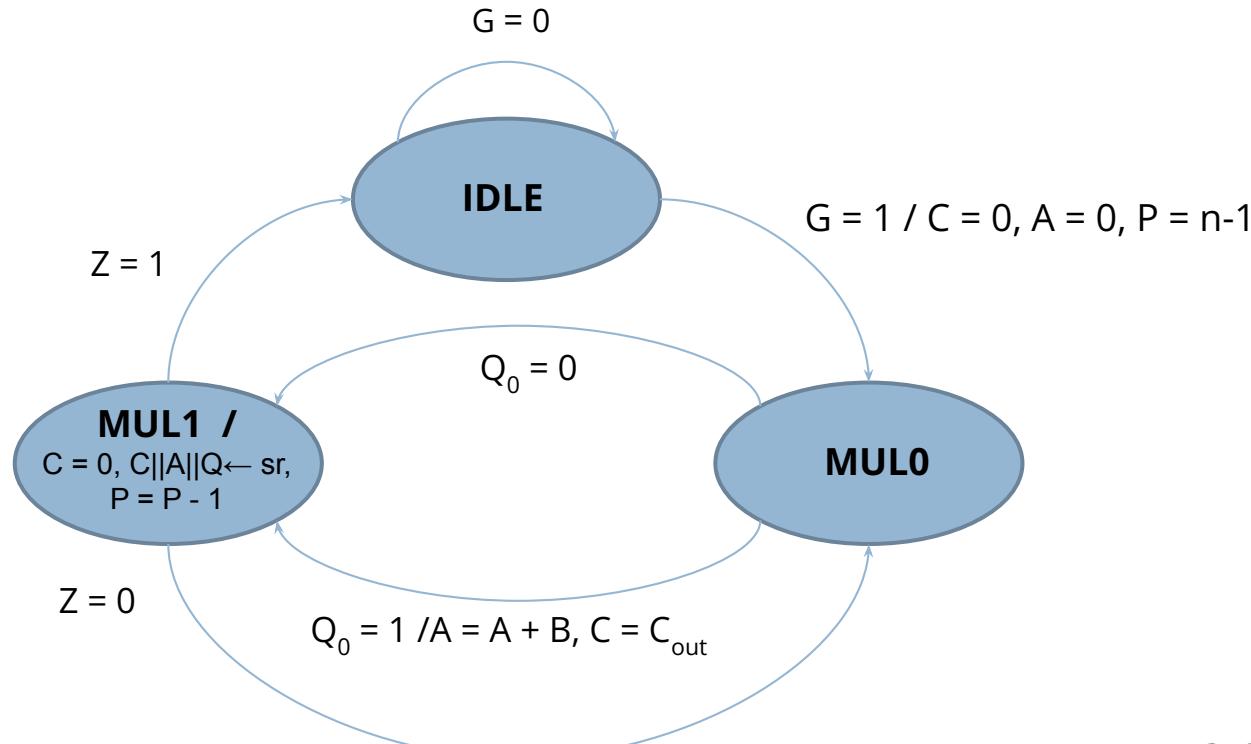


Tabela de Estados

Present state			Inputs		Next state		Decoder Outputs		
Name	M ₁	M ₀	G	Z	M ₁	M ₀	IDLE	MUL0	MUL1
IDLE	0	0	0	×	0	0	1	0	0
	0	0	1	×	0	1	1	0	0
MUL0	0	1	×	×	1	0	0	1	0
MUL1	1	0	×	0	0	1	0	0	1
	1	0	×	1	0	0	0	0	1
—	1	1	×	×	×	×	×	×	×