SD132 – Circuitos Digitais II
A-203 Contadores Síncronos

Autores	
Daniel Muñoz Arboleda (UnB)	Gilmar Silva Beserra (UnB)
Nome 3 (INSTITUIÇÃO)	Nome 4 (INSTITUIÇÃO)

Histórico de revisões		
06/01/2025	V1.0	Versão inicial

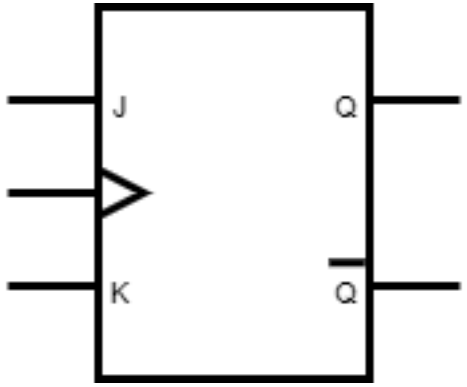
Tópicos

- Revisão de Flip-Flop JK e Flip-Flop D
- Projeto de contadores síncronos
- Exemplo contador sequencial
- Implementação manual do contador em VHDL
- Exemplo sequência aleatória sem repetição
- Implementação manual de sequência aleatória em VHDL
- Implementação estrutural em VHDL

Página em branco.

Aula 1
Contadores Síncronos

Revisão Flip-flop JK

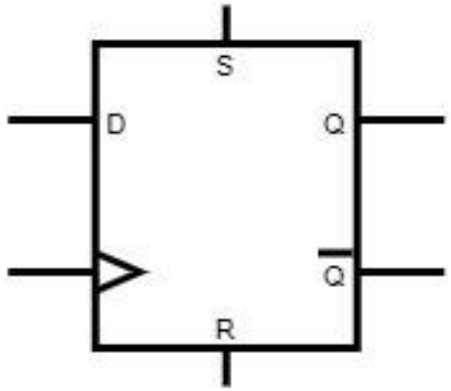


J	K	CLK	Saída - Próximo estado Q(n+1)
0	0	↑	Q(n) (saída não muda)
1	0	↑	1
0	1	↑	0
1	1	↑	Q'(n) (comuta=toggle)

Reinterpretando a tabela característica:

Estado atual Q(n)	Próximo estado Q(n+1)	J	K	Observação
0	0	0	x	Memória (JK=00) ou reset (JK=01)
0	1	1	x	Set (JK=10) ou toggle (JK=11)
1	0	x	1	Reset (JK=01) ou toggle (JK=11)
1	1	x	0	Memória (JK=00) ou set (JK=10)

Revisão Flip-flop D



D Flip Flop Truth Table

CL (Note 1)	D	R	S	Q	$\overline{Q}$
⌐	0	0	0	0	1
⌐	1	0	0	1	0
⌐	x	0	0	Q	$\overline{Q}$
x	x	1	0	0	1
x	x	0	1	1	0
x	x	1	1	1	1

Reinterpretando a tabela característica (desconsiderando as entradas assíncronas Preset e Reset):

Estado atual Q(n)	Próximo estado Q(n+1)	D	CLK
0	0	0	↑
0	1	1	↑
1	0	0	↑
1	1	1	↑

Definições

- Nos contadores **síncronos** todos os Flip-Flops dependem do mesmo sinal de clock. Portanto, todos os Flip-flops mudam simultaneamente de estado.
- Contadores síncronos requerem mais portas lógicas que os contadores assíncronos, mas não requerem usar as entradas assíncronas de Preset, evitando problemas de sincronismo entre os Flip-flops.
- Contadores síncronos são usados em circuitos de alta frequência onde os estados temporários e atrasos produzidos pelos clocks em cascata dos contadores assíncronos são indesejáveis.
- Contadores síncronos podem ser projetados para seguir sequência de dados que não são somente crescimentos ou decrescimentos unitários. O projeto de contadores síncronos segue sempre o mesmo padrão, independente da sequência necessária.

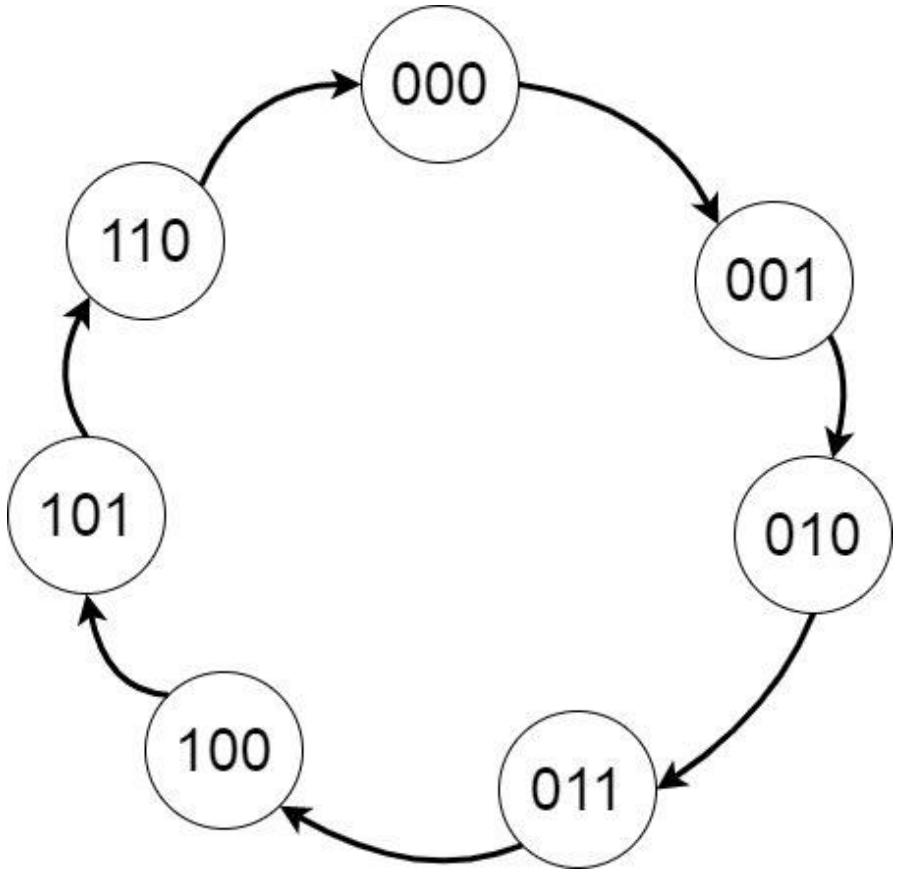
Definição: Diagrama de transição

- Figura constituída por estados e transições entre os estados.
- Os estados representam os valores possíveis da sequência desejada.
- As transições levam o contador de um estado (um valor) para o próximo estado (próximo valor na sequência).

Definição: Diagrama de transição

- Figura constituída por estados e transições entre os estados.
- Os estados representam os valores possíveis da sequência desejada.
- As transições levam o contador de um estado (um valor) para o próximo estado (próximo valor na sequência).

Diagrama de um contador ascendente módulo 7 (conta de 0 a 6):



Definição: Tabela de transição

- É o diagrama de transição em representação tabular.
- Colunas: estado atual e próximo estado em código binário.
- Cada linha contém um valor de estado atual e seu respectivo próximo estado.

Definição: Tabela de transição

- É o diagrama de transição em representação tabular.
- Colunas: estado atual e próximo estado em código binário.
- Cada linha contém um valor de estado atual e seu respectivo próximo estado.

Tabela de transição de contador ascendente módulo 7 (conta de 0 a 6):

Estado atual				**Próximo estado**			
Dec	Q2	Q1	Q0	Q2*	Q1*	Q0*	Dec
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4	1	0	0	1	0	1	5
5	1	0	1	1	1	0	6
6	1	1	0	0	0	0	0

Definições

- Em um contador síncrono, todos os estados possíveis devem produzir um próximo estado específico.
- O próximo estado é determinado pela sequência que o contador deve seguir.
- Se a sequência tiver menos estados do que o máximo de valores possíveis (2^N-1, sendo *N* o número de Flip-flops), então teremos estados não determinados.
- Neste último caso, deve-se especificar o que fazer caso o contador entre em um desses estados, visando garantir que o contador não entre em um loop indesejado.
- A tabela de transição **deve** conter todos os possíveis valores.

Projeto de contadores síncronos

Objetivo: determinar quais saídas do contador devem ser conectadas, através de portas lógicas, nas entradas dos Flip-flops.

- Passo 1: a partir da sequência desejada, obter diagrama de transição (também chamado de diagrama de transição de estados).
- Passo 2: transformar diagrama em uma tabela de transição (estado atual e próximo estado).
- Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip-flops que representam a transição dos bits de estado.
- Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.
- Passo 5: Implementação e validação.

Contador síncrono ascendente módulo 7 com FF D

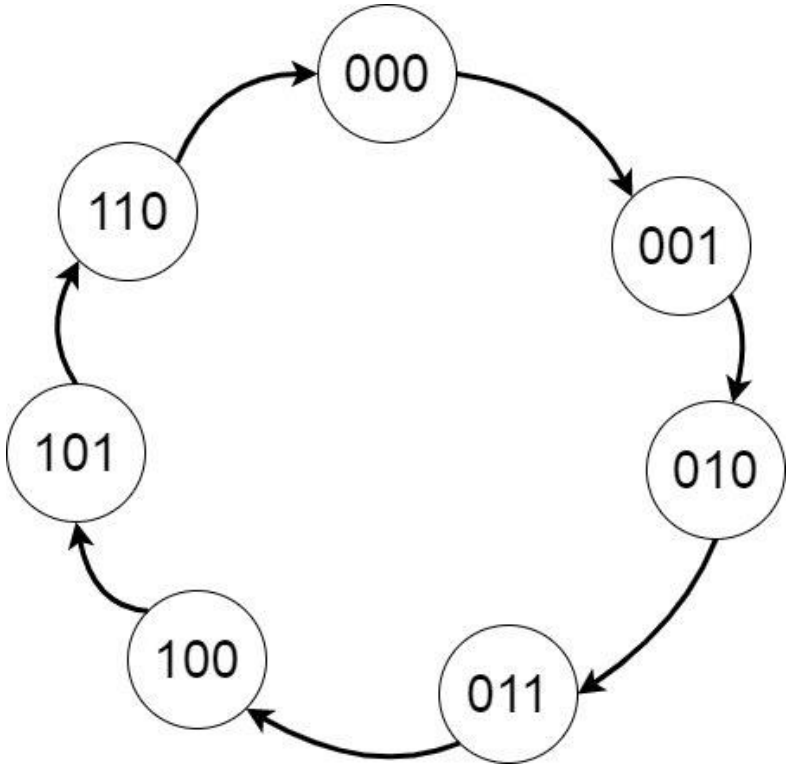
Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 1: Diagrama de transição

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 1: Diagrama de transição



Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 2: Tabela de transição

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 2: Tabela de transição

Estado atual				**Próximo estado**			
Dec	Q2	Q1	Q0	Q2*	Q1*	Q0*	Dec
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4	1	0	0	1	0	1	5
5	1	0	1	1	1	0	6
6	1	1	0	0	0	0	0

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip-flops que representam a transição dos bits de estado.

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip-flops que representam a transição dos bits de estado.

Usando tabela do Flip-flop D:

Estado atual Q(n)	Próximo estado Q(n+1)	D	CLK
0	0	0	↑
0	1	1	↑
1	0	0	↑
1	1	1	↑

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip-flops que representam a transição dos bits de estado.

Estado atual				**Próximo estado**				**Entrada FF D**		
Dec	Q2	Q1	Q0	Q2*	Q1*	Q0*	Dec	D2	D1	D0
0	0	0	0	0	0	1	1	0	0	1
1	0	0	1	0	1	0	2	0	1	0
2	0	1	0	0	1	1	3	0	1	1
3	0	1	1	1	0	0	4	1	0	0
4	1	0	0	1	0	1	5	1	0	1
5	1	0	1	1	1	0	6	1	1	0
6	1	1	0	0	0	0	0	0	0	0

Estado atual Q(n)	**Próximo estado Q(n+1)**	**D**	**CLK**
0	0	0	↑
0	1	1	↑
1	0	0	↑
1	1	1	↑

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual				**Entrada FF D**		
Dec	Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	0	0	0

Mapa K para D2:

Q2 Q1 \ Q0	**0**	**1**
0 0		
0 1		1
1 1		
1 0	1	1

D2 = Q2Q1' + Q2'Q1Q0

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual				**Entrada FF D**		
Dec	Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	0	0	0

Mapa K para D1:

Q2 Q1 \ Q0	**0**	**1**
0 0		1
0 1	1	
1 1		
1 0		1

D1 = Q1'Q0 + Q2'Q1Q0'

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual				**Entrada FF D**		
Dec	Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	0	0	0

Mapa K para D0:

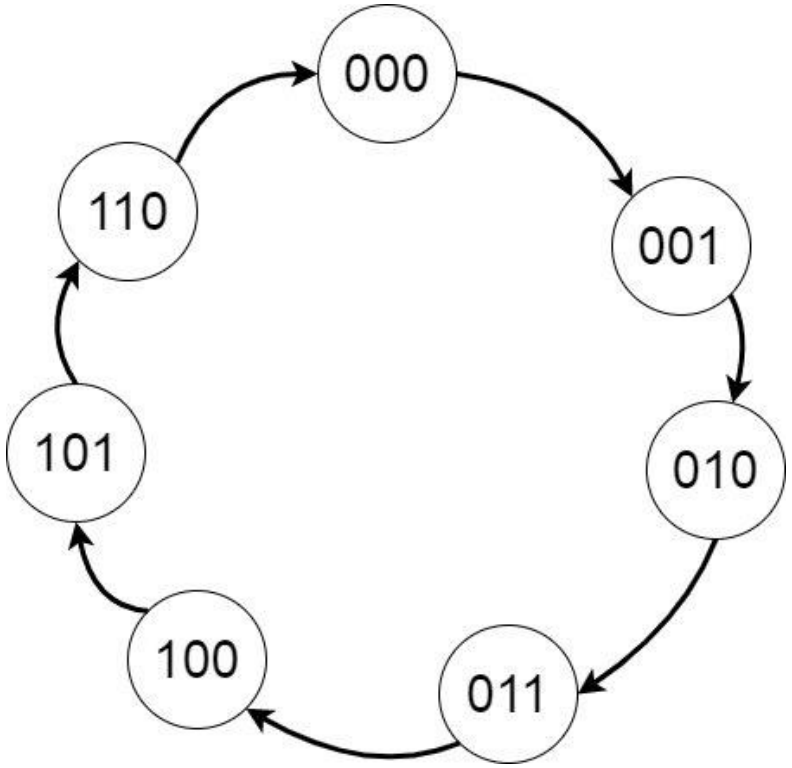
Q2 Q1 \ Q0	**0**	**1**
0 0	1	
0 1	1	
1 1		
1 0	1	

D0 = Q2'Q0' + Q2Q1'Q0'

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.



D2 = Q2Q1' + Q2'Q1Q0

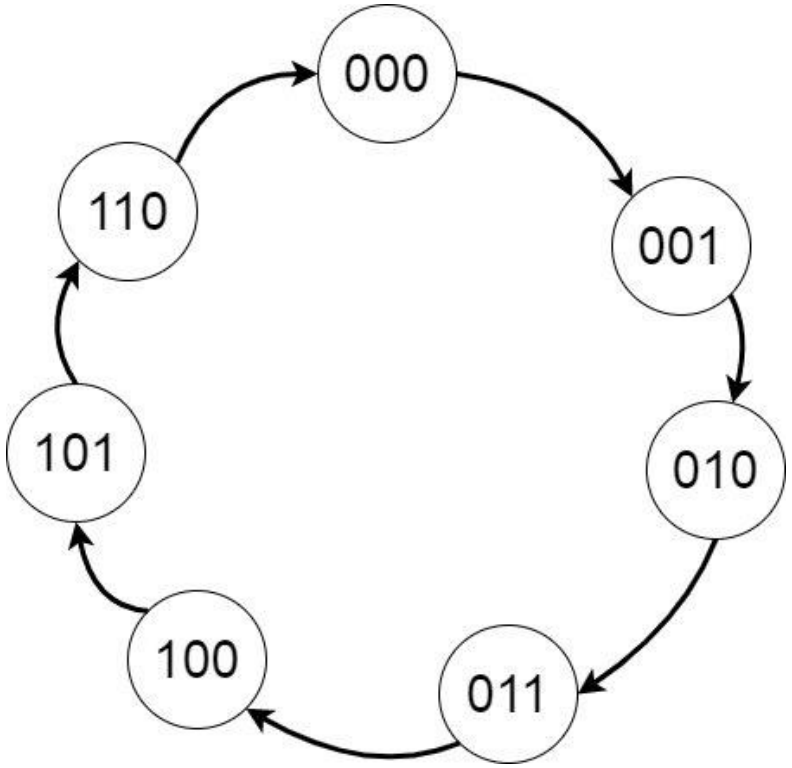
D1 = Q1'Q0 + Q2'Q1Q0'

D0 = Q2'Q0' + Q2Q1'Q0'

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 4b (opcional): Desenhe o circuito equivalente (exercício)



D2 = Q2Q1' + Q2'Q1Q0

D1 = Q1'Q0 + Q2'Q1Q0'

D0 = Q2'Q0' + Q2Q1'Q0'

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 5: Implementação e validação usando a linguagem Verilog:

D2 = Q2Q1' + Q2'Q1Q0

D1 = Q1'Q0 + Q2'Q1Q0'

D0 = Q2'Q0' + Q2Q1'Q0'

```
1   module cnt_asc_mod7(
2       input  reset,
3       input  clk,
4       output  [2:0] led
5   );
6
7       // Internal signals
8       reg [2:0] Q;
9       wire [2:0] D;
10
11      // Sequential logic to update Q
12      always @(posedge clk or posedge reset) begin
13          if (reset) begin
14              Q <= 3'b000;  // Reset Q to 000
15          end else begin
16              Q <= D;       // Update Q with D on rising edge of clk
17          end
18      end
```

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 5: Implementação e validação usando a linguagem Verilog:

D2 = Q2Q1' + Q2'Q1Q0

D1 = Q1'Q0 + Q2'Q1Q0'

D0 = Q2'Q0' + Q2Q1'Q0'

```
20      // Combinatorial logic for D
21      assign D[2] = (Q[2] & ~Q[1]) | (~Q[2] & Q[1] & Q[0]);
22      assign D[1] = (~Q[1] & Q[0]) | (~Q[2] & Q[1] & ~Q[0]);
23      assign D[0] = (~Q[2] & ~Q[0]) | (Q[2] & ~Q[1] & ~Q[0]);
24
25
26      assign led = Q;  // Assign Q to led
27
28  endmodule
```

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 5: Implementação e validação usando a linguagem VHDL:

D2 = Q2Q1' + Q2'Q1Q0

D1 = Q1'Q0 + Q2'Q1Q0'

D0 = Q2'Q0' + Q2Q1'Q0'

```
14  library IEEE;
15  use IEEE.STD_LOGIC_1164.ALL;
16
17  entity cnt_asc_mod7 is
18      Port ( reset : in STD_LOGIC;
19             clk : in STD_LOGIC;
20             led : out STD_LOGIC_VECTOR (2 downto 0));
21  end cnt_asc_mod7;
22
23  architecture Behavioral of cnt_asc_mod7 is
24
25  signal Q : std_logic_vector(2 downto 0) := "000";
26  signal D : std_logic_vector(2 downto 0) := "000";
27
28  begin
```

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 5: Implementação e validação: vamos usar a linguagem VHDL:

D2 = Q2Q1' + Q2'Q1Q0

D1 = Q1'Q0 + Q2'Q1Q0'

D0 = Q2'Q0' + Q2Q1'Q0'

```
30      process(clk,reset)
31      begin
32          if reset='1' then
33              Q <= "000";
34          elsif rising_edge(clk) then
35              Q <= D;
36          end if;
37      end process;
38
39      D(2) <= (Q(2) and not Q(1)) or (not Q(2) and Q(1) and Q(0));
40      D(1) <= (not Q(1) and Q(0)) or (not Q(2) and Q(1) and not Q(0));
41      D(0) <= (not Q(2) and not Q(0)) or (Q(2) and not Q(1) and not Q(0));
42
43      led <= Q;
44
45  end Behavioral;
```

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

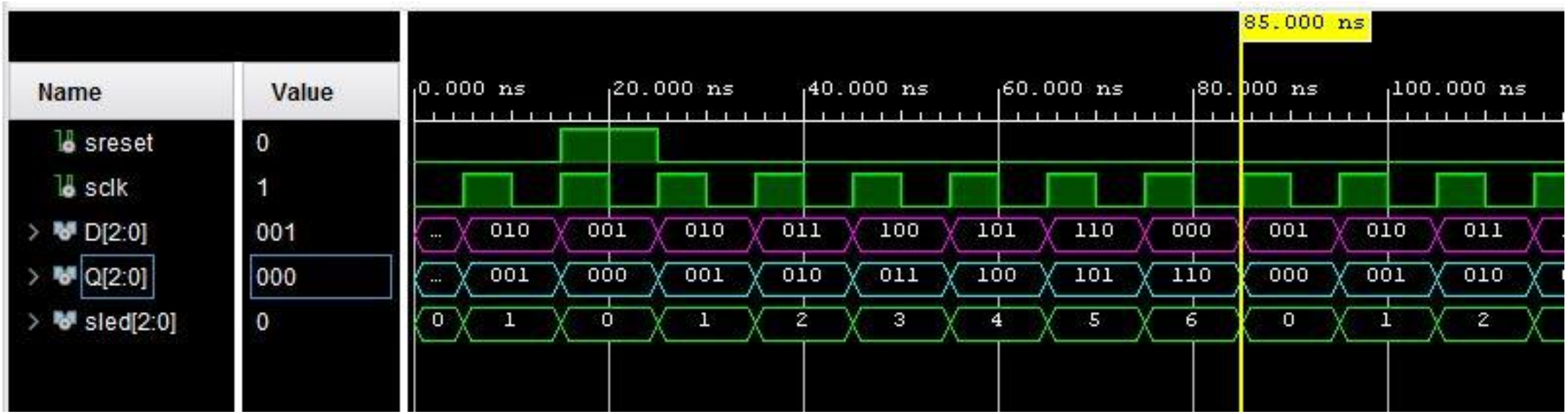
Passo 5: Implementação e validação. Testbench: usando fclk=100MHz, Tclk=10 ns.

```
23  component cnt_asc_mod7 is
24      Port ( reset : in STD_LOGIC;
25             clk : in STD_LOGIC;
26             led : out STD_LOGIC_VECTOR (2 downto 0));
27  end component;
28
29  signal sreset, sclk : std_logic := '0';
30  signal sled : std_logic_vector(2 downto 0) := "000";
31
32  begin
33
34      sclk <= not sclk after 5 ns;
35      sreset <= '0', '1' after 15 ns, '0' after 25 ns;
36
37      uut: cnt_asc_mod7 port map(
38          reset   => sreset,
39          clk     => sclk,
40          led     => sled);
```

Contador síncrono ascendente módulo 7 com FF D

Exemplo: Implemente um circuito sequencial síncrono de um contador módulo 7.

Passo 5: Implementação e validação. Resultado de simulação comportamental:



Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 1: Diagrama de estado (idêntico ao exemplo anterior)

Passo 2: Tabela de transição (idêntico ao exemplo anterior)

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 1: Diagrama de estado (idêntico ao exemplo anterior)

Passo 2: Tabela de transição (idêntico ao exemplo anterior)

Estado atual				**Próximo estado**			
Dec	Q2	Q1	Q0	Q2*	Q1*	Q0*	Dec
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4	1	0	0	1	0	1	5
5	1	0	1	1	1	0	6
6	1	1	0	0	0	0	0

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip flops que representam a transição dos bits de estado.

Estado atual				**Próximo estado**			
Dec	Q2	Q1	Q0	Q2*	Q1*	Q0*	Dec
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4	1	0	0	1	0	1	5
5	1	0	1	1	1	0	6
6	1	1	0	0	0	0	0

Tabela verdade do FF JK

Estado atual Q(n)	**Próximo estado Q(n+1)**	**J**	**K**
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip-flops que representam a transição dos bits de estado.

Estado atual			**Próximo estado**			**Entradas FF JK**		
Q2	Q1	Q0	Q2*	Q1*	Q0*	J2K2	J1K1	J0K0
0	0	0	0	0	1	0x	0x	1x
0	0	1	0	1	0	0x	1x	x1
0	1	0	0	1	1	0x	x0	1x
0	1	1	1	0	0	1x	x1	x1
1	0	0	1	0	1	x0	0x	1x
1	0	1	1	1	0	x0	1x	x1
1	1	0	0	0	0	x1	x1	0x
1	1	1	0	0	0	x1	x1	x1

Tabela verdade do FF JK

Estado atual Q(n)	**Próximo estado Q(n+1)**	**J**	**K**
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Adicionando estado não codificado 111.
Por segurança de 111 vai para 000.

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual			**Entradas FF JK**		
Q2	Q1	Q0	**J2**K2	J1K1	J0K0
0	0	0	0x	0x	1x
0	0	1	0x	1x	x1
0	1	0	0x	x0	1x
0	1	1	1x	x1	x1
1	0	0	x0	0x	1x
1	0	1	x0	1x	x1
1	1	0	x1	x1	0x
1	1	1	x1	x1	x1

Mapa K para J2:

Q2 Q1 \ Q0	**0**	**1**
0 0		
0 1		1
1 1	x	x
1 0	x	x

J2 = Q1Q0

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual			**Entradas FF JK**		
Q2	Q1	Q0	J2**K2**	J1K1	J0K0
0	0	0	0x	0x	1x
0	0	1	0x	1x	x1
0	1	0	0x	x0	1x
0	1	1	1x	x1	x1
1	0	0	x0	0x	1x
1	0	1	x0	1x	x1
1	1	0	x1	x1	0x
1	1	1	x1	x1	x1

Mapa K para K2:

Q2 Q1 \ Q0	**0**	**1**
0 0	x	x
0 1	x	x
1 1	1	1
1 0		

K2 = Q1

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual			**Entradas FF JK**		
Q2	Q1	Q0	J2K2	**J1**K1	J0K0
0	0	0	0x	0x	1x
0	0	1	0x	1x	x1
0	1	0	0x	x0	1x
0	1	1	1x	x1	x1
1	0	0	x0	0x	1x
1	0	1	x0	1x	x1
1	1	0	x1	x1	0x
1	1	1	x1	x1	x1

Mapa K para J1:

Q2 Q1 \ Q0	**0**	**1**
0 0		1
0 1	x	x
1 1	x	x
1 0		1

J1 = Q0

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual			**Entradas FF JK**		
Q2	Q1	Q0	J2K2	J1**K1**	J0K0
0	0	0	0x	0x	1x
0	0	1	0x	1x	x1
0	1	0	0x	x0	1x
0	1	1	1x	x1	x1
1	0	0	x0	0x	1x
1	0	1	x0	1x	x1
1	1	0	x1	x1	0x
1	1	1	x1	x1	x1

Mapa K para K1:

Q2 Q1 \ Q0	**0**	**1**
0 0	x	x
0 1		1
1 1	1	1
1 0	x	x

K1 = Q2 + Q0

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual			**Entradas FF JK**		
Q2	Q1	Q0	J2K2	J1K1	**J0**K0
0	0	0	0x	0x	1x
0	0	1	0x	1x	x1
0	1	0	0x	x0	1x
0	1	1	1x	x1	x1
1	0	0	x0	0x	1x
1	0	1	x0	1x	x1
1	1	0	x1	x1	0x
1	1	1	x1	x1	x1

Mapa K para J0:

Q2 Q1 \ Q0	**0**	**1**
0 0	1	x
0 1	1	x
1 1		x
1 0	1	x

J0 = Q1' + Q2'

Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

Estado atual			**Entradas FF JK**		
Q2	Q1	Q0	J2K2	J1K1	J0**K0**
0	0	0	0x	0x	1x
0	0	1	0x	1x	x1
0	1	0	0x	x0	1x
0	1	1	1x	x1	x1
1	0	0	x0	0x	1x
1	0	1	x0	1x	x1
1	1	0	x1	x1	0x
1	1	1	x1	x1	x1

Mapa K para K0:

Q2 Q1 \ Q0	**0**	**1**
0 0	x	1
0 1	x	1
1 1	x	1
1 0	x	1

K0 = Q0

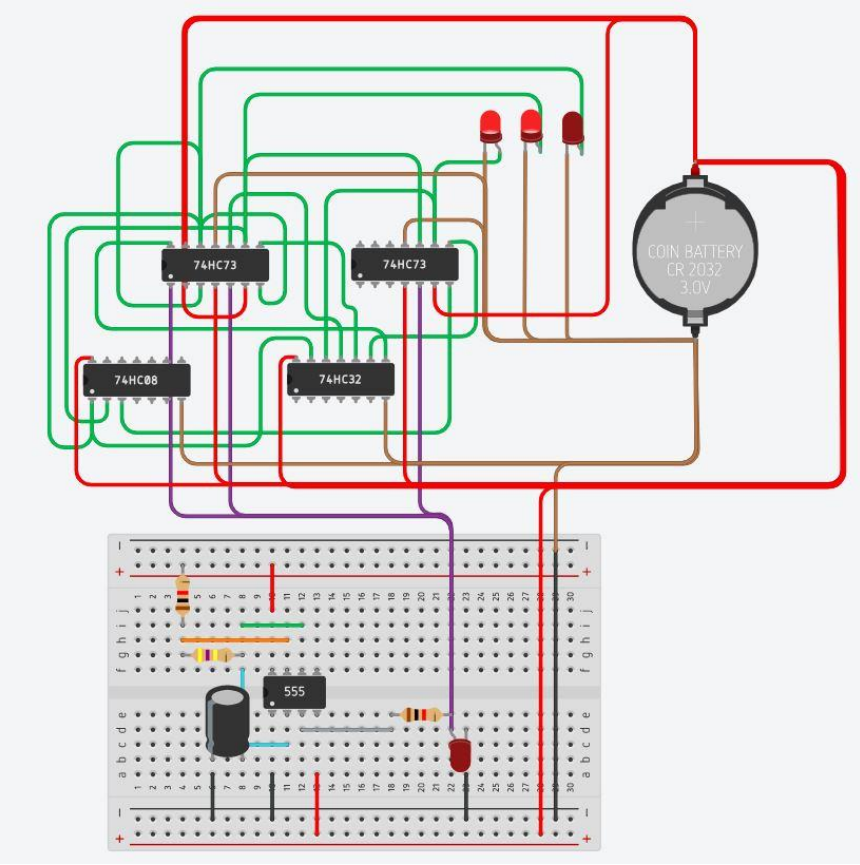
Contador síncrono ascendente módulo 7 com FF JK

Exemplo: Implemente um contador síncrono ascendente módulo 7 com FF JK.

Passo 4: Síntese das equações booleanas usando mapa de Karnaugh.

J2 = Q1Q0 J1 = Q0 J0 = Q1' + Q2'

K2 = Q1 K1 = Q2 + Q0 K0 = Q0



Implementação no Tinkercad disponível:

https://www.tinkercad.com/things/cZneC8GCd3w-contador-ascendente-mod-7ff-jk?sharecode=_vTiQQ0HjxZNhFkbvN00KodIgwgSnTneOHgQMC71-MQ

Contador síncrono descendente módulo 12 com FF D

Exercício 1 em sala de aula: projete um contador síncrono descendente módulo 13 (conta de 12 a 0) usando Flip-flops D. Realize a implementação em Verilog e em VHDL, crie um testbench, e verifique o funcionamento do circuito usando simulação comportamental (RTL). Utilize uma frequência de clock de 100 MHz.

Página em branco.

Aula 2
Contadores síncronos - Sequência aleatória sem repetição

Contador síncrono - sequência aleatória

- Em um contador síncrono, todos os estados possíveis devem produzir um próximo estado específico.
- O próximo estado é determinado pela sequência que o contador deve seguir.
- Se a sequência tiver menos estados do que o máximo de valores possíveis (2^N-1, sendo N o número de Flip-flops), então teremos estados não determinados.
- Neste último caso, deve-se especificar o que fazer caso o contador entre em um desses estados, visando garantir que o contador não entre em um loop indesejado.

Contador síncrono - sequência aleatória

- Uma sequência aleatória é uma sequência de números que não segue uma ordem numérica.
- O mesmo procedimento para projetar contadores síncronos pode ser usado para gerar circuitos que percorrem uma sequência não ordenada de números (sequência previamente definida).
- Observação: não confundir tal circuito com um gerador de números aleatórios.
- Na construção da Tabela de transição, a **primeira coluna** (estado atual) deve conter os valores possíveis **em ordem crescente**. A segunda coluna (próximo estado) deve conter o próximo valor da sequência aleatória.

Contador síncrono - sequência aleatória

Exemplo: implementar um circuito sequencial síncrono que percorra a sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete.

Contador síncrono - sequência aleatória

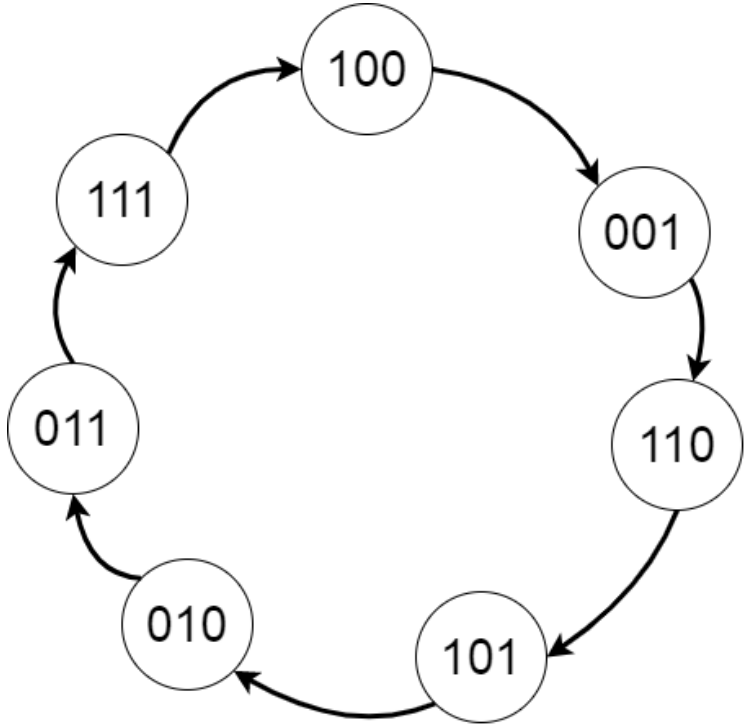
Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 1: diagrama de estados

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 1: diagrama de estados



Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 2: Tabela de transição

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete.

Passo 2: Tabela de transição

Estado atual				**Próximo estado**			
Dec	Q2	Q1	Q0	Q2*	Q1*	Q0*	Dec
0	0	0	0	1	0	0	4
1	0	0	1	1	1	0	6
2	0	1	0	0	1	1	3
3	0	1	1	1	1	1	7
4	1	0	0	0	0	1	1
5	1	0	1	0	1	0	2
6	1	1	0	1	0	1	5
7	1	1	1	0	0	0	0

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip-flops que representam a transição dos bits de estado.

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip-flops que representam a transição dos bits de estado.

Estado atual				**Próximo estado**				**Entradas FF D**		
Dec	Q2	Q1	Q0	Q2*	Q1*	Q0*	Dec	D2	D1	D0
0	0	0	0	1	0	0	4	1	0	0
1	0	0	1	1	1	0	6	1	1	0
2	0	1	0	0	1	1	3	0	1	1
3	0	1	1	1	1	1	7	1	1	1
4	1	0	0	0	0	1	1	0	0	1
5	1	0	1	0	1	0	2	0	1	0
6	1	1	0	1	0	1	5	1	0	1
7	1	1	1	0	0	0	0	0	0	0

Tabela verdade do Flip-flop D

Estado atual Q(n)	**Próximo estado Q(n+1)**	**D**	**CLK**
0	0	0	↑
0	1	1	↑
1	0	0	↑
1	1	1	↑

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 3: Adicionar colunas na tabela com as entradas necessárias dos Flip-flops que representam a transição dos bits de estado.

Estado atual			**Entradas FF D**		
Q2	Q1	Q0	D2	D1	D0
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	0	0	0

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 4: Obter expressões booleanas das entradas dos Flip-flops usando mapa de Karnaugh.

Estado atual			**Entradas FF D**		
Q2	Q1	Q0	D2	D1	D0
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	0	0	0

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 4: Obter expressões booleanas das entradas dos Flip-flops usando mapa de Karnaugh.

Estado atual			**Entradas FF D**		
Q2	Q1	Q0	D2	D1	D0
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	0	0	0

Mapa K para D2:

Q2 Q1 \ Q0	**0**	**1**
0 0	1	1
0 1		1
1 1	1	
1 0		

D2 = Q2'Q1' + Q2'Q0 + Q2Q1Q0'

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 4: Obter expressões booleanas das entradas dos Flip-flops usando mapa de Karnaugh.

Estado atual			**Entradas FF D**		
Q2	Q1	Q0	D2	D1	D0
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	0	0	0

Mapa K para D1:

Q2 Q1 \ Q0	**0**	**1**
0 0		1
0 1	1	1
1 1		
1 0		1

D1 = Q1'Q0 + Q2'Q0 + Q2'Q1

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 4: Obter expressões booleanas das entradas dos Flip-flops usando mapa de Karnaugh.

Estado atual			**Entradas FF D**		
Q2	Q1	Q0	D2	D1	D0
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	0	0	0

Mapa K para D0:

Q2 Q1 \ Q0	**0**	**1**
0 0		
0 1	1	1
1 1	1	
1 0	1	

D0 = Q2'Q1 + Q2Q0'

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 5: Implementação e validação.

D2 = Q2'Q1' + Q2'Q0 + Q2Q1Q0' D1 = Q1'Q0 + Q2'Q0 + Q2'Q1 D0 = Q2'Q1 + Q2Q0'

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 5: Implementação em Verilog.

D2 = Q2'Q1' + Q2'Q0 + Q2Q1Q0' D1 = Q1'Q0 + Q2'Q0 + Q2'Q1 D0 = Q2'Q1 + Q2Q0'

```
module cnt_random_seq(
    input  reset,
    input  clk,
    output  [2:0] led
);

reg [2:0] Q;
wire [2:0] D;
```

```
// Combinational logic for next state
assign D[2] = (~Q[2] & ~Q[1]) | (~Q[2] & Q[0]) | (Q[2] & Q[1] & ~Q[0]);
assign D[1] = (~Q[1] & Q[0]) | (~Q[2] & Q[0]) | (~Q[2] & Q[1]);
assign D[0] = (~Q[2] & Q[1]) | (Q[2] & ~Q[0]);

always @(posedge clk or posedge reset) begin
    if (reset)
        Q <= 3'b000;
    else
        Q <= D;
end

assign led = Q;
```

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 5: Implementação em VHDL.

D2 = Q2'Q1' + Q2'Q0 + Q2Q1Q0' D1 = Q1'Q0 + Q2'Q0 + Q2'Q1 D0 = Q2'Q1 + Q2Q0'

```
25  signal Q : std_logic_vector(2 downto 0) := "000";
26  signal D : std_logic_vector(2 downto 0) := "000";
27
28  begin
29
30      process(clk,reset)
31      begin
32          if reset='1' then
33              Q <= "000";
34          elsif rising_edge(clk) then
35              Q <= D;
36          end if;
37      end process;
38
39      D(2) <= (not Q(2) and not Q(1)) or (not Q(2) and Q(0)) or (Q(2) and Q(1) and not Q(0));
40      D(1) <= (not Q(1) and Q(0)) or (not Q(2) and Q(0)) or (not Q(2) and Q(1));
41      D(0) <= (not Q(2) and Q(1)) or (Q(2) and not Q(0));
42
43      led <= Q;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity cnt_random_seq is
    Port ( reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           led : out STD_LOGIC_VECTOR (2 downto 0));
end cnt_random_seq;

architecture Behavioral of cnt_random_seq is
```

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 5: validação (testbench)

D2 = Q2'Q1' + Q2'Q0 + Q2Q1Q0' D1 = Q1'Q0 + Q2'Q0 + Q2'Q1 D0 = Q2'Q1 + Q2Q0'

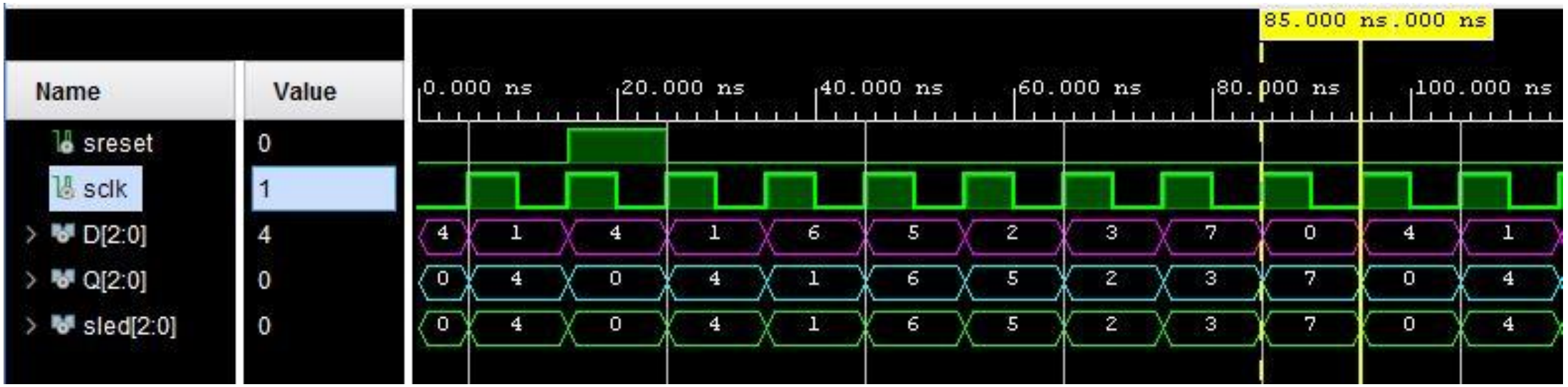
```
23  component cnt_random_seq is
24      Port ( reset : in STD_LOGIC;
25             clk : in STD_LOGIC;
26             led : out STD_LOGIC_VECTOR (2 downto 0));
27  end component;
28
29  signal sreset, sclk : std_logic := '0';
30  signal sled : std_logic_vector(2 downto 0) := "000";
31
32  begin
33
34      sclk <= not sclk after 5 ns;
35      sreset <= '0', '1' after 15 ns, '0' after 25 ns;
36
37      uut: cnt_random_seq port map(
38          reset   => sreset,
39          clk     => sclk,
40          led     => sled);
```

Contador síncrono - sequência aleatória

Exemplo: circuito para sequência aleatória 4- 1- 6- 5- 2- 3- 7- 0, repete

Passo 5: Implementação e validação. Resultado de simulação comportamental.

D2 = Q2'Q1' + Q2'Q0 + Q2Q1Q0' D1 = Q1'Q0 + Q2'Q0 + Q2'Q1 D0 = Q2'Q1 + Q2Q0'



Contador síncrono - sequência aleatória

Exercício 2 em sala de aula: implemente em Verilog em em VHDL a sequência aleatória 0-3-1-5-9-4-2-6-10-repete. Realize o testbench em Verilog e verifique o funcionamento em simulação comportamental (RTL). Use uma frequência de clock de 100 MHz.

Obs1: serão necessários 4 bits para codificar os nove (9) estados da sequência.

Obs2: não todos os estados são codificados.

Dica: use o primeiro valor da sequência como próximo estado dos estados não codificados.

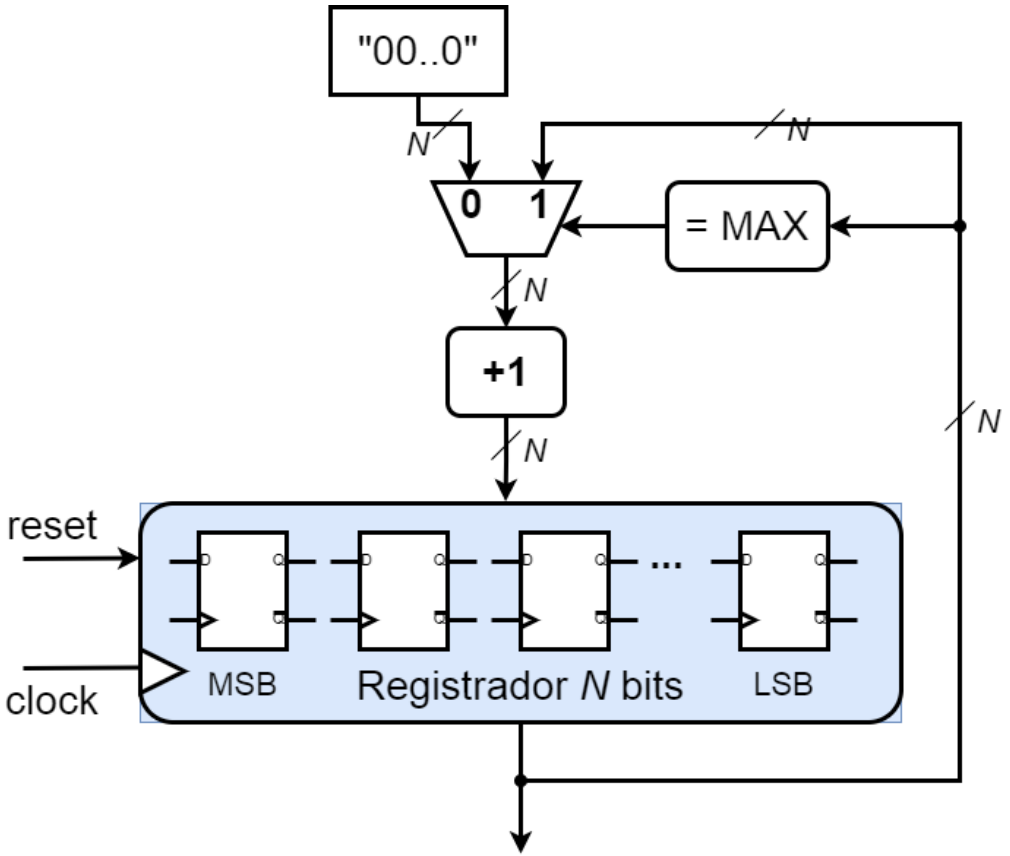
Página em branco.

Aula 3
Implementação em HDL de contadores síncronos

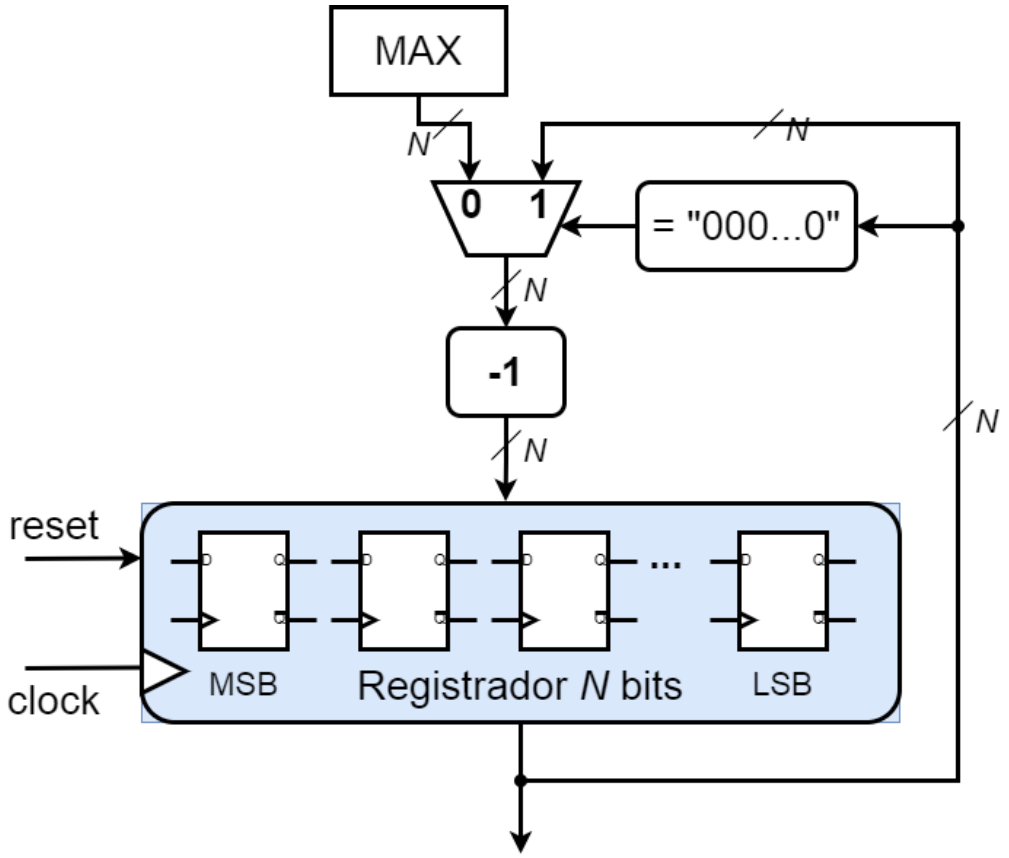
Contadores síncronos em HDL

- Somadores e comparadores podem ser usados para implementar contadores síncronos em VHDL.
- O módulo do contador define o seu valor máximo. Por exemplo, um contador ascendente módulo 100, inicia em zero e termina em 99 (valor máximo). Um contador descendende módulo 100, inicia em 99 (valor máximo) e termina em zero.
- Deve-se utilizar lógica sequencial síncrona para incrementar ou decrementar o contador a cada ciclo de relógio.
- A cada ciclo de clock deve-se comparar se o contador alcançou o valor máximo ou o zero, retornando ao valor valor padrão para iniciar um novo ciclo.

Contadores síncronos em HDL - arquitetura



contador ascendente



contador descendente

Contador síncrono ascendente

Exemplo: implemente em HDL um contador síncrono ascendente módulo 100. Utilize um sinal de clock de 100 MHz. Use reset assíncrono para apagar o registrador. Realize uma simulação comportamental e verifique o funcionamento do circuito.

Contador síncrono ascendente

Exemplo: contador ascendente síncrono módulo 100: implementação em Verilog

```
1   module cnt_asc_mod100 (
2       input reset,
3       input clk,
4       output [6:0] led
5   );
6
7       reg [6:0] cnt;
```

```
9       // Process behavior
10      always @(posedge clk or posedge reset) begin
11          if (reset) begin
12              cnt <= 7'b0000000;  // Reset to 0
13          end else begin
14              if (cnt == 7'b1100011) begin  // 99 in binary
15                  cnt <= 7'b0000000;  // Reset to 0 when 99 is reached
16              end else begin
17                  cnt <= cnt + 1;  // Increment the counter
18              end
19          end
20      end
21
22      // Assign the counter value to the LEDs
23      assign led = cnt;
24
25  endmodule
```

Contador síncrono ascendente

Exemplo: contador ascendente síncrono módulo 100: implementação em VHDL

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity cnt_asc_mod100 is
    Port ( reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           led : out STD_LOGIC_VECTOR(6 downto 0));
end cnt_asc_mod100;

architecture Behavioral of cnt_asc_mod100 is

signal cnt:std_logic_vector(6 downto 0):=(others=>'0');

begin

process(clk,reset)
begin
    if reset='1' then
        cnt <= (others=>'0');
    elsif rising_edge(clk) then
        if cnt = "1100011" then -- 99
            cnt <= (others=>'0');
        else
            cnt <= std_logic_vector(unsigned(cnt)+1);
        end if;
    end if;
end process;

led <= cnt;

end Behavioral;
```

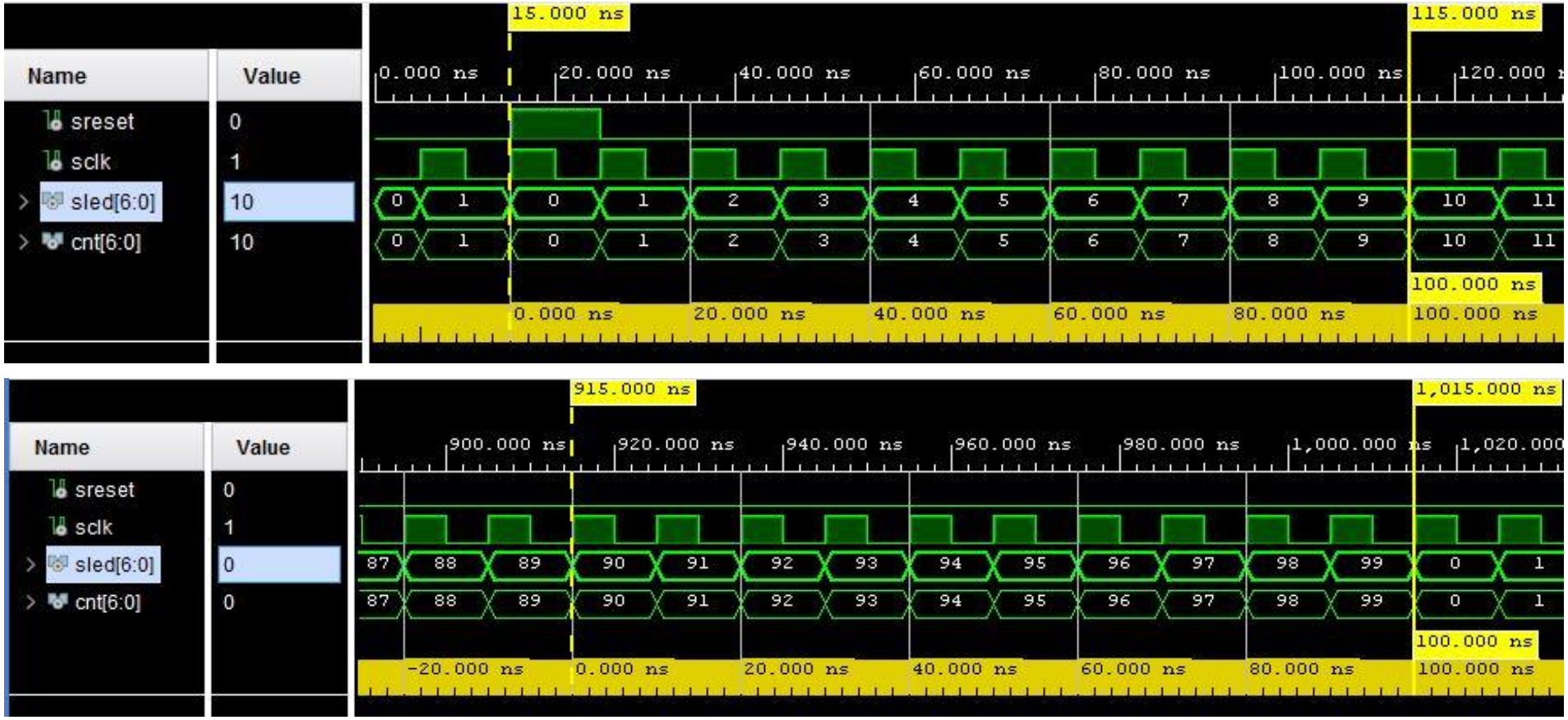
Contador síncrono ascendente

Exemplo: contador ascendente síncrono módulo 100: testbench

```
23  component cnt_asc_mod100 is
24      Port ( reset : in STD_LOGIC;
25             clk : in STD_LOGIC;
26             led : out STD_LOGIC_VECTOR (6 downto 0));
27  end component;
28
29  signal sreset, sclk : std_logic := '0';
30  signal sled : std_logic_vector(6 downto 0) := (others=>'0');
31
32  begin
33
34      sclk <= not sclk after 5 ns;
35      sreset <= '0', '1' after 15 ns, '0' after 25 ns;
36
37      uut: cnt_asc_mod100 port map(
38          reset   => sreset,
39          clk     => sclk,
40          led     => sled);
```

Contador síncrono ascendente

Exemplo: contador ascendente síncrono módulo 100: simulação

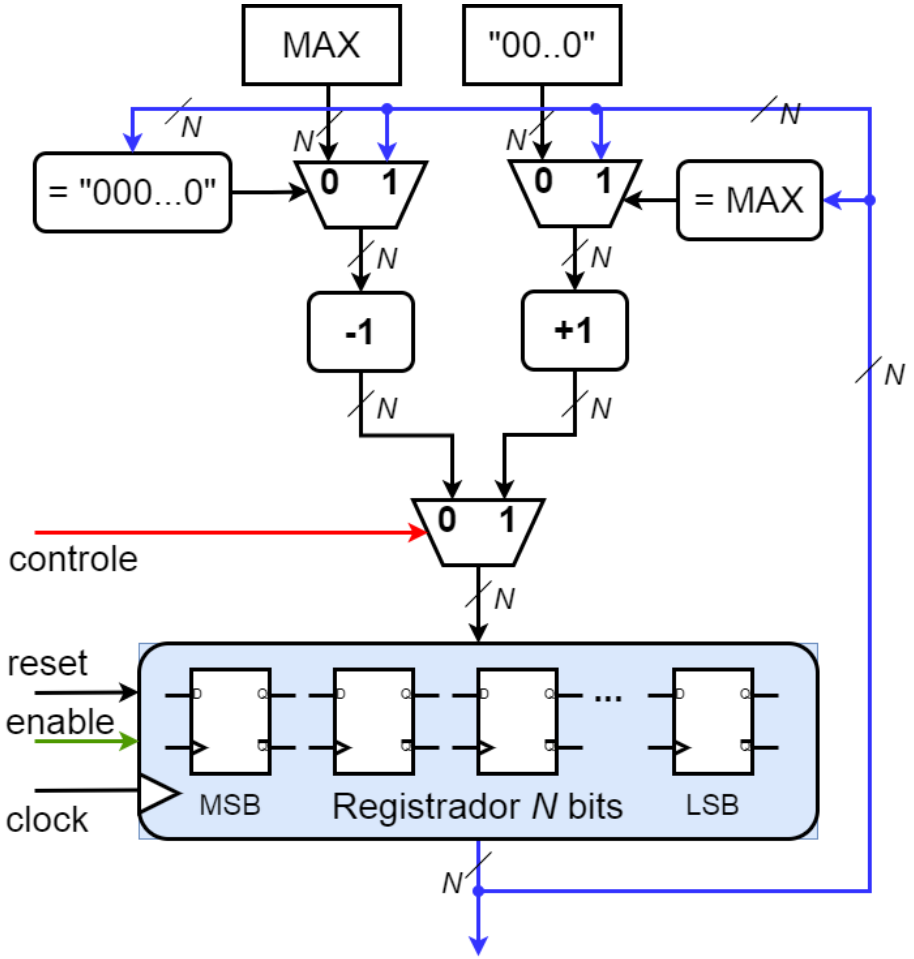


Contador síncrono ascendente

Exercício 3 em sala de aula: implemente em Verilog e em VHDL um contador descendente síncrono módulo 128. Realize um testbench para um clock de 100MHz e verifique o funcionamento em simulação comportamental.

Contador síncrono ascendente/descendente arquitetura

Contador síncrono ascendente/descendente com entradas *controle* e *enable*.



Contador síncrono universal

Exemplo: implemente em HDL um contador síncrono ascendente/descendente módulo 16. Se a entrada Controle = '0' o contador incrementa, caso contrário o contador decrementa. A entrada Enable habilita o contador. Utilize um sinal de clock de 100 MHz. Use reset assíncrono para apagar o registrador. Realize uma simulação comportamental e verifique o comportamento do circuito.

Contador síncrono universal

Exemplo: contador módulo 16 com controle e enable.

```
29  signal cnt : std_logic_vector(3 downto 0) := (others=>'0');
30  begin
31      process(clk,reset)
32      begin
33          if reset='1' then
34              if controle='1' then
35                  cnt <= (others=>'0');
36              else
37                  cnt <= "1111";
38              end if;
39          elsif rising_edge(clk) then
40              if controle='1' then
41                  if cnt = "1111" then -- 15
42                      cnt <= (others=>'0');
43                  else
44                      cnt <= std_logic_vector(unsigned(cnt)+1);
45                  end if;
46              else
47                  if cnt = "0000" then
48                      cnt <= "1111"; -- 15
49                  else
50                      cnt <= std_logic_vector(unsigned(cnt)-1);
51                  end if;
52              end if;
53          end if;
54      end process;
55      led <= cnt;
```

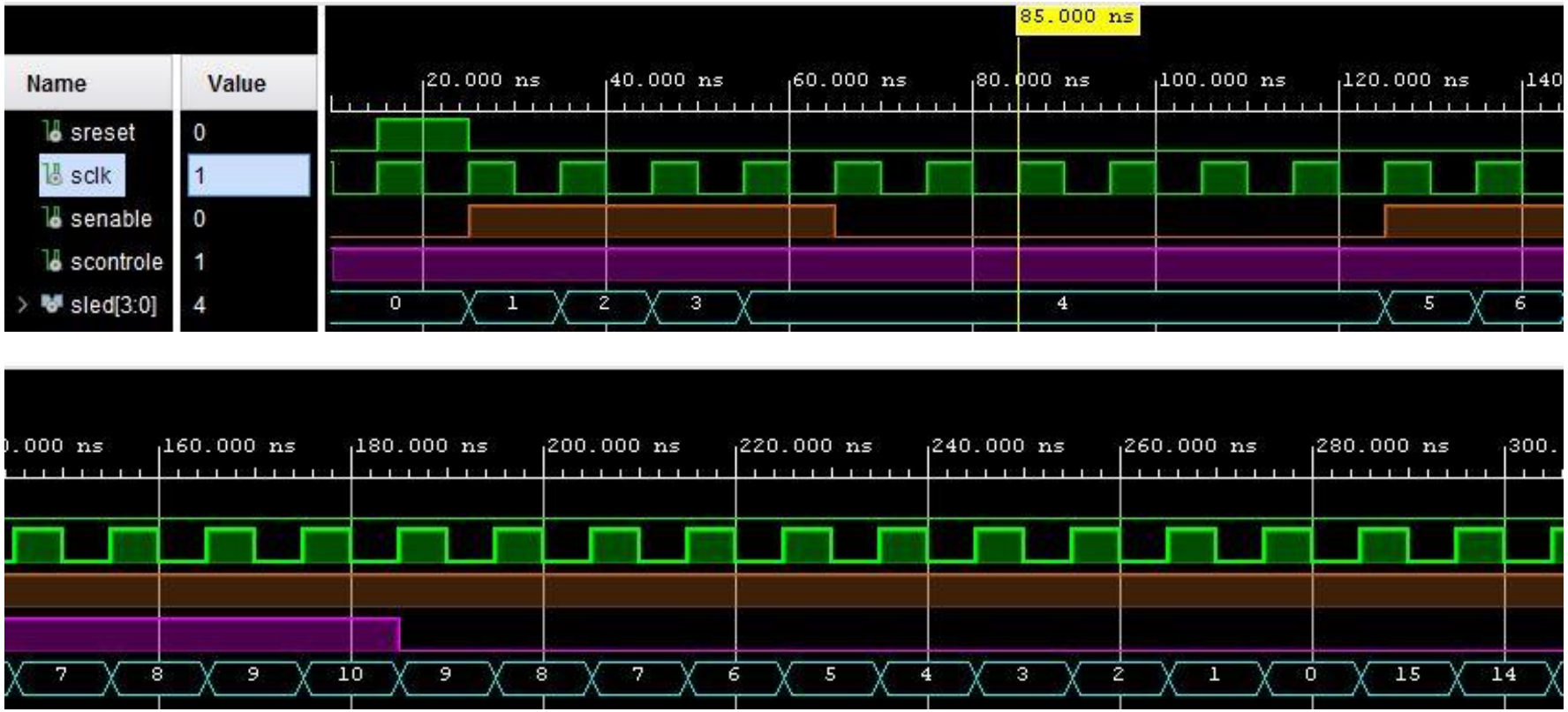
Contador síncrono universal

Exemplo: contador módulo 16 com controle e enable: testbench

```
24  component cnt_asc_dec_mod16 is
25      Port ( reset : in STD_LOGIC;
26             clk : in STD_LOGIC;
27             controle : in STD_LOGIC;
28             enable : in STD_LOGIC;
29             led : out STD_LOGIC_VECTOR (3 downto 0));
30  end component;
31
32  signal sreset, sclk, senable, scontrole: std_logic := '0';
33  signal sled : std_logic_vector(3 downto 0) := (others=>'0');
34
35  begin
36
37      sclk <= not sclk after 5 ns;
38      sreset <= '0', '1' after 15 ns, '0' after 25 ns;
39      senable <= '0', '1' after 25 ns, '0' after 65 ns,
40                      '1' after 125 ns;
41      scontrole <= '1', '0' after 185 ns;
42
43      uut: cnt_asc_dec_mod16 port map(
44          reset    => sreset,
45          clk      => sclk,
46          enable   => senable,
47          controle => scontrole,
48          led      => sled);
```

Contador síncrono universal

Exemplo: contador módulo 16 com controle e enable: simulação



Contador síncrono universal

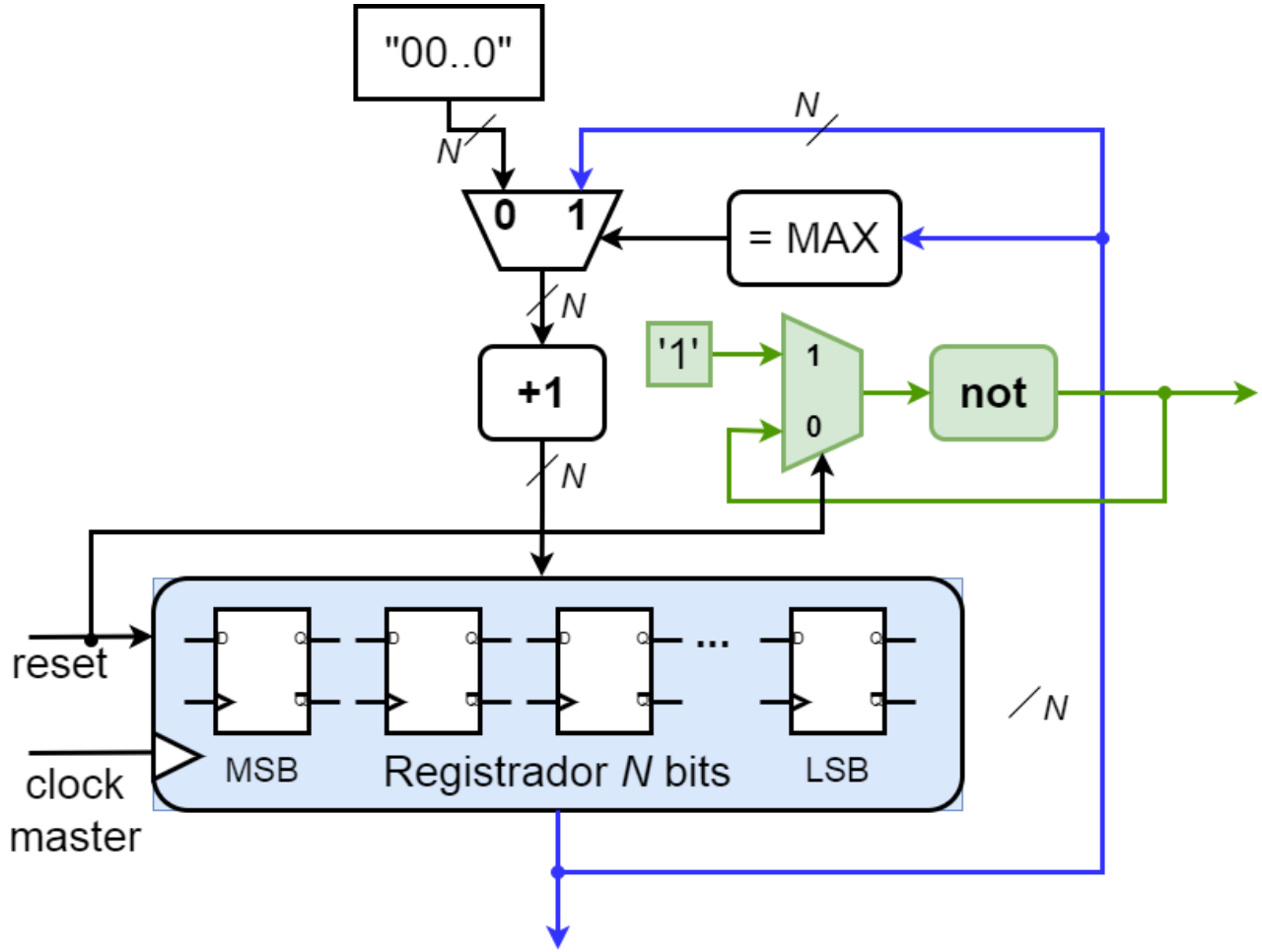
Exercício 4 em sala de aula: Implemente em Verilog o contador síncrono ascendente/descendente módulo 16 com controle e enable, realize um testbench e verifique o comportamento em simulação comportamental. Use uma frequência de clock de 100 MHz.

Página em branco.

Contadores síncronos - divisor de clock

- Contadores síncronos, comparadores e portas inversoras podem ser usados para implementar divisores de clock em HDL.
- Conhecendo a frequência do clock, é possível determinar durante quantos ciclos de clock o contador deve incrementar até alcançar um valor de preset (contador ascendente) ou até alcançar o zero (contador descendente).
- Cada vez que o contador alcançar o preset ou o zero, deve-se inverter um sinal auxiliar que deve ser direcionado como saída do circuito.

Contadores síncronos - arquitetura do divisor de clock



Contadores síncronos - divisor de clock

- Para determinar o valor máximo do contador que vai dividir a frequência de relógio, é necessário conhecer a frequência do clock master.
- Assumindo que desejamos obter um clock com frequência *fd* Hz a partir de um clock master com frequência de 100MHz (T=10ns), é necessário inverter o sinal do clock dividido a cada 1/2fd segundos e sabemos que cada incremento do contador requer 10 ns. Dessa forma, pode-se fazer o seguinte cálculo:

  1 clock = 10 ns

  y=? (1/fd)/2

  y = ((1/fd)/2) / 10E-9

onde, y representa o número de incrementos que o contador precisa realizar para implementar o divisor de clock.

Contador síncrono - divisor de clock

Exemplo: use um contador síncrono para dividir a frequência de um sinal de clock de 100 MHz para 2 Hz. Use as entradas de Reset e Preset de forma síncrona. Use o sinal de clock de 2 Hz no projeto do contador modulo 16 do exemplo anterior. Implemente o circuito na placa de desenvolvimento.

Contadores síncronos - divisor de clock

- Exemplo: use um contador síncrono para dividir a frequência de um sinal de clock de 100 MHz para 2 Hz. Use as entradas de Reset e Preset de forma síncrona. Use o sinal de clock de 2 Hz no projeto do contador modulo 16 do exemplo anterior. Implemente o circuito na placa de desenvolvimento.

Calculando o número de incrementos do contador:

1 clock = 10 ns

y=? (1/2)/2

y = (0.5/2) / 10E-9 = 25.000.000

São necessários 25 milhões de incrementos do contador, requerendo, portanto, 25 bits para implementar o registrador.

Contadores síncronos - divisor de clock

- Exemplo: divisor de clock de 2 Hz a partir de um clock de 100 MHz (Verilog)

```
1   module clkdiv_2Hz (
2       input  reset,
3       input  clk,
4       output  clkdiv
5   );
6
7       // Constant 25 million
8       localparam MAX_CNT = 25_000_000;
9
10      // Counter to keep track of clock cycles
11      reg [24:0] cnt = 0;
12
13      // Auxiliary clock signal
14      reg auxclk = 0;
```

Contadores síncronos - divisor de clock

- Exemplo: divisor de clock de 2 Hz a partir de um clock de 100 MHz (Verilog)

```
16      // Process triggered by rising edge of clock
17      always @(posedge clk or posedge reset) begin
18          if (reset) begin
19              cnt <= 0;
20              auxclk <= 0;
21          end else begin
22              if (cnt == MAX_CNT - 1) begin
23                  cnt <= 0;
24                  auxclk <= ~auxclk; // Toggle aux clk
25              end else begin
26                  cnt <= cnt + 1;
27              end
28          end
29      end
30
31      // Output the auxiliary clock signal as clkdiv
32      assign clkdiv = auxclk;
33
34  endmodule
```

Contadores síncronos - divisor de clock

- Exemplo: divisor de clock de 2 Hz a partir de um clock de 100 MHz (VHDL)

```
14  library IEEE;
15  use IEEE.STD_LOGIC_1164.ALL;
16  use IEEE.NUMERIC_STD.ALL;
17
18  entity clkdiv_2Hz is
19      Port ( reset : in STD_LOGIC;
20             clk : in STD_LOGIC;
21             clkdiv : out STD_LOGIC);
22  end clkdiv_2Hz;
23
24  architecture Behavioral of clkdiv_2Hz is
25
26  constant MAX_cnt : std_logic_vector(24 downto 0) := "1011111010111100001000000";
27  signal cnt : std_logic_vector(24 downto 0) := (others=>'0');
28  signal auxclk : std_logic := '0';
29
30  begin
```

Contadores síncronos - divisor de clock

- Exemplo: divisor de clock de 2 Hz a partir de um clock de 100 MHz (VHDL)

```
32      process(clk,reset)
33      begin
34          if reset='1' then
35              cnt <= (others=>'0');
36              auxclk <= '0';
37          elsif rising_edge(clk) then
38              if cnt = MAX_cnt then -- 25.000.000
39                  cnt <= (others=>'0');
40                  auxclk <= not auxclk;
41              else
42                  cnt <= std_logic_vector(unsigned(cnt)+1);
43              end if;
44          end if;
45      end process;
46
47      clkdiv <= auxclk;
48
49  end Behavioral;
```

Contadores síncronos - divisor de clock

- Exemplo: divisor de clock de 2 Hz. Testbench

```
23  component clkdiv_2Hz is
24      Port ( reset : in STD_LOGIC;
25             clk : in STD_LOGIC;
26             clkdiv : out STD_LOGIC);
27  end component;
28
29  signal sreset, sclk : std_logic := '0';
30  signal sclkdiv : std_logic := '0';
31
32  begin
33
34      sclk <= not sclk after 5 ns;
35      sreset <= '0', '1' after 15 ns, '0' after 25 ns;
36
37      uut: clkdiv_2Hz port map(
38          reset   => sreset,
39          clk     => sclk,
40          clkdiv  => sclkdiv);
```

Contadores síncronos - divisor de clock

- Exemplo: divisor de clock de 2 Hz. Simulação comportamental:

Contadores síncronos - divisor de clock

Exercício 5 em sala de aula: Implemente em Verilog e VHDL um divisor de clock de 0.5 Hz do exemplo anterior. Realize um testbench em Verilog e verifique o funcionamento em simulação comportamental. Inclua os constraints de IO (use um led para a saída) e implemente fisicamente na placa de desenvolvimento.

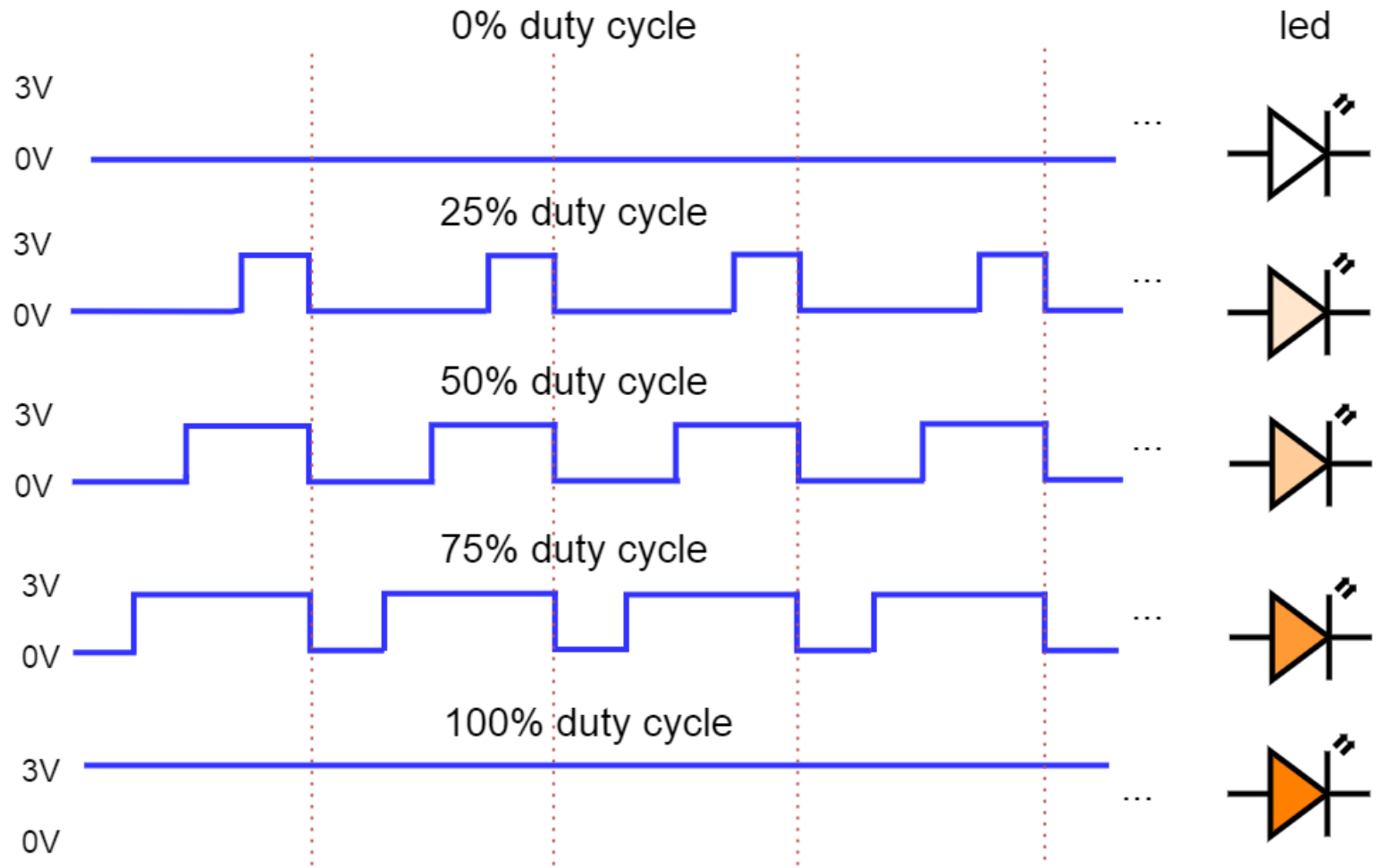
Página em branco.

Aula 4
Implementação em HDL
de um modulador por largura de pulso

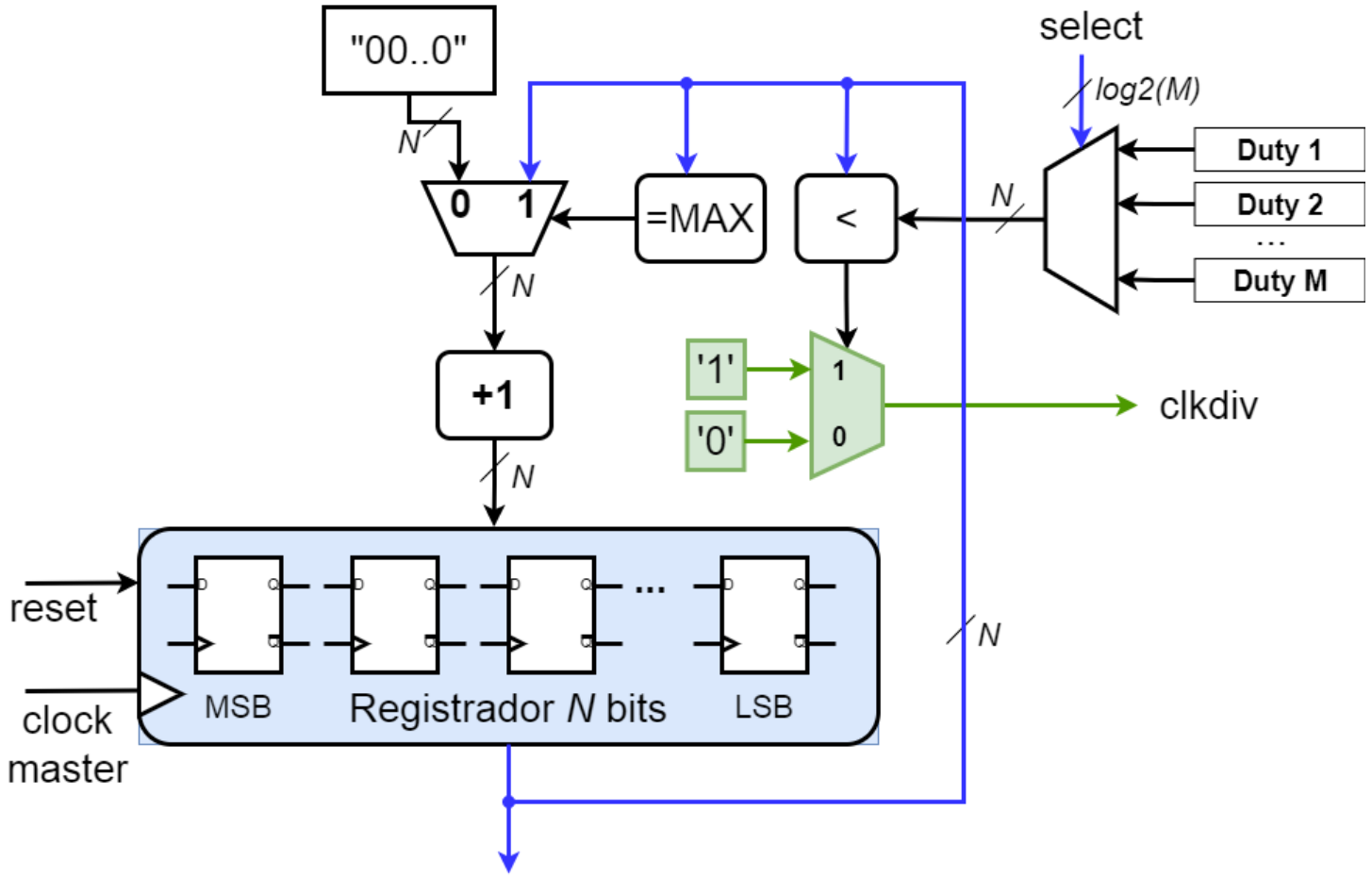
Contador síncrono - modulador PWM

- Modulação por largura de pulso (PWM – *Pulse Width Modulation*) é uma técnica de modulação de informação.
- Utiliza um sinal de período constante, cujo ciclo ativo (duty cycle) é variável no intuito de ajustar o valor médio da onda periódica. Para isto é necessário variar o tempo que o sinal fica em nível lógico alto.
- Variando-se a largura do ciclo ativo entre 0 e 100%, obtém-se uma saída de tensão média de 0 ao valor máximo de alimentação.
- O PWM pode ser aplicado em:
  - Controle de velocidade de motores DC,
  - Controle de iluminação de LEDs,
  - Geração de sinais de áudio,
  - Codificação de informação em sinais analógicos.

Contador síncrono - modulador PWM



Contador síncrono - modulador PWM - arquitetura



Contador síncrono - modulador PWM

Exemplo1: determine o valor de preset do contador de um modulador por largura de pulso com período de 10 us a partir de uma frequência de clock de 125 MHz.

Contador síncrono - modulador PWM

Exemplo1: determine o valor de preset do contador de um modulador por largura de pulso com período de 10 us a partir de uma frequência de clock de 125 MHz.

Calculando o valor máximo do contador:

fclk = 125 MHz = 8 ns = 8 E-9 s

1 clk = 8 ns = 8E-9 s

y=? 10 us = 10E-6 s

y = 1250

Contador síncrono - modulador PWM

Exemplo2: implemente em Verilog um PWM com período de 10 us a partir de uma frequência de clock de 100 MHz. Utilize uma entrada de 4 bits para selecionar 10 valores de duty cycle entre 0% e 100% com passos de 10%.

```
module pwm (
    input clk,
    input reset,
    input [3:0] duty,
    output reg pwm_out
);

// Constant for periodo (999 in binary)
localparam [9:0] PERIODO = 10'b1111100111;

reg [9:0] preset;
reg [9:0] cnt;
```

```
// Mux for preset based on duty cycle
always @(*) begin
    case (duty)
        4'b1010: preset = 10'b1111111111; // 100% = 1023
        4'b1001: preset = 10'b1110011011; // 90% = 923
        4'b1000: preset = 10'b1100110011; // 80% = 819
        4'b0111: preset = 10'b1011001101; // 70% = 717
        4'b0110: preset = 10'b1001100110; // 60% = 614
        4'b0101: preset = 10'b1000000000; // 50% = 512
        4'b0100: preset = 10'b0110011010; // 40% = 410
        4'b0011: preset = 10'b0100110011; // 30% = 307
        4'b0010: preset = 10'b0011001101; // 20% = 205
        4'b0001: preset = 10'b0001100110; // 10% = 102
        default: preset = 10'b0000000000; // 0% (off)
    endcase
end
```

Contador síncrono - modulador PWM

Exemplo2: implemente em Verilog um PWM com período de 10 us a partir de uma frequência de clock de 100 MHz. Utilize uma entrada de 4 bits para selecionar 10 valores de duty cycle entre 0% e 100% com passos de 10%.

```
// Counter process
always @(posedge clk or posedge reset) begin
    if (reset) begin
        cnt <= 10'b0000000000;
    end else begin
        if (cnt == PERIODO) begin
            cnt <= 10'b0000000000;
        end else begin
            cnt <= cnt + 1;
        end
    end
end
```

```
// PWM output process
always @(posedge clk) begin
    if (cnt < preset)
        pwm_out <= 1;
    else
        pwm_out <= 0;
end

endmodule
```

Contador síncrono - modulador PWM

Exemplo3: implemente em VHDL um modulador por largura de pulso com período de 10 us a partir de uma frequência de clock de 100 MHz. Utilize uma entrada de 4 bits para selecionar 10 valores de duty cycle entre 0% e 100% com passos de 10%.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity pwm is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           duty : in STD_LOGIC_VECTOR (3 downto 0);
           pwm_out : out STD_LOGIC);
end pwm;

architecture Behavioral of pwm is
```

Contador síncrono - modulador PWM

Exemplo3: implemente em VHDL um modulador por largura de pulso com período de 10 us a partir de uma frequência de clock de 100 MHz. Utilize uma entrada de 4 bits para selecionar 10 valores de duty cycle entre 0% e 100% com passos de 10%.

```
28  constant periodo : std_logic_vector(9 downto 0) := "1111100111"; -- 0 to 999
29  signal preset: std_logic_vector(9 downto 0) := (others=>'0');
30  signal cnt: std_logic_vector(9 downto 0) := (others=>'0');
31  signal pwm_aux: std_logic := '0';
32
33  begin
34
35      -- mux para determinar preset em funcao do duty cycle (ciclo ativo)
36      with duty select
37          preset <= "1111111111" when "1010", -- 100%=1023
38                    "1110011011" when "1001", -- 90%=923
39                    "1100110011" when "1000", -- 80%=819
40                    "1011001101" when "0111", -- 70%=717
41                    "1001100110" when "0110", -- 60%=614
42                    "1000000000" when "0101", -- 50%=512
43                    "0110011010" when "0100", -- 40%=410
44                    "0100110011" when "0011", -- 30%=307
45                    "0011001101" when "0010", -- 20%=205
46                    "0001100110" when "0001", -- 10%=102
47                    "0000000000" when "0000", -- 0%=desliga
48                    "0000000000" when others; -- 0%=desliga
```

Multiplexador do preset do contador em funcao do duty cycle

Contador síncrono - modulador PWM

Exemplo3: implemente em VHDL um modulador por largura de pulso com período de 10 us a partir de uma frequência de clock de 100 MHz. Utilize uma entrada de 4 bits para selecionar 10 valores de duty cycle entre 0% e 100% com passos de 10%.

```
51      process(clk,reset)
52      begin
53          if reset='1' then
54              cnt <= (others=>'0');
55          elsif rising_edge(clk) then
56              if cnt = periodo then
57                  cnt <= (others=>'0');
58              else
59                  cnt <= std_logic_vector(unsigned(cnt) + 1);
60              end if;
61          end if;
62      end process;
63
64      process(cnt)
65      begin
66          if cnt < preset then
67              pwm_aux <= '1';
68          else
69              pwm_aux <= '0';
70          end if;
71      end process;
```

Contador síncrono de 0 a 999 para implementar período de 10 us

Comparador e multiplexador do sinal de PWM

Contador síncrono - modulador PWM

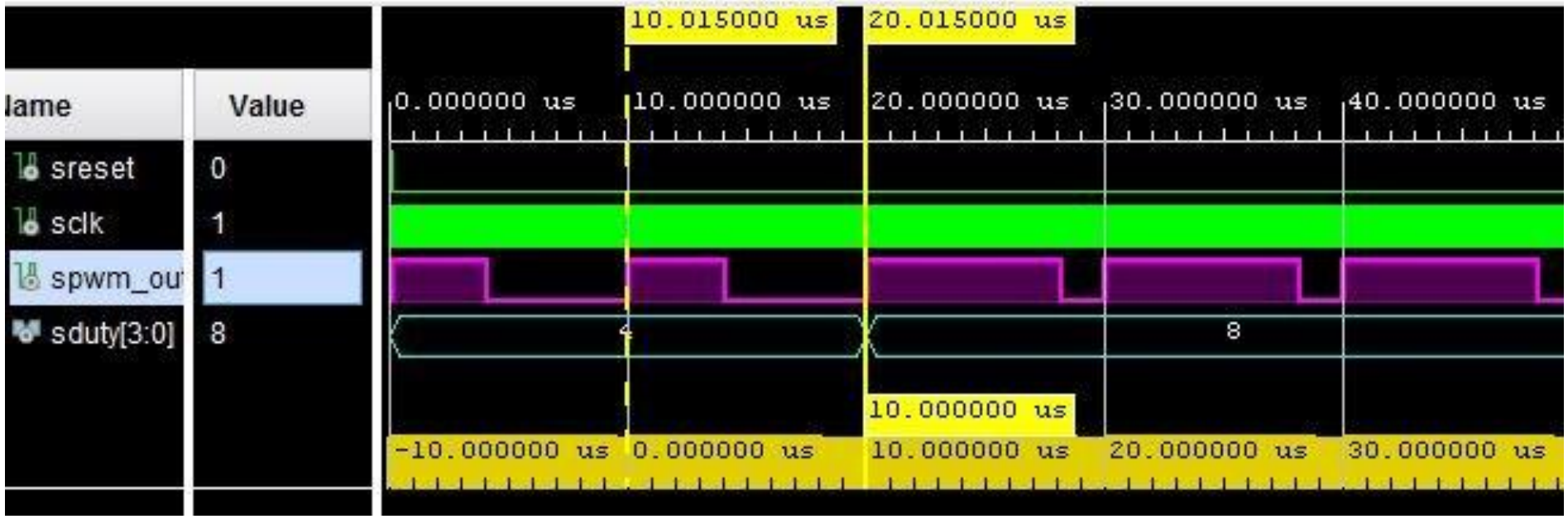
Exemplo3: implemente em VHDL um modulador por largura de pulso com período de 10 us a partir de uma frequência de clock de 100 MHz. Utilize uma entrada de 4 bits para selecionar 10 valores de duty cycle entre 0% e 100% com passos de 10%.

```
component pwm is
    Port ( reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           duty : in STD_LOGIC_VECTOR (3 downto 0);
           pwm_out : out STD_LOGIC);
end component;

signal sreset, sclk : std_logic := '0';
signal spwm_out : std_logic := '0';
signal sduty : std_logic_vector(3 downto 0) := "0000";

begin

    sclk <= not sclk after 5 ns;
    sreset <= '0', '1' after 15 ns, '0' after 25 ns;
    sduty <= "0100", -- 40% de duty cycle
             "1000" after 20 us; -- 80% de duty cycle

    uut: pwm port map(
        reset   => sreset,
        clk     => sclk,
        duty    => sduty,
        pwm_out => spwm_out);
```

Testbench com fclk=100MHz e teste de 40% e 80% de duty cycle.

Contador síncrono - modulador PWM

Exemplo: implemente um modulador por largura de pulso com período de 10 us a partir de uma frequência de clock de 100 MHz. Utilize uma entrada de 4 bits para selecionar 10 valores de duty cycle entre 0% e 100% com passos de 10%.



Contador síncrono - modulador PWM

Exercício 6 em sala de aula: use um contador síncrono para implementar um modulador por largura de pulso com período de 50 Hz. Utilize uma entrada de 2 bits para selecionar 3 valores de duty cycle entre 30% e 100%. Implemente o PWM em VHDL ou Verilog, realize uma simulação comportamental, inclua os constraints de IO (use um led para a saída), implemente e caracterize o circuito e teste na placa de desenvovlimento.

Contador síncrono - modulador PWM

Exercício 7 em sala de aula: Implemente quatro módulos PWM em paralelo, use um periódo de sinal PWM de 50 Hz e um clock master de 100MHz. Cada PWM pode receber um valor diferente de duty cycle entre 25%, 50%, 75% e 100%. Use um entrada de 2 bits para selecionar o duty cycle e outra entrada de dois bits para selecionar o módulo PWM. Implemente o PWM em VHDL ou Verilog, realize uma simulação comportamental, inclua os constraints de IO (use um led para a saída), implemente e caracterize o circuito e teste na placa de desenvolvimento.

Página em branco.