SD132 – Circuitos Digitais II
A-203 Contadores Assíncronos

Execução
Inatel UNIFEI INSTITUTO HARDWARE BR Uema cepedi DIGITAL
Realização
CHIP TECH BRASIL Softex MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

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

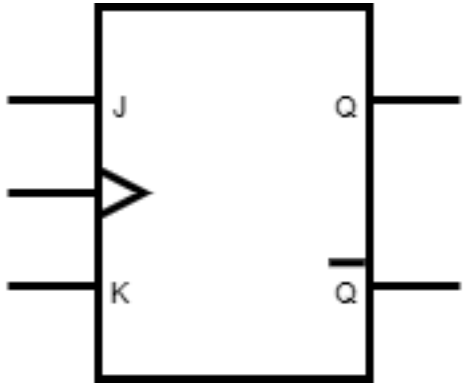
Tópicos

- Revisão de Flip-Flop JK e Flip-Flop D
- Definições
- Contador assíncrono ascendente com FF JK
- Contador assíncrono descendente com FF JK
- Contador assíncrono ascendente com FF D
- Contador assíncrono descendente com FF D
- Desvantagens dos contadores assíncronos
- Implementação em VHDL

Página em branco.

Aula 1
Contadores Assíncronos

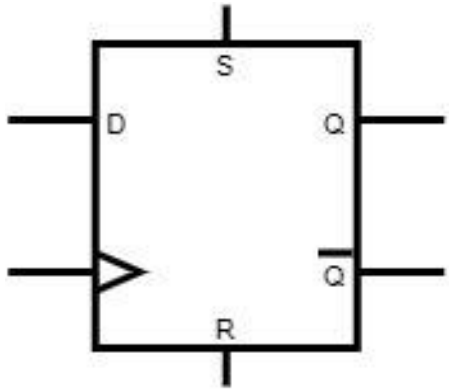
Revisão Flip-flop JK



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)

J: entrada de set.
K: entrada de reset.
Clk: clock
Q (Saída): estado atual do flip-flop.
Q' (Saída complementar): estado oposto ao de Q.

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

Clk: clock
D: entrada de dado.
Set: entrada assíncrona, força valor em '1'.
Reset: entrada assíncrona, força valor em '0'.
Q (Saída): estado atual do flip-flop.
Q' (Saída complementar): estado oposto ao de Q.

Definições

- Contadores são circuitos digitais baseados em registradores, usados para contar eventos.
- Se o registrador usa *N* Flip-flops, temos 2^N estados possíveis (valores possíveis). Por exemplo: um contador com 3 Flip-flops tem $2^3 = 8$ valores possíveis. Um contador com 4 Flip-flops tem $2^4 = 16$ valores possíveis.
- O **módulo** de um contador é o valor x que o contador pode contar. Por exemplo: contador módulo 6 (conta de 0 a 5). Contador módulo 10 conta de 0 a 9 (contador BCD).
- O número de Flip-flops e a forma como estão conectados vai determinar o módulo e a sequência específica que o contador vai realizar.

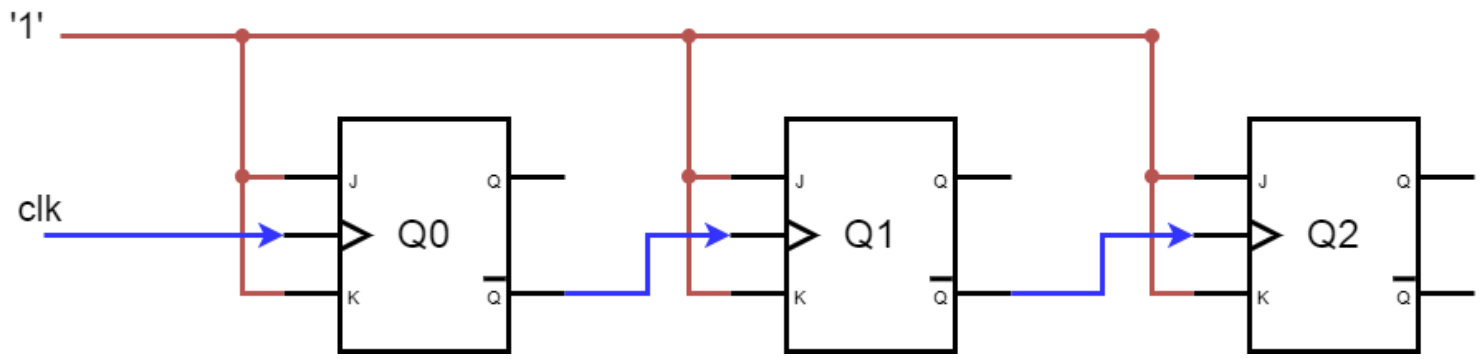
Definições

- Contadores podem ser síncronos ou assíncronos.
- Nos contadores **síncronos** todos os Flip-Flops dependem do mesmo sinal de clock. Portanto, todos os Flip-flops mudam simultaneamente de estado.
- Nos contadores **assíncronos** a saída de um Flip-Flop alimenta a entrada de clock do seguinte Flip-flop. Portanto, o tempo de propagação depende do número de Flip-flops (do módulo) do contador.

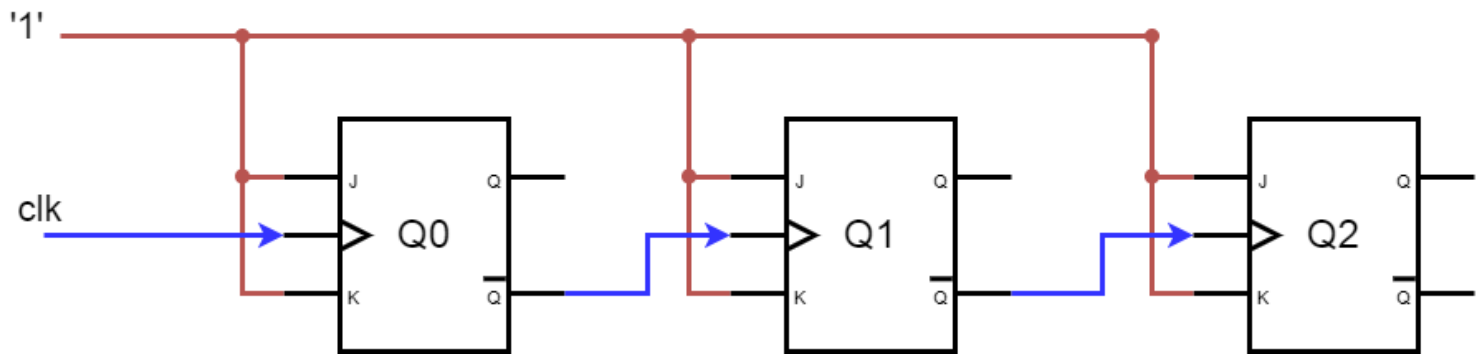
Definições

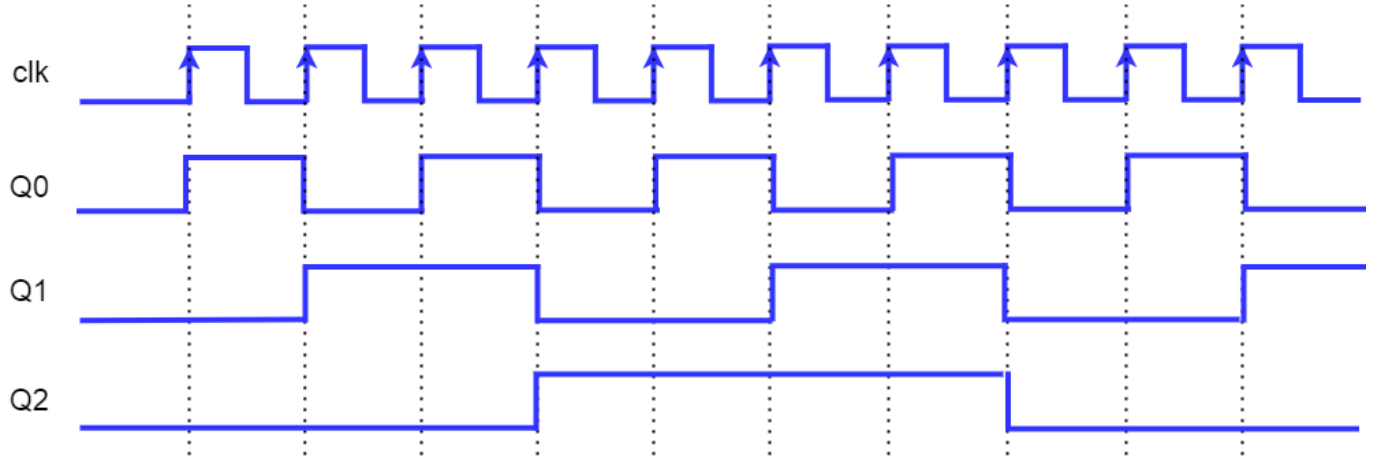
- Contadores ascendentes contam de 0 a um valor máximo.
- Contadores descendentes contam de um valor máximo até 0.
- Contadores bidirecionais contam em ambos os sentidos.
- Além de Flip-flops, os contadores usam portas lógicas AND, OR, XOR e circuitos de decodificação.
- Características importantes: módulo, velocidade, precisão, capacidade de cascateamento, sincronismo, consumo de energia.
- Aplicações: contagem de tempo (relógios digitais), divisão de clock, contagem de eventos, sistemas de controle, etc.

Contador assíncrono ascendente com FF JK



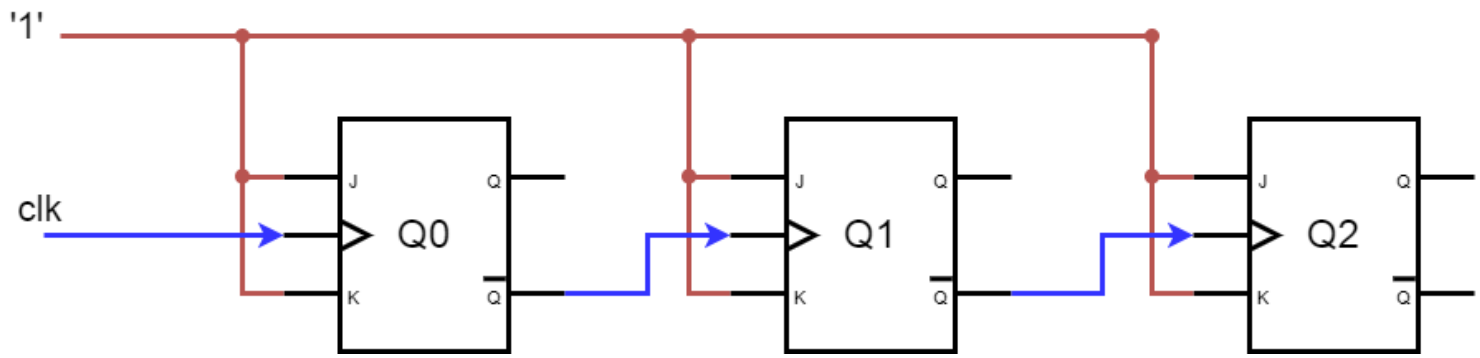
Contador assíncrono ascendente com FF JK

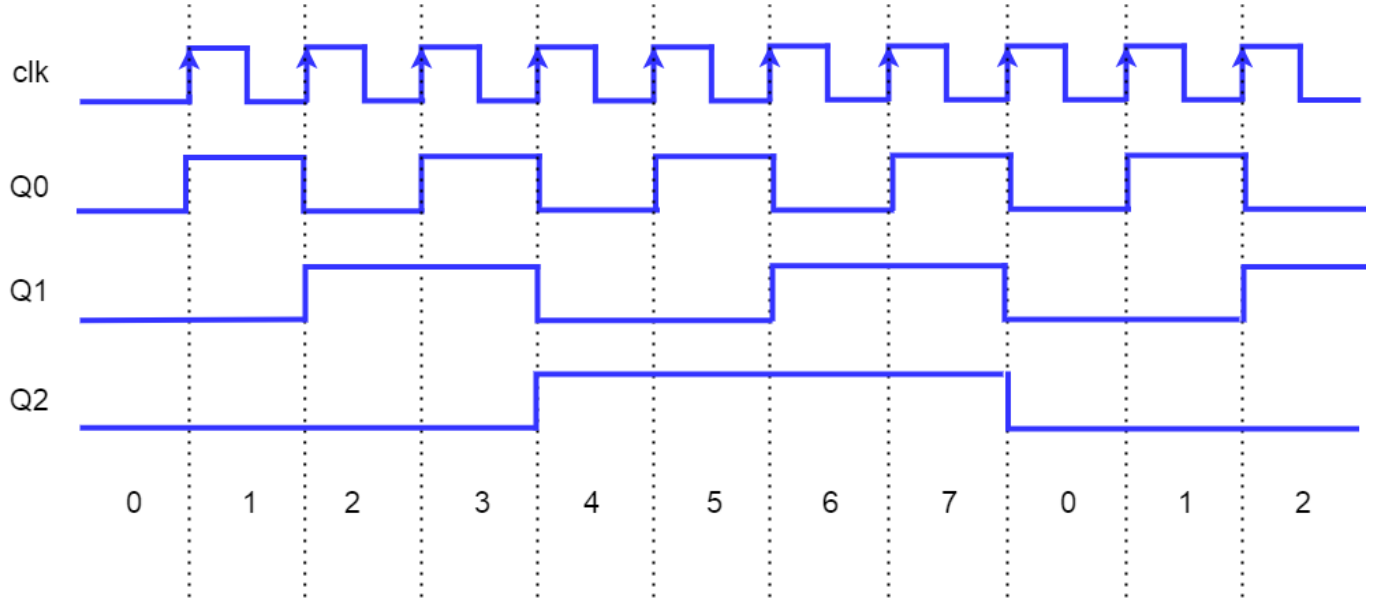




Qual o módulo?

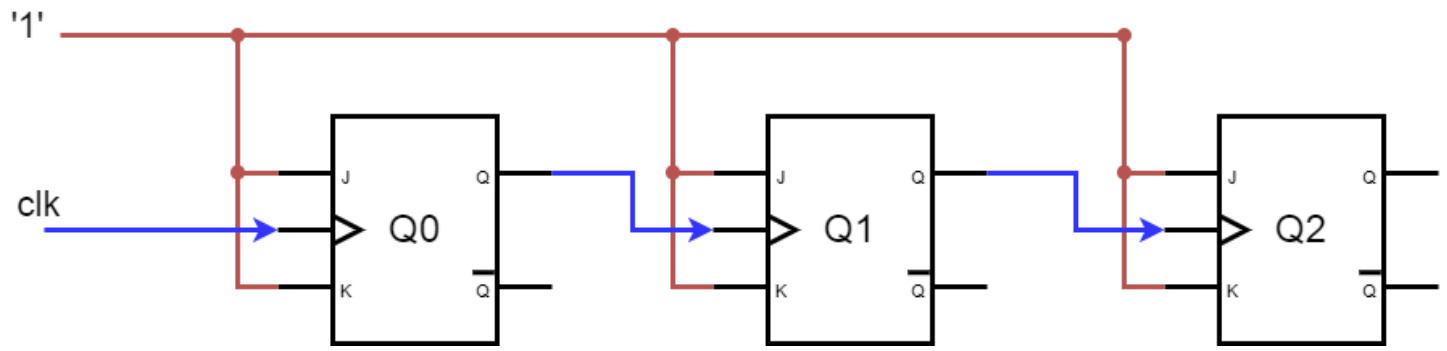
Contador assíncrono ascendente com FF JK



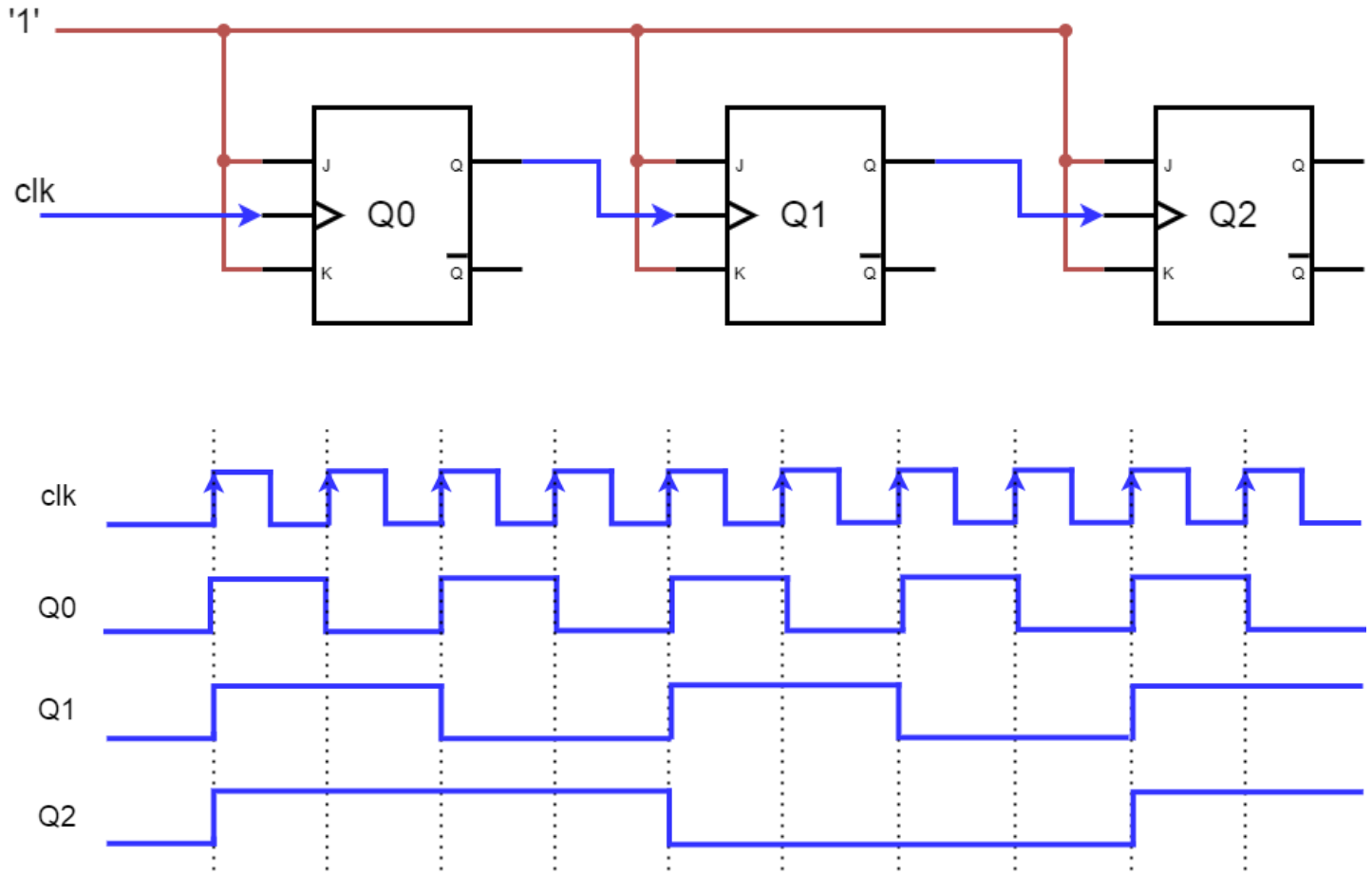


Contador módulo 8

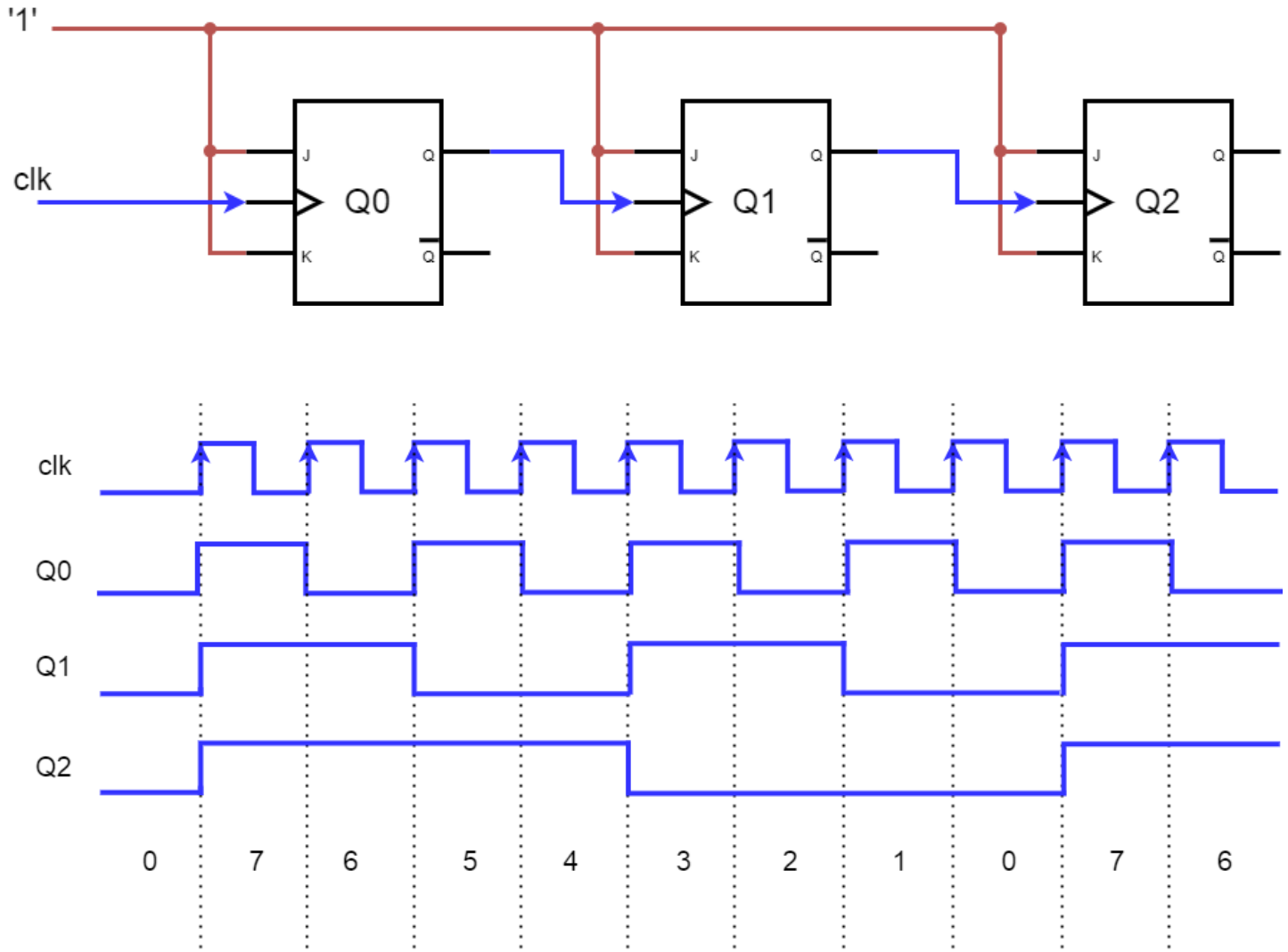
Contador assíncrono descendente com FF JK



Contador assíncrono descendente com FF JK

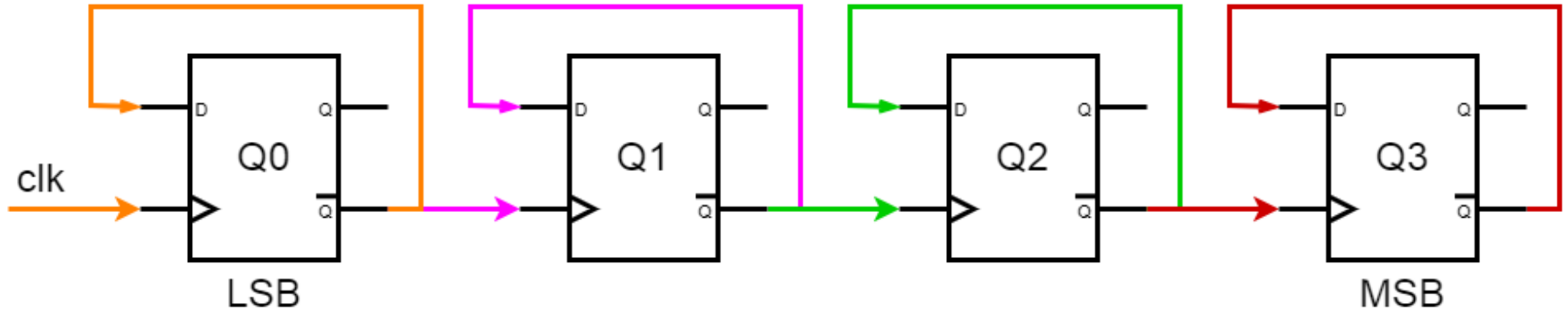


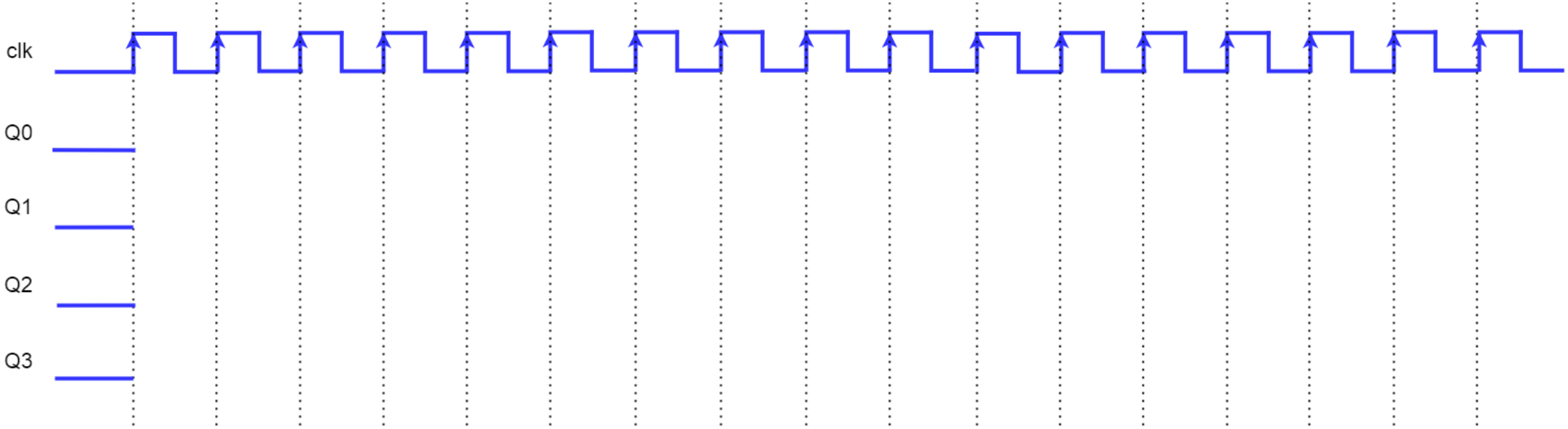
Contador assíncrono descendente com FF JK



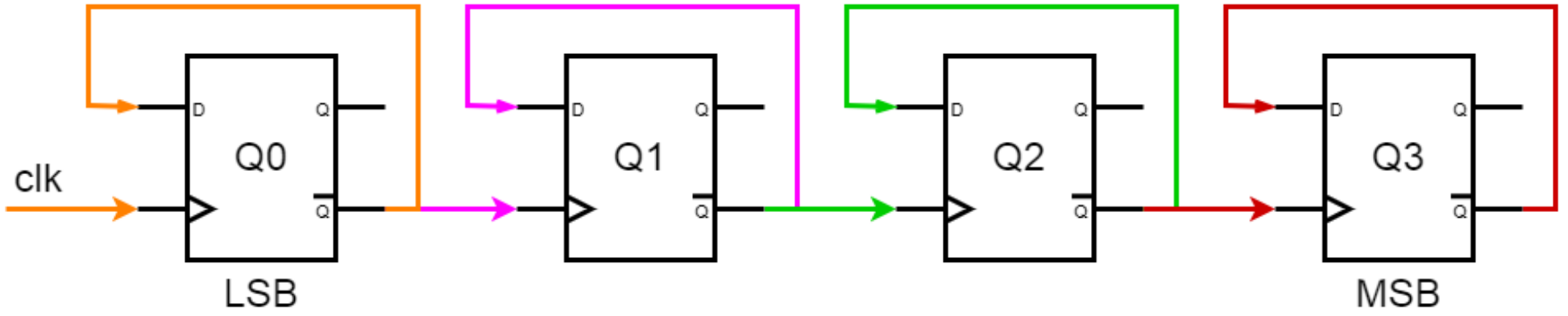
Página em branco.

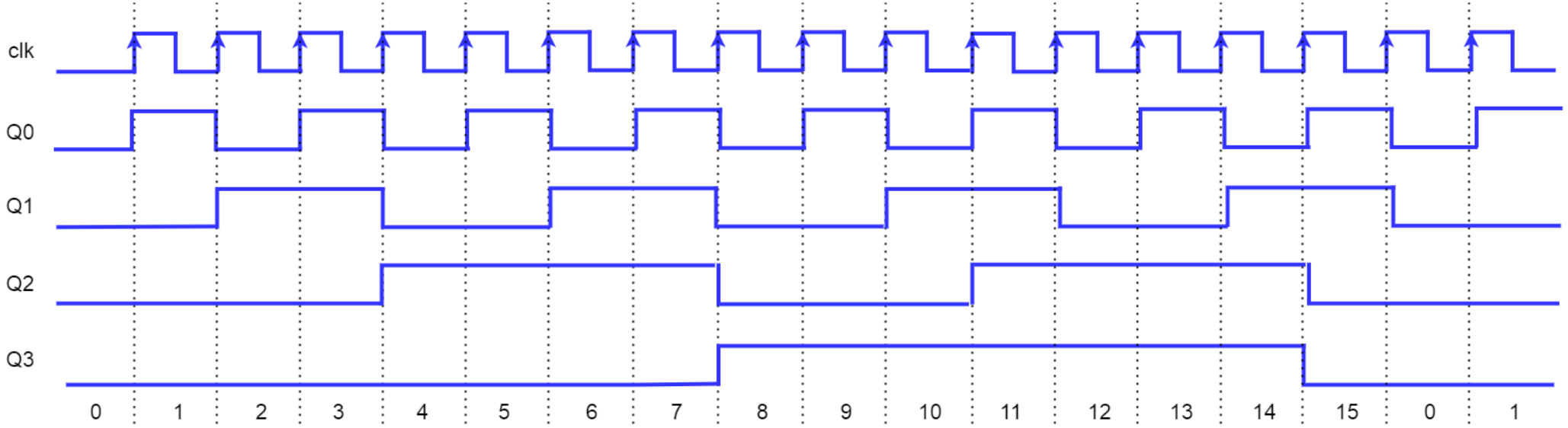
Contador assíncrono ascendente de 4 bits com FFD



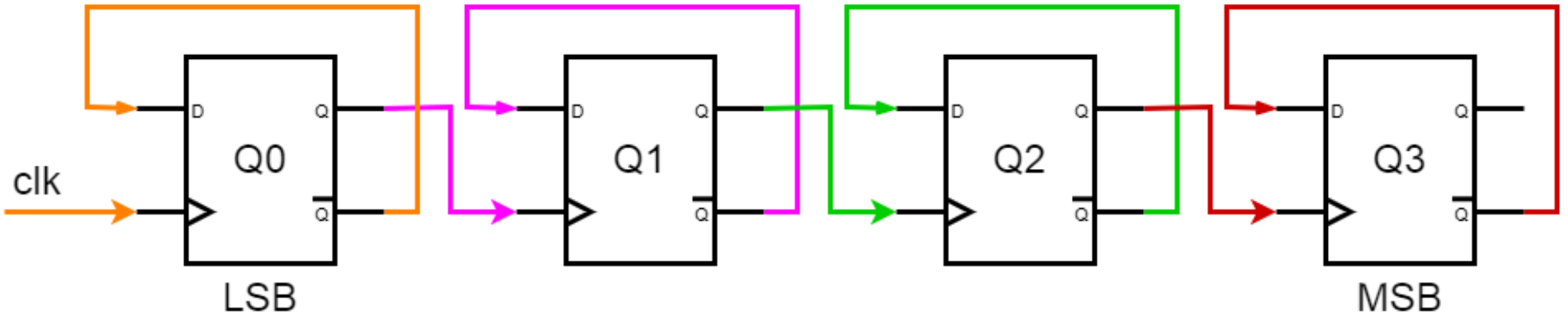


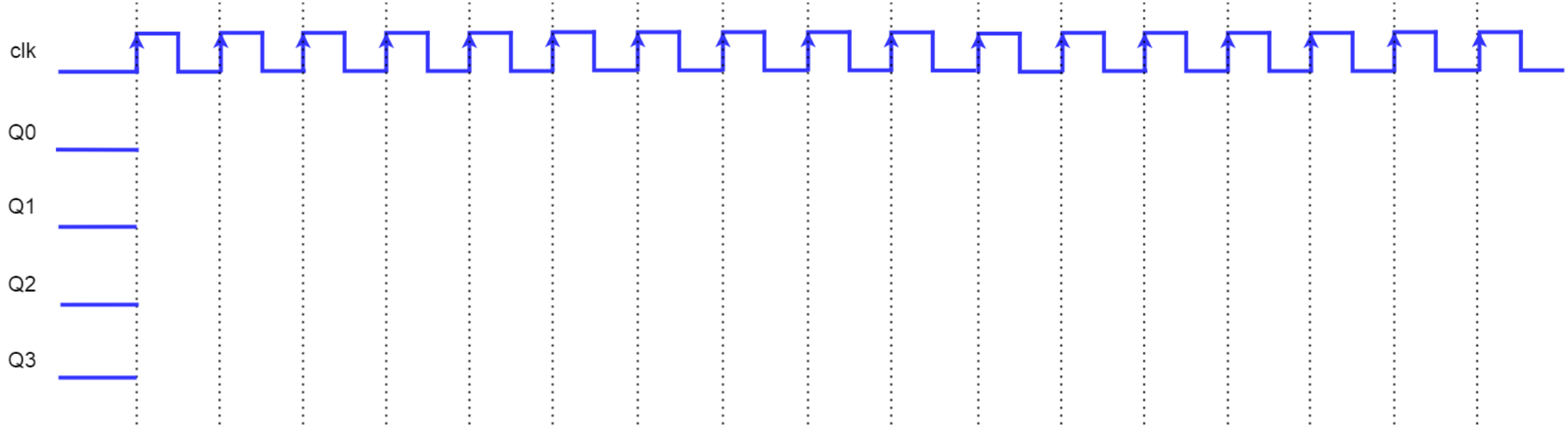
Contador assíncrono ascendente de 4 bits com FFD



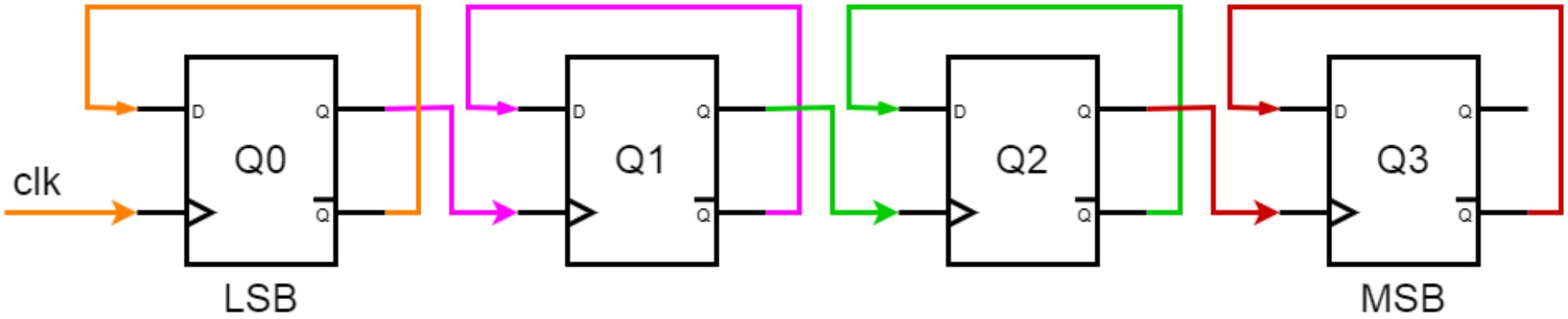


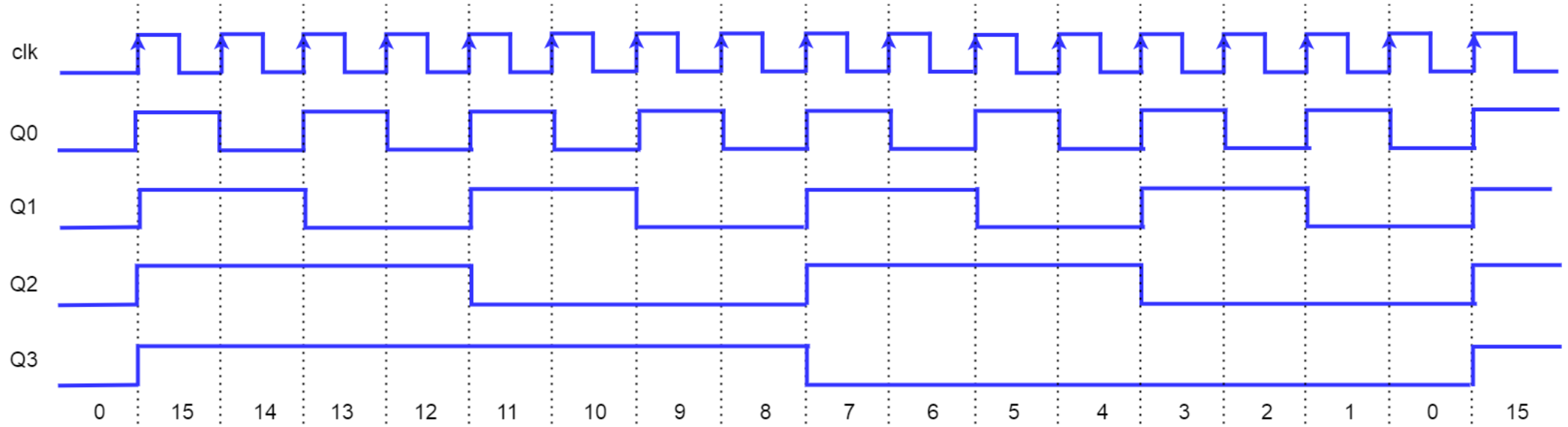
Contador assíncrono descendente de 4 bits com FFD



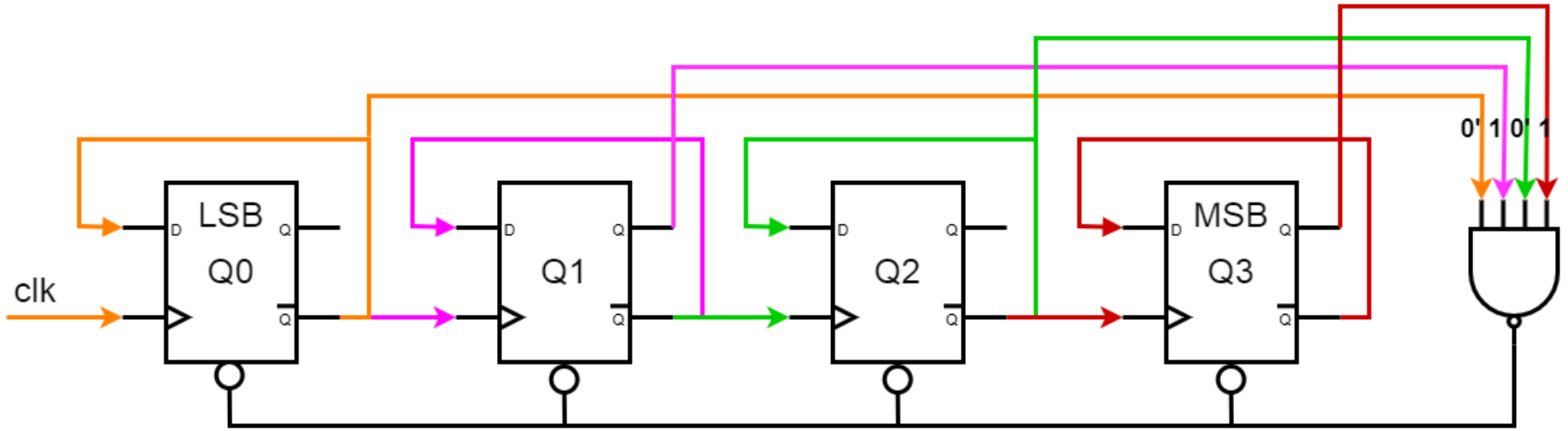


Contador assíncrono descendente de 4 bits com FFD



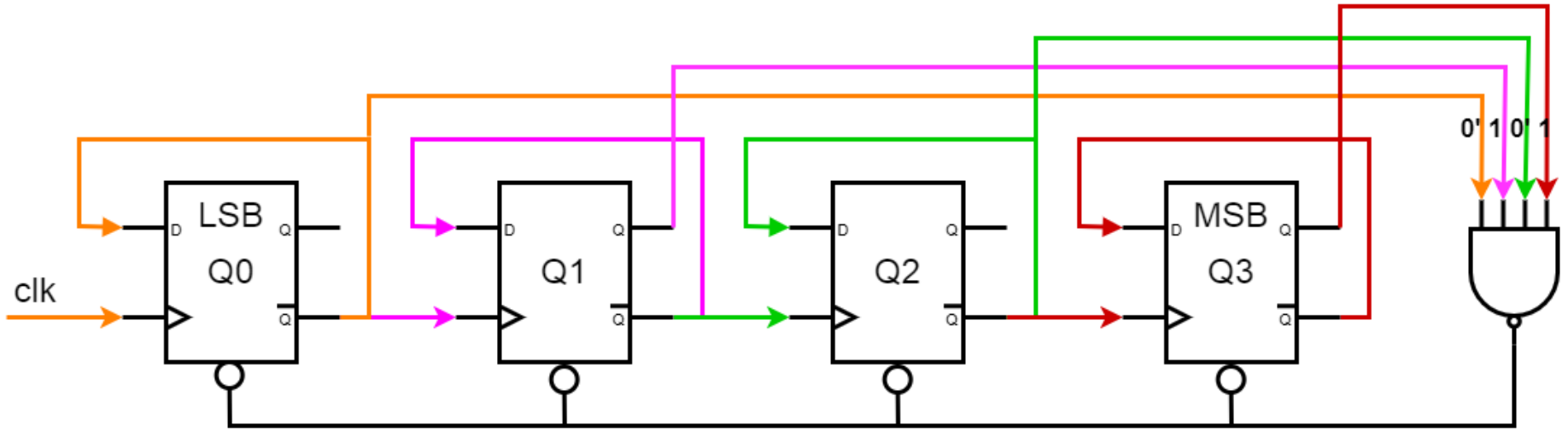


Contador assíncrono BCD com FFD



Q3	Q2	Q1	Q0
0	0	0	0

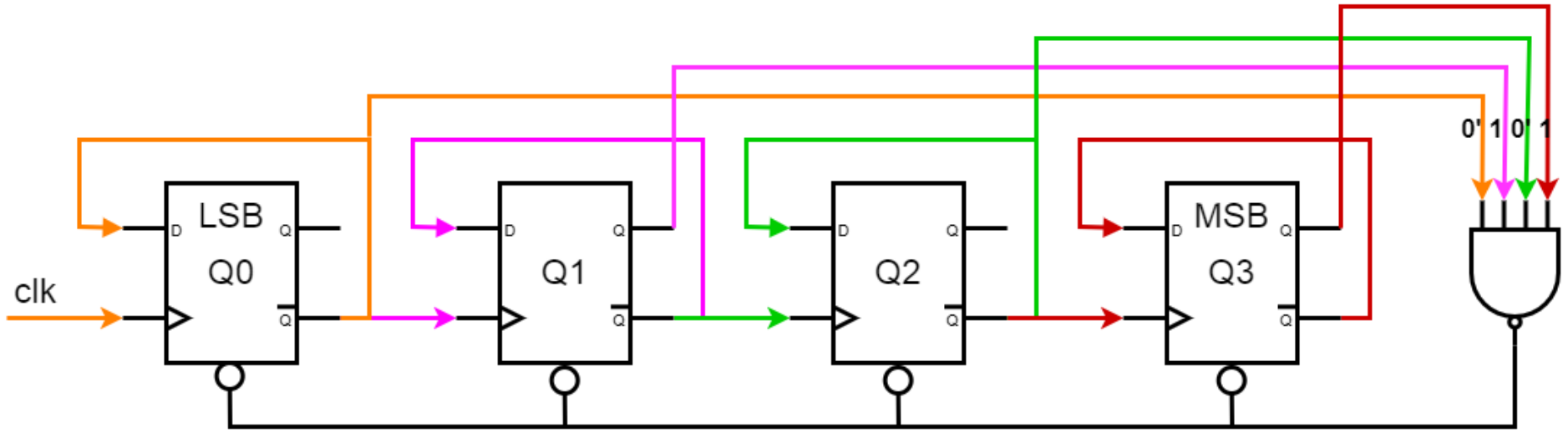
Contador assíncrono BCD com FFD



Q3	Q2	Q1	Q0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0

Estado com duração muito pequena
aproximadamente o atraso da porta NAND (ps)

Contador assíncrono BCD com FFD



- Contador BCD (0 a 9).
- Os FFs armazenam por um instante muito pequeno o valor 1010 (10_{dec}), mas a porta NAND produz o sinal de reset (assíncrono), apagando o valor de todos os Flip-flops, iniciando um novo ciclo.

Página em branco.

Vantagens dos contadores assíncronos

- Implementação simples.
- Facilmente escalonáveis para contadores maiores.
- Menor número de componentes se comparado com contadores síncrono.
- Menor consumo de energia se comparado com contadores síncronos.

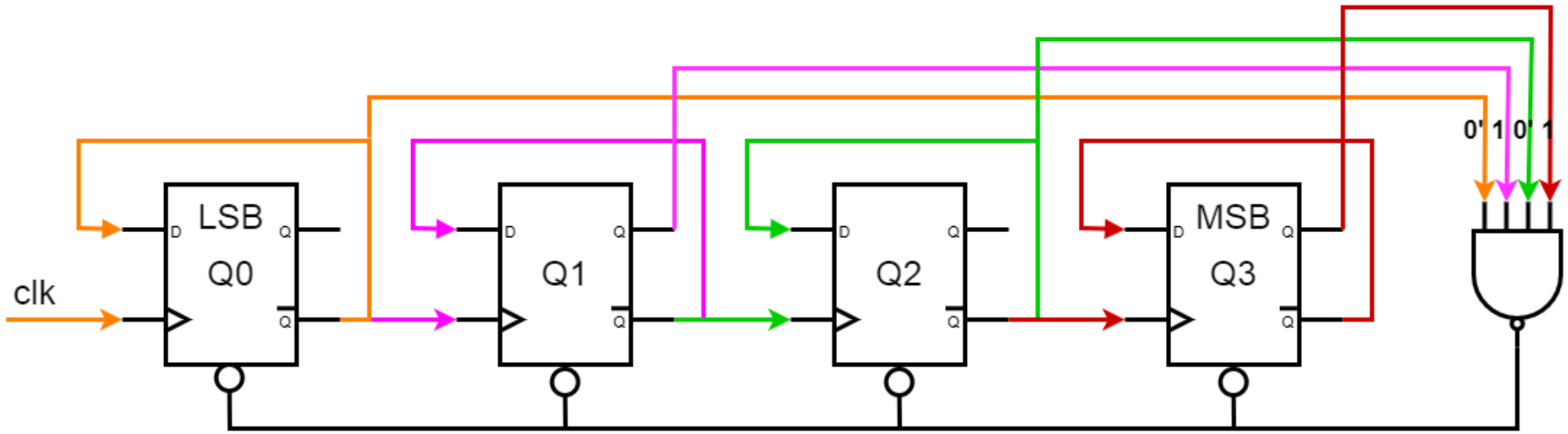
Desvantagens dos contadores assíncronos

- Problemas de sincronismo: os estados não mudam simultaneamente.
- O tempo de propagação de cada Flip-flop se acumula.
- Erros de contagem por pequenas frações de tempo.
- Velocidade limitada: dificuldade de contar a altas frequências de clock.
- Isto é uma desvantagem para aplicações que requerem altas taxas de processamento.

Página em branco.

Implementação em HDL de contadores assíncronos

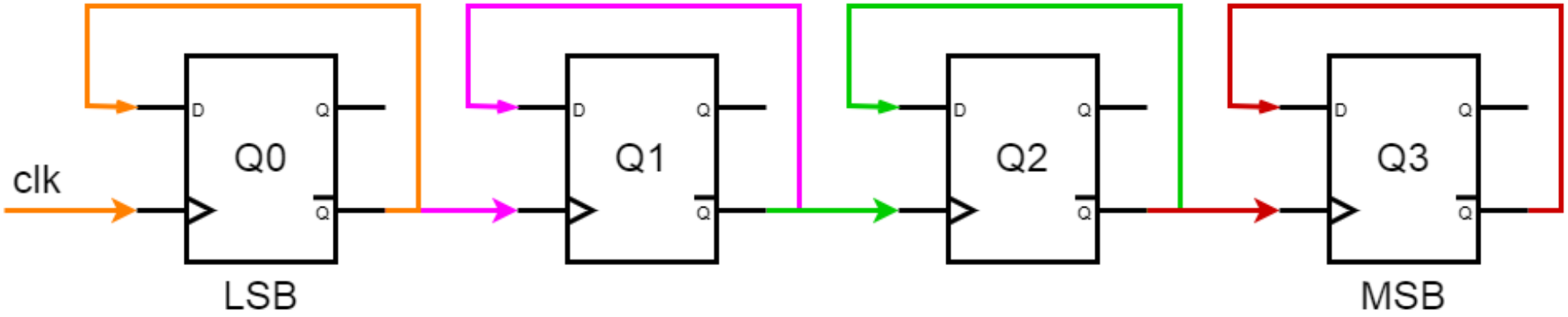
- Exemplo contador BCD



Aula 2
Implementação de contadores assíncronos em HDLs

Implementação de um contador assíncrono de 4 bits

Embora seja possível implementar contadores assíncronos em HDLs, a sua aplicação deve ser muito bem analisada a fim de verificar que os atrasos não produzam saídas inesperadas.



Implementação Verilog de um contador assíncrono mod16

```
1  module cnt_assincrono_asc_4bits (
2      input reset,
3      input clk,
4      output [3:0] led
5  );
6
7      reg [3:0] Q;
8      reg [3:0] Qn;
9      reg [3:0] D;
10
11     // Generate the inverse of Q
12     always @(*) begin
13         Qn = ~Q;
14     end
15
16     // Logic for D based on Qn
17     always @(*) begin
18         D[0] = Qn[0];
19         D[1] = Qn[1];
20         D[2] = Qn[2];
21         D[3] = Qn[3];
22     end
```

Implementação Verilog de um contador assíncrono mod16

```
24      // Process for Q(0)
25      always @(posedge clk or posedge reset) begin
26          if (reset) begin
27              Q[0] <= 0;
28          end else begin
29              Q[0] <= D[0];
30          end
31      end
32
33      // Process for Q(1)
34      always @(posedge Qn[0] or posedge reset) begin
35          if (reset) begin
36              Q[1] <= 0;
37          end else begin
38              Q[1] <= D[1];
39          end
40      end
41
```

```
42      // Process for Q(2)
43      always @(posedge Qn[1] or posedge reset) begin
44          if (reset) begin
45              Q[2] <= 0;
46          end else begin
47              Q[2] <= D[2];
48          end
49      end
50
51      // Process for Q(3)
52      always @(posedge Qn[2] or posedge reset) begin
53          if (reset) begin
54              Q[3] <= 0;
55          end else begin
56              Q[3] <= D[3];
57          end
58      end
59
60      // Assign Q to led output
61      assign led = Q;
62
63  endmodule
```

Implementação Verilog de um contador assíncrono mod16

Observe que apenas o primeiro FF é sensível à borda de subida do clock.

```
24      // Process for Q(0)
25      always @(posedge clk or posedge reset) begin
26          if (reset) begin
27              Q[0] <= 0;
28          end else begin
29              Q[0] <= D[0];
30          end
31      end
32
33      // Process for Q(1)
34      always @(posedge Qn[0] or posedge reset) begin
35          if (reset) begin
36              Q[1] <= 0;
37          end else begin
38              Q[1] <= D[1];
39          end
40      end
41
```

```
42      // Process for Q(2)
43      always @(posedge Qn[1] or posedge reset) begin
44          if (reset) begin
45              Q[2] <= 0;
46          end else begin
47              Q[2] <= D[2];
48          end
49      end
50
51      // Process for Q(3)
52      always @(posedge Qn[2] or posedge reset) begin
53          if (reset) begin
54              Q[3] <= 0;
55          end else begin
56              Q[3] <= D[3];
57          end
58      end
59
60      // Assign Q to led output
61      assign led = Q;
62
63  endmodule
```

Os outros FFs dependem da atualização da saída do FF anterior, produzindo um atraso.

Implementação VHDL de um contador assíncrono mod16

```
16  library IEEE;
17  use IEEE.STD_LOGIC_1164.ALL;
18
19  entity cnt_assincrono_asc_4bits is
20      Port ( reset : in STD_LOGIC;
21             clk : in STD_LOGIC;
22             led : out STD_LOGIC_VECTOR (3 downto 0));
23  end cnt_assincrono_asc_4bits;
24
25  architecture Behavioral of cnt_assincrono_asc_4bits is
26
27  signal Q, Qn : std_logic_vector(3 downto 0) := "0000";
28  signal D : std_logic_vector(3 downto 0) := "0000";
```

```
32      Qn <= not Q;
33
34      D(0) <= Qn(0);
35      D(1) <= Qn(1);
36      D(2) <= Qn(2);
37      D(3) <= Qn(3);
38
```

Implementação VHDL de um contador assíncrono mod16

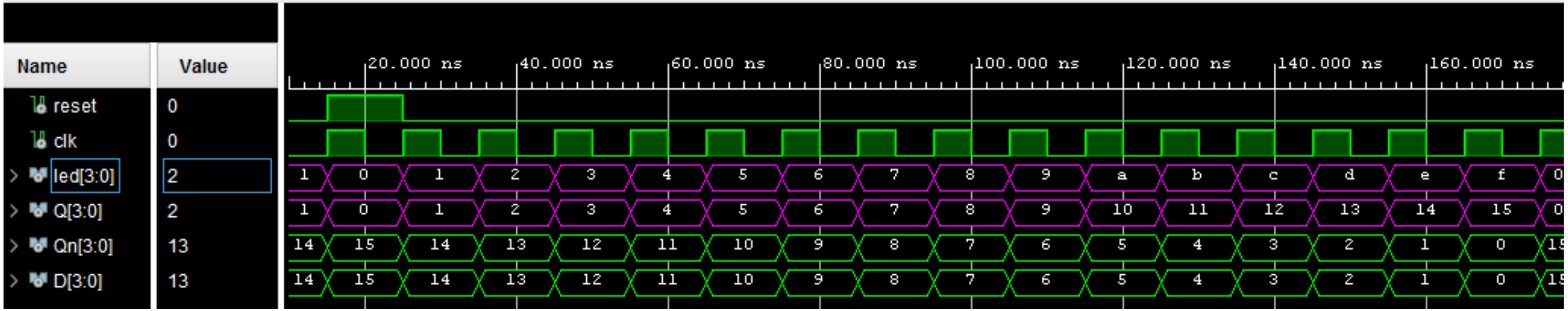
```
39      process(clk,reset)
40      begin
41          if reset='1' then
42              Q(0) <= '0';
43          elsif rising_edge(clk) then
44              Q(0) <= D(0);
45          end if;
46      end process;
47
48      process(Qn(0),reset)
49      begin
50          if reset='1' then
51              Q(1) <= '0';
52          elsif rising_edge(Qn(0)) then
53              Q(1) <= D(1);
54          end if;
55      end process;
```

```
57      process(Qn(1),reset)
58      begin
59          if reset='1' then
60              Q(2) <= '0';
61          elsif rising_edge(Qn(1)) then
62              Q(2) <= D(2);
63          end if;
64      end process;
65
66      process(Qn(2),reset)
67      begin
68          if reset='1' then
69              Q(3) <= '0';
70          elsif rising_edge(Qn(2)) then
71              Q(3) <= D(3);
72          end if;
73      end process;
74
75      led <= Q;
```

Contador assíncrono mod16: Testbench (VHDL).

```
25  component cnt_assincrono_asc_4bits is
26      Port ( reset : in STD_LOGIC;
27             clk : in STD_LOGIC;
28             led : out STD_LOGIC_VECTOR (3 downto 0));
29  end component;
30
31  signal reset,clk : std_logic := '0';
32  signal led : std_logic_vector(3 downto 0) := "0000";
33
34  begin
35
36      uut: cnt_assincrono_asc_4bits port map(
37          reset   => reset,
38          clk     => clk,
39          led     => led);
40
41      clk <= not clk after 5 ns;
42      reset <= '0', '1' after 15 ns, '0' after 25 ns;
```

Contador assíncrono de mod16: Simulação.



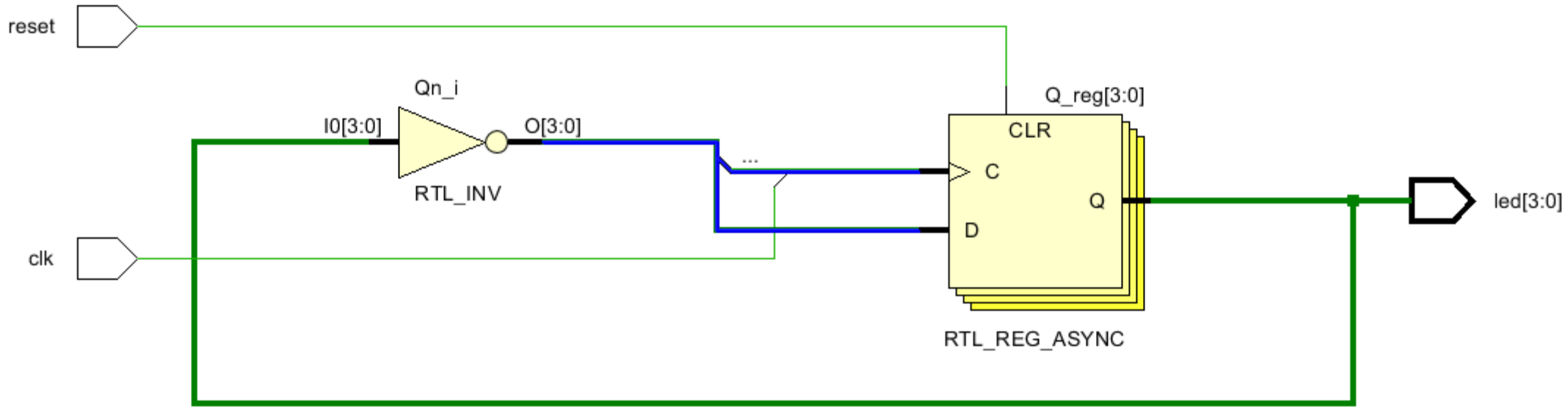
A simulação comportamental não leva em consideração atrasos na propagação dos sinais. Serve apenas para aferir a corretude lógica do circuito implementado em HDLs.

A simulação de timing (pós-implementação) permite analisar o efeito dos atrasos do sinal de clock e de dados entre os Flip-flops.

Página em branco.

Implementação de um contador assíncrono mod16

Após a especificação dos constraints de IO (reset, clock e leds), podemos observar o diagrama esquemático de alto nível.

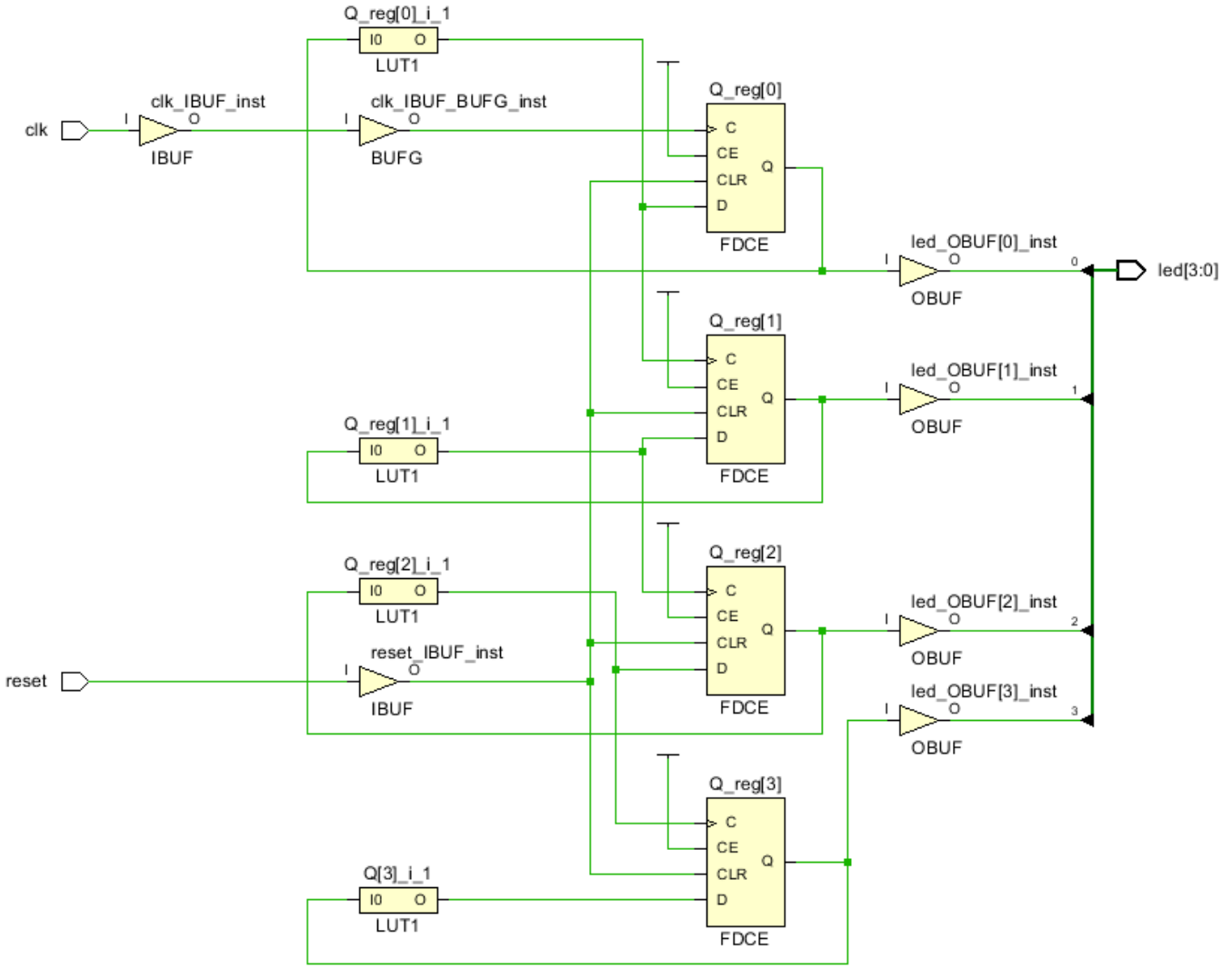


Implementação de um contador assíncrono mod16

Esquemático de síntese lógica:

Após a síntese lógica temos a definição das portas lógicas (implementadas em LUTs) e sua conexão com Flip-flops e portas de entrada e a saída.

Observe que as LUTs devem implementar a negação D <= not Q



Implementação de um contador assíncrono mod16

Esquemático de síntese lógica:

Após a síntese lógica podemos estimar o consumo de recursos em termos de LUTs (Look-up Tables), Flip-flops, pinos de IO, e outros recursos.

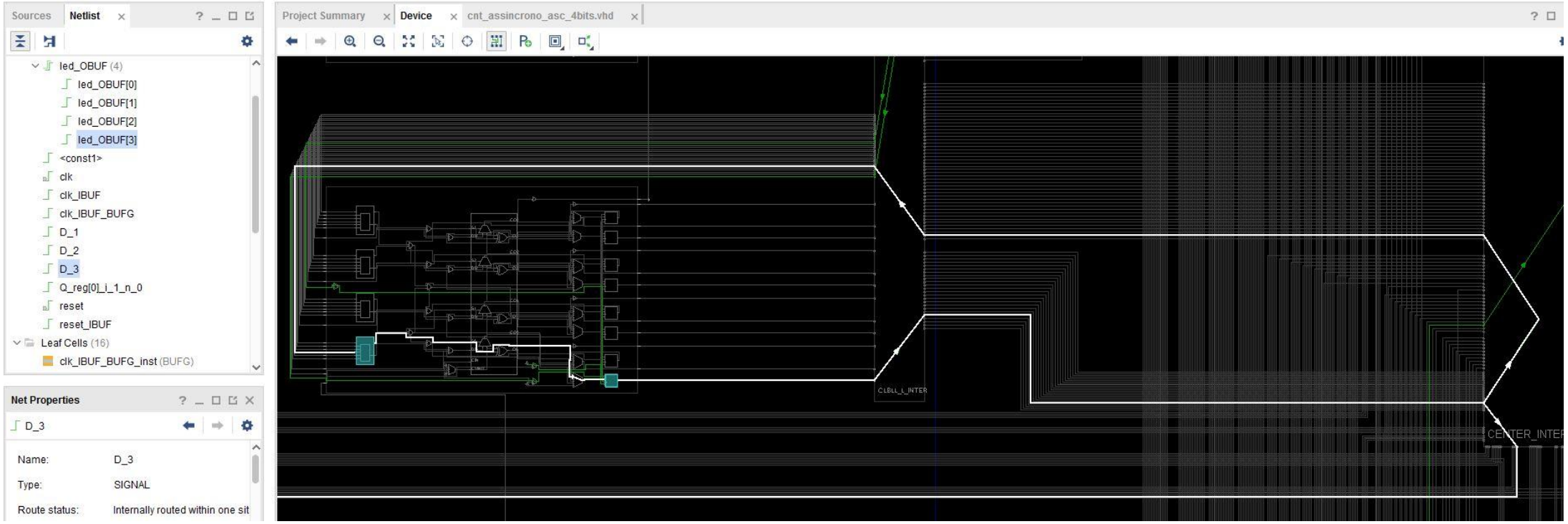
Utilization **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	4	20800	0.02
FF	4	41600	0.01
IO	6	106	5.66
BUFG	1	32	3.13

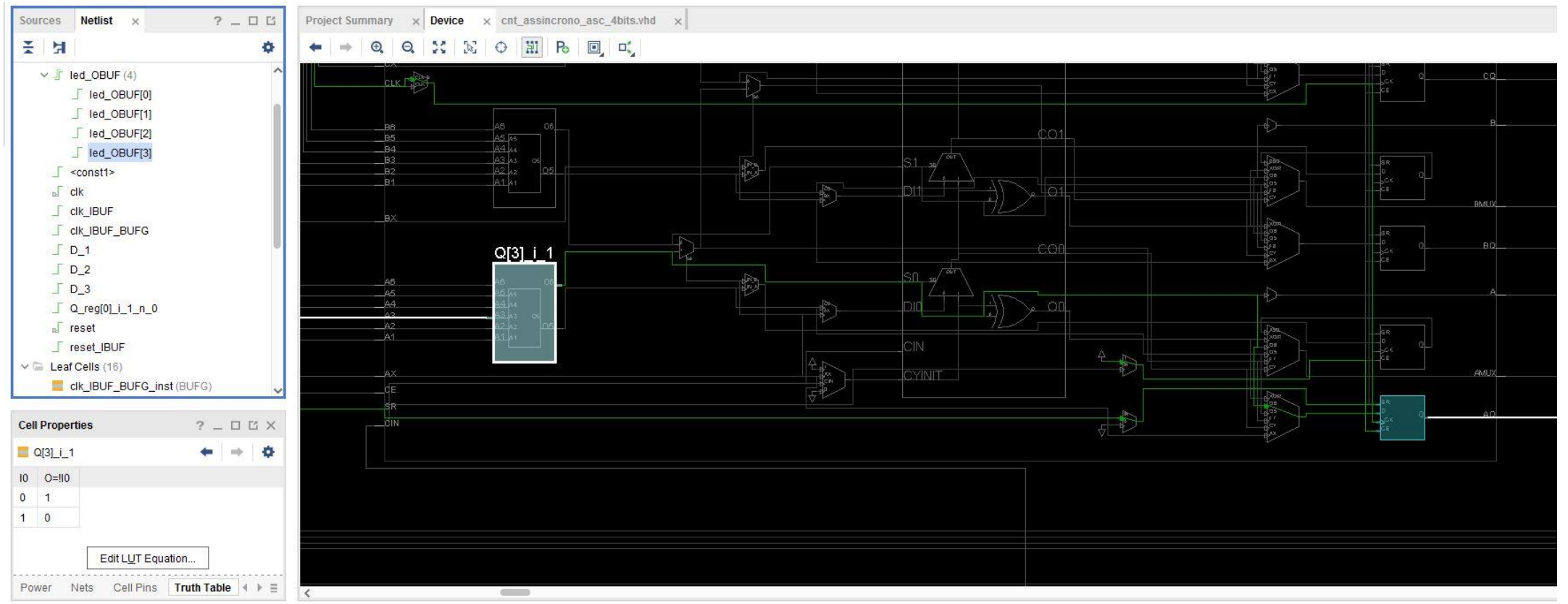
Implementação de um contador assíncrono mod16

Após o processo Place and Route (PAR) observa-se o roteamento do circuito no FPGA. A LUT implementa D(3) <= not Q(3). Observe o comprimento do caminho, o qual pode ser diferente para os outros Flip-flops, produzindo delays diferentes.

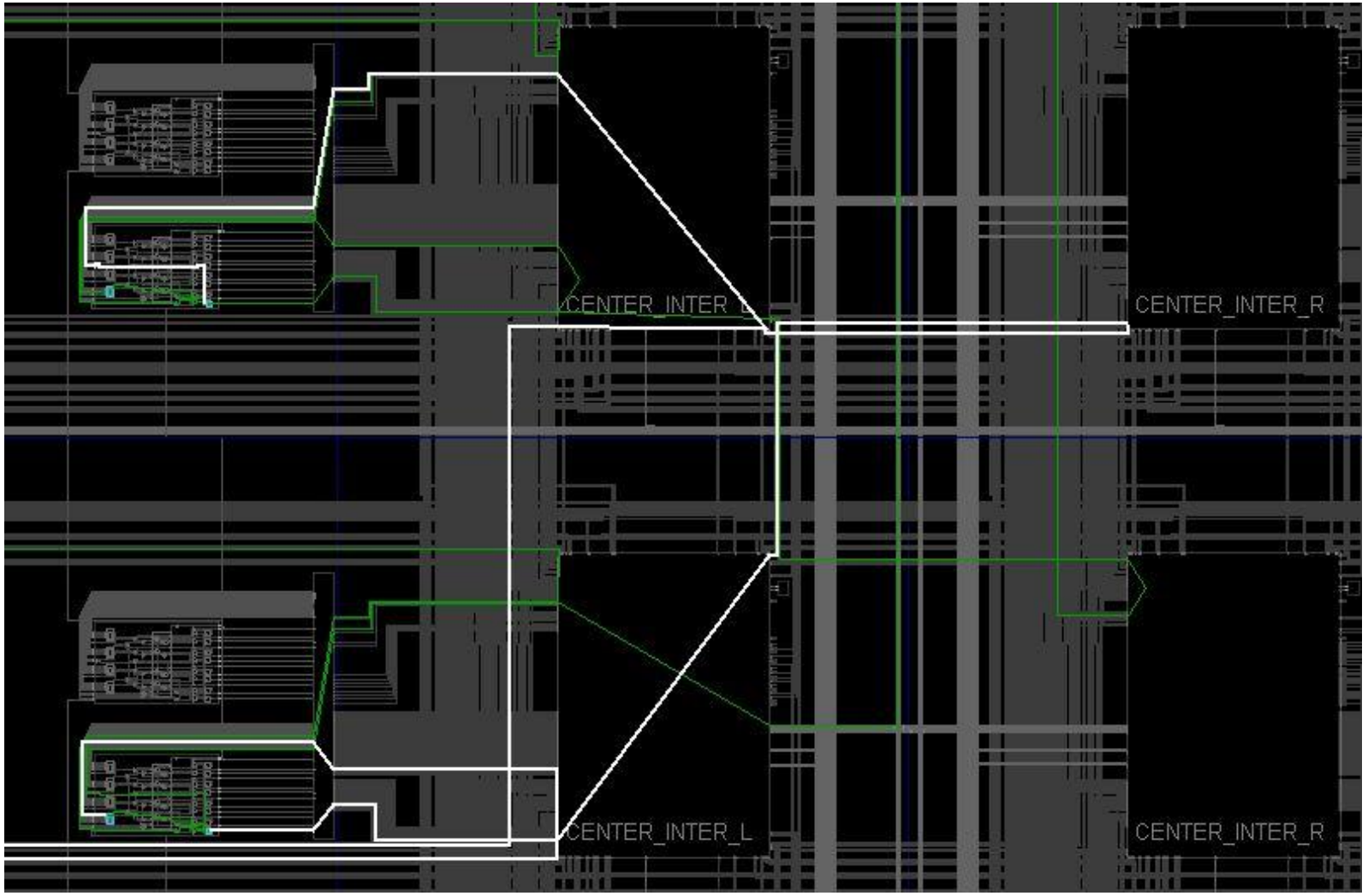


Implementação de um contador assíncrono mod16

Análise do Bloco Lógico Configurável para D(3) <= not Q(3). Observe a tabela verdade da LUT (Look-up Table)



Implementação de um contador assíncrono mod16



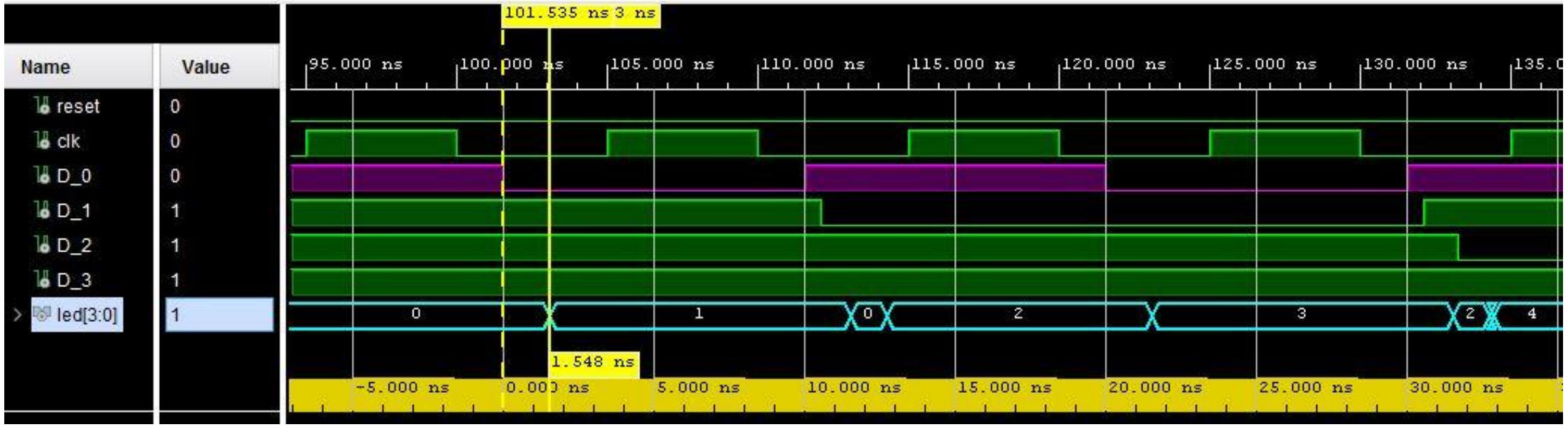
Análise clock do Flip-flop do bit 1

Q(1) = Qn(0).

Observe os caminhos que produzem atrasos no clock deste flip-flop do contador assíncrono.

Implementação de um contador assíncrono mod16

Simulação de timing após o processo PAR:



Observe o atraso entre a borda de subida do clk e D(0)=Qn(0)

Observe o atraso entre a borda de descida de D(0)=Qn(0)=clk FF1 e a saída led.

Página em branco.