SD132 – Circuitos Digitais II
A-202 Registradores de Deslocamento

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
09/04/2025	V1.1	Revisão de conteúdo, organização de exercícios

Tópicos

- Definição
- Registrador SIPO (serial-paralelo)
- Registrador PISO (paralelo-serial)
- Registrador de deslocamento universal
- Registrador de deslocamento em anel
- Implementação em HDL

Página em branco.

Aula 1
Registradores de deslocamento

Definições

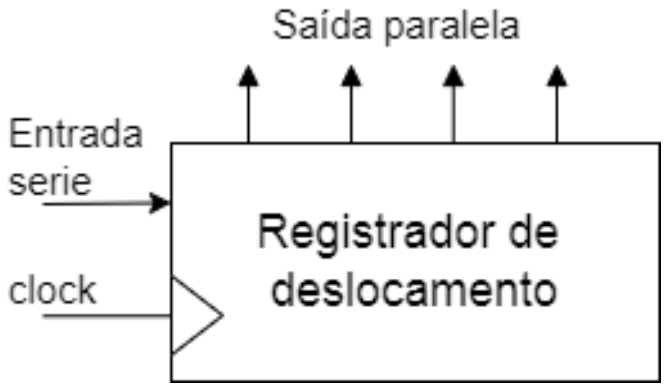
- São circuitos sequenciais síncronos.
- Conjunto de Flip-flops que podem ser interligados para deslocar a informação à esquerda ou à direita a cada ciclo de relógio.
- Podem ser usados para rotacionar (deslocamento em anel) a cada ciclo de clock uma palavra de *N* bits.
- Aplicações: criptografia, aritmética computacional (multiplicar ou dividir por potencias de dois), serialização e paralelização de dados, sincronismo de eventos, comunicações digitais, processamento de sinais, etc.

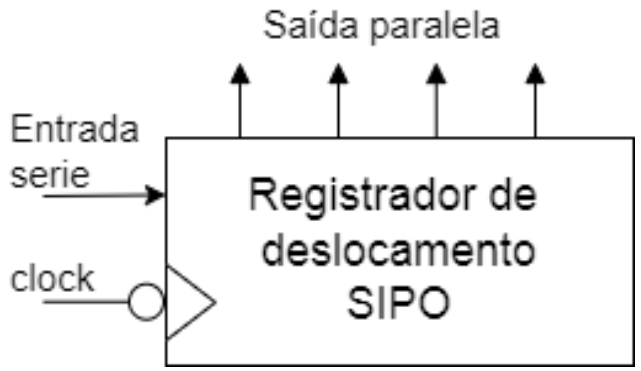
Definições

- Registradores com entrada serie: a informação binaria é formada por bits que chegam sequencialmente, um após o outro.
- Registradores com entrada paralela: a informação binária é formada por bits que se apresentam simultaneamente.
- Registradores de deslocamento podem ser usados para serializar informação (conversor paralelo – série) ou paralelizar informação (conversor séria - paralelo).

Registrador SIPO

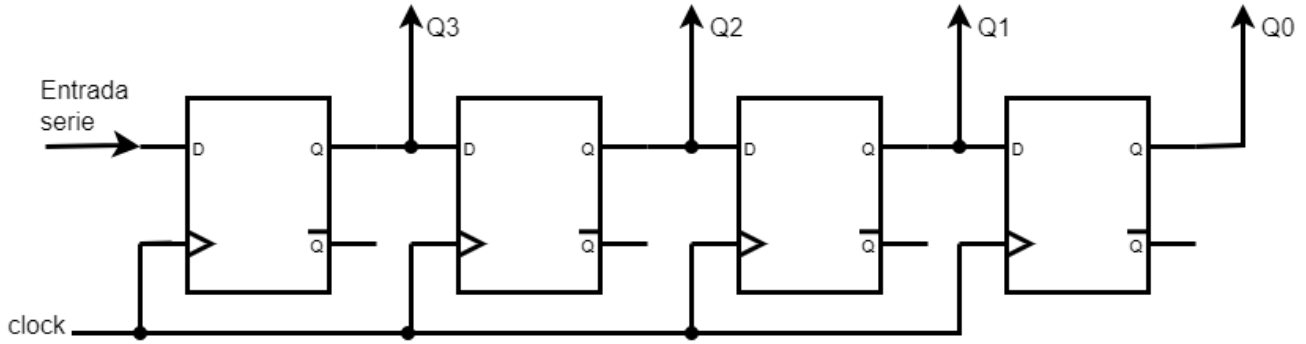
- Entrada serial - saída paralela.
- Serve como conversor série - paralelo.
- Exemplo de registrador SIPO de 4 bits:



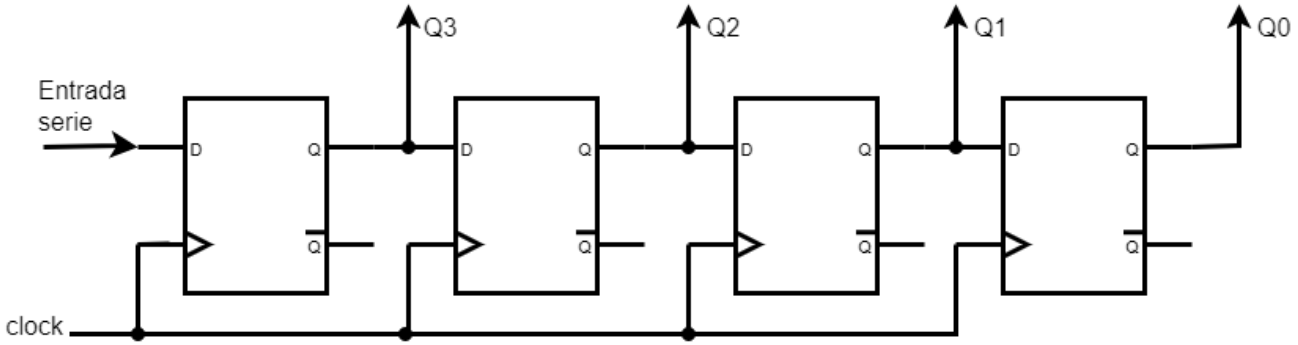


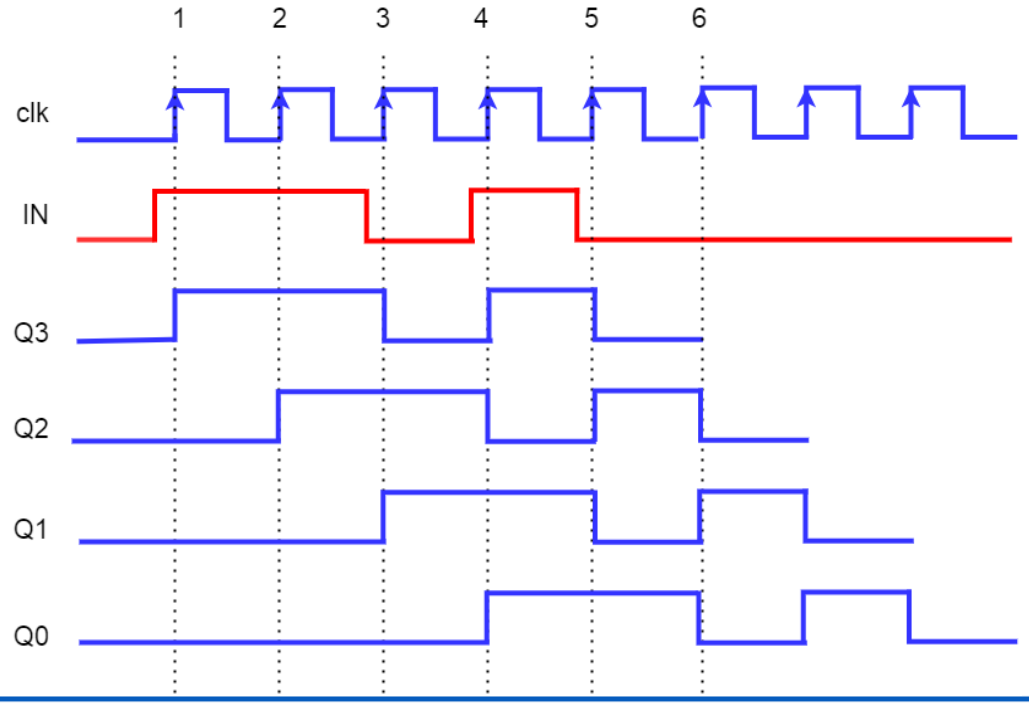
Registrador SIPO (entrada serial - saída paralela)

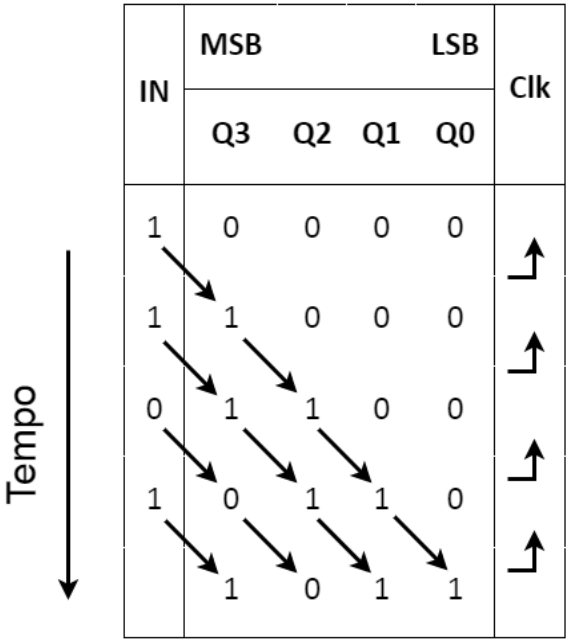
- Exemplo de registrador SIPO de 4 bits (deslocamento à direita)



Registrador SIPO 4 bits (entrada serial - saída paralela)





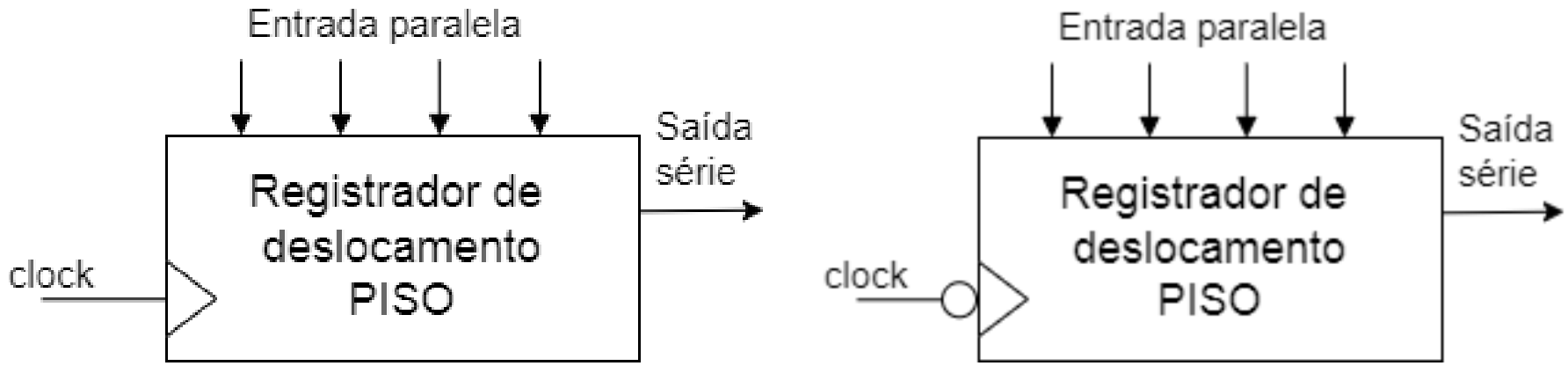


- Quatro clocks para carregar a palavra de entrada.
- Quatro clocks para esvaziar o registrador.

Página em branco.

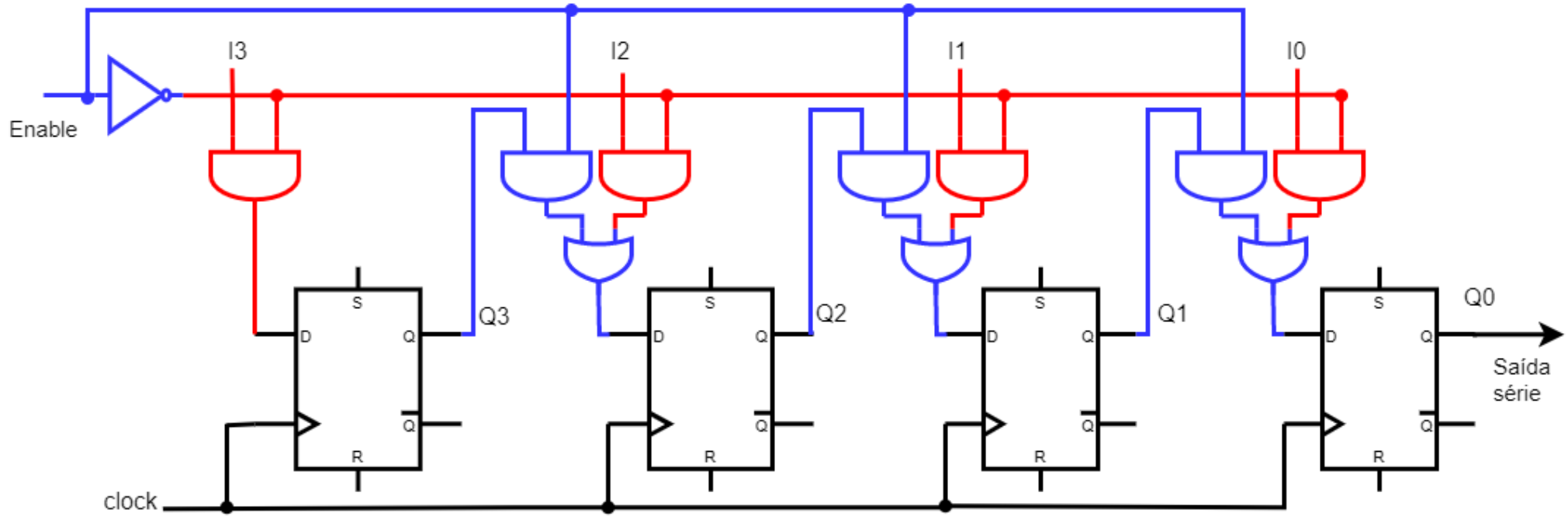
Registrador PISO

- Entrada paralela - saída serial.
- Serve como serializador de dados.
- Exemplo de registrador PISO de 4 bits:



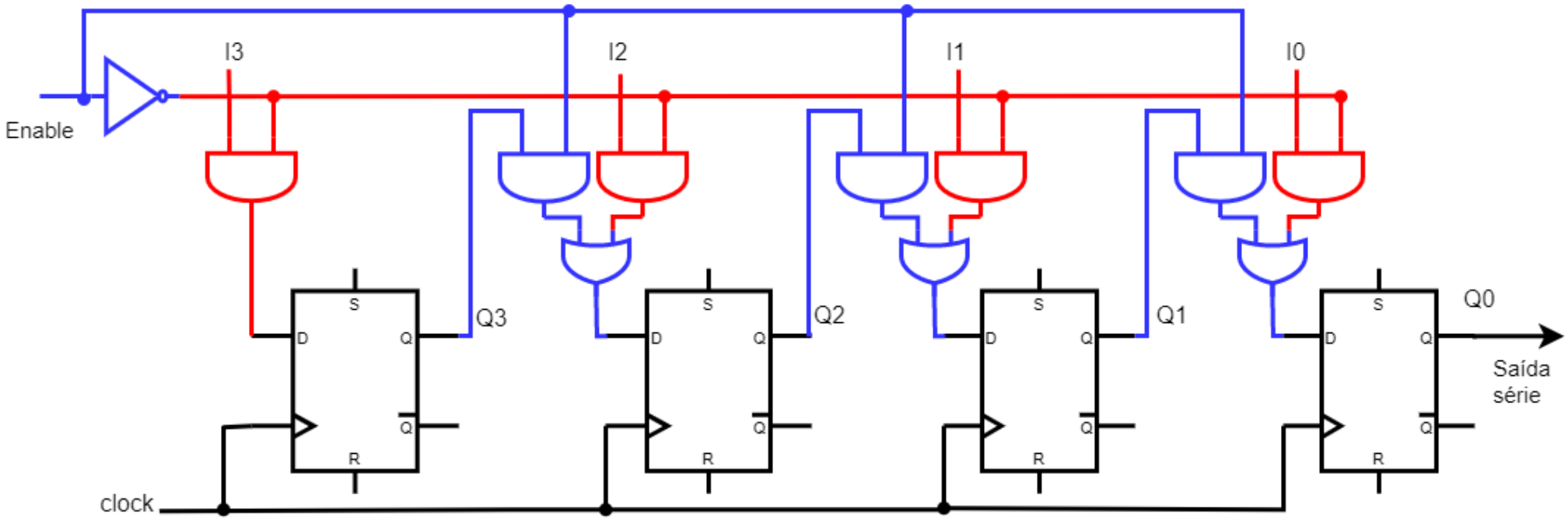
Registrador PISO de 4 bits (entrada paralela saída série)

- Exemplo de registrador PISO de 4 bits com lógica combinacional para carregar a palavra. Desloca à direita.



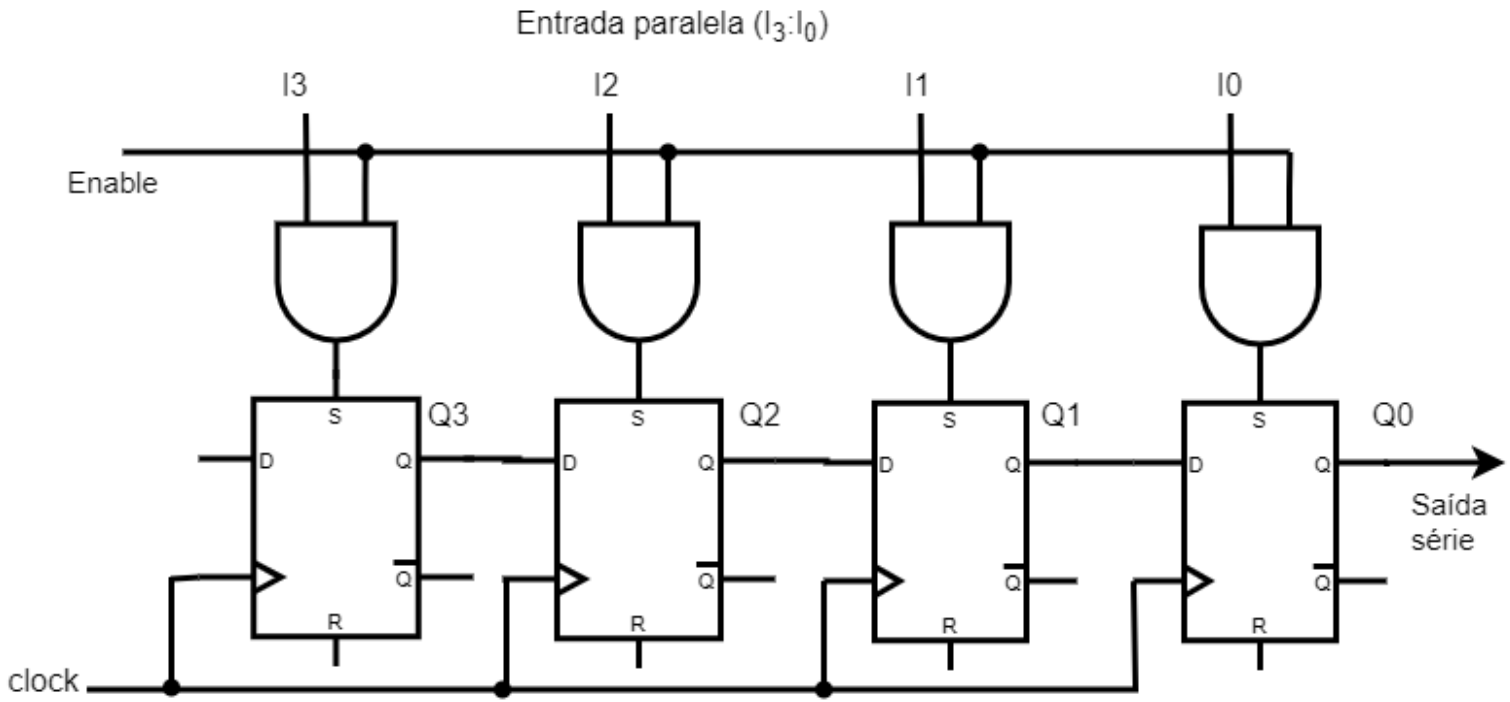
Registrador PISO de 4 bits (entrada paralela saída série)

1. Enable='0' para carregar de forma paralela a palavra I3:I0 no registrador.
2. Enable='1' para habilitar o deslocamento.
3. A cada borda de subida do clock, o registrador será deslocado à direita. A palavra aparece serialmente na saída Q(0).



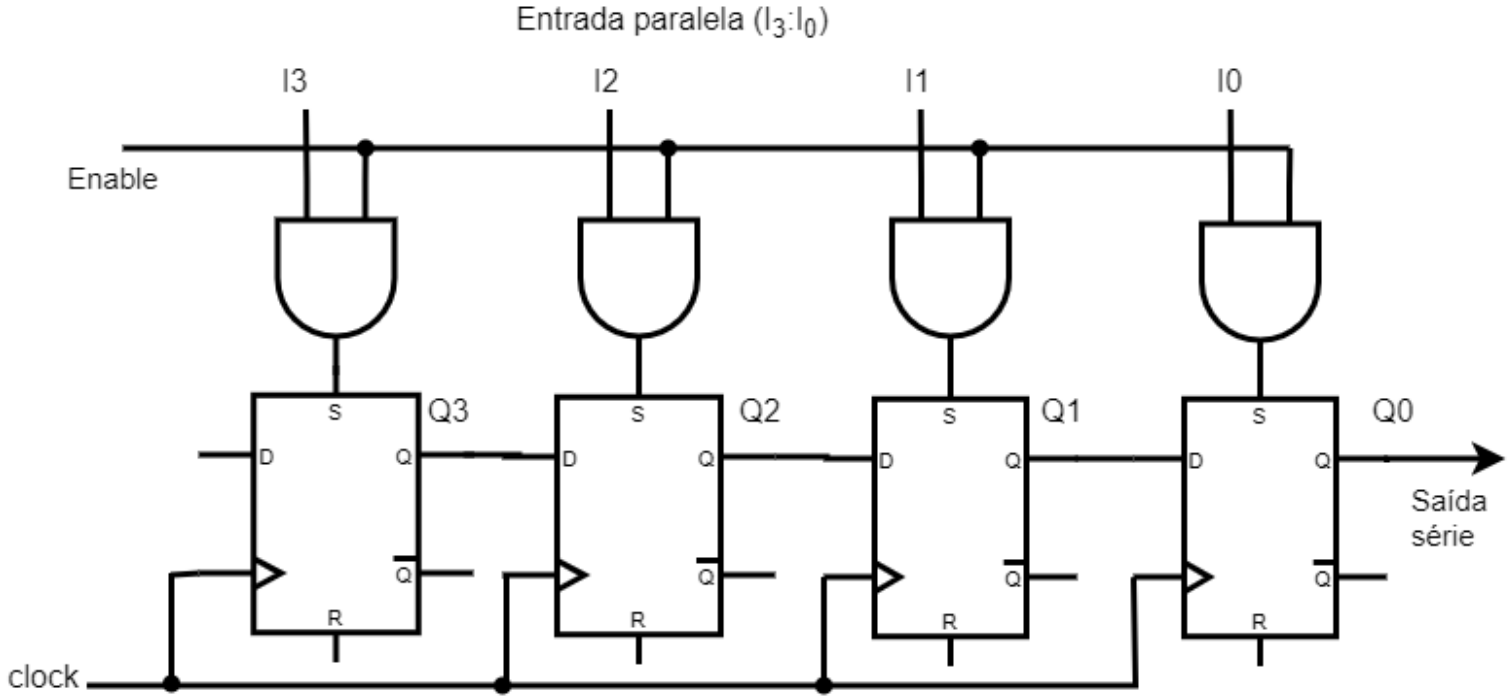
Registrador PISO de 4 bits (entrada paralela saída série)

- Exemplo de registrador PISO de 4 bits com entrada assíncrona Set para carregar a palavra.

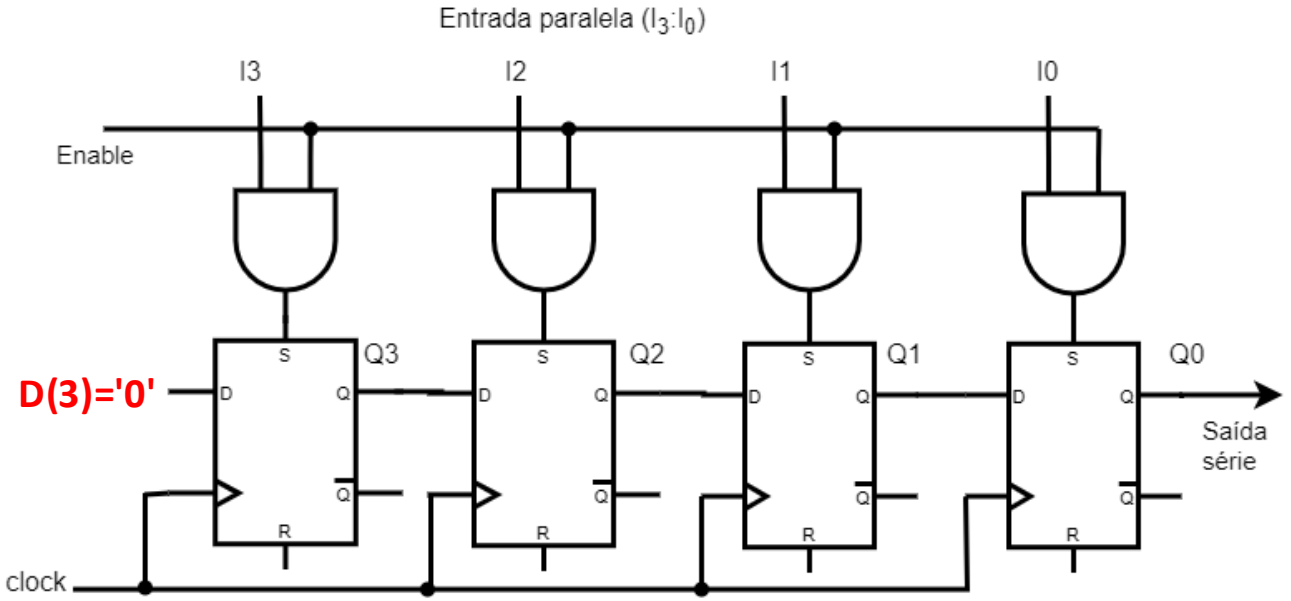


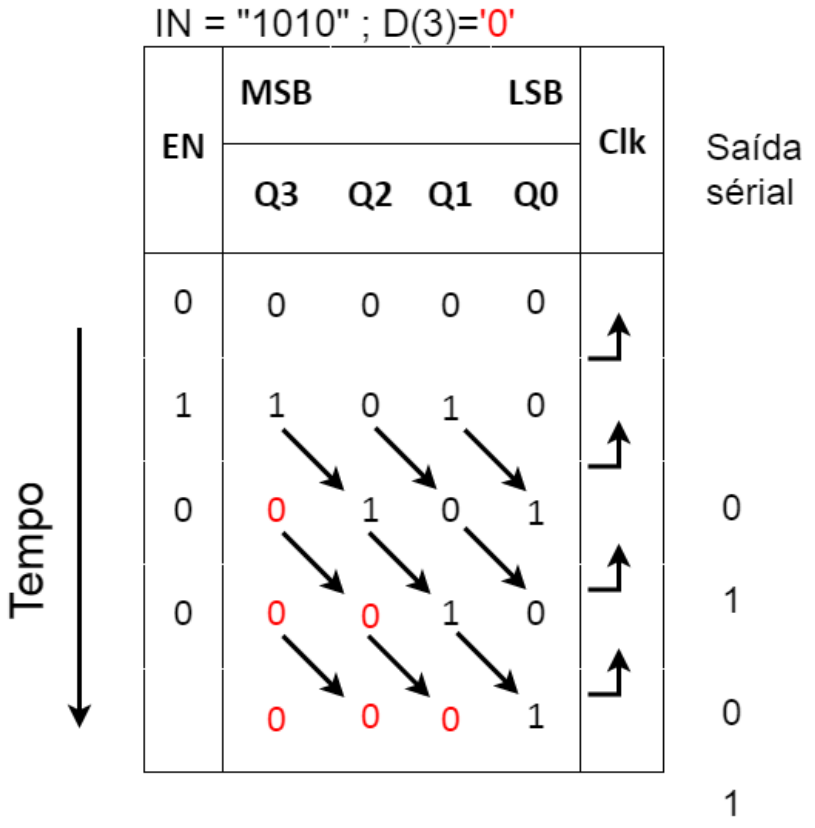
Registrador PISO de 4 bits (entrada paralela saída série)

1. Enable='1' e Set='1' para carregar de forma paralela a palavra no registrador.
2. Enable='0' e Set='0' para habilitar o deslocamento.
3. A cada borda de subida do clock, o registrador será deslocado à direita. Na saída Q(0) vai aparecer serialmente a palavra registrada.



Registrador PISO (entrada paralela saída serial)



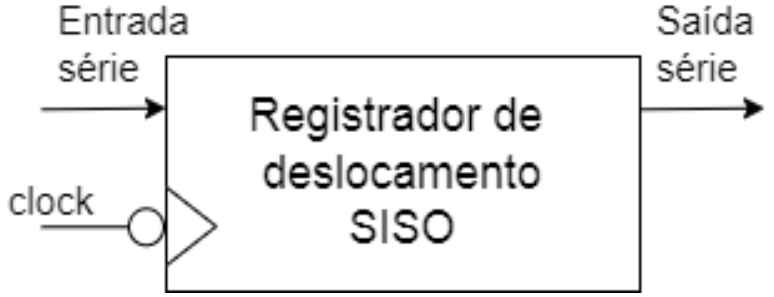


- A cada borda de subida do clock, a entrada D(3) no Flip-flop MSB desloca a palavra à direita em uma posição.
- A palavra registrada aparece serialmente na saída do Flip-flop LSB.

Página em branco.

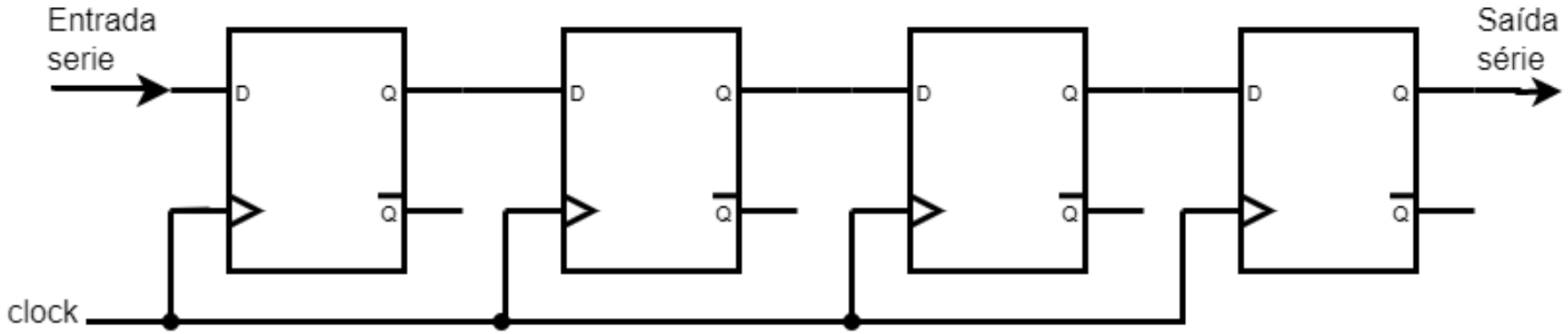
Registrador SISO

- Entrada serial- saída serial
- Serve para realizar transferência serial de dados.



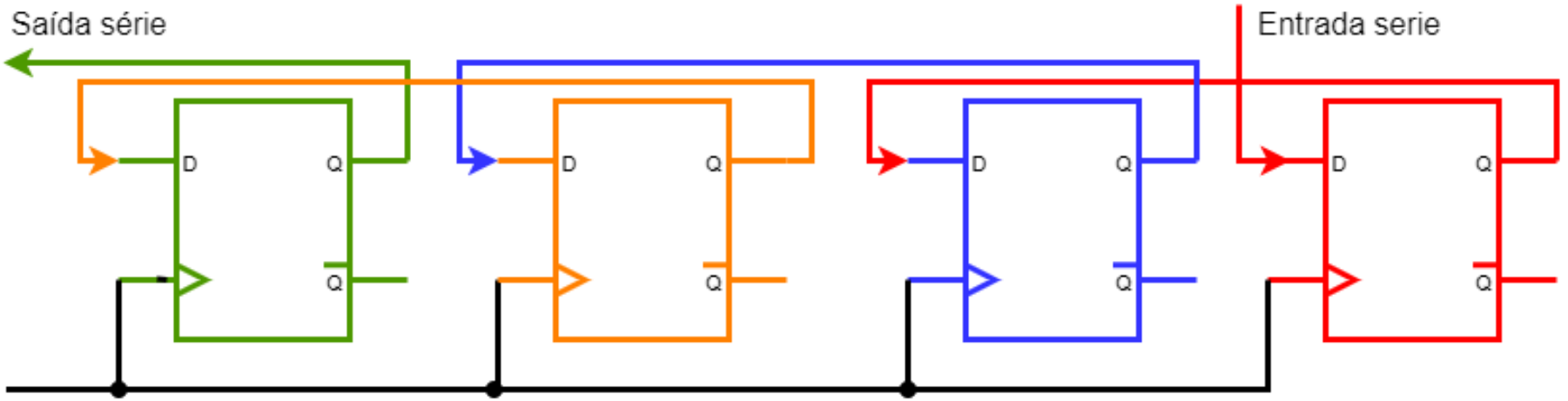
Registrador SISO (Entrada serial - saída serial)

- Exemplo de registrador SISO de 4 bits com deslocamento à direita.



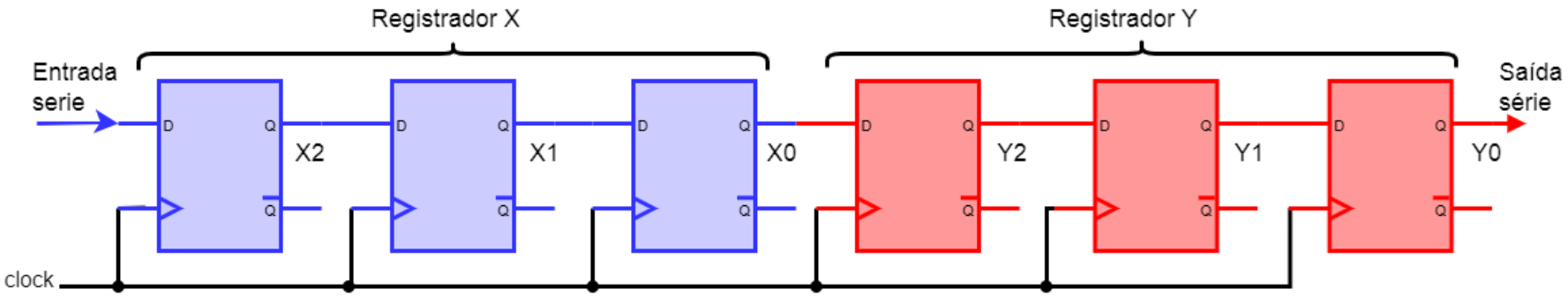
Registrador SISO (Entrada serial - saída serial)

- Exemplo de registrador SISO de 4 bits com deslocamento à esquerda.



Registrador SISO (Entrada serial - saída serial)

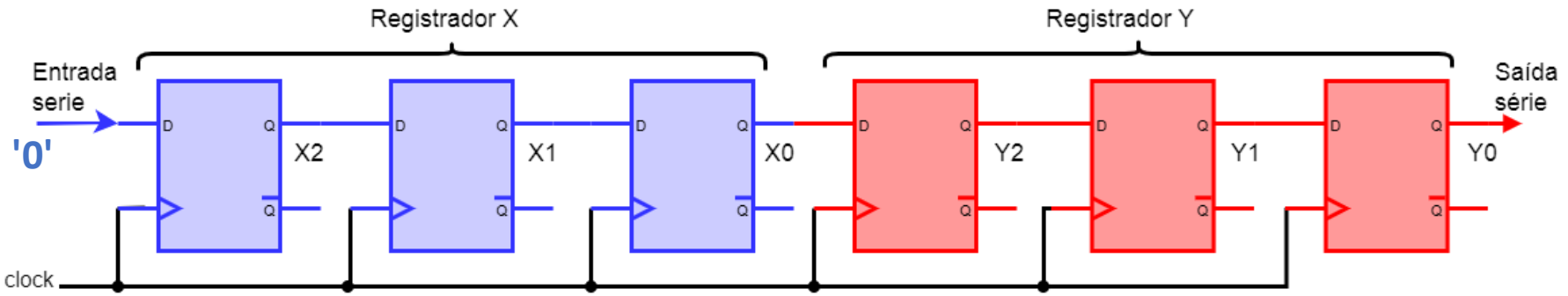
- Exemplo de um SISO para transferência bit serial entre dois registradores.



- Inicialmente a entrada serial é usada para carregar a informação desejada no registrador X.
- Neste exemplo o registrador X é carregado em três (3) ciclos de relógio.

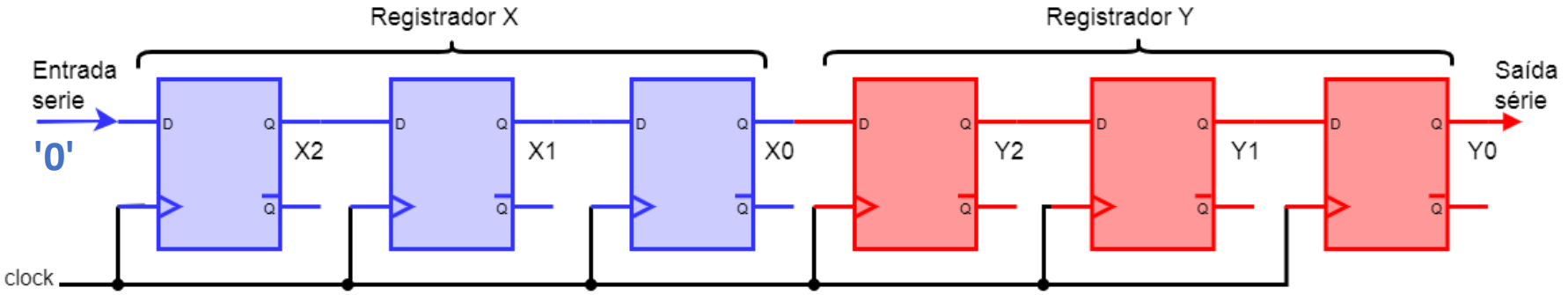
Registrador SISO (Entrada serial - saída serial)

- Exemplo de um SISO para transferência serial de dados entre dois registradores.



- Após o carregamento do registrador X, a entrada série é colocada em '0' e a cada novo ciclo de relógio transfere bit a bit a informação no registrador Y.
- Após 3 ciclos de relógio a informação foi transferida ao registrador Y.

Registrador SISO (Entrada serial - saída serial)



X2	X1	X0	Y2	Y1	Y0
1	0	1	0	0	0
0	1	0	1	0	0
0	0	1	0	1	0
0	0	0	1	0	1

Condição após RegX ser carregado

Depois do primeiro pulso

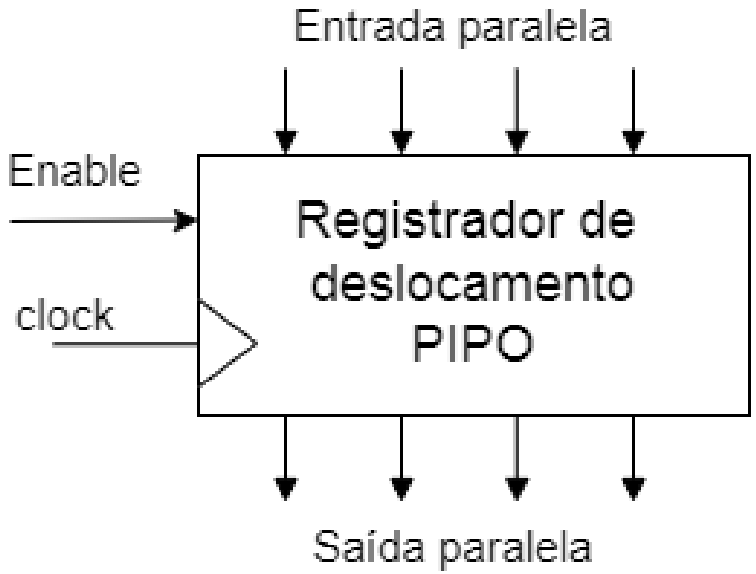
Depois do segundo pulso

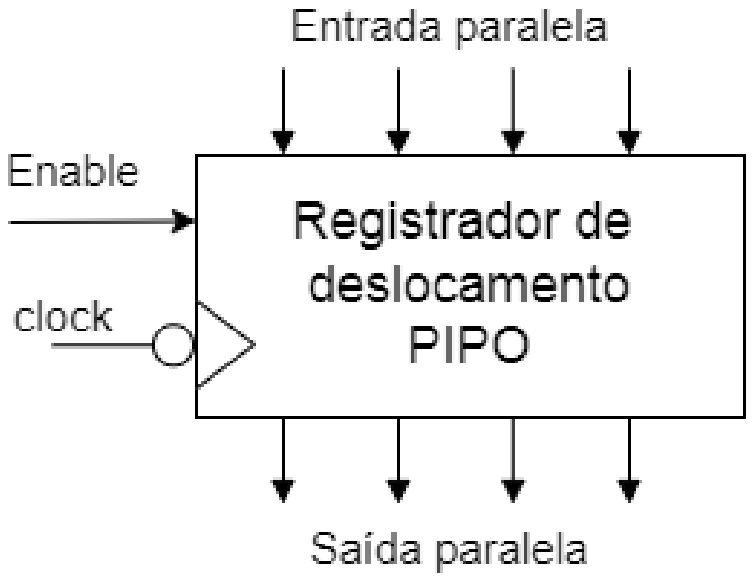
Depois do terceiro pulso

Página em branco.

Registrador PIPO

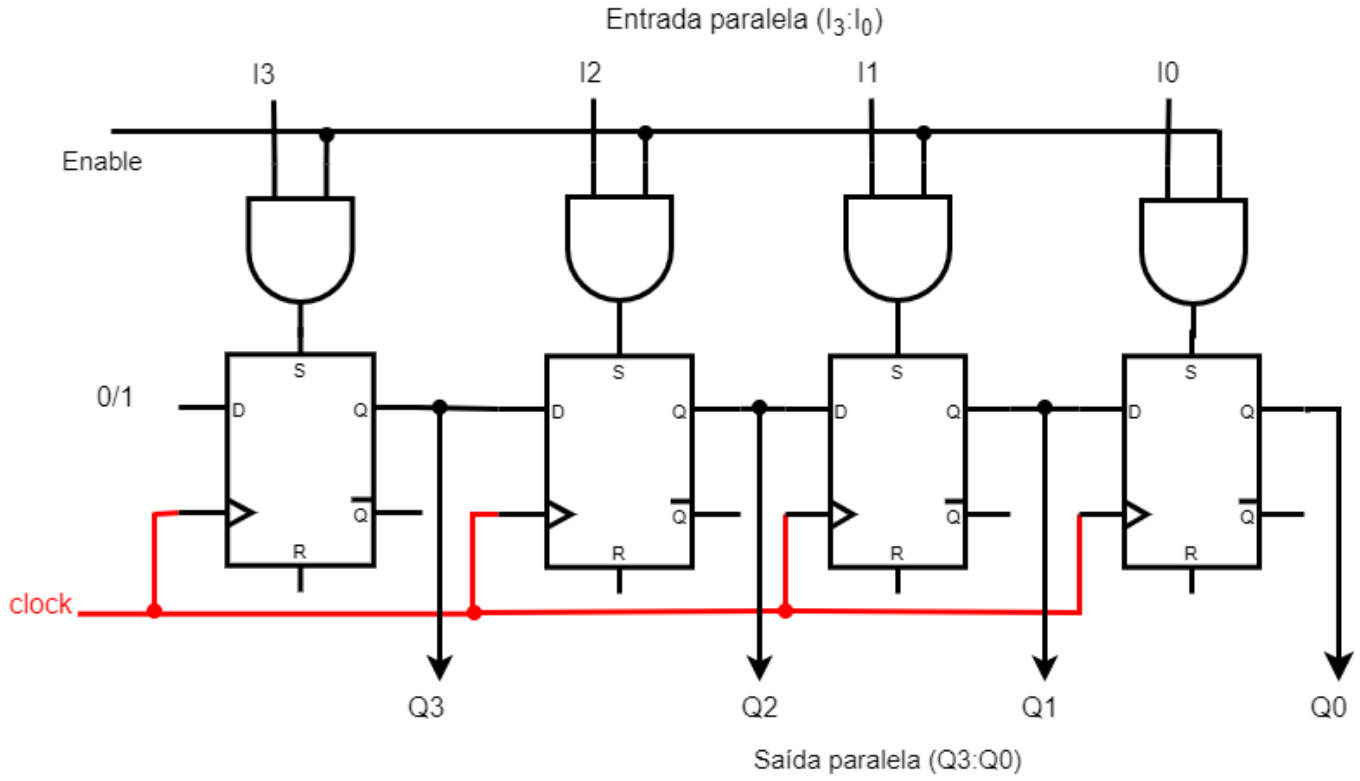
- Entrada paralela - saída paralela
- A entrada *enable* permite carregar a palavra no registrador.





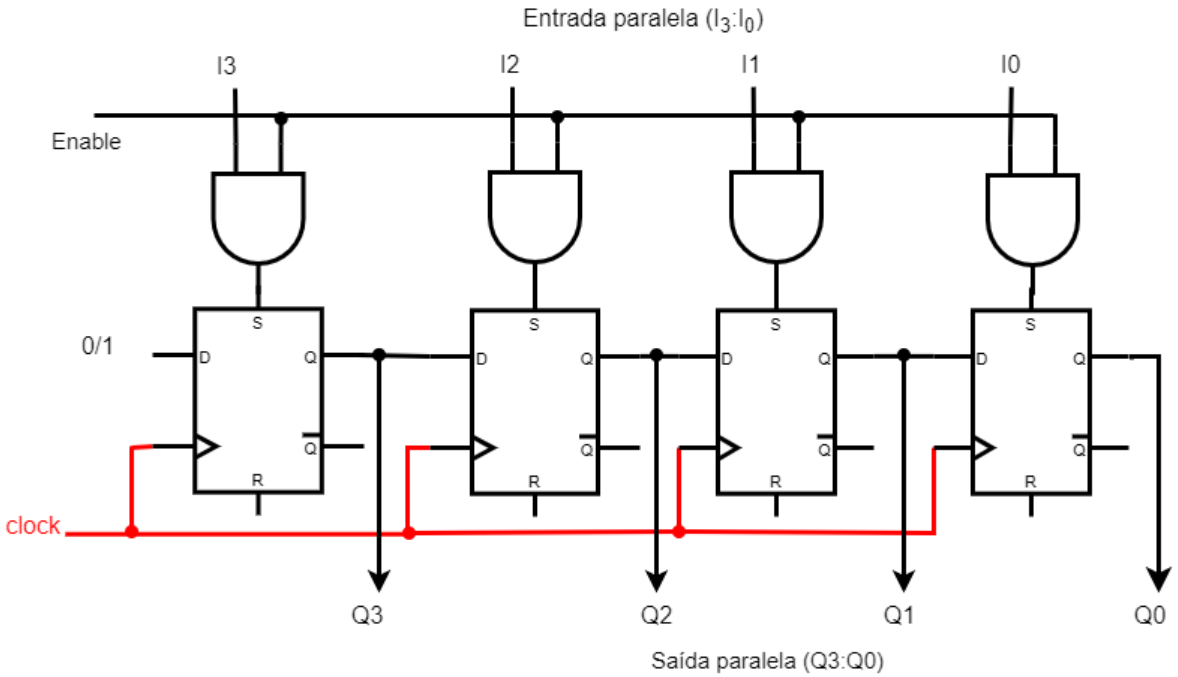
Registrador PIPO (entrada paralela - saída paralela)

- Exemplo de PIPO de 4 bits à direita.
- Deslocamento à direita: O bit descartado é o resto da divisão.



Registrador PIPO (entrada paralela - saída paralela)

- Exemplo de PIPO de 4 bits à direita.
- Deslocando com '0' é feita uma divisão por 2.

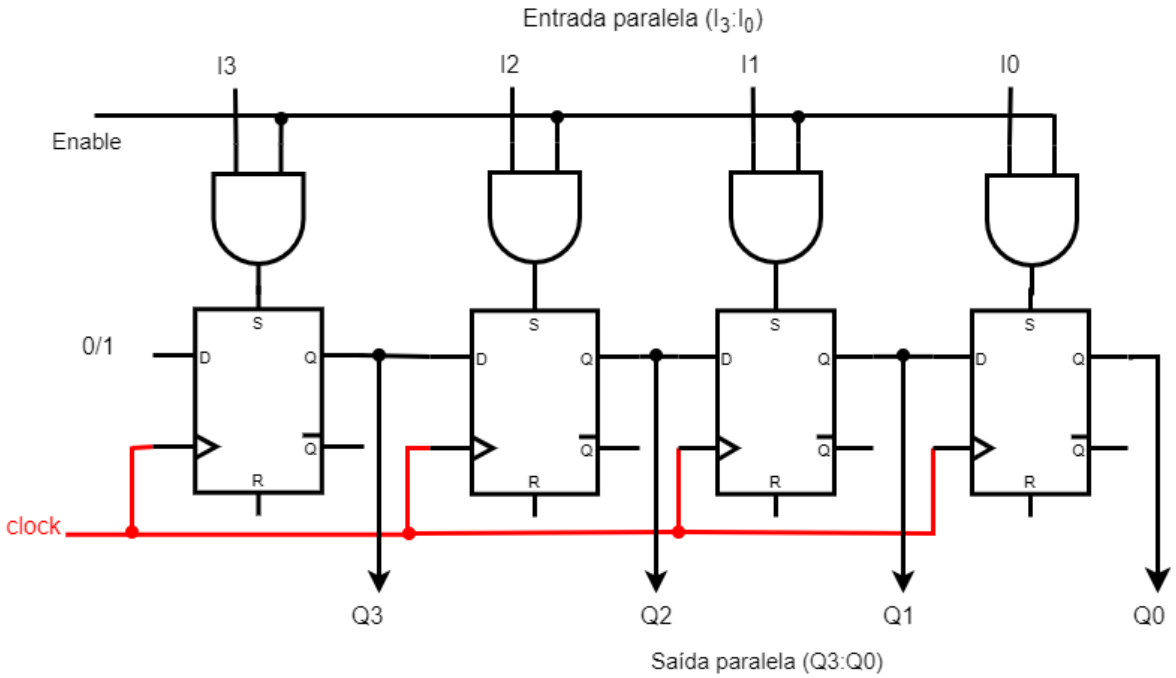


Tempo

D(3)	Q3	Q2	Q1	Q0	Resto	Dec
x	1	1	0	1	x	13
0	0	1	1	0	1	6
0	0	0	1	1	0	3
0	0	0	0	1	1	1
0	0	0	0	0	1	0

Registrador PIPO (entrada paralela - saída paralela)

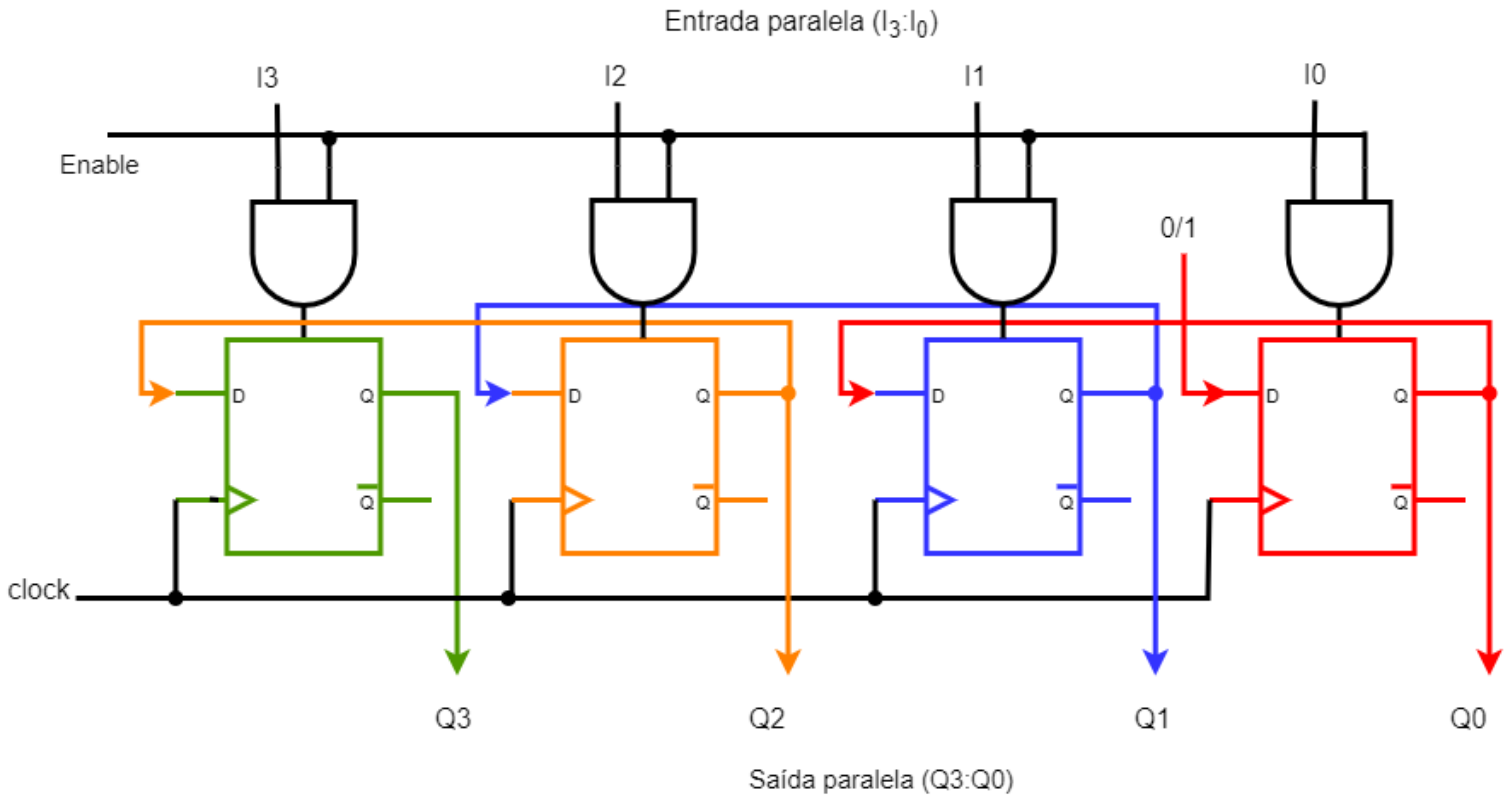
- Exemplo de PIPO de 4 bits à direita: divisão por 2.
- Preservação de sinal: deslocar com '0' se a palavra for positiva. Deslocar com '1' se a palavra for negativa.



D(3)	Q3	Q2	Q1	Q0	Resto	Dec
x	1	0	1	0	x	-6
1	1	1	0	1	0	-3
1	1	1	1	0	1	-2
1	1	1	1	1	0	-1

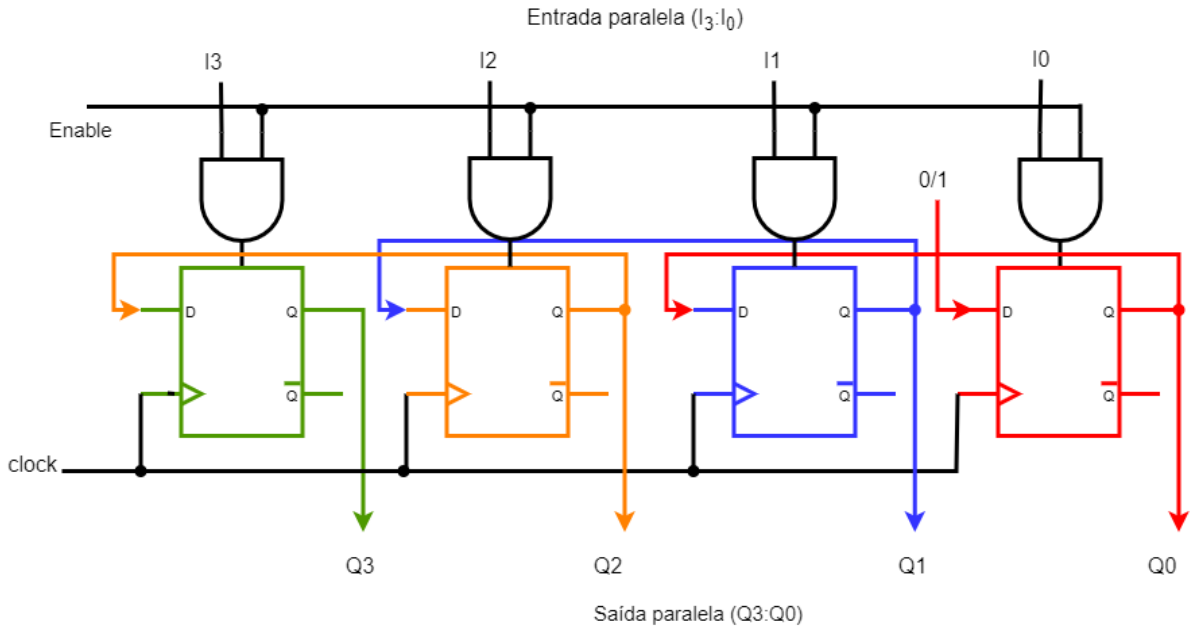
Registrador PIPO (entrada paralela - saída paralela)

- Exemplo de PIPO de 4 bits à esquerda
- Deslocamento à esquerda com '0': multiplicação por 2.



Registrador PIPO (entrada paralela - saída paralela)

- Exemplo de PIPO de 4 bits à esquerda.
- Deslocamento à esquerda com '0': multiplicação por 2. O bit descartado pode ser usado como carry.

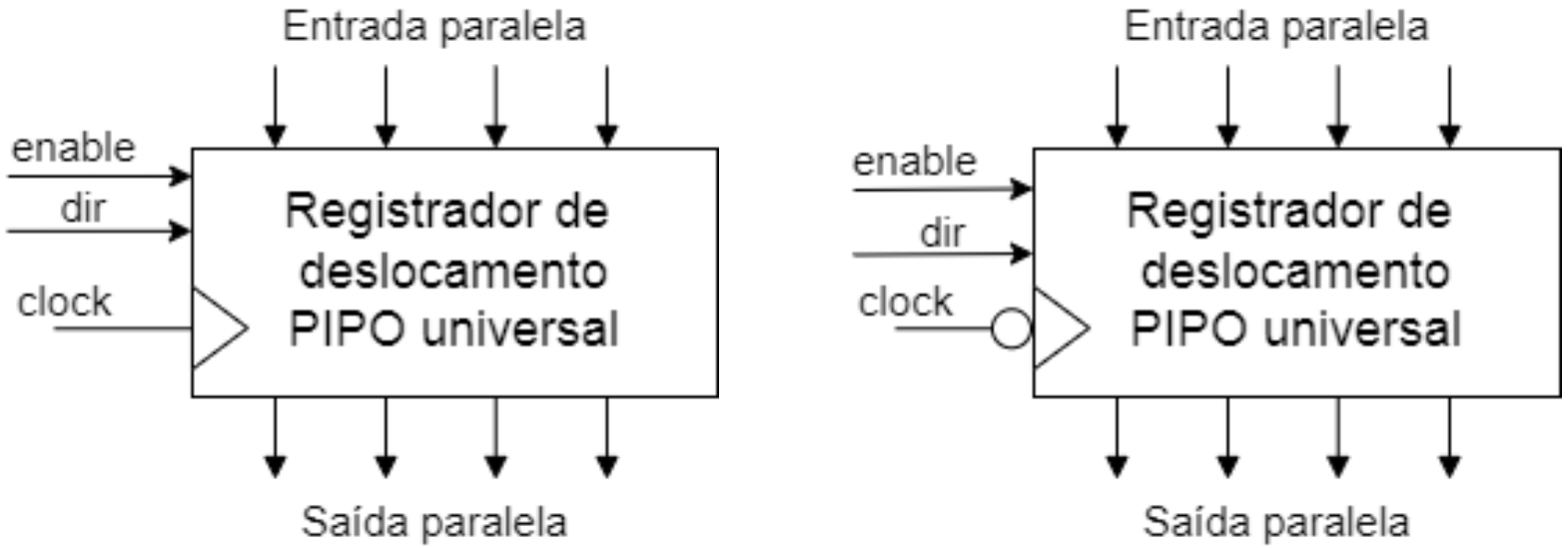


Tempo

D(0)	Q3	Q2	Q1	Q0	Dec
x	0	0	1	1	3
0	0	1	1	0	6
0	1	1	0	0	12
0	1	0	0	0	8, C=1 (24)

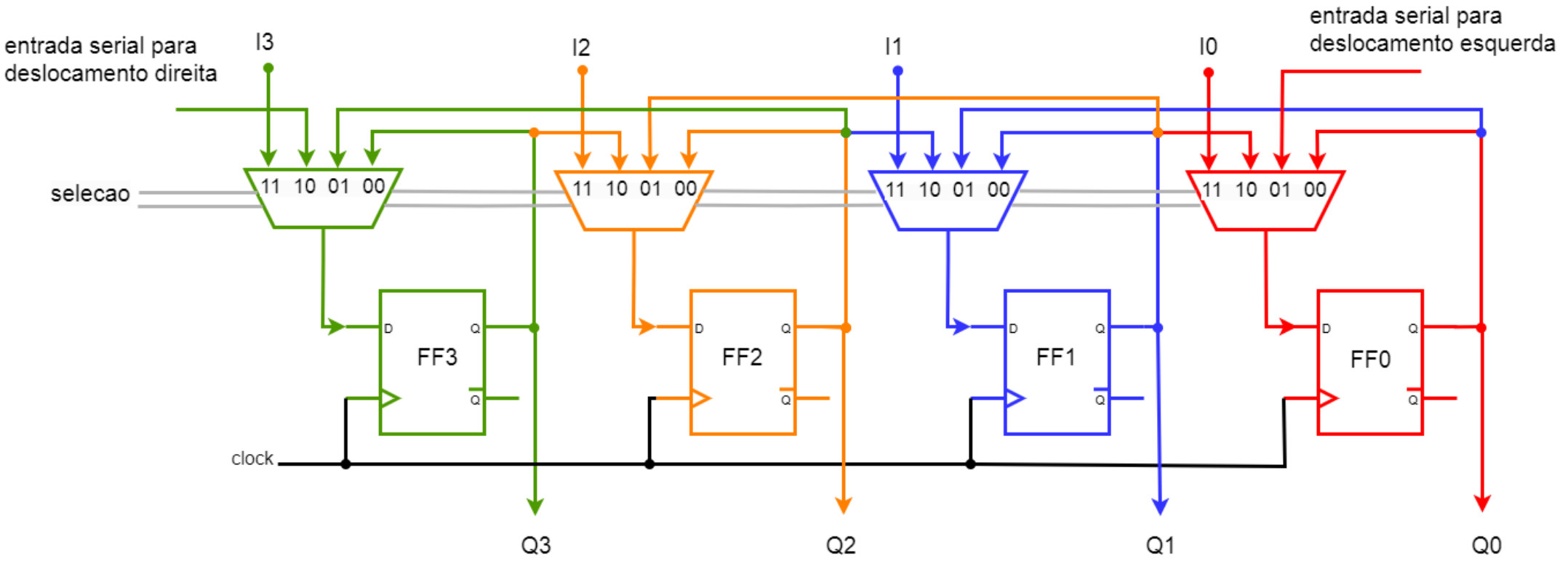
Registrador PIPO universal

- Entrada paralela - saída paralela
- A entrada dir permite escolher a direção do deslocamento (dir='0' : esquerda; dir='1': direita).
- Também é chamado de registrador de deslocamento universal.



Registrador PIPO universal

- Utiliza um multiplexador 4x1 na entrada de dado de cada Flip-Flop para direcionar a informação.



Exercícios registradores de deslocamento

- Exercício 1: Desenhe o esquemático de um registrador SIPO de 8 bits com deslocamento à esquerda.
- Exercício 2: Desenhe o esquemático de um registrador PISO de 8 bits com deslocamento à esquerda. Inclua entrada Reset para apagar o conteúdo do registrador.
- Exercício 3: Desenhe o esquemático de um registrador SIPO de 8 bits com deslocamento à esquerda e à direita. Inclua a entrada *dir* para selecionar a direção de deslocamento. Se *dir*='0' desloca à direita, caso contrário, desloca à esquerda.
- Exercício 4: Desenhe o esquemático de um registrador PISO de 8 bits com deslocamento à esquerda e à direita. Inclua a entrada *dir* para selecionar a direção de deslocamento. Se *dir*='0' desloca à direita, caso contrário, desloca à esquerda.

Página em branco.

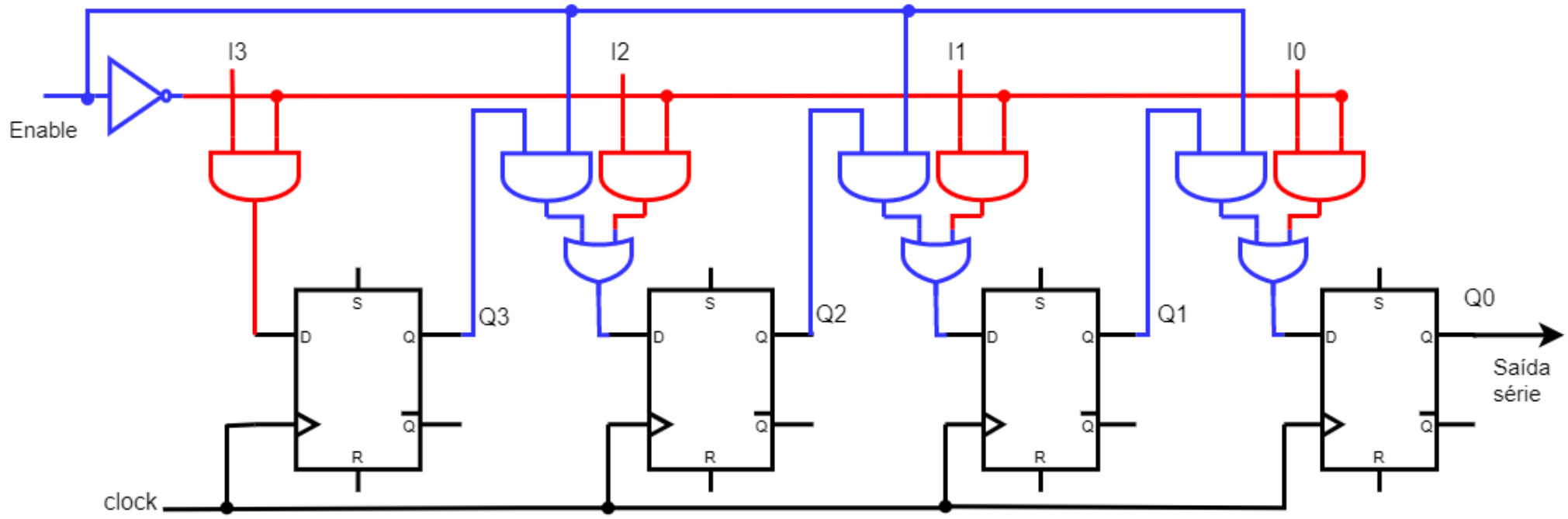
Aula 2
Implementação em HDL de registradores de deslocamento

Implementação VHDL de registrador PISO de *N* bits

Exemplo 1: implementar um registrador PISO de *N* bits. Use uma entrada genérica para indicar o número de bits. O número máximo de bits deve ser de 32. Use uma entrada 'dir' para indicar a direção de deslocamento, uma entrada 'load' para carregar a palavra e um reset assíncrono para apagar o registrador.

Implementação HDL de registrador PISO de N bits

Lembrando a aula passada: arquitetura de um PISO de 4 bits.



Implementação Verilog de registrador PISO de N bits

```
1   module PISO_reg #(parameter num_bits = 8) (
2       input reset,
3       input clk,
4       input load,
5       input dir,
6       input [num_bits-1:0] data_in,
7       output data_out
8   );
9
10      // Internal register to hold the data
11      reg [num_bits-1:0] reg_data = 0;
12
13      // Assign the output data based on the direction
14      assign data_out = (dir == 1'b0) ? reg_data[0] : reg_data[num_bits-1];
```

Implementação Verilog de registrador PISO de N bits

```
16      // Process to handle the data movement
17      always @(posedge clk or posedge reset) begin
18          if (reset) begin
19              reg_data <= 0;  // Reset the register to 0
20          end else if (load) begin
21              reg_data <= data_in;  // Load the input data into the register
22          end else if (dir) begin
23              reg_data <= {reg_data[num_bits-2:0], 1'b0};  // Shift left (with a 0 inserted)
24          end else begin
25              reg_data <= {1'b0, reg_data[num_bits-1:1]};  // Shift right (with a 0 inserted
26          end
27      end
28
29  endmodule
```

Implementação VHDL de registrador PISO de N bits

```
14  library IEEE;
15  use IEEE.STD_LOGIC_1164.ALL;
16  use IEEE.STD_LOGIC_UNSIGNED.ALL;
17  use IEEE.NUMERIC_STD.ALL;
18
19  entity PISO_reg is
20      generic(num_bits : integer range 1 to 32);
21      Port ( reset : in STD_LOGIC;
22             clk : in STD_LOGIC;
23             load : in STD_LOGIC;
24             dir : in STD_LOGIC;
25             data_in : in STD_LOGIC_VECTOR (num_bits-1 downto 0);
26             data_out : out STD_LOGIC);
27  end PISO_reg;
28
29  architecture Behavioral of PISO_reg is
30
31  signal reg : unsigned(num_bits-1 downto 0):= (others=>'0');
```

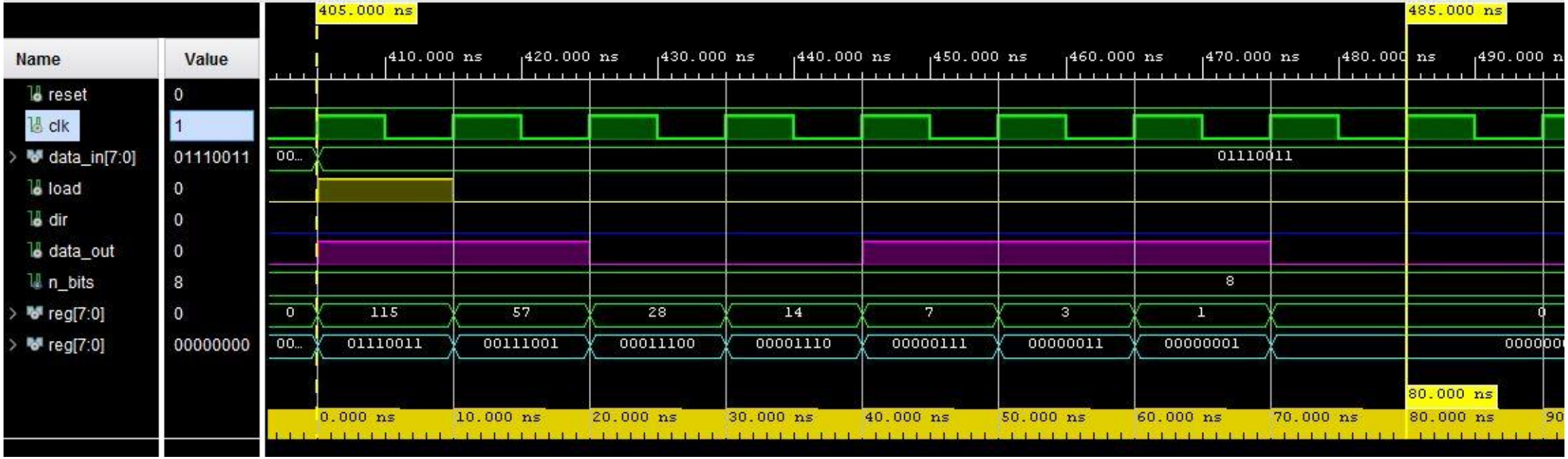
Implementação VHDL de registrador PISO de N bits

```
35      with dir select
36         data_out <= reg(0) when '0',
37                     reg(num_bits-1) when others;
38
39      process(clk,reset)
40      begin
41          if reset='1' then
42              reg <= (others=>'0');
43          elsif rising_edge(clk) then
44              if load = '1' then
45                  reg <= unsigned(data_in);
46              elsif dir='1' then
47                  reg <= reg(num_bits-2 downto 0) & '0'; -- deslocamento a esquerda
48                  --reg <= shift_left(reg,1);             -- solucao alternativa
49              elsif dir='0' then
50                  reg <= '0'& reg(num_bits-1 downto 1);  -- deslocamento a direita
51                  --reg <= shift_right(reg,1);            -- solucao alternativa
52              end if;
53          end if;
54      end process;
55
56  end Behavioral;
```

Registrador PISO: testbench para 8 bits.

```
49      uut: PISO_reg
50      generic map(num_bits => n_bits)
51      port map(
52          reset       => reset,
53          clk         => clk,
54          load        => load,
55          dir         => dir,
56          data_in     => data_in,
57          data_out    => data_out);
58
59      clk <= not clk after 5 ns;
60      reset <= '0', '1' after 15 ns, '0' after 35 ns;
61      load  <= '0', '1' after 45 ns, '0' after 55 ns,
62                   '1' after 145 ns, '0' after 155 ns,
63                   '1' after 405 ns, '0' after 415 ns,
64                   '1' after 515 ns, '0' after 525 ns,
65                   '1' after 805 ns, '0' after 815 ns,
66                   '1' after 915 ns, '0' after 925 ns;
67      dir  <= '0', '1' after 45 ns, '0' after 145 ns,
68                   '0' after 405 ns, '1' after 515 ns,
69                   '1' after 805 ns, '0' after 915 ns;
70      data_in <= "00000000", "00001011" after 45 ns, "01110011" after 405 ns, "01110001" after 805 ns;
```

Registrador PISO de 8 bits: simulação, desloca direita

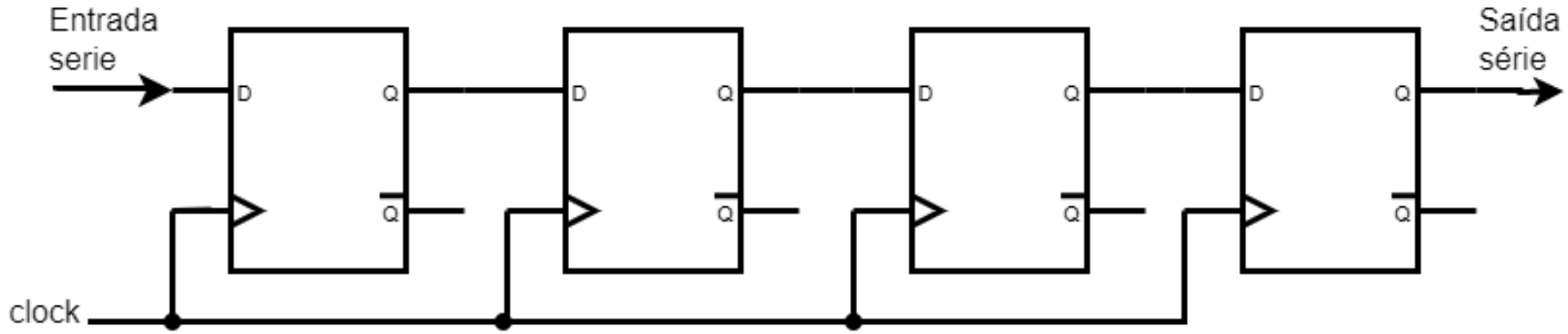


Registrador PISO de 8 bits: simulação, desloca esquerda

Página em branco.

Implementação VHDL de registrador SISO de *N* bits

Lembrando a aula passada: exemplo de SISO de 4 bits.



Implementação Verilog de registrador SISO de N bits

```
1   module SISO_reg #(parameter num_bits = 8) (
2       input reset,
3       input clk,
4       input enable,
5       input dir,
6       input data_in,
7       output data_out
8   );
9
10      // Internal register to hold the data
11      reg [num_bits-1:0] reg_data = 0;
12
13      // Assign the output data based on direction
14      assign data_out = (dir == 1'b0) ? reg_data[0] : reg_data[num_bits-1];
```

Implementação Verilog de registrador SISO de N bits

```
16      // Process to handle the data movement
17      always @(posedge clk or posedge reset) begin
18          if (reset) begin
19              reg_data <= 0;  // Reset the register to 0
20          end else if (enable) begin
21              if (dir) begin
22                  reg_data <= {reg_data[num_bits-2:0], data_in};  // Shift left
23              end else begin
24                  reg_data <= {data_in, reg_data[num_bits-1:1]};  // Shift righ
25              end
26          end
27      end
28
29  endmodule
```

Implementação VHDL de registrador SISO de N bits

```
14  library IEEE;
15  use IEEE.STD_LOGIC_1164.ALL;
16  use IEEE.STD_LOGIC_UNSIGNED.ALL;
17  use IEEE.NUMERIC_STD.ALL;
18
19  entity SISO_reg is
20      generic(num_bits : integer range 1 to 32);
21      Port ( reset : in STD_LOGIC;
22             clk : in STD_LOGIC;
23             enable : in STD_LOGIC;
24             dir : in STD_LOGIC;
25             data_in : in STD_LOGIC;
26             data_out : out STD_LOGIC);
27  end SISO_reg;
28
29  architecture Behavioral of SISO_reg is
30
31  signal reg : unsigned(num_bits-1 downto 0):= (others=>'0');
```

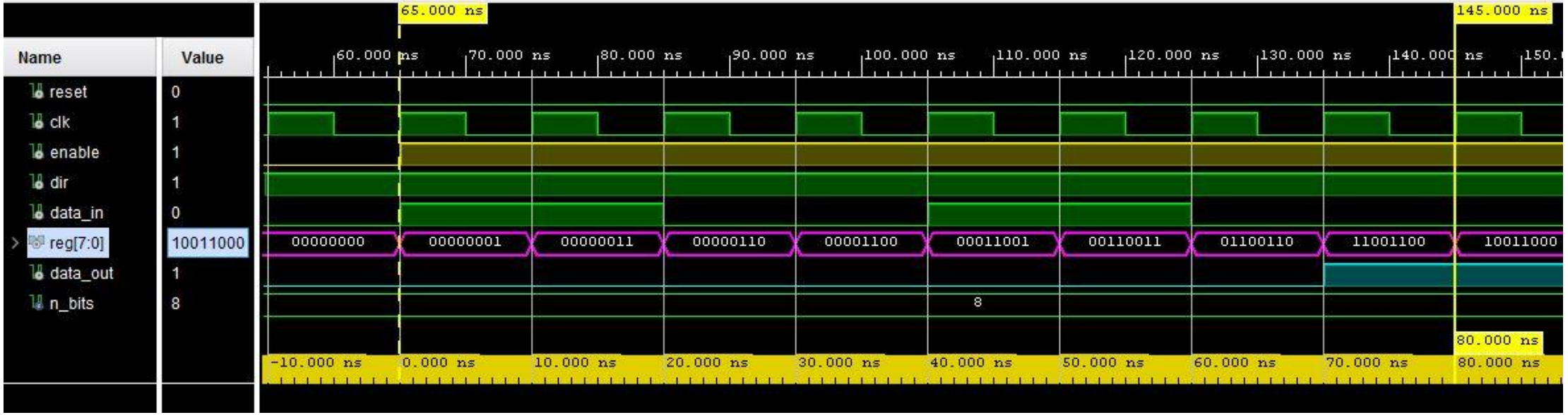
Implementação VHDL de registrador SISO de N bits

```
35      with dir select
36         data_out <= reg(0) when '0',
37                     reg(num_bits-1) when others;
38
39      process(clk,reset)
40      begin
41          if reset='1' then
42              reg <= (others=>'0');
43          elsif rising_edge(clk) then
44              if enable = '1' then
45                  if dir='1' then
46                      reg <= reg(num_bits-2 downto 0) & data_in; -- deslocamento a esquerda
47                      --reg <= shift_left(reg,1);                -- solucao alternativa
48                  else
49                      reg <= data_in & reg(num_bits-1 downto 1);  -- deslocamento a direita
50                      --reg <= shift_right(reg,1);                -- solucao alternativa
51                  end if;
52              end if;
53          end if;
54      end process;
```

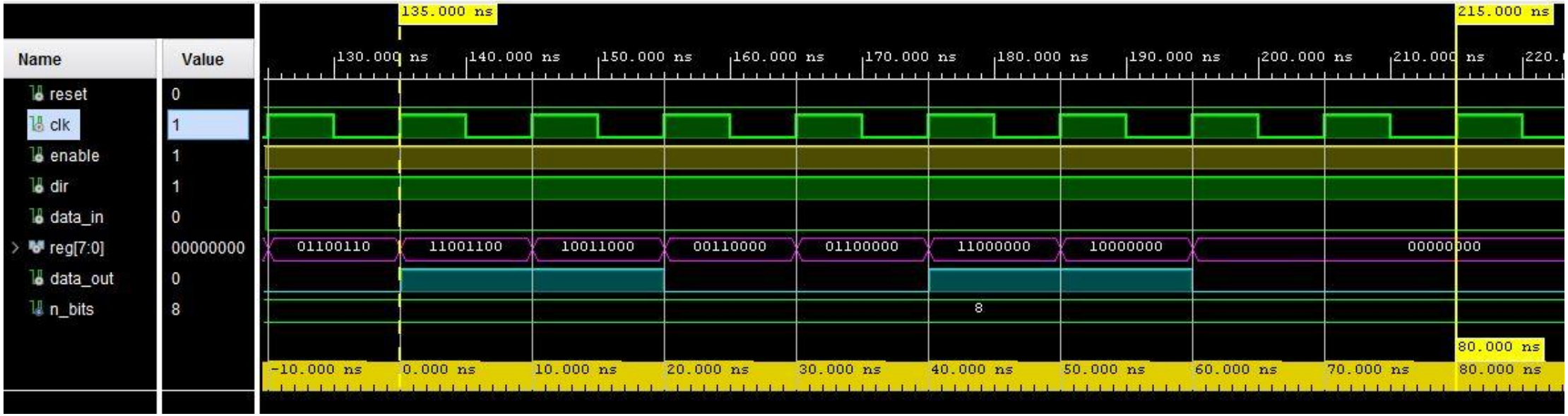
Registrador SISO. Testbench para SISO de 8 bits.

```
55      uut: SISO_reg
56      generic map(num_bits => n_bits)
57      port map(
58          reset       => reset,
59          clk         => clk,
60          enable      => enable,
61          dir         => dir,
62          data_in     => data_in,
63          data_out    => data_out);
64
65      clk <= not clk after 5 ns;
66      reset <= '0', '1' after 15 ns, '0' after 35 ns;
67      enable <= '0', '1' after 65 ns;
68      dir  <= '0', '1' after 45 ns, -- 65 ns + 8 clocks para preencher reg
69                   '1' after 145 ns,-- 8 clocks para esvaziar reg
70                   '0' after 245 ns,-- 8 clocks para preencher reg
71                   '0' after 345 ns;-- 8 clocks para esvaziar reg
72      data_in <= '0', '1' after 65 ns, '0' after 85 ns, '1' after 105 ns, '0' after 125 ns, --11001100
73                     '0' after 145 ns, -- espera que esvazie o registrador
74                     '1' after 245 ns, '0' after 255 ns, '1' after 265 ns, '0' after 275 ns,
75                     '1' after 285 ns, '0' after 295 ns, '1' after 305 ns, '0' after 315 ns; --01010101
```

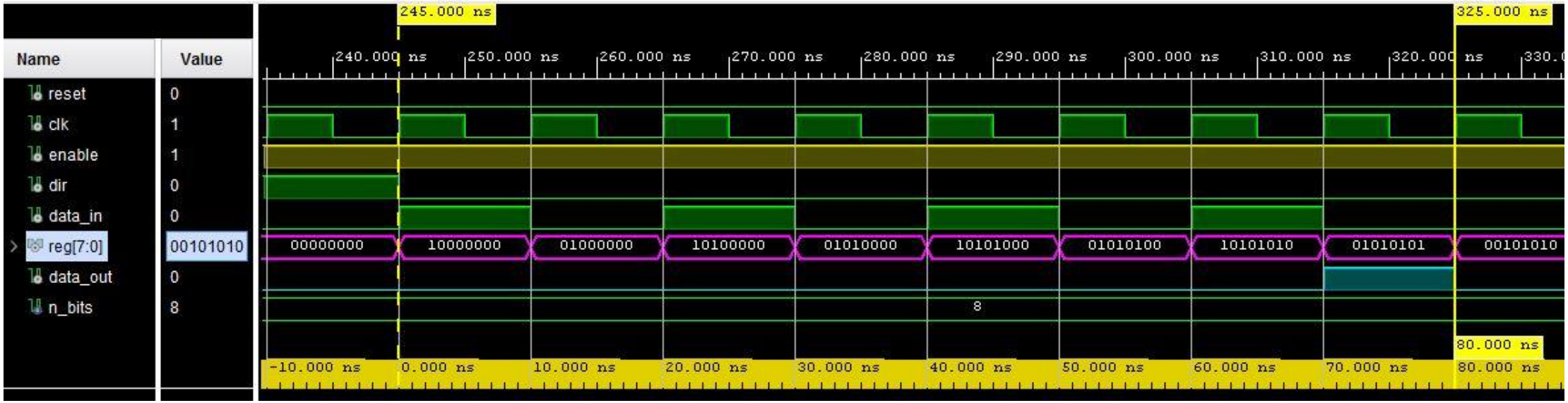
Registrador SISO de 8 bits. Simulação: desloca esquerda.



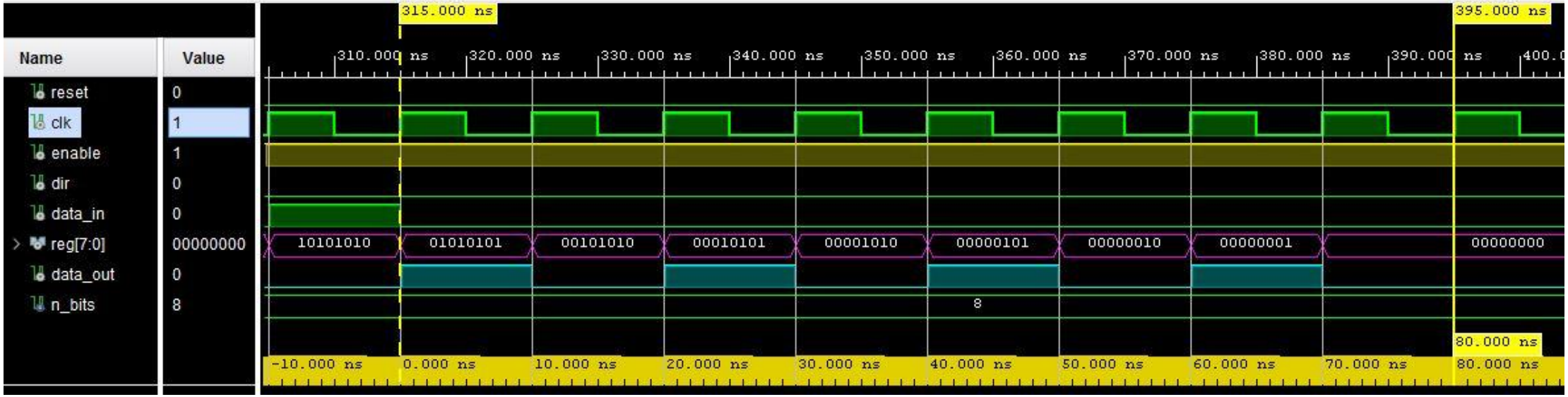
Registrador SISO de 8 bits. Simulação: desloca esquerda.



Registrador SISO de 8 bits. Simulação: desloca direita.



Registrador SISO de 8 bits. Simulação: desloca direita.



Página em branco.

Implementação de registrador PIPO em anel de *N* bits

Exemplo 3: implementar um registrador PIPO de *N* bits com deslocamento em anel. Use uma entrada genérica para indicar o número de bits. O número máximo de bits deve ser de 32. Use uma entrada 'dir' para indicar a direção de deslocamento, uma entrada 'load' para carregar a palavra e um reset assíncrono para apagar o registrador.

Deslocamento em anel:

- À direita: o LS bit entra no MS bit e desloca à direita a palavra.
- À esquerda: o MS bit entra no LS bit e desloca à esquerda a palavra.

Implementação Verilog de registrador PIPO em anel

```verilog
module PIPO_reg_anel #(parameter num_bits = 32)(
    input reset,
    input clk,
    input load,
    input dir,
    input [num_bits-1:0] data_in,
    output [num_bits-1:0] data_out
);

    reg [num_bits-1:0] reg_data;

    always @(posedge clk or posedge reset) begin
        if (reset)
            reg_data <= {num_bits{1'b0}};
        else if (load)
            reg_data <= data_in;
        else if (dir)
            // Shift left with wrap-around
            reg_data <= {reg_data[num_bits-2:0], reg_data[num_bits-1]};
        else
            // Shift right with wrap-around
            reg_data <= {reg_data[0], reg_data[num_bits-1:1]};
    end

    assign data_out = reg_data;

endmodule
```

Implementação VHDL de registrador PIPO em anel

```
14  library IEEE;
15  use IEEE.STD_LOGIC_1164.ALL;
16  use IEEE.STD_LOGIC_UNSIGNED.ALL;
17  use IEEE.NUMERIC_STD.ALL;
18
19  entity PIPO_reg_anel is
20      generic(num_bits : integer range 1 to 32);
21      Port ( reset : in STD_LOGIC;
22             clk : in STD_LOGIC;
23             load : in STD_LOGIC;
24             dir : in STD_LOGIC;
25             data_in : in STD_LOGIC_VECTOR (num_bits-1 downto 0);
26             data_out : out STD_LOGIC_VECTOR (num_bits-1 downto 0));
27  end PIPO_reg_anel;
28
29  architecture Behavioral of PIPO_reg_anel is
30
31  signal reg : unsigned(num_bits-1 downto 0):= (others=>'0');
32
33  begin
```

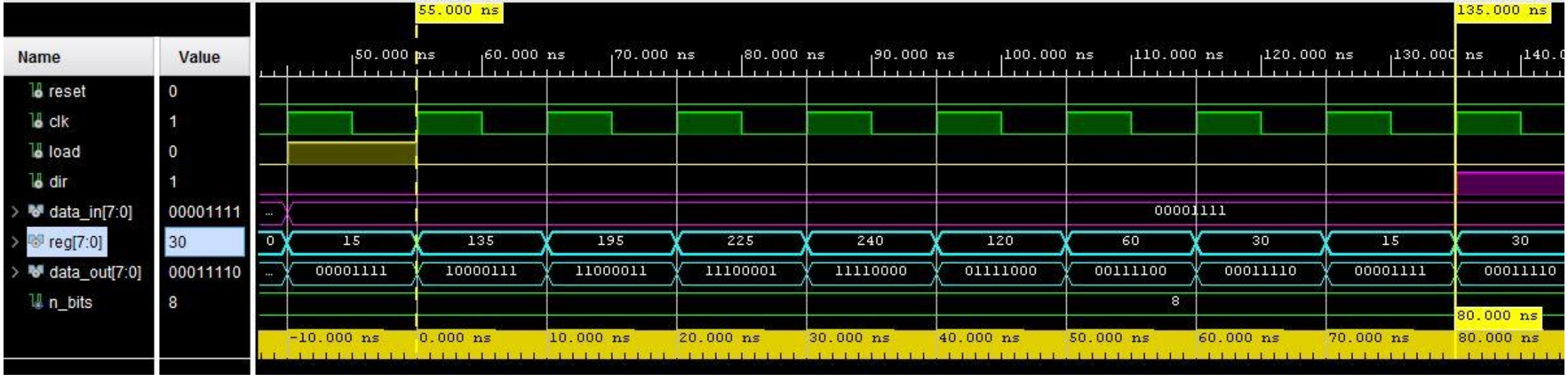
Implementação VHDL de registrador PIPO em anel

```
35      data_out <= std_logic_vector(reg);
36
37      process(clk,reset)
38      begin
39          if reset='1' then
40              reg <= (others=>'0');
41          elsif rising_edge(clk) then
42              if load = '1' then
43                  reg <= unsigned(data_in);
44              elsif dir='1' then
45                  reg <= reg(num_bits-2 downto 0)&reg(num_bits-1); --esquerda
46              elsif dir='0' then
47                  reg <= reg(0)&reg(num_bits-1 downto 1);  --direita
48              end if;
49          end if;
50      end process;
51
52  end Behavioral;
```

Registrador PIPO em anel: testbench

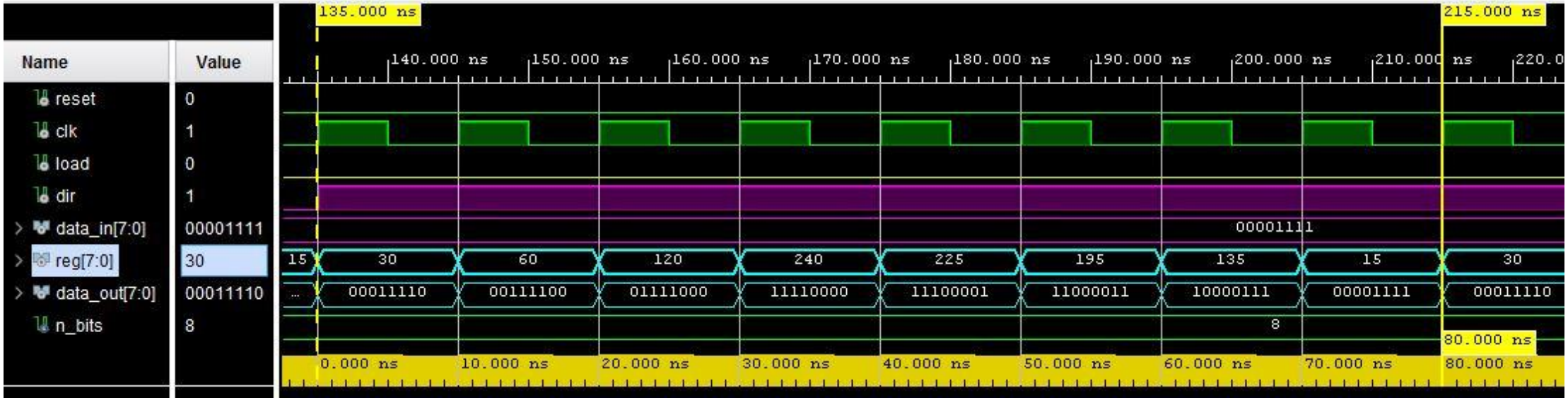
```
43  constant n_bits : integer range 0 to 32 := 8;
44  signal reset,clk,load,dir : std_logic := '0';
45  signal data_in,data_out : std_logic_vector(n_bits-1 downto 0):= (others=>'0');
46
47  begin
48
49      uut: PIPO_reg_anel
50      generic map(num_bits => n_bits)
51      port map(
52          reset       => reset,
53          clk         => clk,
54          load        => load,
55          dir         => dir,
56          data_in     => data_in,
57          data_out    => data_out);
58
59      clk <= not clk after 5 ns;
60      reset <= '0', '1' after 15 ns, '0' after 35 ns;
61      load  <= '0', '1' after 45 ns, '0' after 55 ns;
62      dir   <= '0', '1' after 135 ns;
63      data_in <= "00000000", "00001111" after 45 ns;
64
65  end Behavioral;
```

Registrador PIPO em anel: simulação desloca esquerda



Observe que o registrador de 8 bits é carregado com o valor "00001111" e depois de 8 clocks (80 ns) o registrador volta a conter o mesmo valor.

Registrador PIPO em anel: simulação desloca esquerda



Observe que o registrador de 8 bits é carregado com o valor "00001111" e depois de 8 clocks (80 ns) o registrador volta a conter o mesmo valor.

Exercícios: registradores deslocamento em HDLs

Exercício 1: Realize um testbench em Verilog que instancie dois registradores PISO de N bits, um deve realizar deslocamentos à direita e o outro deve realizar deslocamentos à esquerda. A palavra a ser carregada deve ser comum a ambos registradores. Realize a síntese lógica, obtenha o esquemático RTL e a ocupação de recursos. Verifique o funcionamento a partir de uma simulação comportamental. Repita o testbench e a simulação em VHDL.

Nota: envie para o instrutor os códigos HDLs, testbench, esquemático RTL, ocupação de recursos e prints de simulação.

Exercícios: registradores deslocamento em HDLs

Exemplo 2: implementar um registrador SISO de *N* bits com entrada enable. Use uma entrada genérica para indicar o número de bits. O número máximo de bits deve ser de 32. Use uma entrada 'dir' para indicar a direção de deslocamento, uma entrada 'enable' para carregar a palavra, e um reset assíncrono para apagar o registrador. Realize a síntese lógica, obtenha o esquemático RTL e a ocupação de recursos.

Nota: envie para o instrutor os códigos HDLs, esquemático RTL, e a ocupação de recursos.

Exercícios: registradores deslocamento em HDLs

Exercício 3: Realize um testbench em Verilog e outro em VHDL que instancie dois registradores SISO do exercício anterior. O primeiro deve realizar deslocamentos à direita e o outro deve realizar deslocamentos à esquerda. A palavra a ser carregada deve ser comum a ambos registradores. Realize a síntese lógica, obtenha o esquemático RTL e a ocupação de recursos. Verifique o funcionamento a partir de uma simulação comportamental.

Nota: envie para o instrutor os códigos HDLs, testbench, esquemático RTL, ocupação de recursos e prints de simulação.

Página em branco.

Aula 3
Implementação física de um registrador de deslocamento PIPO (entrada paralela e saída paralela) universal

Implementação física de registrador PIPO

Exercício: implemente em Verilog ou VHDL um registrador PIPO de *N* bits para divisão e multiplicação por potências de dois. O circuito deve usar uma entrada de *reset* para apagar o registrador, uma entrada *enable* para carregar a palavra, e uma entrada *dir* para indicar a direção de deslocamento. Requisitos:

1. Escolha o tamanho em bits *N* dependendo dos leds disponíveis na placa.
2. Utilize um clock de 5 Hz. Para obter esse sinal pode usar um divisor de clock a partir do clock master da placa de desenvolvimento.
3. O deslocamento a esquerda deve ser feito com '0'. O deslocamento a direita deve ser feito com MSB bit a fim de preservar o sinal da palavra.
4. Realize um testbench e verifique o funcionamento do circuito.
5. Realize síntese lógica, analise o diagrama esquemático e consumo de recursos.
6. Conecte os pinos de IO e realize a implementação física do circuito. Teste na placa de desenvolvimento. Caracterize em termos de ocupação de recursos.

Página em branco.

Aula 4
Implementação física de um registrador de deslocamento SIPO (entrada serial e saída paralela) universal

Implementação física de registrador SIPO universal

Exercício: implemente em Verilog ou VHDL um registrador SIPO de *N* bits. O circuito deve usar uma entrada de *reset* para apagar o registrador e uma entrada *dir* para indicar a direção de deslocamento. Requisitos:

1. Escolha o tamanho em bits *N* dependendo dos leds disponíveis na placa.
2. Utilize um clock de 1 Hz. Para obter esse sinal pode usar um divisor de clock a partir do clock master da placa de desenvolvimento.
3. O deslocamento a esquerda deve ser feito com '0'. O deslocamento a direita deve ser feito com MSB bit a fim de preservar o sinal da palavra.
4. Realize um testbench e verifique o funcionamento do circuito.
5. Realize síntese lógica, analise o diagrama esquemático e consumo de recursos.
6. Conecte os pinos de IO e realize a implementação física do circuito. Teste na placa de desenvolvimento. Caracterize em termos de ocupação de recursos.