

Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Engenharia Elétrica - DEE

Nome: Alysson Machado de Oliveira Barbosa

Email: alysson.barbosa@ee.ufcg.edu.br

Disciplina: Laboratório de Circuitos Lógicos

Professora: Fernanda Cecília Correia Lima Loureiro

Experimento 06 - Flip-flops, Registradores e Contadores com HDL (Verilog)

Objetivo Geral

Este experimento consiste na realização de quatro partes específicas e possui como objetivo geral o estudo de Flip-Flops, Registradores e Contadores, bem como o projeto e implementação destes circuitos lógicos utilizando Verilog. Para tanto, é realizada a implementação e a verificação do funcionamento correspondente aos seguintes experimentos específicos:

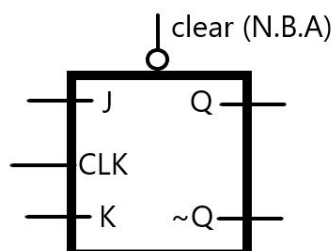
- **Flip-flop JK;**
- **Registrador de deslocamento com entradas paralela e saídas paralela e serial;**
- **Contador assíncrono crescente/decrescente;**
- **Contador binário síncrono;**

Objetivo 1

Especificação e implementação de um Flip-Flop JK, com entrada clear assíncrona, com o projeto realizado utilizando a linguagem Verilog para implementação na Placa DE2.

❖ A - I)

Figura 01 - Bloco lógico de um flip-flop JK (entrada clear N.B.A assíncrono)



Fonte: Imagem Autoral.

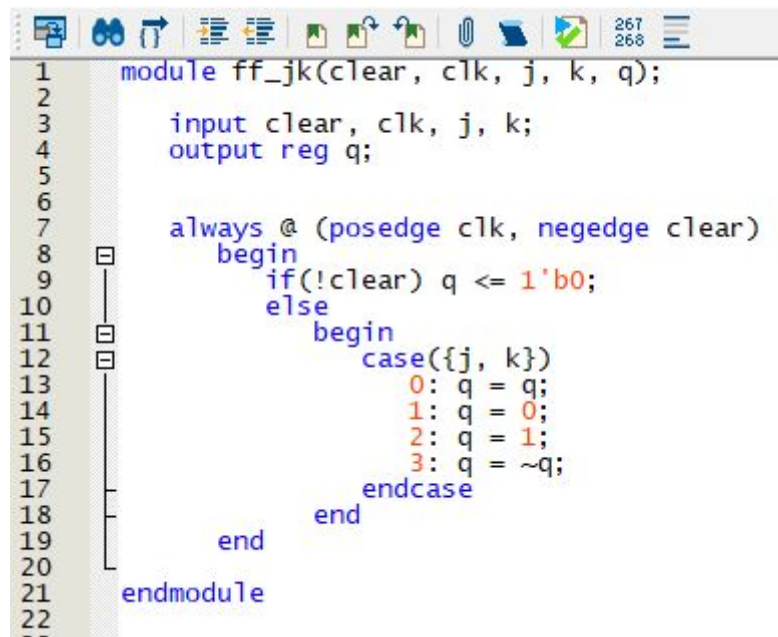
❖ A-II)

Observando o bloco lógico acima, é possível identificar que, especificamente, a combinação $J = 1, K = 0$ é um comando para ativar (set) a saída do flip-flop. A combinação $J = 0, K = 1$ é um comando para desativar (reset) a saída do flip-flop; e a combinação $J = K = 1$ é um comando para inverter o flip-flop, trocando o sinal de saída pelo seu complemento. Fazendo $J = K$ o flip-flop J-K se torna um flip-flop T.

❖ B-I)

O projeto do Flip-Flop JK foi realizado com as seguintes entradas: CLOCK (CK) sensível à borda de subida e CLEAR (CLR) assíncrono e Nível Baixo Ativo (NBA). Tal módulo foi arquitetado utilizando a linguagem de Descrição de Hardware Verilog. Observe o código abaixo:

Figura 02 - Implementação de um Flip-Flop JK em Verilog



```
1 module ff_jk(clear, clk, j, k, q);
2
3     input clear, clk, j, k;
4     output reg q;
5
6
7     always @ (posedge clk, negedge clear)
8     begin
9         if(!clear) q <= 1'b0;
10        else
11            begin
12                case({j, k})
13                    0: q = q;
14                    1: q = 0;
15                    2: q = 1;
16                    3: q = ~q;
17                endcase
18            end
19        end
20    endmodule
21
22
23
```

Fonte: Imagem Autoral.

Observe que o módulo implementado é sensível a borda de subida e a entrada clear (N.B.A), responsável por tornar a saída nível baixo. Além disso, a entrada clear ela é assíncrona em relação ao clock.

Para testar o módulo feito, vamos implementá-lo em uma placa FPGA. Observe as implementações abaixo:

Figura 03 - Criando um módulo de teste para o Flip-Flop JK

```

1  module mod_test1(SW, LEDR);
2
3      input [3:0]SW;
4      output LEDR;
5
6      ff_jk ff1(SW[3], SW[2], SW[1], SW[0], LEDR);
7
8  endmodule
9
10

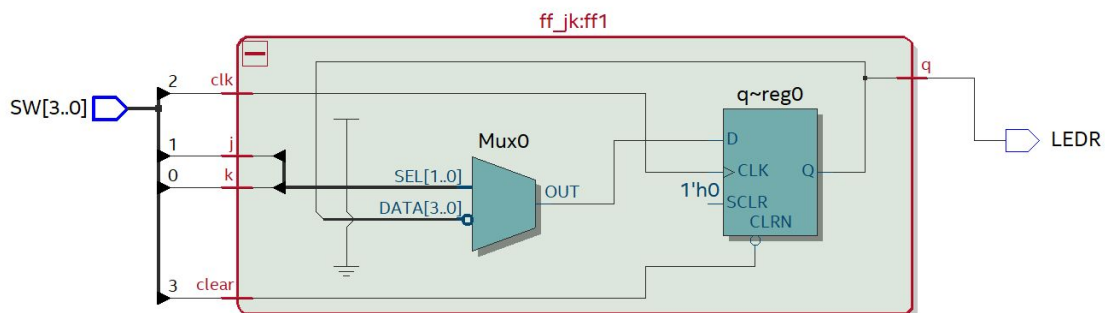
```

Fonte: Imagem Autoral.

Descrição das entradas e saídas do módulo de teste:

- SW[3] = Referente a entrada clear (N.B.A);
- SW[2] = Referente ao clock;
- SW[1] = Referente a entrada J do Flip-Flop;
- SW[0] = Referente a entrada K do Flip-Flop;
- LEDR = Dado de saída do Flip-Flop;

Figura 04 - Netlist do módulo de teste do Flip-Flop JK



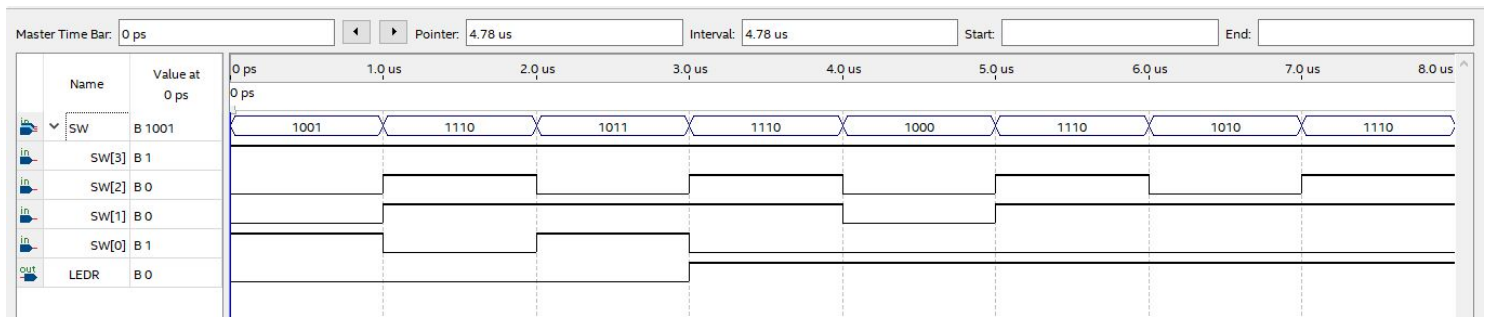
Fonte: Imagem Autoral.

❖ C)

Abaixo é possível analisar a tabela verdade deste Flip-Flop JK, juntamente com a WaveForm do módulo de teste. Observe como o andamento da WaveForm é condizente com a tabela Verdade do Flip-Flop JK.

Clear	J	K	Q
0	X	X	0
1	0	0	Q(anterior)
1	0	1	0
1	1	0	1
1	1	1	~Q(anterior)

Figura 05 - Análise da WaveForm



Fonte: Imagem Autoral.

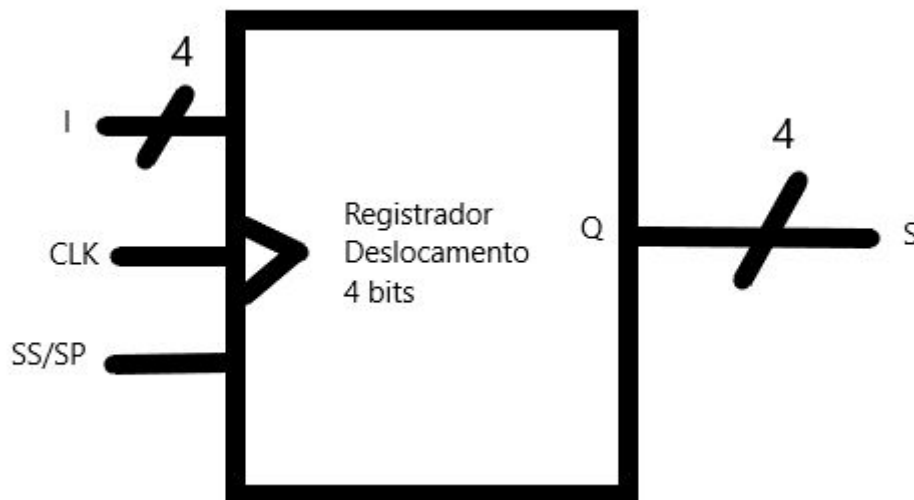
Objetivo 2

Especificação e implementação de um Registrador de deslocamento para a direita, com entrada paralela e saídas paralela e serial, de 4 bits, com o projeto realizado utilizando a linguagem Verilog para implementação na Placa DE2.

❖ A-I)

Foi realizado a especificação de um Registrador de entrada paralela e saídas paralela e serial, de 4 bits, com carregamento paralelo síncrono, e deslocamento para a direita. Observe o bloco lógico abaixo do circuito implementado:

Figura 06 - Bloco Lógico do Registrador de Deslocamento 4 bits



Fonte: Imagem Autoral.

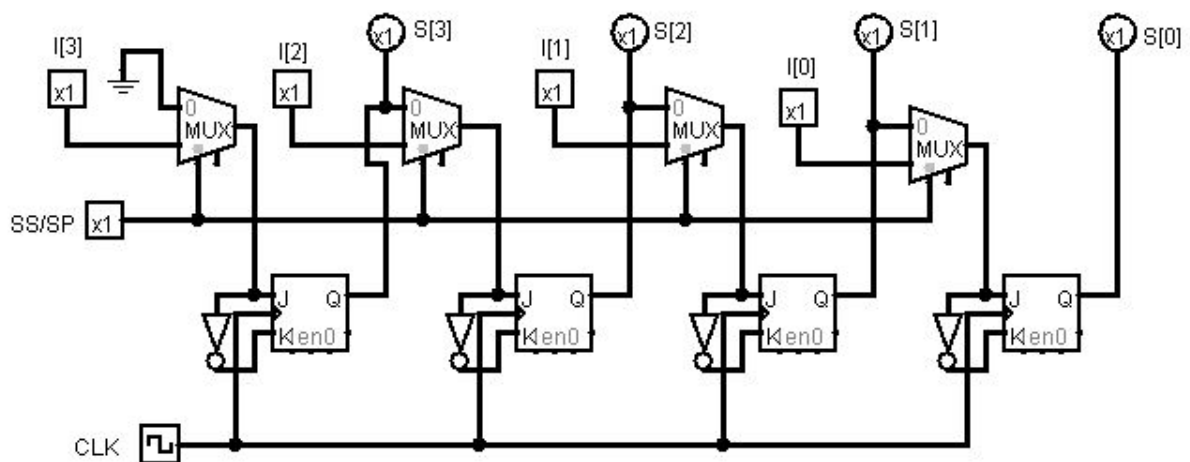
❖ A-II)

Observe a descrição das entradas de dados e saída:

- CLK = Entrada de clock do circuito;
- I[3:0] = Entrada de dados paralela;
- SS/SP = Enable que permuta entre saída serial e saída paralela;
- S[3:0] = Saída dos dados paralela (SS/SP = 1) ou serial (SS/SP = 0);

❖ A-III)

Figura 07 - Diagrama Lógico do Registrador de Deslocamento 4 bits

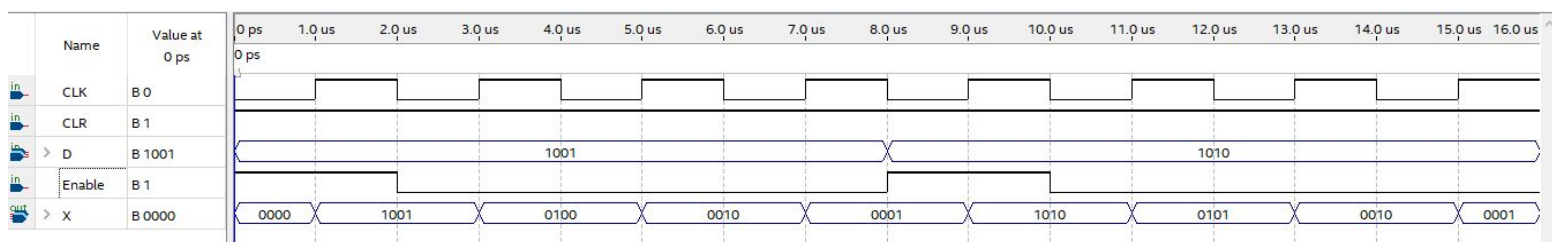


Fonte: Imagem Autoral.

❖ A-IV)

Observe o diagrama de tempo abaixo, ele ilustra a ideia de ter um registrador de deslocamento com saída serial e paralela, a partir do dado de entrada de forma paralela.

Figura 08 - Diagrama de Tempo do Registrador de Deslocamento de 4 bits

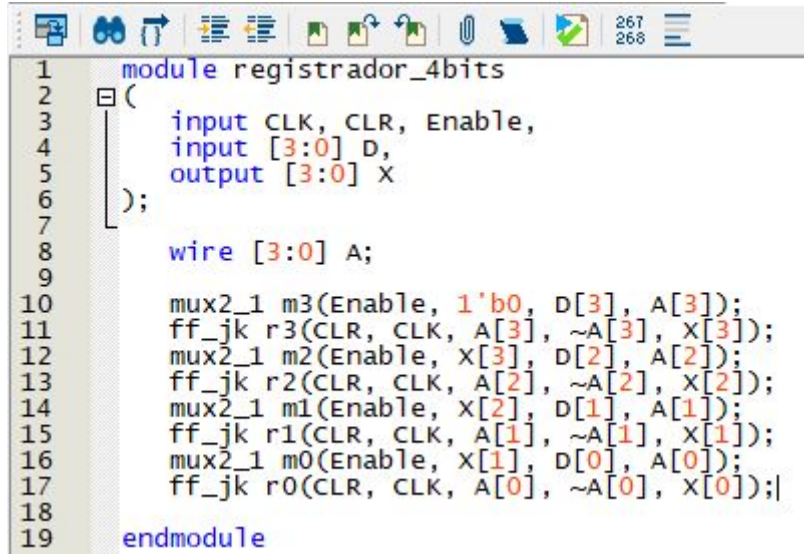


Fonte: Imagem Autoral.

B)

Agora que o circuito está estabelecido, foi realizada a seguinte arquitetura desse circuito usando a linguagem de descrição de hardware verilog:

Figura 09 - Implementação do Registrador de 4 bits em Verilog

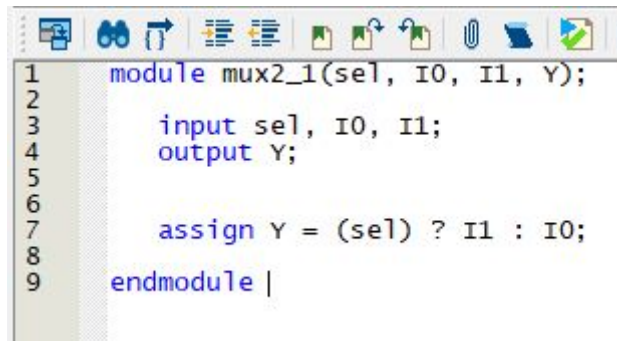
The image shows a screenshot of a Verilog code editor. The code defines a module named 'registrador_4bits'. It has three inputs: 'CLK', 'CLR', and 'Enable', and one input bus '[3:0] D'. It has one output bus '[3:0] X'. Inside the module, a wire '[3:0] A' is declared. The logic consists of four 2-to-1 multiplexers (mux2_1 m0 to m3) and four J-K flip-flops (ff_jk r0 to r3). Each flip-flop's 'D' input is connected to the output of a multiplexer. The multiplexers select between the current value of 'A' and the input 'D' based on the 'Enable' signal. The flip-flops are clocked by 'CLK' and cleared by 'CLR'. The output 'X' is connected to the output of the flip-flops.

```
1 module registrador_4bits
2 (
3     input CLK, CLR, Enable,
4     input [3:0] D,
5     output [3:0] X
6 );
7
8     wire [3:0] A;
9
10    mux2_1 m3(Enable, 1'b0, D[3], A[3]);
11    ff_jk r3(CLR, CLK, A[3], ~A[3], X[3]);
12    mux2_1 m2(Enable, X[3], D[2], A[2]);
13    ff_jk r2(CLR, CLK, A[2], ~A[2], X[2]);
14    mux2_1 m1(Enable, X[2], D[1], A[1]);
15    ff_jk r1(CLR, CLK, A[1], ~A[1], X[1]);
16    mux2_1 m0(Enable, X[1], D[0], A[0]);
17    ff_jk r0(CLR, CLK, A[0], ~A[0], X[0]);
18
19 endmodule
```

Fonte: Imagem Autoral.

Observe que houve a necessidade de implementar um Multiplexador para realizar o controle da saída serial e saída paralela.

Figura 10 - Implementação de um Multiplexador com operador ternário

The image shows a screenshot of a Verilog code editor. The code defines a module named 'mux2_1'. It has three inputs: 'sel', 'I0', and 'I1', and one output 'Y'. The logic is implemented using a ternary operator: 'assign Y = (sel) ? I1 : I0;'. The module is enclosed in 'module' and 'endmodule' keywords.

```
1 module mux2_1(sel, I0, I1, Y);
2
3     input sel, I0, I1;
4     output Y;
5
6
7     assign Y = (sel) ? I1 : I0;
8
9 endmodule |
```

Fonte: Imagem Autoral.

Para testar o módulo do registrador de deslocamento de 4 bits, foi implementado um módulo de teste compatível com o FPGA. Assim como está na figura abaixo.

Figura 11 - Módulo de teste para o registrador de deslocamento de 4 bits

```

1  module mod_test2(SW, LEDR);
2
3      input [6:0]SW;
4      output [3:0]LEDR;
5
6      registrador_4bits rd1(SW[6], SW[5], SW[4], SW[3:0], LEDR);
7
8  endmodule
9

```

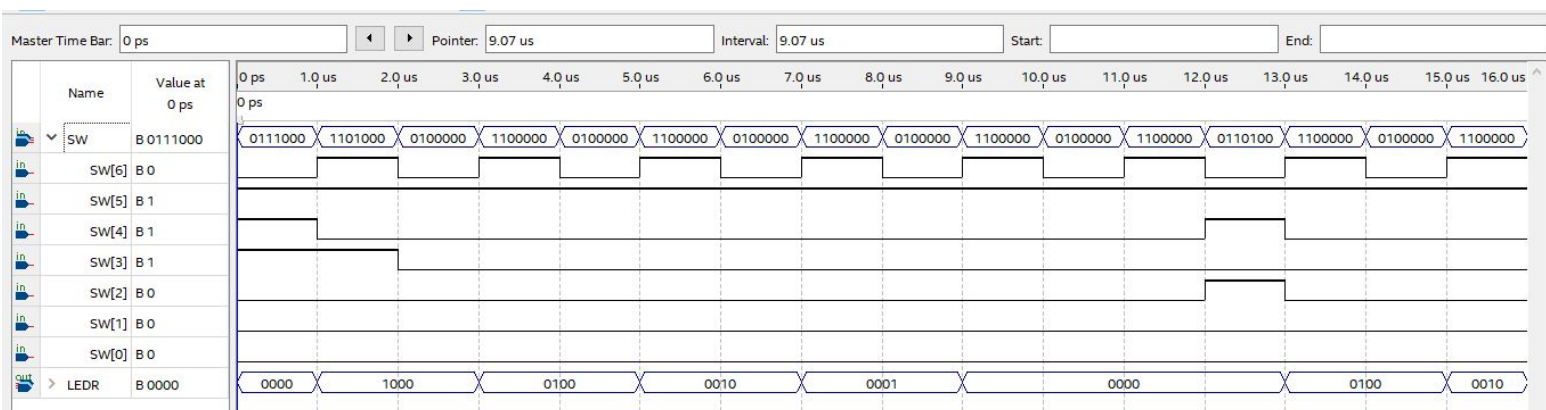
Fonte: Imagem Autoral.

Descrição das entradas e saídas do módulo de teste:

- SW[6] = Entrada do clock do circuito;
- SW[5] = Entrada clear (N.B.A);
- SW[4] = Enable que permuta entre saída serial e saída paralela;
- SW[3:0] = Entrada de Dados paralela;
- LEDR[3:0] = Saída dos dados paralela (SW[4] = 1) ou serial (SW[4] = 0);

Abaixo é possível visualizar o diagrama de tempo para esse módulo de teste. Nele é possível observar quando o registrador tem saída serial e quando tem saída paralela.

Figura 12 - Diagrama de tempo do módulo de teste para o registrador



Fonte: Imagem Autoral.

❖ C)

Abaixo é possível visualizar a tabela verdade para o registrador de deslocamento de 4 bits, implementado com flip-flops jk, clock em borda de subida e clear com nível baixo ativo.

❖ A-II)

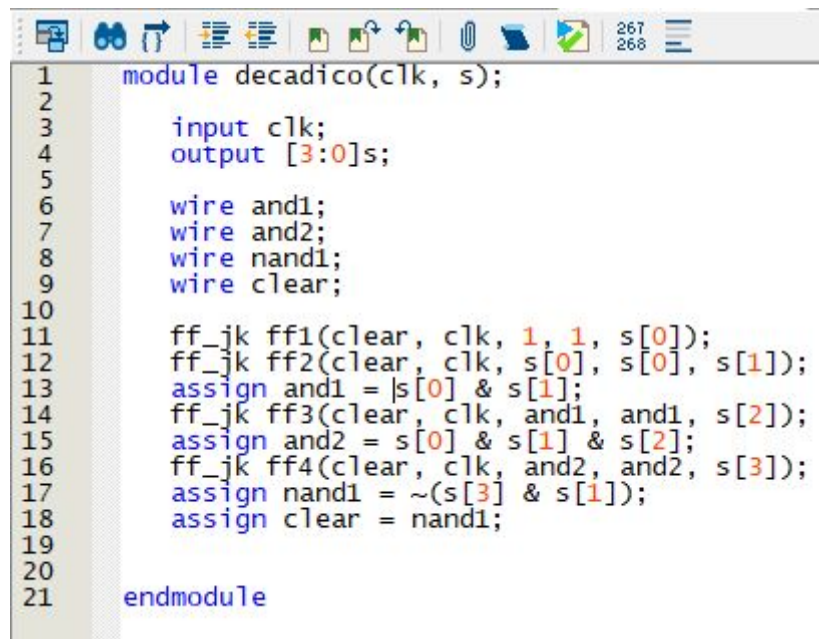
Descrição das entradas de dados e saída:

- CLK = Clock do circuito em nível alto ativo;
- S[3:0] = Saída de dados do circuito [0000, 1001];

❖ B)

Foi implementado o contador decádico síncrono na linguagem de descrição de hardware verilog, assim com está no módulo abaixo.

Figura 14 - Contador Decádico Síncrono em verilog

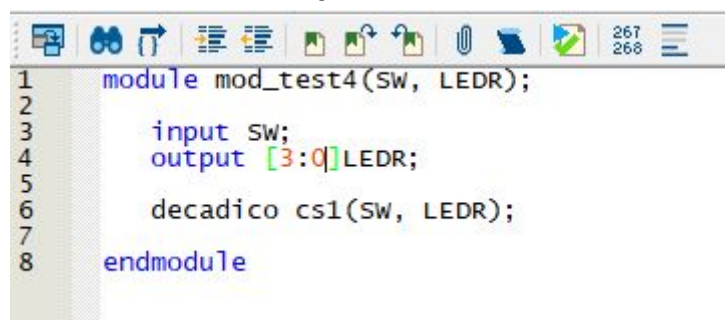


```
1 module decadico(clk, s);
2
3     input clk;
4     output [3:0]s;
5
6     wire and1;
7     wire and2;
8     wire nand1;
9     wire clear;
10
11     ff_jk ff1(clear, clk, 1, 1, s[0]);
12     ff_jk ff2(clear, clk, s[0], s[0], s[1]);
13     assign and1 = s[0] & s[1];
14     ff_jk ff3(clear, clk, and1, and1, s[2]);
15     assign and2 = s[0] & s[1] & s[2];
16     ff_jk ff4(clear, clk, and2, and2, s[3]);
17     assign nand1 = ~(s[3] & s[1]);
18     assign clear = nand1;
19
20
21 endmodule
```

Fonte: Imagem Autoral.

Para testar o módulo, foi implementado um módulo de teste compatível com uma placa FPGA.

Figura 15 - Módulo de teste para o contador síncrono decádico



```
1 module mod_test4(SW, LEDR);
2
3     input SW;
4     output [3:0]LEDR;
5
6     decadico cs1(SW, LEDR);
7
8 endmodule
```

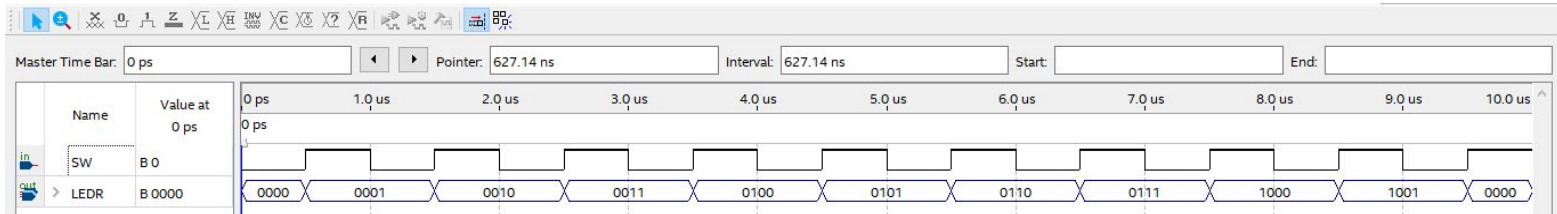
Fonte: Imagem Autoral.

Especificação do módulo de teste:

- SW = entrada de clock do circuito;
- LEDR[3:0] = Saída do circuito;

Por fim, foi implementado o diagrama de tempo para o contador síncrono decádico. Assim como representado na figura abaixo.

Figura 16 - Diagrama de Tempo do Contador Decádico Síncrono



Fonte: Imagem Autoral.

Desse modo, a saída dos dados a cada clock obedece ao seguinte padrão:

CLK	LEDR[3]	LEDR[2]	LEDR[1]	LEDR[0]
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0
11	0	0	0	1