



**EESC - USP<sup>®</sup>**

Prática N°9 - SEL0606

Alunos (Bancada 5):  
Vitor Alexandre Garcia Vaz - 14611432  
Gabriel Dezejácomo Maruschi - 14571525

# Sumário

<b>1</b>	<b>Introdução</b>	<b>3</b>
1.1	<i>Objetivos</i>	3
1.2	<i>Contador</i>	3
<b>2</b>	<b>Materiais e métodos</b>	<b>4</b>
2.1	<i>Interface</i>	4
2.2	<i>Contador</i>	5
2.3	<i>Flip-Flop JK</i>	6
<b>3</b>	<b>Conclusão</b>	<b>8</b>
3.1	<i>Círculo RTL</i>	8
3.2	<i>Número de células lógicas</i>	9
3.3	<i>Funcionamento do contador síncrono</i>	10
3.4	<i>Funcionamento da FPGA</i>	12

## Listas de Imagens

1	Esquemático de contador síncrono de 4 bits . . . . .	3
2	Tabela verdade do FF-JK . . . . .	4
3	Kit DE10-LITE . . . . .	4
4	Código da interface . . . . .	5
5	Código do contador assíncrono . . . . .	6
6	Código do FF tipo JK . . . . .	7
7	Visualização do circuito referente à interface . . . . .	8
8	Visualização do circuito referente ao registrador . . . . .	8
9	Visualização do circuito referente ao Flip-Flop tipo JK . . . . .	9
10	Resumo de funcionamento . . . . .	9
11	Simulação do contador síncrono de 4 bits e sem ativação de clear . . . . .	10
12	Simulação do contador síncrono de 5 bits e sem ativação de clear . . . . .	10
13	Simulação do contador síncrono de 5 bits e com ativação de clear . . . . .	11
17	Implementação de contador no dispositivo FPGA . . . . .	12

# 1 Introdução

## 1.1 Objetivos

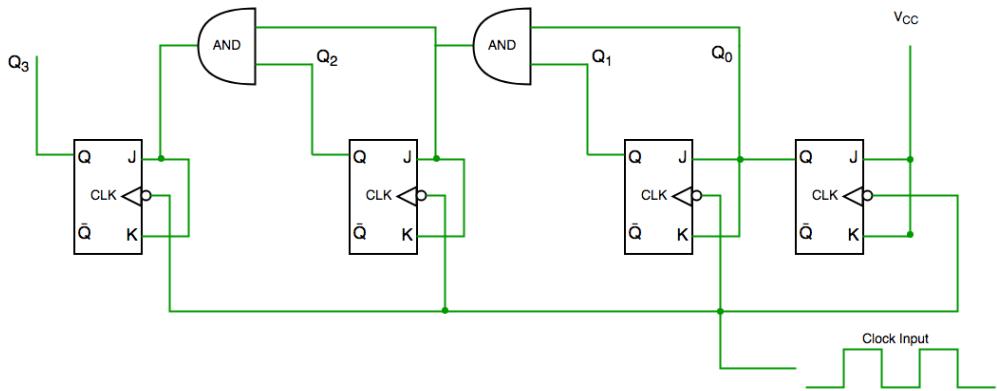
Nesta prática do Laboratório de Sistemas Digitais, implementamos um contador binário, síncrono e de N bits (parametrizável), com clock e enable, utilizando a linguagem VHDL no software Quartus, e testamos no kit DE10-LITE (MAX 10 10M50DAF484C7G) bem como no software ModelSim.

Para isso, visamos elaborar o registrador através de uma associação de flip-flops conforme a fig.1, em conjunto com uma lógica combinacional entre as saídas dessas unidades sequenciais. Assim, vale ressaltar que cada flip-flop gerará um bit da saída final Q.

## 1.2 Contador

De maneira geral, os contadores são circuitos sequenciais que fazem parte de componentes como relógios, cronômetros e computadores, tendo como principal função a contagem de dados e geração de diferentes frequências de clock.

Figure 1: Esquemático de contador síncrono de 4 bits



Fonte: [Site da internet](#)

Além disso, tais contadores funcionam conforme um sinal de clock, podendo possuir sinais de reset e enable para ditarem o seu funcionamento a partir de um certo momento. Nesse sentido, tomando como base a fig.1 para elaboração do contador, utilizamos flip-flops tipo JK em sua implementação.

### FF JK

O Flip-Flop JK funciona com a lógica de *Set*, *Reset* e *Toggle* da saída **Q**.

Possui as entradas **J**, **K** e **Clk** (clock). A tabela verdade abaixo evidencia a relação das entradas com a saída:

Figure 2: Tabela verdade do FF-JK

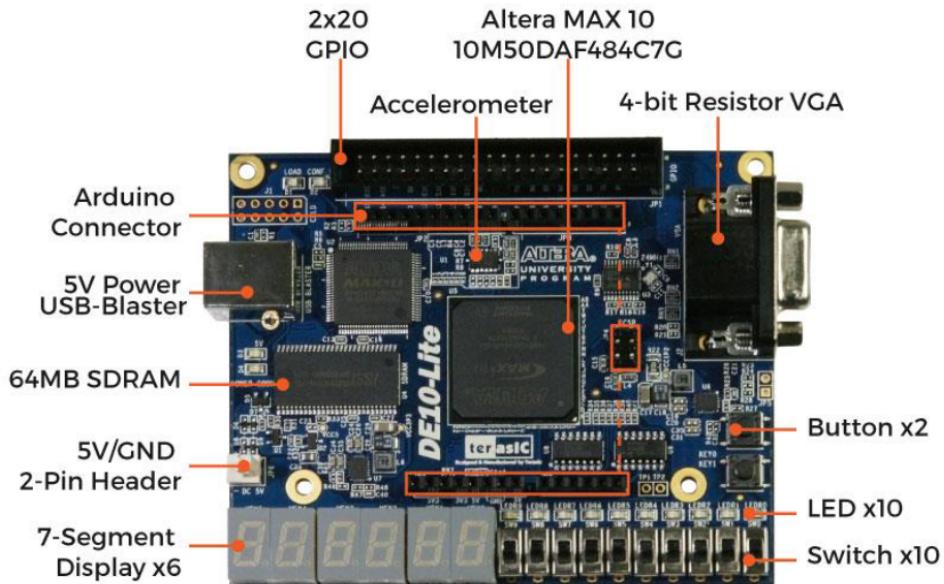
<i>J</i>	<i>K</i>	<i>Q</i>	<i>C</i>	<i>Q*</i>
0	0	0	↑	0 Hold
0	0	1	↑	1
0	1	0	↑	0 Reset
0	1	1	↑	0
1	0	0	↑	1 Set
1	0	1	↑	1
1	1	0	↑	1 Toggle
1	1	1	↑	0

Fonte: V. P. Nelson, *Digital Logic Circuit analysis and design*. 1995

## 2 Materiais e métodos

O código utilizado para desenvolver os módulos dos Flip-Flops e do contador foi escrito em VHDL e compilado no Quartus. A fim de testá-los no DE10-LITE, desenvolvemos também um interface cujo objetivo foi redirecionar os sinais dos pinos para os módulos e vice-versa. Utilizamos também o software ModelSim da Intel para visualizar as ondas de entrada e saída do Contador binário desenvolvido.

Figure 3: Kit DE10-LITE



Fonte: DE10-Lite User Manual

### 2.1 Interface

A interface, codificada como é mostrado abaixo, direciona as chaves para as entradas do módulo do contador e as saídas para um sinal auxiliar. Este sinal é exibido nos LEDs e também é redirecionado como entrada do módulo do display de binário para 7 segmentos, um decodificador desenvolvido em práticas passadas.

Figure 4: Código da interface

```

1  ENTITY DE10_LITE_counter IS
2      PORT (
3          KEY : IN BIT_VECTOR(1 DOWNTO 0);
4          LEDR: OUT BIT_VECTOR(9 DOWNTO 0);
5          HEX0: OUT BIT_VECTOR(7 DOWNTO 0)
6      );
7
8  END DE10_LITE_counter;
9
10 ARCHITECTURE interface OF DE10_LITE_counter IS
11
12     SIGNAL aux : BIT_VECTOR(3 DOWNTO 0);
13
14 BEGIN
15     -- contador de 4 bits (0000_2 a 1111_2)
16     count: entity work.counter
17         GENERIC MAP(n => 4)
18         PORT MAP(clk => KEY(0), clrn => KEY(1), Q => aux);
19
20     LEDR(9 DOWNTO 6) <= aux;
21
22     -- conecta saída do contador com display de sete segmentos
23     display: entity work.hex27seg
24         PORT MAP(hexa => aux, segments => HEX0);
25
26 END interface;

```

Fonte: os autores

Para a chamada do módulo do contador, as entradas de Clock e Clear, `clk` e `clrn`, respectivamente, foram direcionados para os botões 0 e 1 do kit. A saída pôde ser visualizada nos LEDs 9 a 6 do kit e no display HEX0. Vale ressaltar o **GENERIC MAP** especificado na chamada do contador binário. Nele, definimos o tamanho do contador ao fornecer o valor 4 para a variável genérica `n`. Isso será tratado em detalhes na seção do contador.

## 2.2 Contador

O Contador binário desenvolvido em VHDL conta com o uso do loop FOR-GENERATE disponível na linguagem para fazer quantas chamadas forem necessárias da entidade de um Flip-Flop JK.

Para isso, generalizamos com o inteiro `n`. Este possibilita que chamadas do módulo do registrador escolham o barramento deste. O valor *default* de `n` é 4.

Há também no contador o sinal de clear assíncrono. Nesta implementação, `clrn` = 0 zera a saída `Q`, reiniciando o contador. Note: `clrn` é *active-low*.

A construção do código utiliza a estrutura recorrente do contador para generalizar sua arquitetura para qualquer `n` natural e diferente de zero.

Figure 5: Código do contador assícrono

```

1  --Copyright:
2  --Date:      13/11/24
3  --Version:   1.0
4  --Owners:    Gabriel D. Maruschi
5  --           Vitor Alexandre Garcia Vaz
6
7  entity counter is
8      generic (
9          n      : integer := 4
10     );
11
12     port(
13         clk : IN BIT;
14         clrn : IN BIT;
15         Q    : OUT BIT_VECTOR (n-1 downto 0)
16     );
17 END counter;
18
19
20 ARCHITECTURE estrutural OF counter IS
21
22     signal qaux : bit_vector (n-1 downto 0);
23     signal qand : bit_vector (n-1 downto 0);
24
25 BEGIN
26
27     FF0: entity work.jk_ff PORT MAP (clk => clk, cr => clrn, j => '1', k => '1', q => qaux(0));
28
29     qand(0) <= qaux (0);
30
31     abc: FOR i IN 1 TO n-1 GENERATE
32
33         FF: entity work.jk_ff PORT MAP ( clk => clk, cr => clrn , j => qand(i-1), k => qand(i-1), q => qaux(i));
34         qand(i) <= qaux(i) AND qand(i-1);
35
36     END GENERATE abc;
37
38     Q <= qaux;
39
40 end estrutural;

```

Fonte: os autores

## 2.3 Flip-Flop JK

O Flip-Flop JK possui as entradas **j**, **k**, **clk** e **cr**, além de saída **q**, todos de 1 bit. A depender da ocorrência da borda crescente do clock, a saída responde aos sinais de maneira que J é um *SET*, K é um *RESET*, e os dois ao mesmo tempo geram a comutação da saída **q** (*TOGGLE*).

Figure 6: Código do FF tipo JK

```
● ● ●

1 --Copyright:
2 --Date:      23/10/24
3 --Version:   1.0
4 --Owners:    Gabriel D. Maruschi
5 --           Vitor Alexandre Garcia Vaz
6
7 ENTITY jk_ff IS
8     PORT
9     (
10        cr      : IN BIT;
11        j, k    : IN BIT;
12        clk : IN BIT;
13        q      : OUT BIT
14    );
15 END jk_ff;
16
17 ARCHITECTURE jkff_module OF jk_ff IS
18
19 BEGIN
20
21     PROCESS(clk, cr)
22     VARIABLE qs : BIT;
23     BEGIN
24
25         IF(cr = '0') THEN qs := '0';
26
27         ELSIF(clk'EVENT and clk = '1') THEN
28             qs := (j AND NOT qs) OR (NOT k AND qs);
29
30         END IF;
31
32         q <= qs;
33
34     END PROCESS;
35
36 END jkff_module;
```

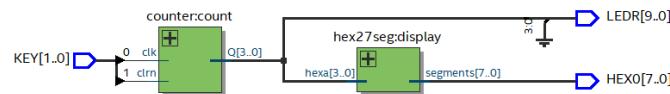
Fonte: os autores

### 3 Conclusão

#### 3.1 Circuito RTL

A partir do código em VHDL feito em laboratório, e usando a ferramenta de síntese disponibilizada pelo quartus, o seguinte circuito, referente à interface entre o contador e os pinos da FPGA, foi obtido:

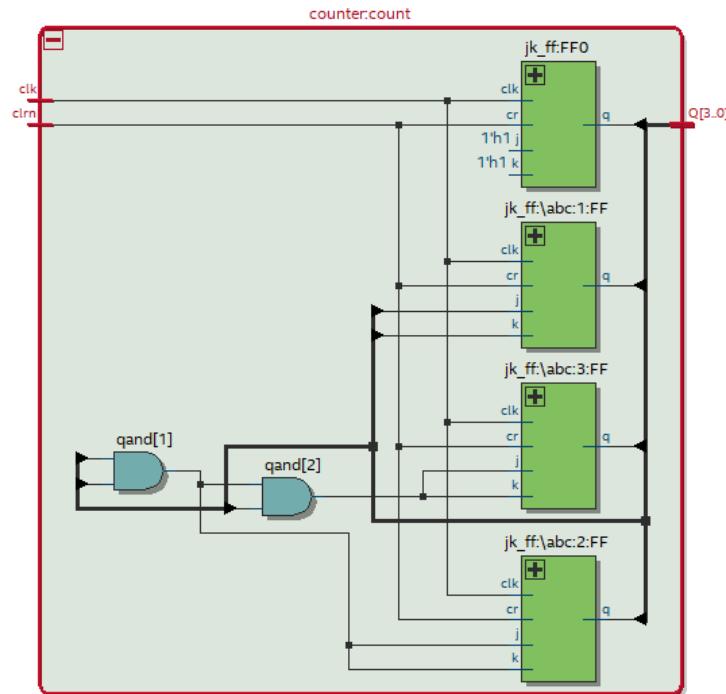
Figure 7: Visuzalização do circuito referente à interface



Fonte: os autores

Por outro lado, referentemente à estrutura interna do contador síncrono, o seguinte circuito foi obtido:

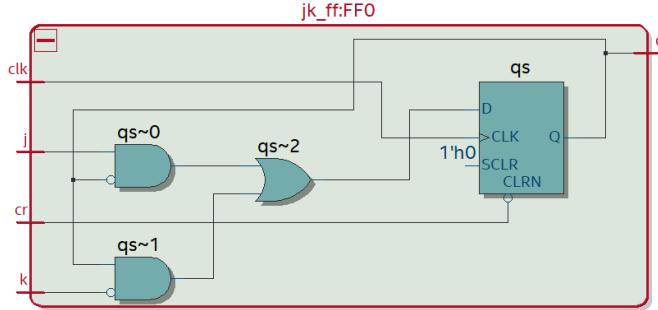
Figure 8: Visuzalização do circuito referente ao registrador



Fonte: os autores

Ademais, para cada flip-flop tipo JK associado associado á formação do contador, obtemos o seguinte circuito:

Figure 9: Visualização do circuito referente ao Flip-Flop tipo JK

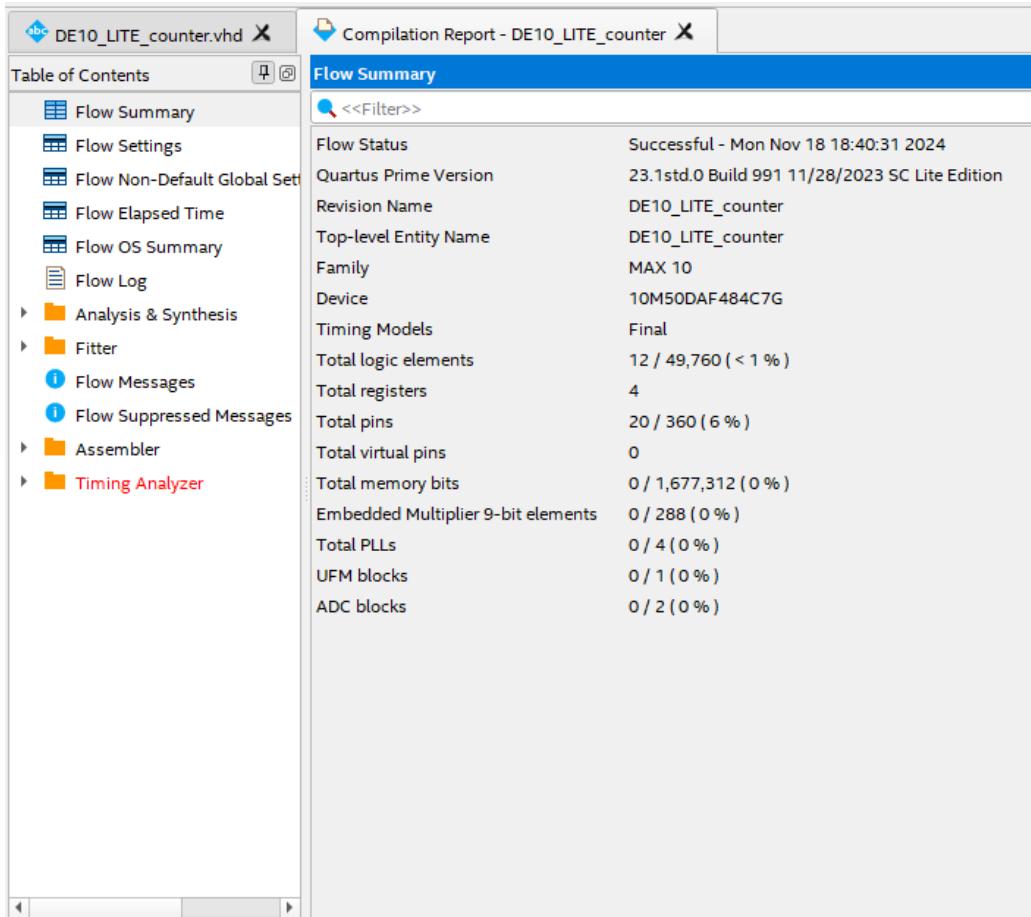


Fonte: os autores

### 3.2 Número de células lógicas

Ademais, por meio do resumo do funcionamento e constituição do circuito descrito em VHDL no software (fig10), chegamos à conclusão de que o circuito sintetizado apresentou um uso de 12 elementos lógicos e 20 pinos do dispositivo reconfigurável (FPGA).

Figure 10: Resumo de funcionamento



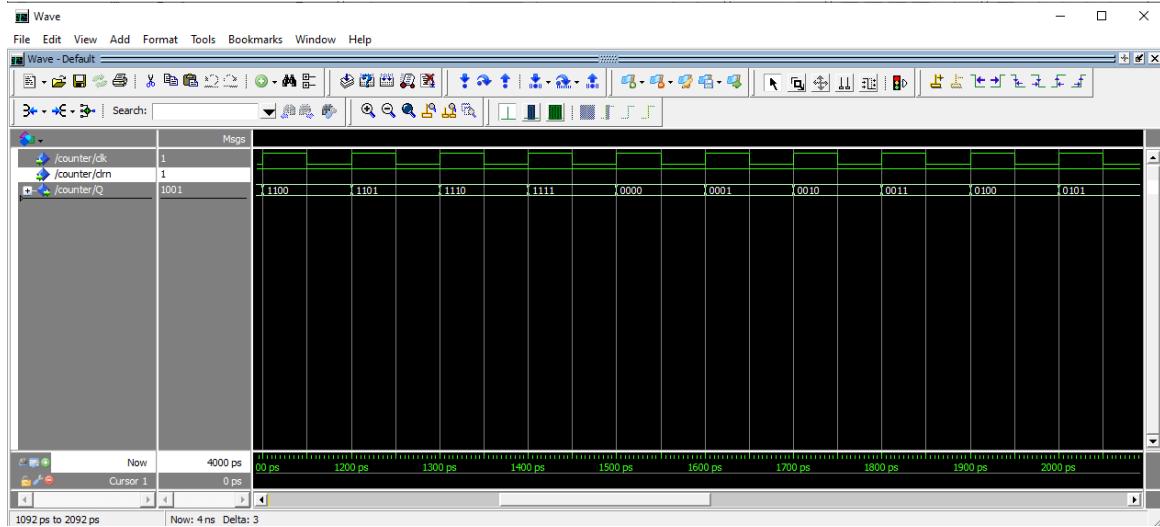
Fonte: os autores

### 3.3 Funcionamento do contador síncrono

A respeito do funcionamento do contador síncrono elaborado pelo grupo, utilizamos o software de ModelSim para simulá-lo em conjunto com a implementação em dispositivo reconfigurável (FPGA), para comprovarmos e validarmos a funcionalidade desse circuito.

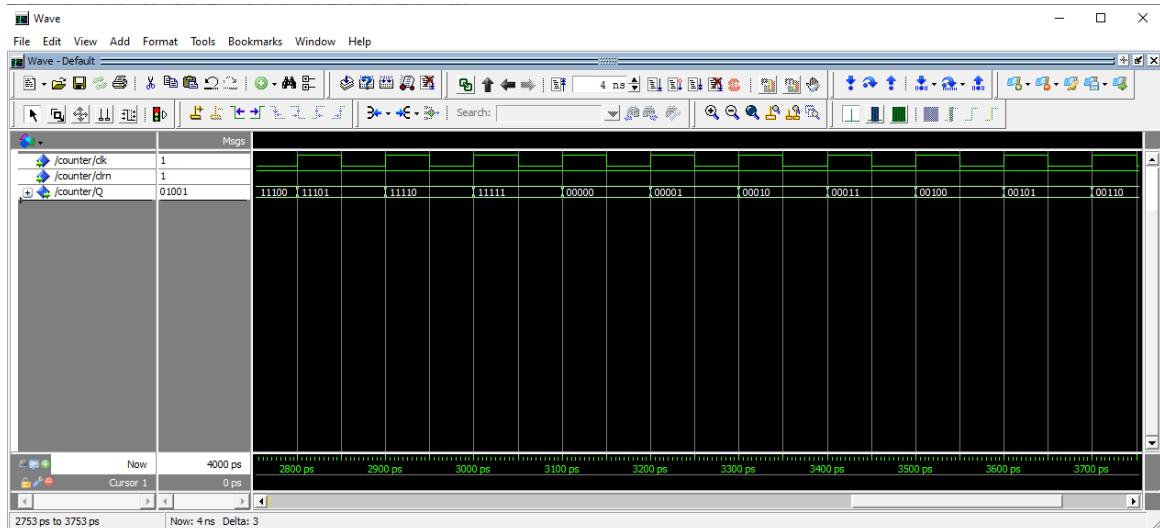
Nesse sentido, o circuito referente ao contador implementado funcionou da forma esperada para diferentes extensões (em bits) de contagem. Assim, tal fato pode ser comprovado pelas simulações feitas e expostas nas figuras fig.11 e fig.12. Isso se deve ao fato de que - na primeira simulação (de 4 bits) - a cada ciclo de clock é realizada a contagem que se estende de  $0(0000_2)$  a  $2^4 - 1(1111_2)$  e um posterior reinício, enquanto que - na segunda simulação (de 5 bits) - o mesmo fato ocorre, porém com a contagem se estendendo de  $0(00000_2)$  a  $2^5 - 1(11111_2)$ , também com um reinício posterior.

Figure 11: Simulação do contador síncrono de 4 bits e sem ativação de clear



Fonte: os autores

Figure 12: Simulação do contador síncrono de 5 bits e sem ativação de clear

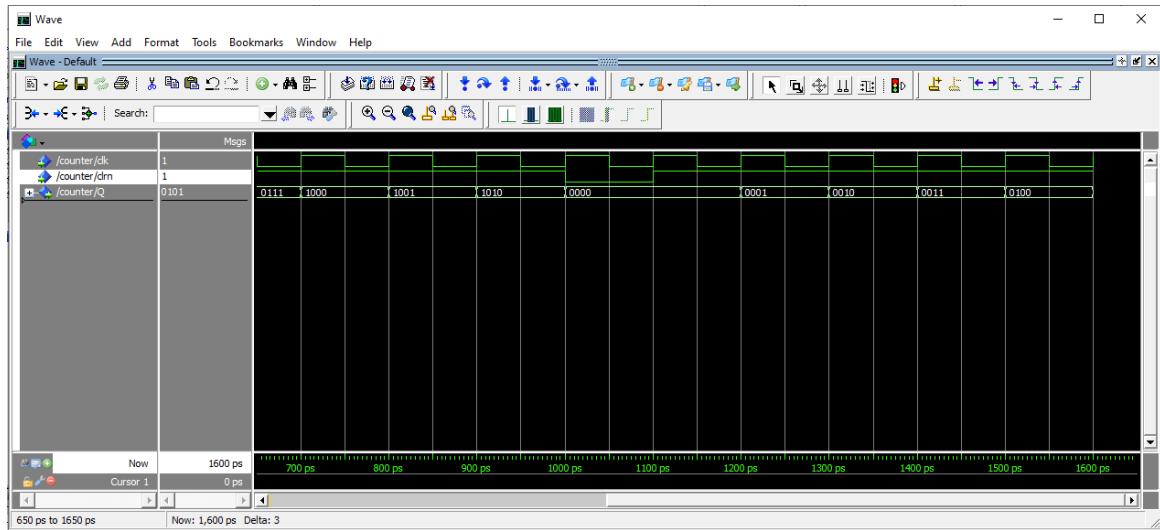


Fonte: os autores

Além disso, também pudemos comprovar a eficiência do clear (clr) do circuito implementado através da simulação da fig.13 , já que definimos os flip-flops JK com reset (cr) negativo e, durante a simulação feita, quando o clear é ativado, a saída Q do contador passa a ser nula e a contagem é reiniciada. Assim, tal fato

mostra-se coerente com a implementação realizada pelo grupo, tendo em vista que a entrada de reset de cada flip-flop JK (componente do contador) está conectada ao sinal de entrada clear do contador assíncrono, conforme o código do contador (fig.5).

Figure 13: Simulação do contador síncrono de 5 bits e sem ativação de clear



Fonte: os autores

### 3.4 Funcionamento da FPGA

Portanto, segue as imagens do funcionamento do projeto no dispositivo reconfigurável (FPGA) após a implementação do circuito elaborada, com parametrização de 4 bits e ao longo de 15 bordas positivas de clock (correspondentes à ativação do botão KEY(0) 15 vezes).

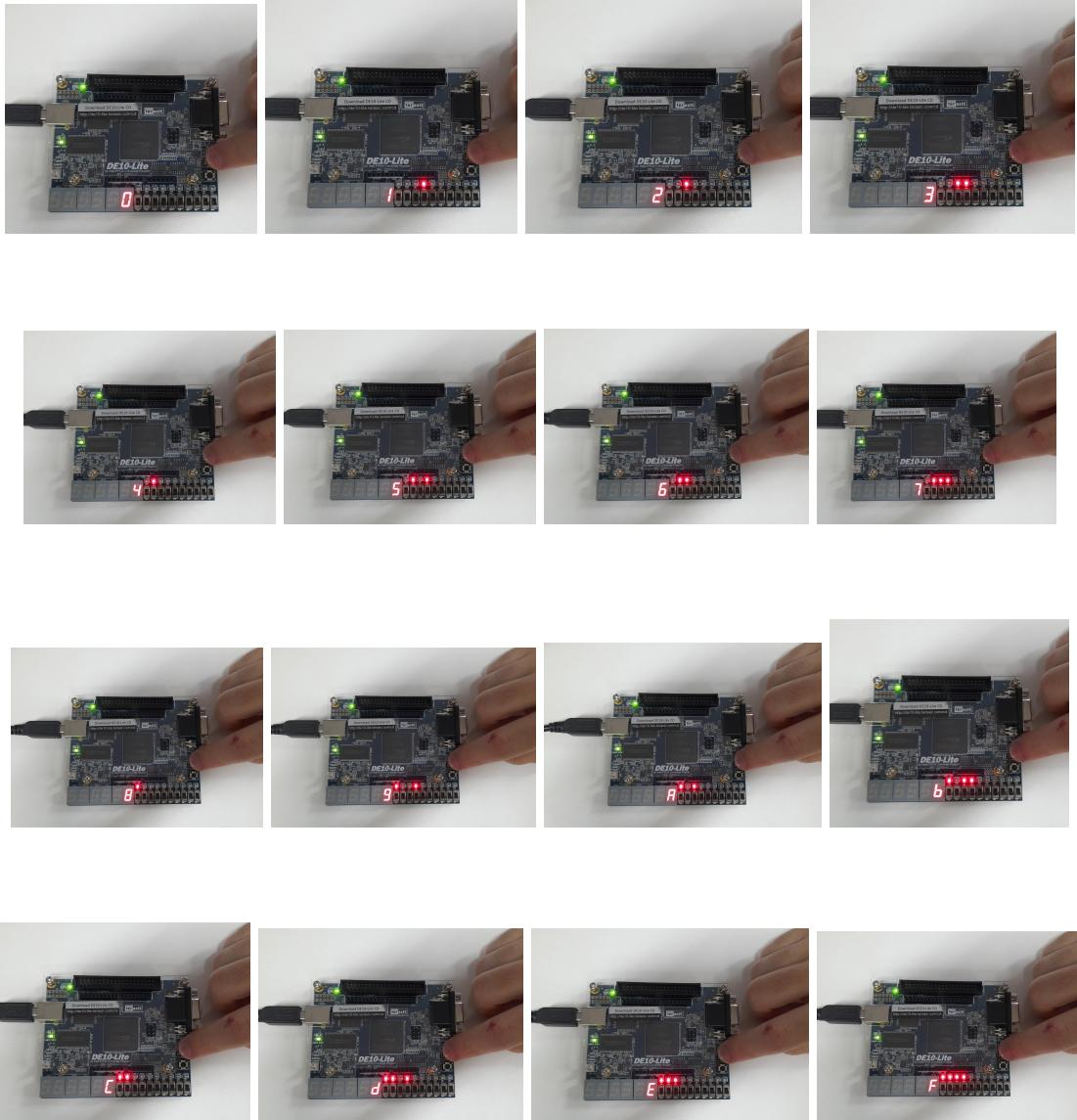


Figure 17: Implementação de contador no dispositivo FPGA

Fonte: os autores