Implementação de um Circuito da Sequência de Fibonacci utilizando a interface Xilinx

Lucas André Bringmann e Víctor Florence de Quadros

Departamento de Engenharias, Arquitetura e Computação
Universidade de Santa Cruz do Sul (UNISC) – Campus Santa Cruz do Sul
Caixa Postal 188-96.815 – Santa Cruz do Sul – RS – Brasil

{lucasabringmann, victorquadros}@mx2.unisc.br

Abstract. *This work aims to implement a circuit that shows the Fibonacci sequence from one to thirteen in a seven segment Led display using hex base. Will also be implemented an asynchronous counter witch will increment by one every time an overflow occurs in Fibonacci Sequence. In Other words, every time it reaches thirteen..*

Resumo. *Este trabalho visa implementar um circuito que exiba a sequência de Fibonacci até o número 13 em base hexa decimal em um display de Led de 7 seguimentos. Em paralelo, será desenvolvido um contador assíncrono que incremente em um toda vez que houver um overflow na sequência de Fibonacci, ou seja, toda vez que chegar ao número 13.*

1. Introdução

A sequência de Fibonacci é uma sucessão de números que retoma do século 13, onde o matemático italiano Leonardo de Pisa descreveu que o crescimento de uma população de coelhos aumentava seguindo essa sequência. Ela normalmente começa no 0 e 1 sendo que os próximos números sempre são uma suma dos seus dois antecessores. Dessa forma ficou mais simples elaborar um circuito que começa no seu segundo número e chega até o seu oitavo, no caso o número 13.

O Circuito conta com duas representações do número da sequência, sendo de maneira binária pelos 4 primeiros LEDs disponíveis na placa e por um painel de 7 segmentos. Além disso, também conta com um contador binário, representado pelos 4 LEDs restantes na placa, de quantas vezes a sequência atingiu seu limite e foi resetada, mas essa representação de estouro tem o limite de até 15, quando também acaba resetando a zero.

Para a criação do circuito, foram utilizados os conceitos de projeto aprendidos e desenvolvidos ao longo da disciplina utilizando os softwares da Xilinx e também visando a placa disponibilizada, sendo ela a Spartan-3, modelo XC3S1000.

2. Implementação

Para a implementação dos circuitos, foi utilizado a ferramenta Xilinx ISE na sua versão 14.7 onde foi construído toda a lógica de funcionamento, em representação de esquemático, a partir de portas lógicas, flip flops e um divisor de frequência disponibilizado pelo professor, também foi testado pelo test bench disponibilizado a

partir da mesma ferramenta. Além dela foi utilizado o Xilinx PlanAhead que possibilita o mapeamento das portas de entrada e saída do Circuito para a placa Spartan-3.

2.1. Divisor de frequência:

Esse circuito tem como o objetivo dividir o clock utilizado pelo FPGA Spartan-3 que é de 50MHz, com o objetivo de ser possível visualizar as mudanças ocorridas.

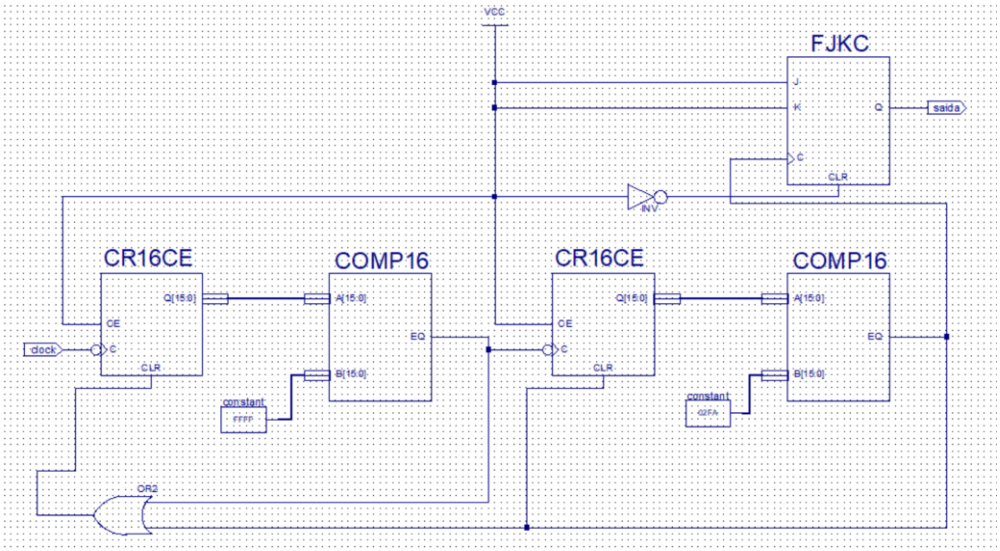


Imagem 1. Print do esquemático do divisor de frequência.

2.2. Máquina de estados para sequência de Fibonacci:

Esse circuito tem como função fazer as lógicas necessárias para passos da sequência, ou seja, transmitir a partir a cada pulso de clock o próximo número da sequência da sucessão por uma representação binária.

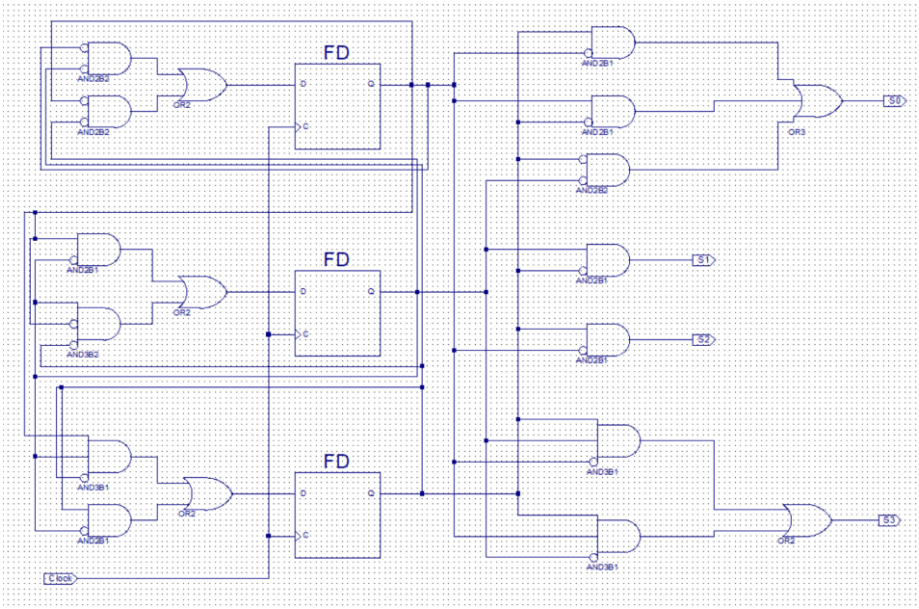


Imagem 2. Print do esquemático da máquina de estados de Fibonacci.

Sequência de Fibonacci

	Q2/Q1/Q0	S3	S2	S1	S0	Próximo estado E=1	Próximo estado E=0
1	000	0	0	0	1	001	001
1	001	0	0	0	1	010	010
2	010	0	0	1	0	011	011
3	011	0	0	1	1	100	100
5	100	0	1	0	1	101	101
8	101	1	0	0	0	110	110
13	110	1	1	0	1	000	000

S0 = Q2'.Q1'.Q0' + Q2'.Q1'.Q0 + Q2'.Q1.Q0 + Q2.Q1'.Q0' + Q2.Q1.Q0'

	00	01	11	10
0	1	1	1	0
1	1	0	0	1

S0 = Q2.Q0' + Q2'.Q0 + Q2'.Q1'

S1 = Q2'.Q1.Q0' + Q2.Q1.Q0

	00	01	11	10
0	0	0	1	1
1	0	0	0	0

S1 = Q2'.Q1

S2 = Q2.Q1'.Q0' + Q2.Q1.Q0'

	00	01	11	10
0	0	0	0	0
1	1	0	0	1

S2 = Q2.Q0'

S3 = Q2.Q1'.Q0 + Q2.Q1.Q0'

	00	01	11	10
0	0	0	0	0
1	0		0	1

S3 = Q2.Q1'.Q0 + Q2.Q1.Q0'

D0 = Q2'.Q1'.Q0' + Q2'.Q1.Q0' + Q2.Q1'.Q0'

	00	01	11	10
0	1	0	0	1
1	1	0	0	0

D0 = Q2'.Q0' + Q1'.Q0'

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

	00	01	11	10
0	0	1	0	1
1	0	1	0	0

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

D2 = Q2'.Q1.Q0 + Q2.Q1'.Q0' + Q2.Q1'.Q0

	00	01	11	10
0	0	0	0	1
1	1	1	0	0

D2 = Q2'.Q1.Q0 + Q2.Q1'

Imagem 3. Print dos cálculos da máquina de estados de Fibonacci.

2.3. Contador assíncrono:

Um simples contador assíncrono, ou seja, que o clock dos flip flops não são compartilhados e a saída é ligada no clock do próximo para realizar a contagem. Como o objetivo é chegar até o número 15 não é necessário utilizar o clear para recomeçar a contagem, pois ela estoura automaticamente indo do 15 para o 0.

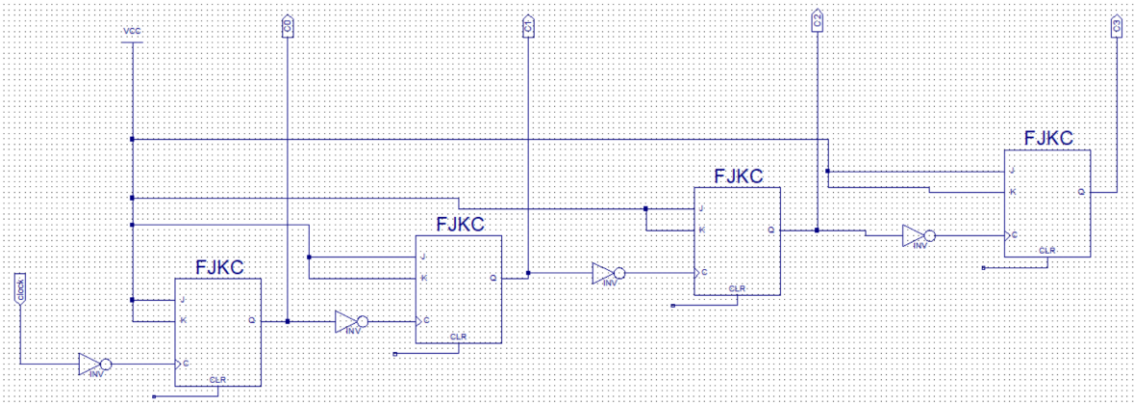


Imagem 4. Print dos esquemático do Contador assíncrono.

2.4. Tradutor para o painel de 7 segmentos:

Como a saída da máquina de estado que representa a sequência de Fibonacci é binária devemos transformar sua saída de 4 sinais para os 7 esperados pelo painel para representar de maneira Hexadecimal os números até 13 na base decimal.

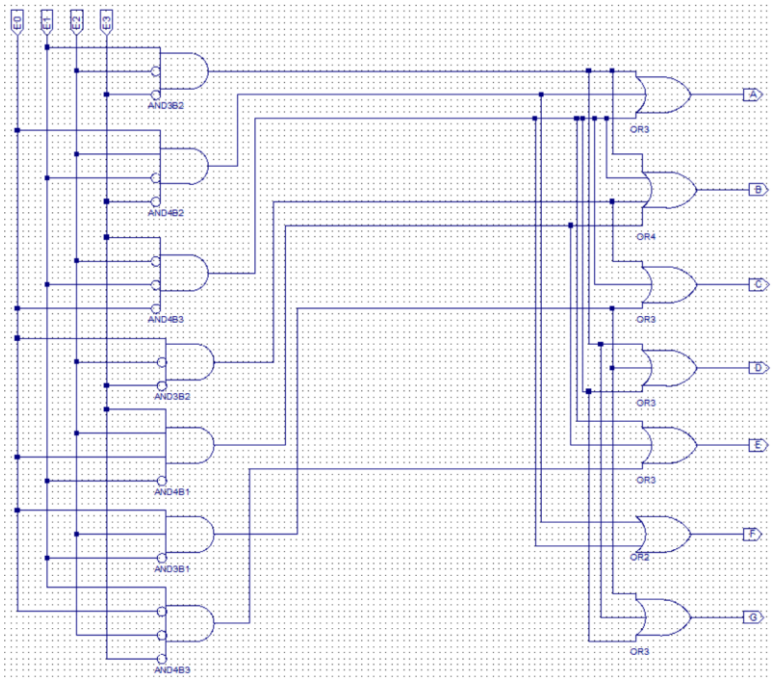


Imagem 5. Print dos esquemático do tradutor para o painel de 7 segmentos.

2.5. Combinação dos circuitos:

Por fim, a combinação de todos os circuitos em representação de esquemático contendo o clock do sistema, o divisor de frequência, a máquina responsável pela sequência de Fibonacci (Schematic_MqDeEstado), o tradutor para o painel de 7 segmentos (SevenSchematic), o contador assíncrono (SchematicAssincCount), uma porta "and" para sinalizar positivo quando a sucessão chega a 13 ao contador e as respectivas saídas.

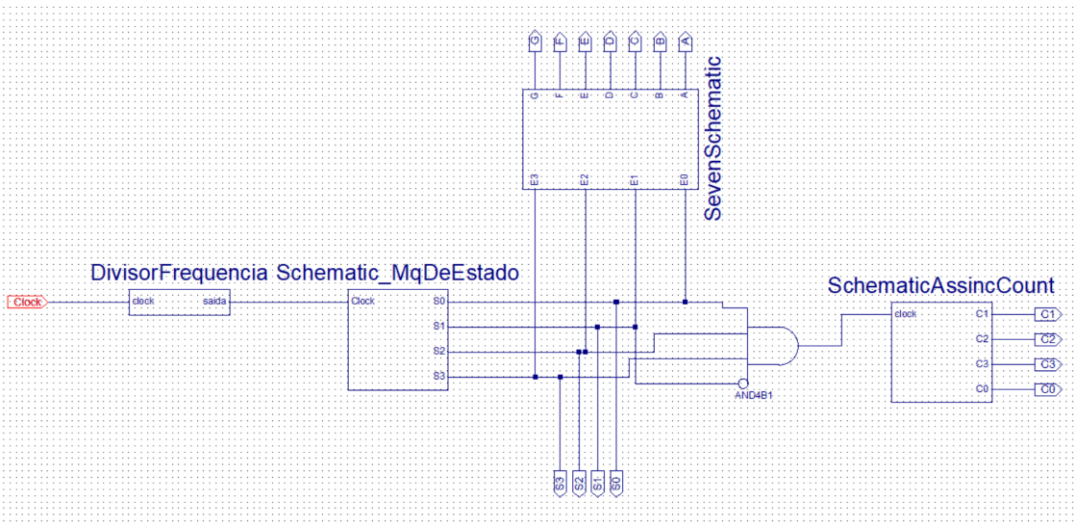


Imagem 6. Print dos esquemático da combinação dos circuitos.

3. Testes

A partir do Test Bench foi possível similar o comportamento do circuito de maneira real alternado o clock entre zero e um, mas para isso foi preciso remover o divisor de frequência, pois ele somente é utilizado para a exportação para a placa real. E com sua construção foi possível ver visualmente por meio de um gráfico de ondar o comportamento quando cada saída está ativa em cada variação de clock

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY TrabSchematic_TrabSchematic_sch_tb IS
END TrabSchematic_TrabSchematic_sch_tb;
ARCHITECTURE behavioral OF TrabSchematic_TrabSchematic_sch_tb IS

   COMPONENT TrabSchematic
   PORT( S0 :  OUT   STD_LOGIC;
         S1   :  OUT   STD_LOGIC;
         S2   :  OUT   STD_LOGIC;
         S3   :  OUT   STD_LOGIC;
         A :  OUT   STD_LOGIC;
         B :  OUT   STD_LOGIC;
         C :  OUT   STD_LOGIC;
         D :  OUT   STD_LOGIC;
         E :  OUT   STD_LOGIC;
         F :  OUT   STD_LOGIC;
         G :  OUT   STD_LOGIC;
         C0   :  OUT   STD_LOGIC;
         C3   :  OUT   STD_LOGIC;
         C2   :  OUT   STD_LOGIC;
         C1   :  OUT   STD_LOGIC;
         Clock    :  IN STD_LOGIC);
   END COMPONENT;

   SIGNAL S0    :  STD_LOGIC;
   SIGNAL S1    :  STD_LOGIC;
   SIGNAL S2    :  STD_LOGIC;
   SIGNAL S3    :  STD_LOGIC;
   SIGNAL A :  STD_LOGIC;
   SIGNAL B :  STD_LOGIC;
   SIGNAL C :  STD_LOGIC;
   SIGNAL D :  STD_LOGIC;
   SIGNAL E :  STD_LOGIC;
   SIGNAL F :  STD_LOGIC;
   SIGNAL G :  STD_LOGIC;
   SIGNAL C0    :  STD_LOGIC;
   SIGNAL C3    :  STD_LOGIC;
   SIGNAL C2    :  STD_LOGIC;
   SIGNAL C1    :  STD_LOGIC;
   SIGNAL Clock    :  STD_LOGIC;

BEGIN

   UUT: TrabSchematic PORT MAP(
      S0 => S0,
      S1 => S1,
      S2 => S2,
      S3 => S3,
      A => A,
      B => B,
      C => C,
      D => D,
      E => E,
      F => F,
      G => G,
      C0 => C0,
      C3 => C3,
      C2 => C2,
      C1 => C1,
      Clock => Clock
   );

-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      Clock<='1';
         WAIT for 10 ns; -- will wait forever
      Clock<='0';
         WAIT for 10 ns;
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

Imagem 7. Print do test bench.

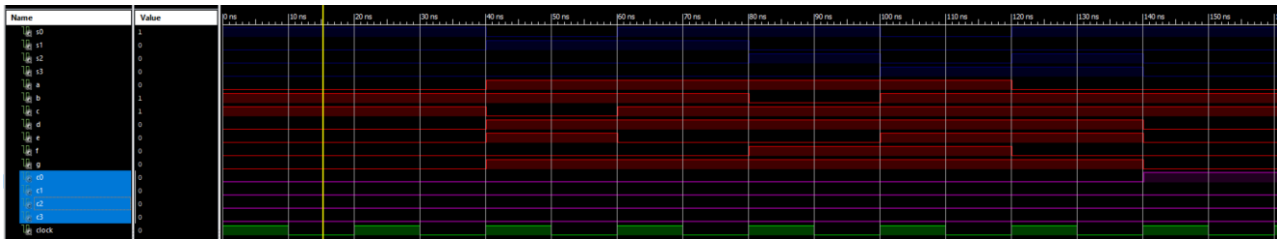
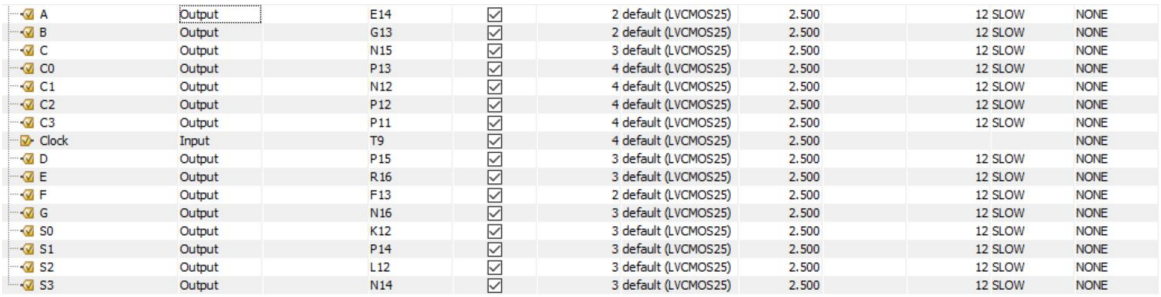


Imagem 8. Print dos gráfico de ondas gerado pelo test bench.

4. Mapeamento das saídas e do clock

Por fim, foi realizado o mapeamento pelo PlanAHead da entrada do circuito sistema e das saidas para a placa real a fim de se utilizar do clock da placa para a entrada, dos LEDs e do painel de 7 segmentos.



A	Output	E14	☑	2 default (LVCMOS25)	2.500	12 SLOW	NONE
B	Output	G13	☑	2 default (LVCMOS25)	2.500	12 SLOW	NONE
C	Output	N15	☑	3 default (LVCMOS25)	2.500	12 SLOW	NONE
C0	Output	P13	☑	4 default (LVCMOS25)	2.500	12 SLOW	NONE
C1	Output	N12	☑	4 default (LVCMOS25)	2.500	12 SLOW	NONE
C2	Output	P12	☑	4 default (LVCMOS25)	2.500	12 SLOW	NONE
C3	Output	P11	☑	4 default (LVCMOS25)	2.500	12 SLOW	NONE
Clock	Input	T9	☑	4 default (LVCMOS25)	2.500		NONE
D	Output	P15	☑	3 default (LVCMOS25)	2.500	12 SLOW	NONE
E	Output	R16	☑	3 default (LVCMOS25)	2.500	12 SLOW	NONE
F	Output	F13	☑	2 default (LVCMOS25)	2.500	12 SLOW	NONE
G	Output	N16	☑	3 default (LVCMOS25)	2.500	12 SLOW	NONE
S0	Output	K12	☑	3 default (LVCMOS25)	2.500	12 SLOW	NONE
S1	Output	P14	☑	3 default (LVCMOS25)	2.500	12 SLOW	NONE
S2	Output	L12	☑	3 default (LVCMOS25)	2.500	12 SLOW	NONE
S3	Output	N14	☑	3 default (LVCMOS25)	2.500	12 SLOW	NONE

Imagem 9. Print do mapeamento da entrada e das saídas do circuito.

5. Conclusão

A partir do circuito projetado, foi possível validar os conceitos estudados ao longo da primeira etapa da disciplina. Entre eles, podemos destacar a máquina de estados que é o centro de toda a proposta do desenvolvimento quanto dessa etapa da disciplina, por meio da sequência de Fibonacci foi possível demonstrar de maneira quase real uma utilização dos conceitos e técnicas aprendidas até o momento.

Vale destacar a importância de se aprender os conceitos básicos principalmente dos testes que são escritos em VHDL, mesmo que gerados automaticamente a partir do esquemático, essa linguagem de descrição de hardware é altamente utilizada pela indústria na elaboração dos chips utilizados em todo sistema de computação, assim, criando uma premissa para os próximos passos da disciplina. Para ter ao circuito na integra ele acompanha em anexo e basta acessar: https://github.com/VictordeQuadros/Fibonacci7Seg .

Referências

Xilinx, Inx “Spartan-3 Starter Kit Board User Guide”. Disponível: https://www.xilinx.com/, Maio/2022.