Relatório do experimento 5: um algoritmo como um circuito lógico

Henrique Koji Miyamoto (RA 169614) e Pedro Luís Azevedo Costa (RA175857)

1 INTRODUÇÃO

Esse experimento teve como objetivo a construção de um processador de propósito único que calcula números da série de Fibonacci.

Um processador pode ser definido como um circuito que recebe entradas, faz operações e registra (armazena) os resultados. Um processador de propósito único é um tal circuito que realiza uma única tarefa de processamento – como calcular números da sequência de Fibonacci, no nosso caso. Os resultados armazenados foram os três últimos números da sequência em um determinado instante. Nesse tipo de processador, a funcionalidade desejada é implementada diretamente no *hardware* (VAHID, 2008).

A sequência de Fibonacci é a sequência iniciada por $1, 1, 2, 3, 5, 8, 13, 21, ...$ que se caracteriza por cada termo, após os dois primeiros, ser a soma dos dois anteriores. Ela é atribuída a Leonardo de Pisa, conhecido como Fibonacci, como solução do problema de quantos casais de coelhos haveria após um dado número de meses em uma população que se iniciasse com apenas um par, considerando que, em cada mês, cada casal gera um novo casal que se torna produtivo a partir do segundo mês (LUCHETTA, 2015).

Nesse projeto, estamos interessados na implementação do algoritmo de obtenção dos números da sequência. Utilizamos o algoritmo clássico, conforme apresentado por Tavares e Covre (2015), cuja implementação em circuitos lógicos utilizou CIs da família 7400 e registrador disponibilizado por Tavares (2015a).

2 MATERIAIS E PROCEDIMENTOS

O experimento foi projetado e simulado no programa Quartus II e testado na placa FPGA. Para o projeto, foram utilizados um contador binário de 4 bits (7469), um decodificador de 2 para 4 linhas (74139), um somador completo de 4 bits (7483) e três registradores, projetados para facilitar a implementação desse experimento (TAVARES, 2015a).

O algoritmo a ser implementado está escrito abaixo em pseudocódigo. A ideia é que o último e o penúltimo valor da série sejam armazenados, de modo que sua soma possa gerar o próximo valor. A cada ciclo, o que era o último número é agora armazenado como penúltimo; o que era o atual torna-se último; e a soma do último com o penúltimo gera o novo atual. Nessas transições, o que era o penúltimo número é perdido. O início da sequência deve ser definido manualmente.

```
anterior1 = 0
anterior2 = 0
atual = 1
N = (valor desejado)
n = 2
enquanto (verdadeiro):
  se (N < 3) ou (n == N):
    retorna (atual)
  caso contrário:
    anterior1 = anterior2
    anterior2 = atual
    atual = anterior1 + anterior2
    n = n + 1
```

Figura 1. Pseudocódigo de um programa para calcular números da sequência de Fibonacci (TAVARES e COVRE, 2015).

A implementação desse algoritmo como circuito lógico funciona da seguinte maneira: o contador recebe o sinal do *clock* e envia um número (binário, 2 bits) para o decodificador. São usadas três saídas do decodificador: cada uma delas corresponde a uma ação de um ciclo (conforme descrito acima, no algoritmo). Os três registradores do circuito correspondem aos números atual $F(n)$, último $F(n-1)$ e penúltimo $F(n-2)$. Quando o decodificador recebe o correspondente a 0_{10}, a primeira ação do ciclo é realizada: o número que estava armazenado no registrador do último número, passa para o do penúltimo. Quando o decodificador recebe 1_{10}, o número atual passa a ser armazenado como último. Quando o decodificador recebe 2_{10}, o número que é correspondente à soma do último e penúltimo (operação realizada pelo circuito somador) é atribuído como atual. O número 3_{10} no decodificador não corresponde a nenhuma ação. O circuito possui uma entrada *reset*, que pode ser acionada quando se deseja reiniciar a contagem da série. Ela define novamente o estado inicial nos registradores. O diagrama de blocos a seguir ilustra essa implementação.

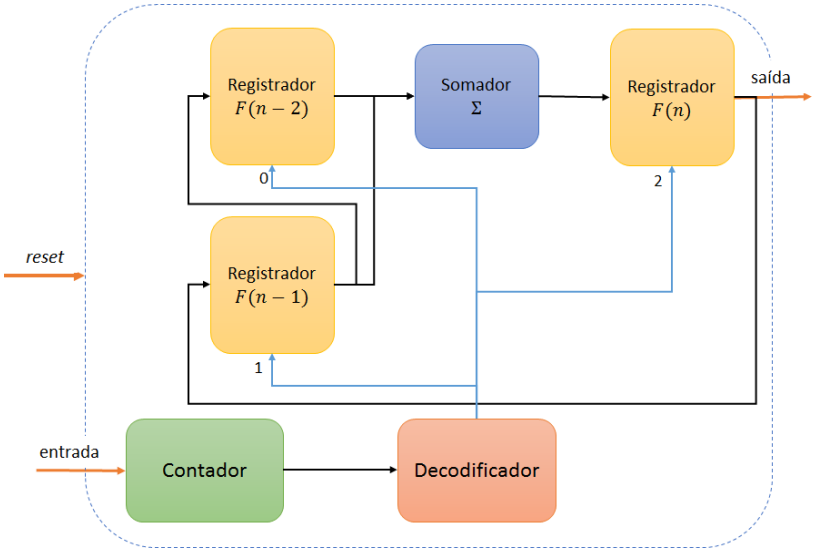
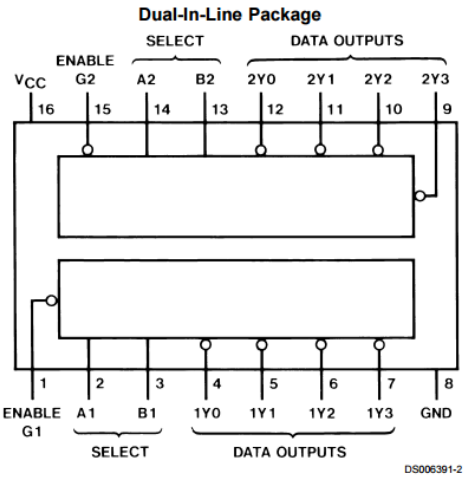


Figura 2. Diagrama de blocos do circuito projetado

A seguir, apresentamos uma breve descrição de cada CI utilizado e uma descrição com mais detalhes da implementação do algoritmo apresentado.

O contador binário 7469 recebe um sinal de *clock* externo e conta quatro números (de 00 a 11). Esses números entram no decodificador 74139 (FAIRCHILD SEMICONDUCTOR, 2015) e acionam, um de cada vez, a saída cujo número decimal corresponde ao número binário formado pelo sinal da entrada. Como cada ciclo do algoritmo tem três passos e o contador e decodificador são de dois bits (quatro números), um número não ativará nenhuma ação (o número $3_{10} = 11_2$). Os outros três ativarão os registradores, um de cada vez. Em cada saída do decodificador é necessária uma porta lógica inversora, pois tal componente segue a lógica inversa, conforme tabela abaixo.



LS139

Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = High Level, L = Low Level, X = Don't Care

Figura 3. Diagrama do decodificador 74139 e tabela que representa seu funcionamento (adaptado de FAIRCHILD SEMICONDUCTOR, 2015).

Os registradores utilizados foram projetados e disponibilizados por Tavares (2015a), de modo a facilitar a implementação desse projeto. Eles possuem barramentos para entrada de dados **d[3..0]** e para saída de dados **out[3..0]**. Têm entrada do *clock* e duas entradas de controle (r0 e r1), que definem o modo de operação: a entrada 00 ativa o modo 1, que copia os dados da entrada para a saída, ao receber sinal de *clock*; a entrada 10 (modo 2) força a saída em 0001_2; e a entrada 01 (modo 3) força a saída em 0000_2.

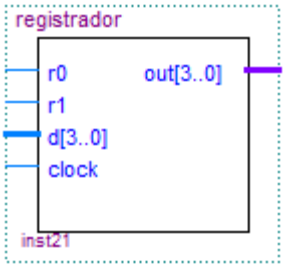


Figura 4. Um bloco do registrador de Tavares (2015a) no Quartus II

As entradas de controle dos registradores estão ligadas a uma entrada do circuito de *reset*, para reiniciar a contagem da série de Fibonacci. Essa ligação é feita de forma tal que, quando essa entrada é acionada, os registradores $F(n-2)$ e $F(n-1)$ têm seus valores definidos como 0_{10} e o

registrador $F(n)$, como 1_{10}. No circuito, temos uma porta lógica inversora após a entrada, pois, na placa FPGA, o *push button*, utilizado para essa entrada, torna 0 o valor na entrada ao ser acionado.

O circuito somador total de 4 bits 7483 (MOTOROLA, 2015) é utilizado para somar os dois últimos valores, de modo a obter o próximo. Ele recebe como entrada os dois valores dos registradores $F(n-2)$ e $F(n-1)$ e sua saída vai para o registrador registrador $F(n)$. É importante notar que a entrada de *carry-in* desse somador está permanentemente definida em 0 (GND), para que a soma seja corretamente operada.

A seguir, o diagrama do circuito completo. A saída do circuito é o número atual da sequência de Fibonacci, armazenado no registrador $F(n)$.

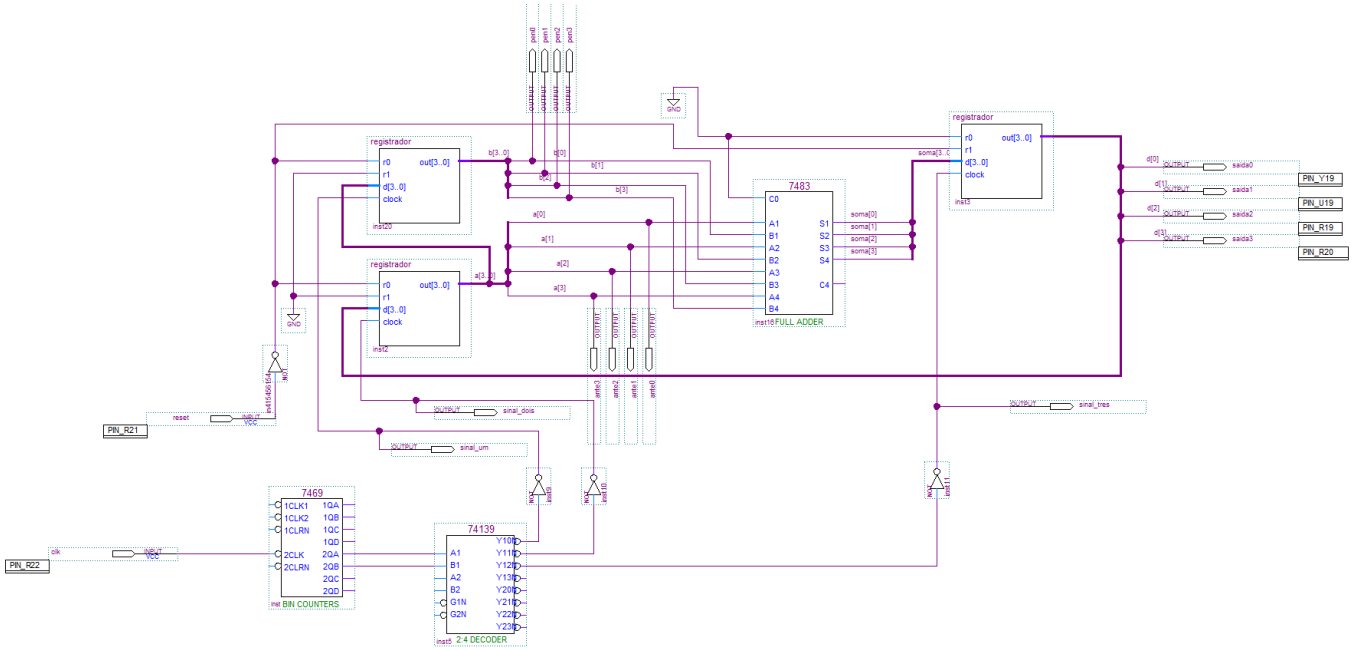


Figura 5. Diagrama completo do circuito do processador que calcula números da sequência de Fibonacci

O nosso circuito tem saídas extras de controle, que foram incluídas para podermos conferir e visualizar o resultado em diferentes etapas do ciclo do algoritmo, o que pode ser visto na seção 3 (Resultados). Isso foi feito devido aos erros encontrados durante o projeto (ver seção 4 – Discussão).

É interessante notar que, como o circuito foi projetado com base em números binários de 4 bits, há uma limitação na representação dos números da sequência de Fibonacci. Com 4 bits, o maior número decimal que pode ser representado é 15_{10} (1111_2). Assim, o maior número da sequência que pode ser representado por nosso circuito é 13_{10}. A partir do próximo, 21_{10}, haverá um *carry-out*, que não poderá ser representado (pois são necessários mais bits).

Para teste na placa FPGA, a entrada do *clock* foi associada a um *push button*, assim como a entrada *reset*; as saídas foram associadas a LEDs, conforme mostrado na próxima seção.

3 RESULTADOS

O circuito foi bem sucedido em executar o algoritmo que calcula a sequência de Fibonacci. Os sete primeiros números foram calculados e representados com sucesso (como explicado anteriormente, os números seguintes não podem ser representados, devido ao *carry-out*).

A seguir, imagem que representa simulação funcional do circuito. Todas as saídas adicionais são mostradas e a saída do circuito, que representa o valor atual da sequência, está destacada.

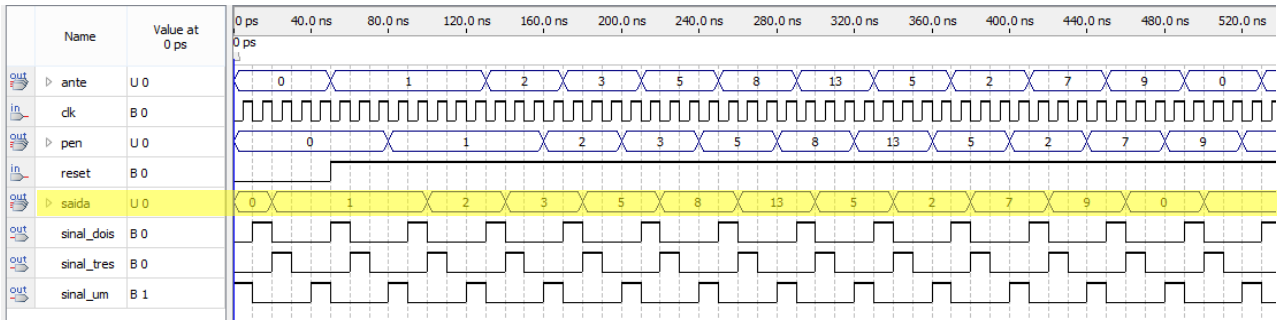


Figura 6. Resultado da simulação funcional do circuito

Abaixo imagens com as saídas, representadas nos LEDs da placa FPGA, para os números que podem ser representados com até 4 bits (até o 13_{10}). O quarto LED da esquerda para a direita representa o algarismo menos significativo e o primeiro, o mais significativo.

Figura 7. Saída representando o número 1_{10}

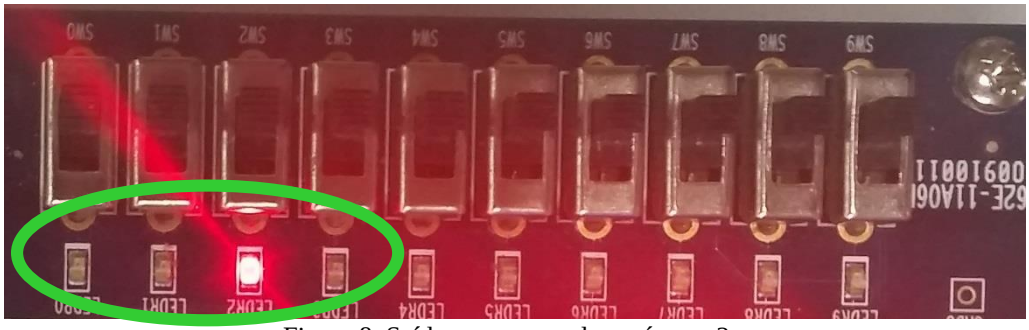


Figura 8. Saída representando o número 2_{10}

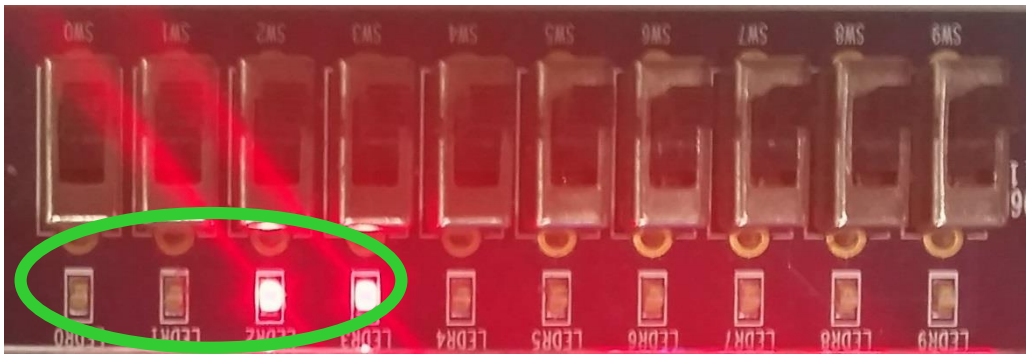


Figura 9. Saída representando o número 3_{10}

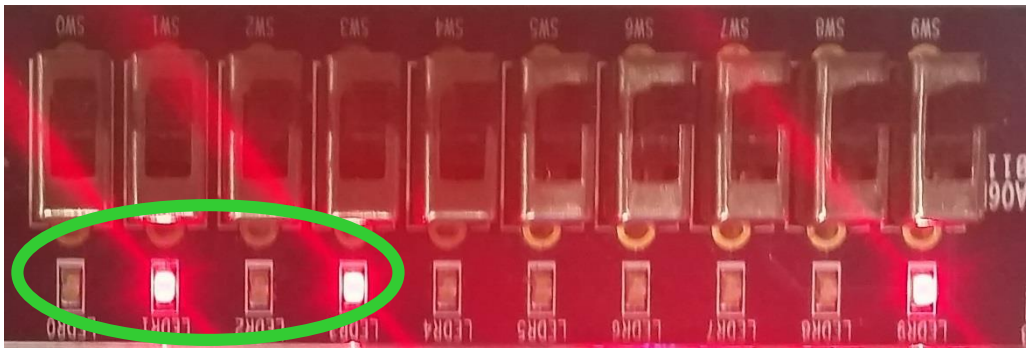


Figura 10. Saída representando o número 5_{10}

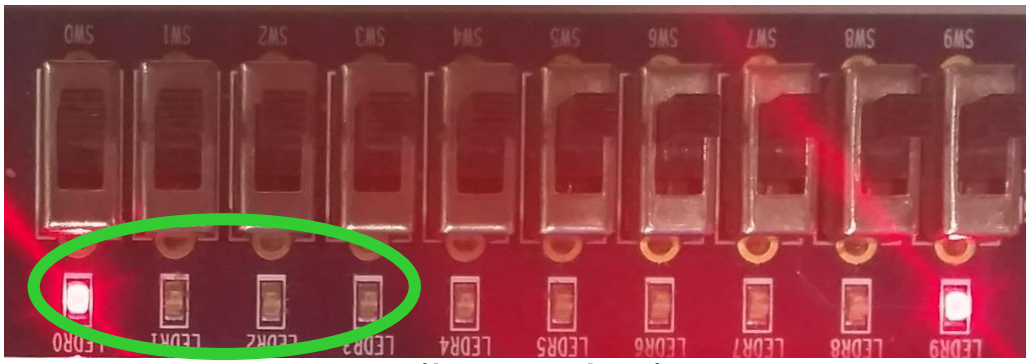


Figura 11. Saída representando o número 8_{10}

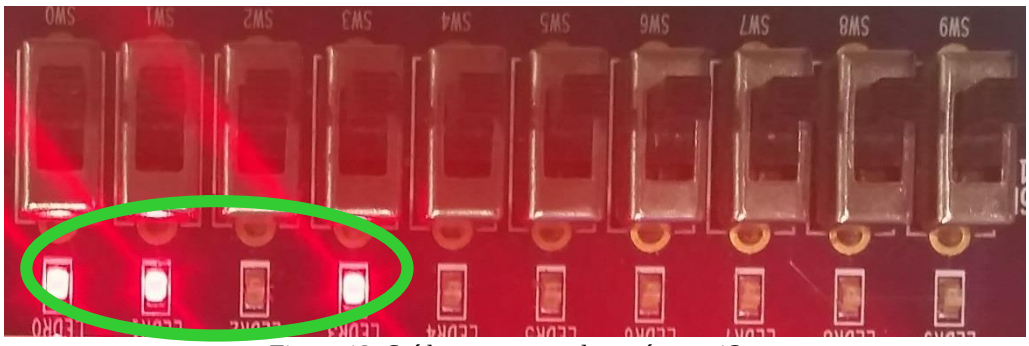


Figura 12. Saída representando o número 13_{10}

4 DISCUSSÃO

Durante a montagem do circuito, tivemos dificuldades, pois a simulação funcional apresentava erros que não deveriam ocorrer e cuja origem não foi totalmente esclarecida. Para contornar esse problema, trocamos alguns componentes do circuito por equivalentes, como o contador.

No teste da placa FPGA, eventualmente, alguns LEDs, que não estavam sendo usados e que não foram configurados se acendiam também por motivos desconhecidos. No entanto, isso não afetou o resultado do experimento.

O *push button* correspondente ao *clock* na placa FPGA, precisa ser pressionado quatro vezes para o circuito passar para o próximo número da sequência, porque o contador é de 2 bits (4 números) e é preciso passar pelas quatro ações do ciclo: registrar o penúltimo número, o último, o atual, e o estado que não corresponde a nenhuma ação. O botão *reset* basta ser pressionado uma vez.

Nosso projeto, apesar de funcionar conforme especificações, não está bem otimizado, devido a limitações de tempo de execução. Por exemplo, na parte do circuito que envolve o contador e o decodificador, há um valor sobrando (são 4 valores disponíveis, mas apenas 3 usados). Isso poderia ser resolvido de algumas formas, como modificar o contador para ele se tornar um contador módulo-3. No início do projeto, tentamos encontrar um contador módulo-3, mas não encotramos um que nos atendesse.

Uma outra proposta de como melhorar o projeto é aumentar o número de bits do circuito, pois com apenas 4 bits, a quantidade de números da sequência que podem ser representados é bem limitado.

5 REFERÊNCIAS

FAIRCHILD SEMICONDUTOR. *DM74LS138, DM74LS139 Decoders/Demultiplexers*. Disponível em: < http://ecee.colorado.edu/~mcclurel/dm74ls138.pdf>. Acesso em: 03 nov. 2015.

LUCHETTA, Valéria Ostete Jannis. *Leonardo de Pisa (Fibonacci)*. Disponível em: <http://www.ime.usp.br/~leo/imatica/historia/fibonacci.html>. Acesso em: 03 nov. 2015.

MOTOROLA. *4-bit binary full adder with fast carry*. Disponível em: <http://pdf.datasheetcatalog.com/datasheet/motorola/SN74LS83D.pdf>. Acesso em: 03 nov. 2015.

TAVARES, Tiago. *Um projeto de registrador: passo a passo*. Disponível em: <http://www.dca.fee.unicamp.br/~tavares/courses/2015s2/relatorio-exemplo.pdf>. Acesso em: 03 nov. 2015a.

_____. *Aula - Implementando um Algoritmo num Circuito Lógico*. 2015b. Vídeo na internet (4min31s). Disponível em: <https://vimeo.com/141446628>. Acesso em: 03 nov. 2015b.

TAVARES, Tiago e COVRE, Marcos. *5 – Um Algoritmo como um Circuito Lógico*. Disponível em: <http://www.dca.fee.unicamp.br/~tavares/courses/2015s2/ea773-5.pdf>. Acesso em: 03 nov. 2015.

VAHID, Franklin. *Sistemas Digitais: projeto, otimização e HDLs*. Bookman: Porto Alegre, 2008.