Relatório do experimento 3: Unidade Lógica e Aritmética

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

1 INTRODUÇÃO

Nesse experimento, projetamos e testamos uma Unidade Lógica e Aritmética (ULA). Trata-se de "um dispositivo que realiza operações lógicas e aritméticas sobre números representados em um circuito lógico" (TAVARES e COUVRE, 2015). As operações que nossa ULA é capaz de realizar são soma e XOR bit a bit. O dispositivo recebe uma entrada de dados e uma entrada de controle para definir qual operação será realizada e gera, na saída, o resultado da operação selecionada e algumas *flags* definidas. Para tanto, são necessárias estruturas para o controle de fluxo de dados assim como os circuitos que implementam as funções desejadas, conforme especificado ao longo do texto.

2 MATERIAIS E PROCEDIMENTOS

Nesse experimento, projetamos os circuitos no ambiente virtual Quartus II, assim como no anterior. No entanto, dessa vez, pudemos testá-lo usando uma placa FPGA (*Field-Programmable Gate Array*), que possui LEDs e botoeiras, associados a saídas e entradas do circuito, respectivamente.

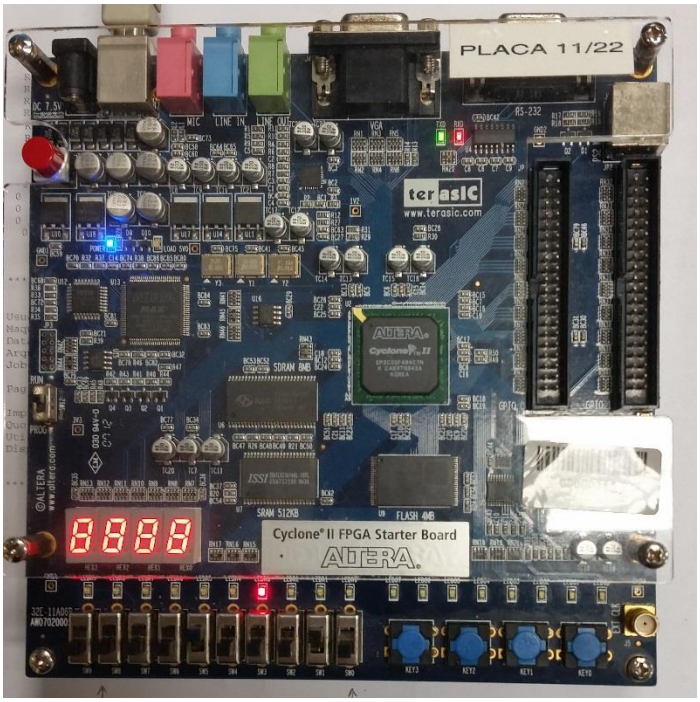

Figura 1. Foto da placa FPGA

Para a montagem da unidade lógica e aritmética, precisamos de: (a) estruturas de controle de fluxo de dados – que foram implementados utilizando multiplexadores e demultiplexadores, a partir de componentes *tri-state* – e (b) circuitos que implementem a soma e a operação XOR bit a bit. Abaixo, apresentamos o esquema simplificado do nosso circuito lógico e, mais adiante, discutimos em mais detalhes cada parte dele.

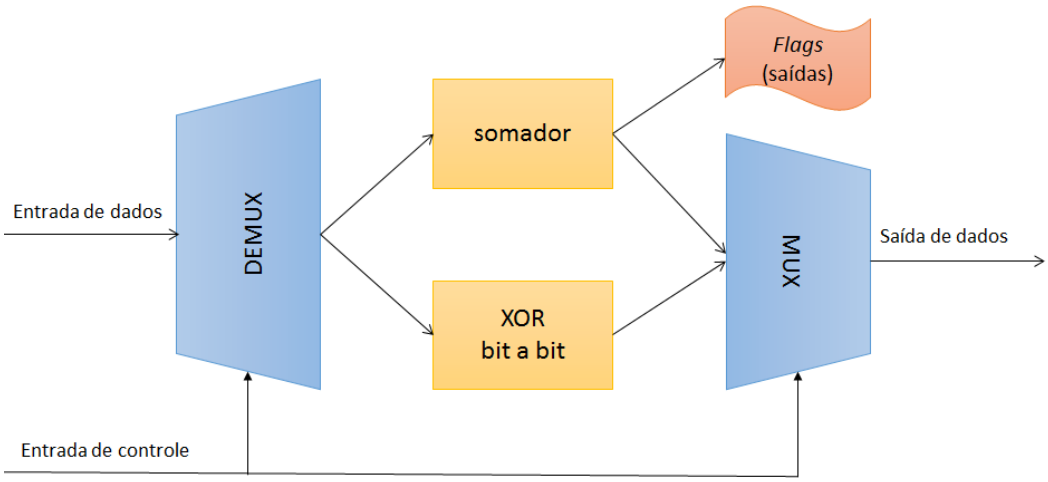

Figura 2. Esquema simplificado da nossa ULA

2.1 DEMULTIPLEXADOR

Um demultiplexador (DEMUX) ou distribuidor é um sistema combinacional que copia os dados de entrada para a saída selecionada pela entrada de controle e deixa todas as outras saídas em zero (ERCEGOVAC et al., 2000, p.234).

Na nossa implementação, primeiramente montamos um demultiplexador de 1 bit, com uma entrada de dados e duas saídas, a partir de componentes *tri-state*. Um *buffer tri-state* tem uma entrada adicional, de controle, que, quando habilitada, permite a passagem de dados da entrada para a saída e, quando desabilitada, força a saída para um estado de alta impedância, e o componente passa a se comportar como uma chave aberta. A imagem abaixo mostra o diagrama do demultiplexador de 1 bit.

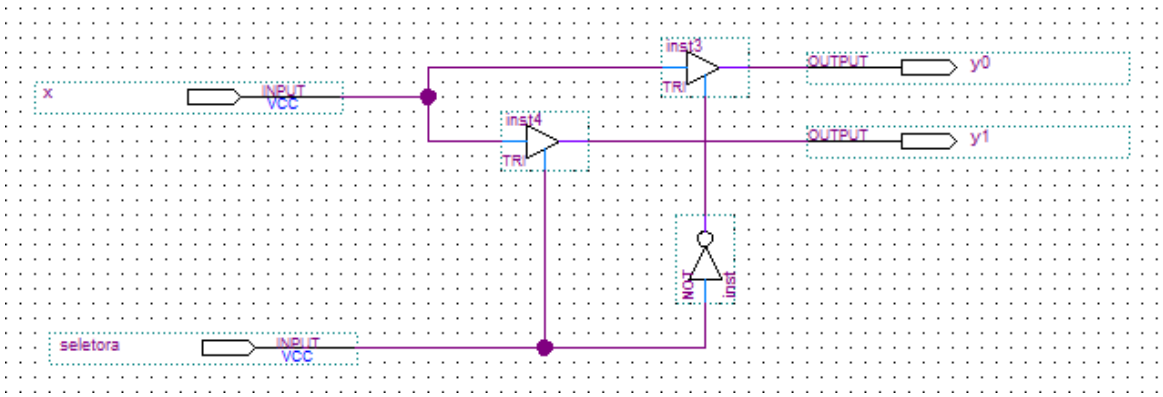

Figura 3. Diagrama do demultiplexador de 1 bit

A seguir, realizamos o encapsulamento do nosso demultiplexador de 1 bit e, com quatro deles, montamos um demultiplexador de 4 bits.

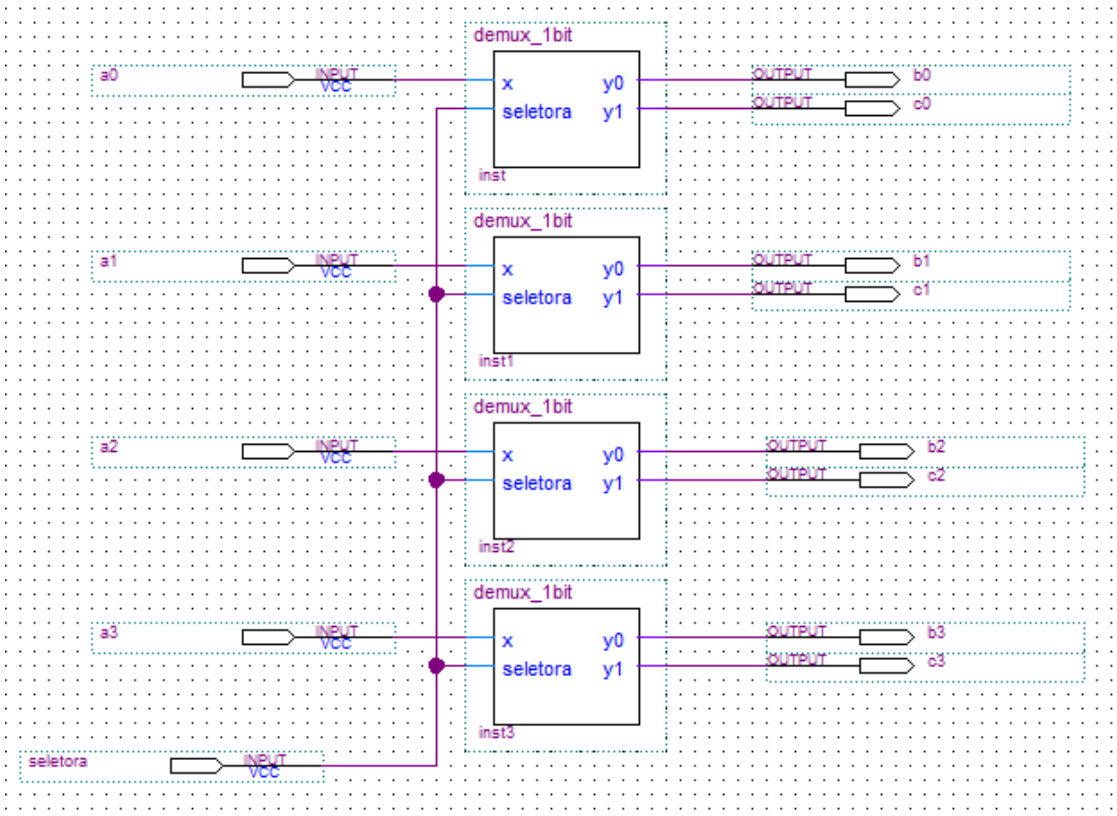

Figura 4. Diagrama do demultiplexador de 4 bits

Esse circuito também foi encapsulado, resultando no seguinte bloco lógico.

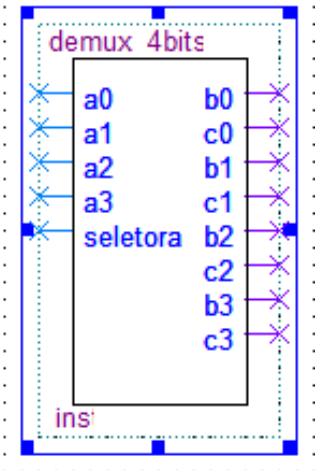

Figura 5. Bloco lógico demultiplexador de 4 bits

2.2 MULTIPLEXADOR

Um multiplexador (MUX) ou seletor é um sistema combinacional que copia em sua saída os dados da entrada selecionada pela entrada de controle (ERCEGOVAC et al., 2000, p.229). Da mesma forma que com o demultiplexador, primeiramente projetamos um multiplexador de 1 bit (figura 6), a partir do qual projetamos um multiplexador de 4 bits (figura 7), que foi encapsulado em um bloco lógico (figura 8).

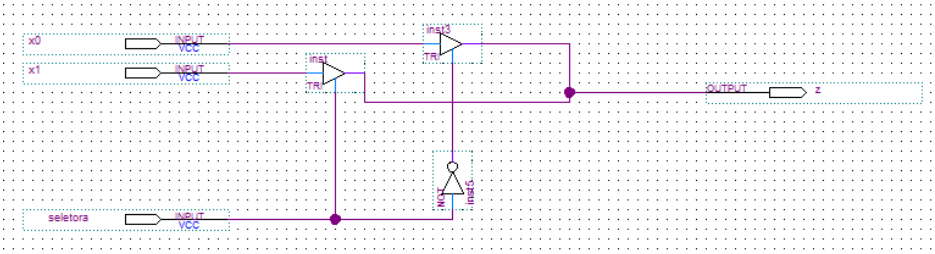

Figura 6. Diagrama do multiplexador de 1 bit

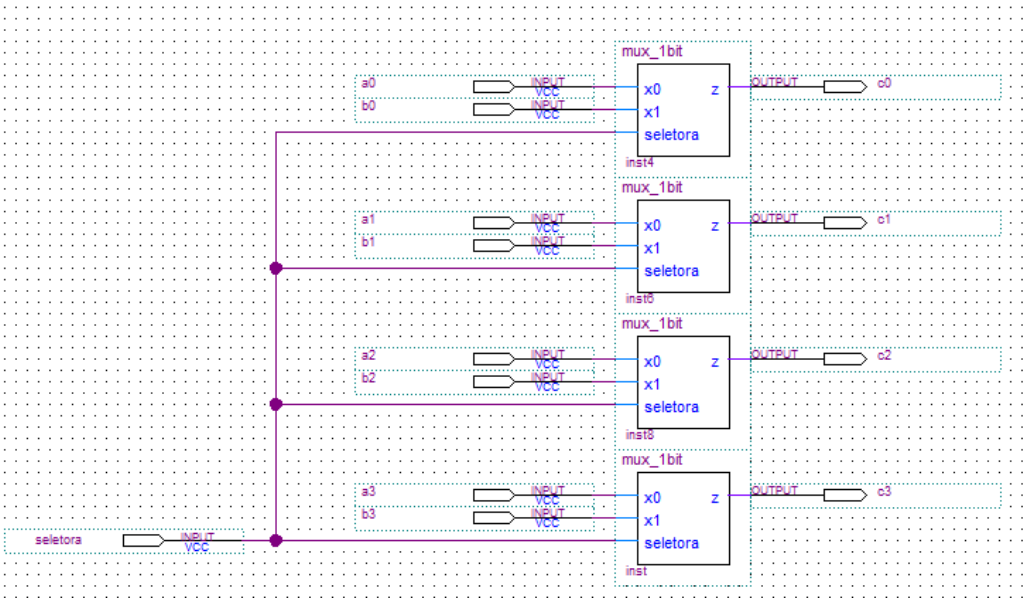

Figura 7. Diagrama do multiplexador de 4 bits

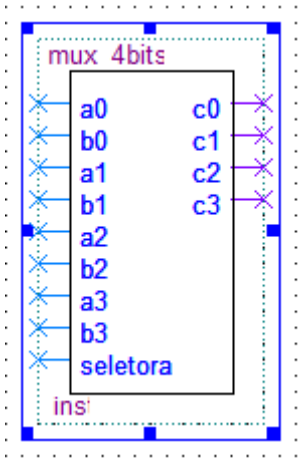

Figura 8. Bloco lógico do multiplexador de 4 bits

2.3 SOMADOR

Um somador binário é um sistema combinacional com duas entradas que são vetores de bits e uma saída que representa a soma desses vetores de bits. Sinais de entrada e de saída adicionais, *carry-in* e *carry-out*, respectivamente, são utilizados para facilitar a implementação de somadores maiores, como no nosso caso de somadores de 4 bits. A estratégia aqui utilizada para aumentar o número de níveis do somador foi a de transporte propagado (*carry ripple*), em que o *carry-out* de um nível é somado ao próximo, na forma de *carry-in*. Uma especificação de alto nível para tal sistema é mostrada a seguir (ERCEGOVAC et al., 2000, p.250).

Entrada

$x_i, y_i, c_i \in \{0, 1\}$

Saída

$z_i, c_{i+1} \in \{0, 1\}$

Função de saída

$$z_i = (x_i + y_i + c_i) \bmod 2$$

$$c_{i+1} = \begin{cases} 1 & \text{se } (x_i + y_i + c_i) \geq 2 \\ 0 & \text{caso contrário} \end{cases}$$

Figura 9. Especificação de alto nível para somador de transporte propagado

Mais uma vez, primeiramente projetamos o módulo somador de 1 bit (figura 10), que foi encapsulado para projetarmos o módulo somador de 4 bits (figura 11), que foi, por sua vez, transformado em um bloco lógico único (figura 12). O somador de 1 bit segue a implementação com meio-somadores, apresentada por Ercegovac et al. (p. 253).

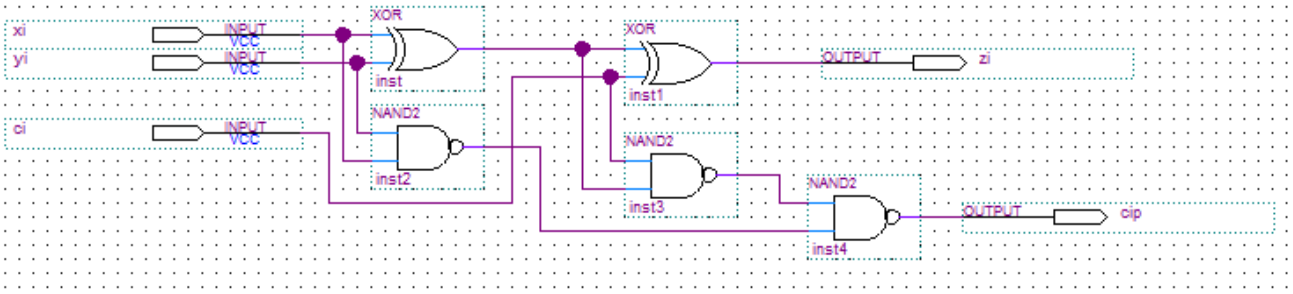

Figura 10. Diagrama do somador de 1 bit

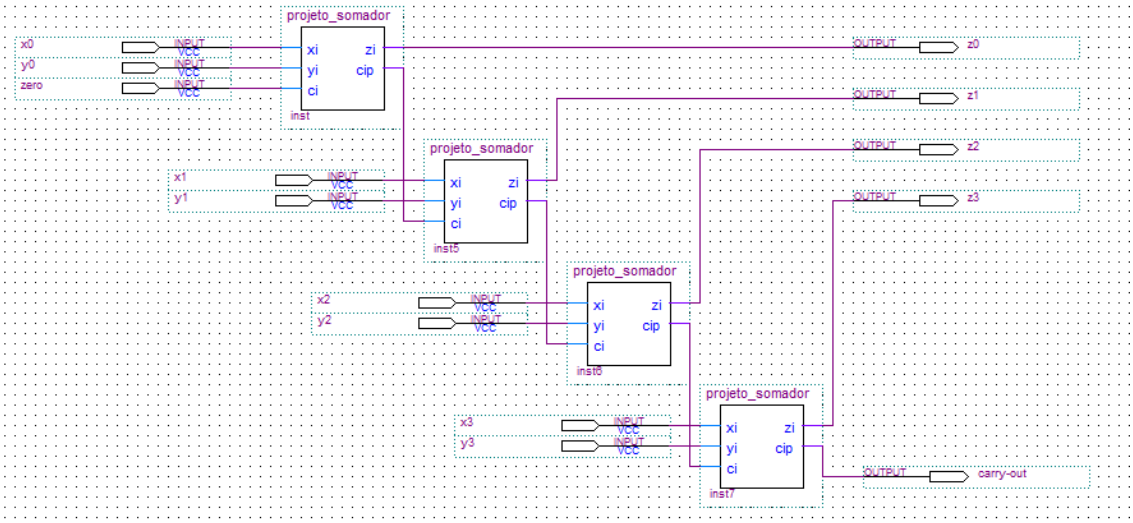

Figura 11. Diagrama do somador de 4 bits

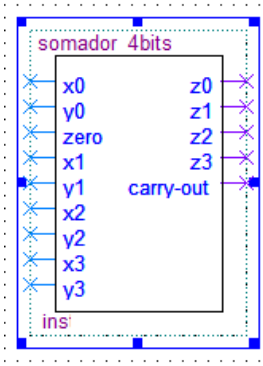

Figura 12. Bloco lógico somador de 4 bits

A entrada "zero" do somador deve estar ligada permanentemente a um sinal desabilitado. Isso porque trata-se da entrada *carry-in* do primeiro nível de soma. Além da saída de 4 bits que representa o resultado da soma binária, há a saída *carry-out*, que indica se houve "vai um" no bit mais significativo.

2.4 OPERADOR XOR BIT A BIT

O operador lógico "ou exclusivo" ou "*exclusive or*" (XOR) fornece 1 à saída quando as duas entradas forem diferentes, conforme tabela abaixo (IDOETA et al., 2006, p. 68).

Tabela 1. Tabela-verdade da função XOR

A	B	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Nosso operador lógico que realiza a função XOR em cada um dos 4 bits de entrada foi montado e encapsulado conforme os diagramas a seguir.

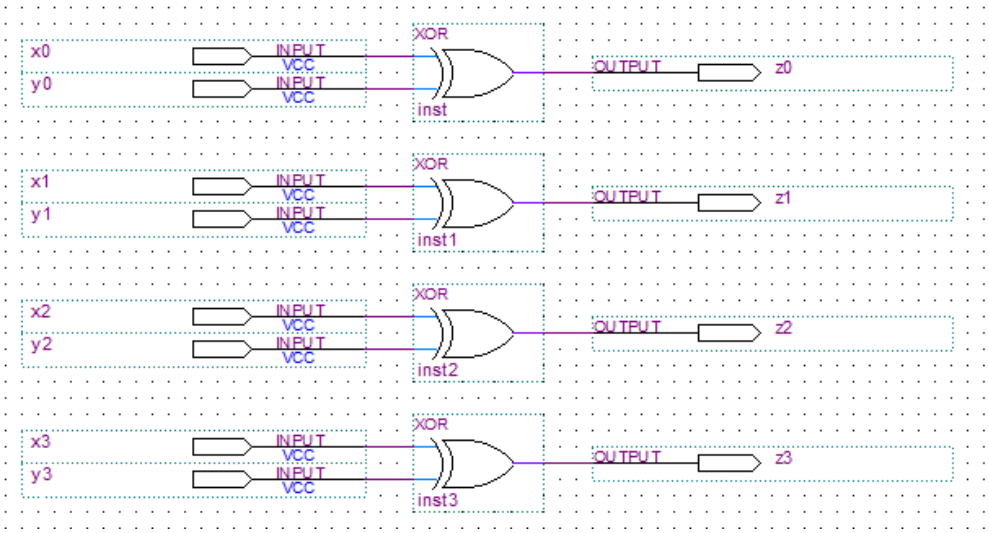

Figura 13. Diagrama do operador XOR bit a bit

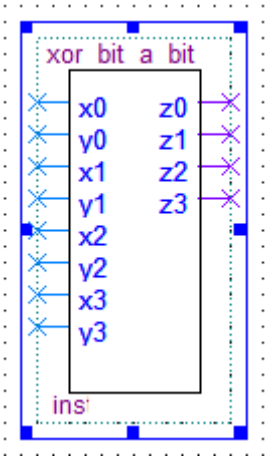

Figura 14. Bloco lógico XOR bit a bit

2.5 *FLAGS*

As *flags* são bits de saída que indicam algum status do circuito. Nosso circuito possui duas *flags*, ambas associadas ao circuito somador: uma para indicar se o resultado da soma for zero (*flag* Z) e outra para indicar se houve *carry-out* no bit mais significativo do operador (*flag* C). Tais *flags* foram associadas a LEDs da placa FPGA, que se acendem quando as situações referentes ocorrem no circuito.

Para a implementação da *flag* C bastou ligar a saída do *carry-out* do somador a uma saída do circuito. Para *flag* Z, ligamos todas as saídas dos somadores a uma porta NOR e a saída desta se tornou a *flag*. Assim, se todos os bits forem zero (valor 0 em decimal), a saída será ativa.

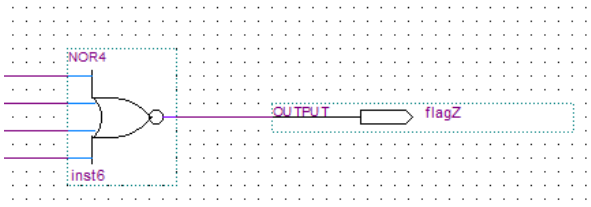

Figura 15. Diagrama da porta NOR para implementação da *flag* Z

2.6 CIRCUITO COMPLETO

A montagem final do circuito está mostrada no diagrama a seguir.

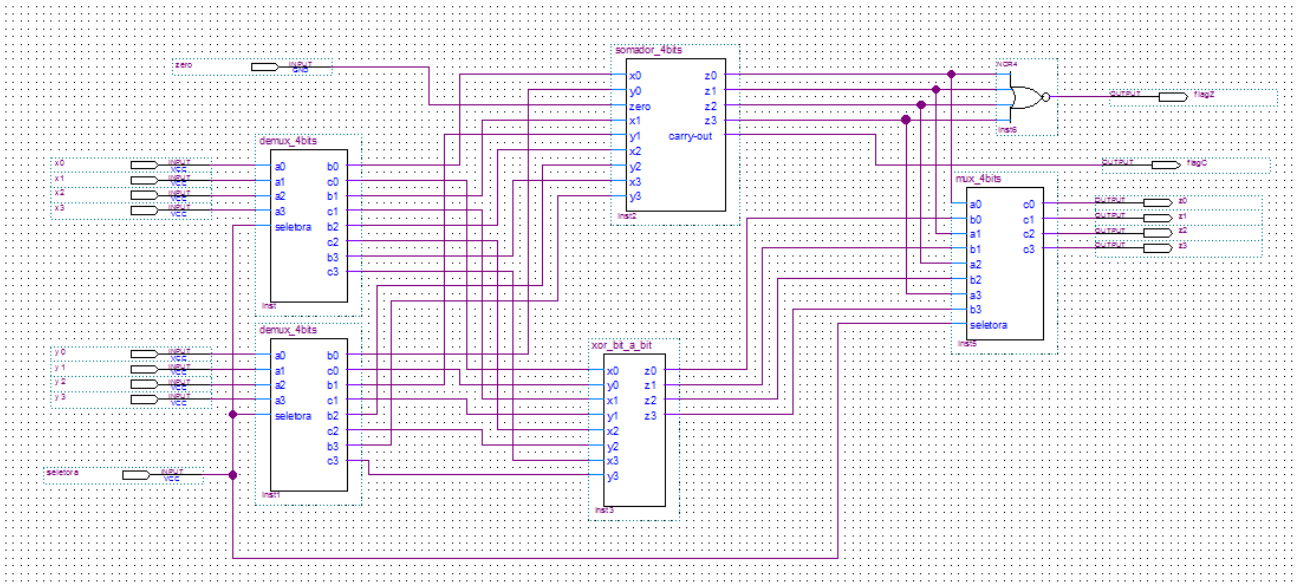

Figura 16. Diagrama do circuito completo

Para o teste, associamos cada entrada e saída a uma chave ou LED, respectivamente, do circuito. Para tanto, o circuito completo foi encapsulado em um único bloco. Foram adicionadas, ainda, duas portas AND para fazer com que as *flags* só sejam ativadas quando a operação soma tiver sido selecionada.

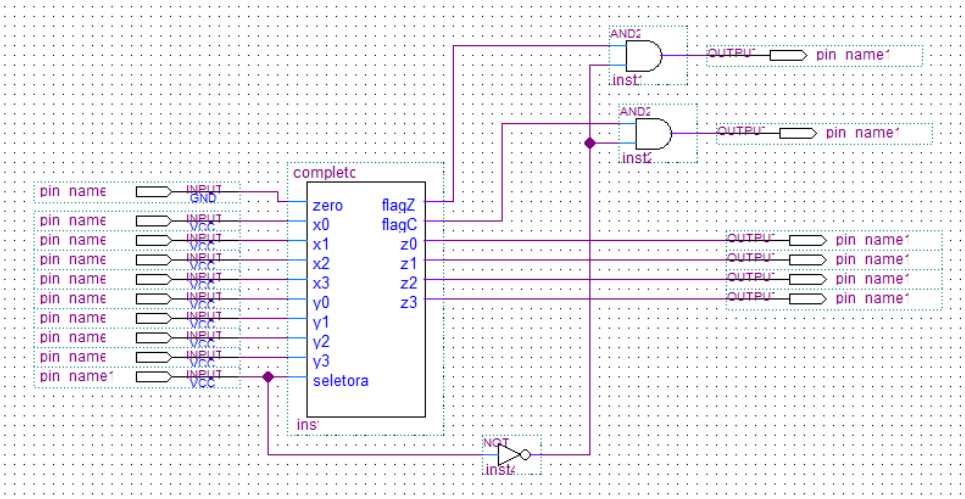

Figura 17. Bloco lógico do circuito completo

3 RESULTADOS

Para realizar os testes, configuramos a placa FPGA da seguinte maneira: o primeiro botão da direita para esquerda é o *carry-in* (entrada "zero", deve ficar sempre desabilitada), os botões de 2 a 5 e de 6 a 9 correspondem a números de 4 bits com algarismos mais significativos à esquerda (vetores $\bar{x}$ e $\bar{y}$). O décimo botão é a entrada seletora (0 para soma e 1 para XOR bit a bit). O primeiro LED à esquerda corresponde à *flag* Z, o segundo, à *flag* C e, os quatro seguintes, ao vetor de bits da saída.

O circuito funcionou como desejado. Apresentamos, a seguir, resultados de diversos testes.

Teste 1: Soma $0000 + 0000 = 0000$, flagC $= 0$, flagZ $= 1$

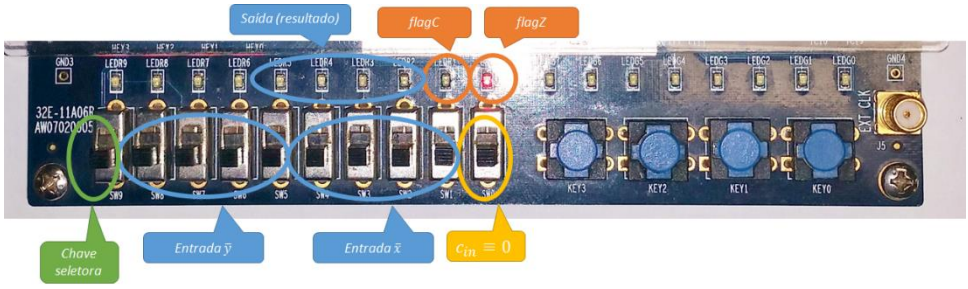

Figura 18. Foto do teste 1

Teste 2: Soma $0001 + 0001 = 0010$, flagC $= 0$, flagZ $= 0$

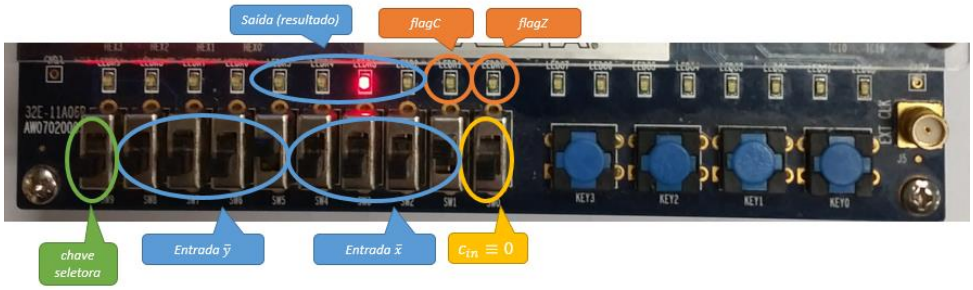

Figura 19. Foto do teste 2

Teste 3: Soma $0101 + 1110 = 0011$, flagC $= 1$, flagZ $= 0$

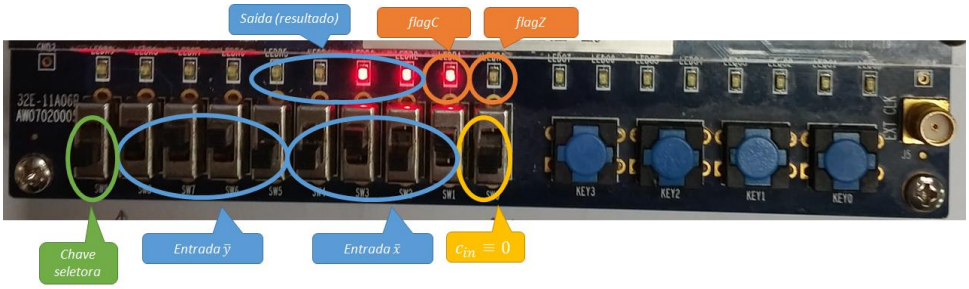

Figura 20. Foto do teste 3

Teste 4: Soma $1111 + 0001 = 0000$, flagC $= 1$, flagZ $= 1$

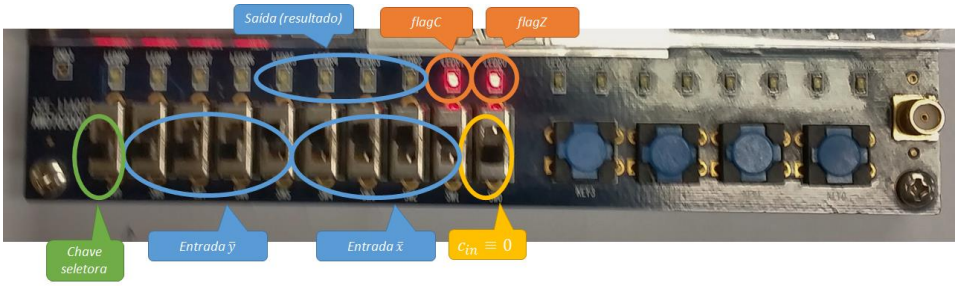

Figura 21. Foto do teste 4

Teste 5: XOR bit a bit $0011 \oplus 0000 = 0011$

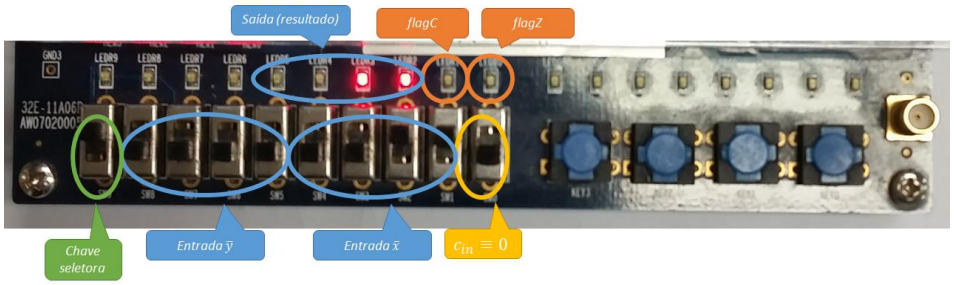

Figura 22. Foto do teste 5

Teste 6: XOR bit a bit $1010 \oplus 0011 = 1001$

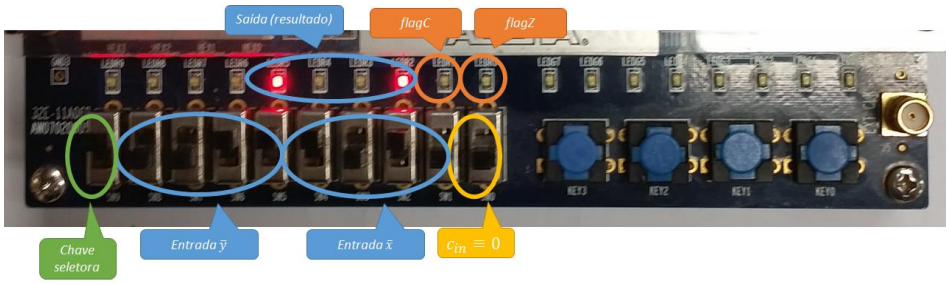

Figura 23. Foto do teste 6

Nos testes 5 e 6, os LEDs das *flag* Z e *flag* C não acendem, pois essas saídas só são referentes à operação soma.

4 DISCUSSÃO

Nesse relatório, procuramos descrever o processo de projetar e montar uma ULA, conforme apresentado na seção 1. Para tanto, dividimos o problema em subproblemas menores (DEMUX, MUX, somador, XOR bit a bit), encapsulando cada etapa, para facilitar o projeto. Isso se mostrou vantajoso, pois podíamos alterar cada parte separadamente, se necessário, e a visualização do

circuito se tornava mais simples no ambiente do Quartus II. Além disso, cada subetapa foi dividida em elementos que realizam a operação com apenas 1 bit de entrada, a partir dos quais foram montados os blocos lógicos que operam com entradas de 4 bits.

Um problema que surgiu durante o projeto foi que, no multiplexador de 1 bit, tentamos ligar as saídas dos dois *buffers* com uma porta OR, o que gerou problemas no circuito. Descobrimos, então, que isso não era necessário, uma vez que o *buffer* destivado estaria em alta impedância. Outra ocorrência foi que, ao fim do circuito, primeiramente, não havíamos inserido as portas AND citadas em 2.6, o que provocou resultados indesejáveis nas *flags*, quando a operação XOR era soliciada. Como descrito em 2.6, as portas AND fazem com que as *flags* só sejam ativadas durante a operação soma. Uma última dificuldade a ser citada foi a configuração das entradas e saídas da placa FPGA para carregarmos o circuito nela, o que ocorreu apesar dos tutoriais, devido à inexperiência com o processo.

Ao fim, o experimento se mostrou muito proveitoso, pois, devido ao grande número de partes em que foi dividido, foi possível colocar em prática uma diversidade de conceitos e técnicas aprendidas.

5 REFERÊNCIAS

TAVARES, Tiago; COUVRE, Marcos. *3 - Unidade Lógica e Aritmética*. Disponível em: <http://www.dca.fee.unicamp.br/~tavares/courses/2015s2/ea773-3.pdf>. Acesso em: 20 ago. 2015.

ERCEGOVAC, Milos D. ; LANG, Tomás ; MORENO, Jaime H. *Introdução aos sistemas digitais*. Tradução: José Carlos Barbosa dos Santos. Porto Alegre: Bookman, 2000.

IDOETA, Ivan Valeije; CAPUANO, Francisco Gabriel. *Elementos de eletrônica digital*. 37ª. Edição. São Paulo: Érica, 2006.