EEL480 - Laboratório de Sistemas Digitais Relatório 01 Unidade Lógica e Aritmética (ULA) de 4 bits desenvolvida em Hardware Description Language (VHDL)

Karen Silva Pacheco[1], **João Pedro Martins Filipe Figueiredo Matos Menezes Pereira**[2]

[1]Escola Politécnica - Universidade Federal do Rio de Janeiro

Resumo. *Este relatório descreve a metodologia definida para a construção de uma unidade logica e aritmética (ULA) de 4 bits com oito operações, através de uma linguagem de descrição de hardware (VHDL - Hardware Description Language), produzido para a matéria de sistemas digitais, ministrada pelo professor Roberto Pacheco.*

1. Introdução

Este trabalho tem como objetivo desenvolver, em VHDL, uma ULA (Unidade Lógica e Aritmética) de 4 bits e 8 operações utilizando o conhecimento adquirido nas aulas teóricas e práticas da disciplina.

VHDL significa VHSIC (Very High Speed Integrated Circuits) Hardware Description Language (Linguagem de Descrição de Hardware). O código em VHDL permite descrever em software o comportamento e a estrutura do hardware de qualquer sistema digital.

Através da utilização de uma rede combinacional, selecionaremos as entradas e operações executadas na ULA, as quais são: soma, subtração em complemento de 2, incremento de 1, e decremento de 1 ,troca de sinal, multiplicação por 2, divisão por 2 e comparação de números naturais. Para a exibição dos resultados, além de 4 flags (Zero, sinal, carry out e overflow).

2. Implementação de módulos combinacionais

Inicialmente, foi construído 4 módulos lógicos, pensados para cada um das operações citadas acimas, de tal forma que fosse necessário utilizar a menor quantidade de portas lógicas possíveis e, ao mesmo tempo ter um sistema modular que pudesse realizar todas as operações de forma otimizada.

2.1. Somador Paralelo de 4 bits

Este módulo foi construído utilizado com base em somadores completos de 1 bit, sendo possível aglutinar 4 módulos somadores de 1 bit para construir um somador paralelo de 4 bits de entrada, sendo 1 bit de sinal.

O circuito somador completo (Full Adder) recebe três bits de entrada A, B e Cin (o último correspondendo ao eventual carry gerado na operação com bits menos significativos) e produz dois bits de saída: o Bit de Soma (S = A + B) e o Bit de Carry (Cout).

Componentes utilizados: 3 portas ANDs, 1 porta OR e 1 XOR

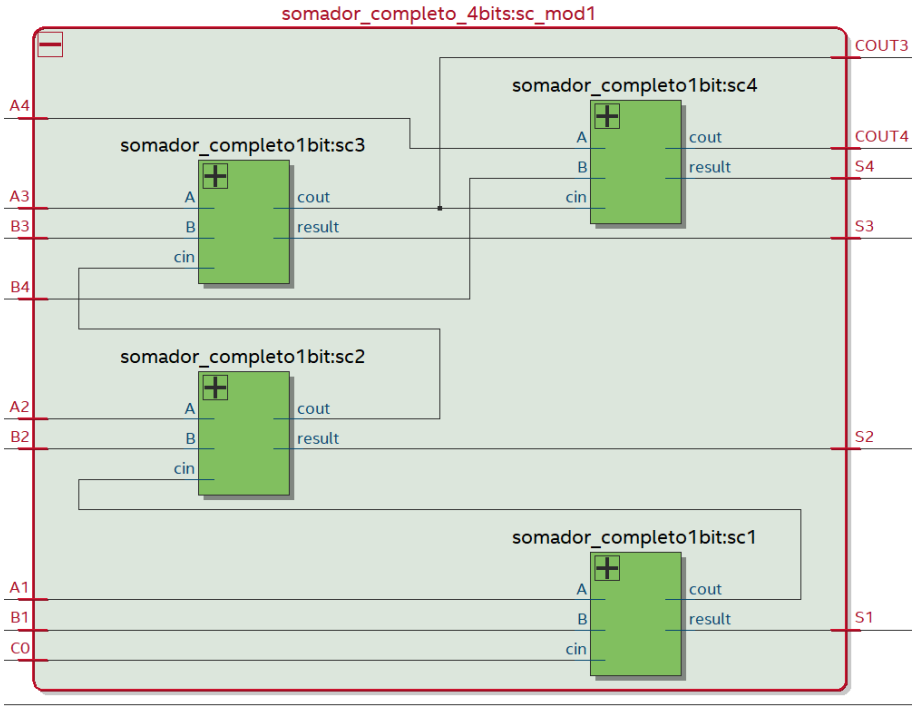

Figura 1. Módulo Somador Paralelo de 4 bits

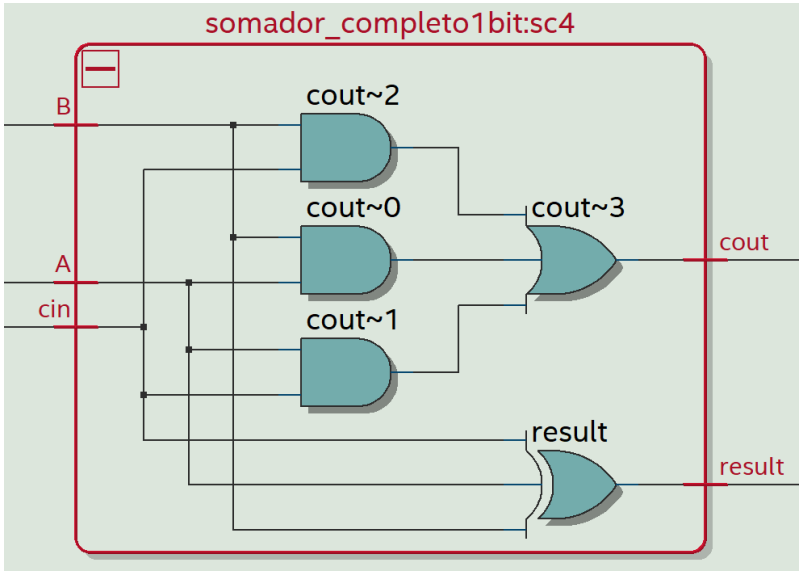

Figura 2. Portas lógicas do módulo Somador completo 1 bit

Suas vantagens geram em torno da simplicidade e a modularidade do circuito; sua desvantagem, porém, é ser muito lento: seu atraso é aproximadamente igual à soma dos atrasos das saídas “vai-um” dos circuitos de soma completa.

2.2. Complementador

A operação de complemento a = COMPL(B, K), sendo B a entrada de um dos operandos da ALU, e K a entrada de controle deste módulo, é definida como:

$$a_i = \begin{cases} b_i & \text{se } K = 0 \\ b'_i & \text{se } K = 1 \end{cases}$$

Figura 3. Relação entre entrada e saída do circuito Complementador

O objetivo desse circuito é inverter os bits de entrada, realizando o complemento de 1, através da entrada de controle (Kx/Ky) que define se as portas de saída sairão invertidas ou não.

No projeto, foi alocado dois complementadores, um para cada operando da ALU.

Componentes utilizados: 4 portas XOR, alocadas entre os bits de entrada de dados e a entrada de controle.

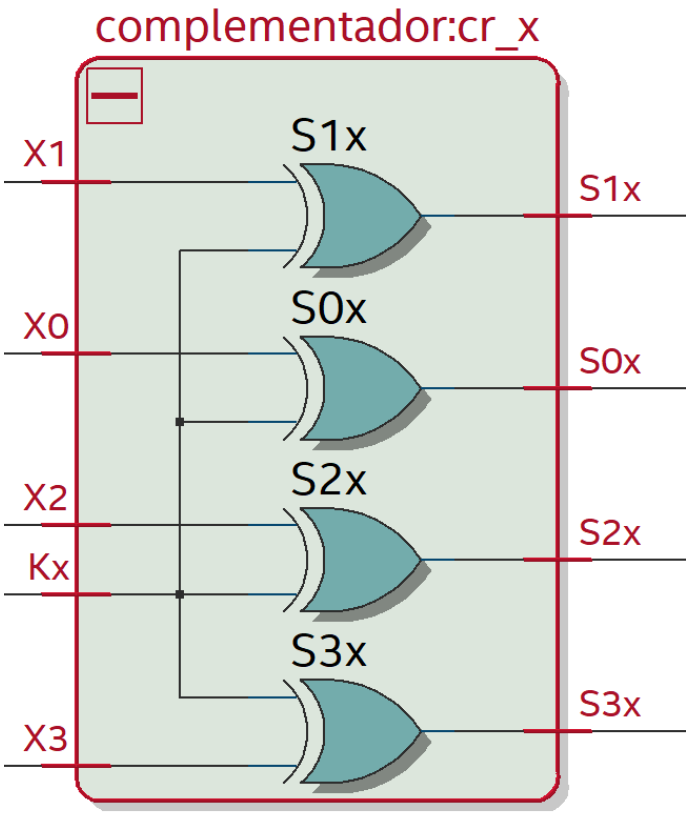

Figura 4. Portas lógicas - complementador

2.3. Deslocador

Um deslocador é um circuito combinacional com entradas x = (xn, xn-1, ..., x0,x-1) com (n+2) bits e uma saída y = (yn-1, ..., y0). Além disso, há duas entradas adicionais de controle:

- d para indicar a direção do deslocamento, ou seja, esquerda ou direita
- s para indicar se há ou não deslocamento

Este módulo desloca os dados de entrada em um bit, ou para esquerda ou para direita, dependendo do valor de d, ou entrega os dados da entrada sem modificação, conforme s.

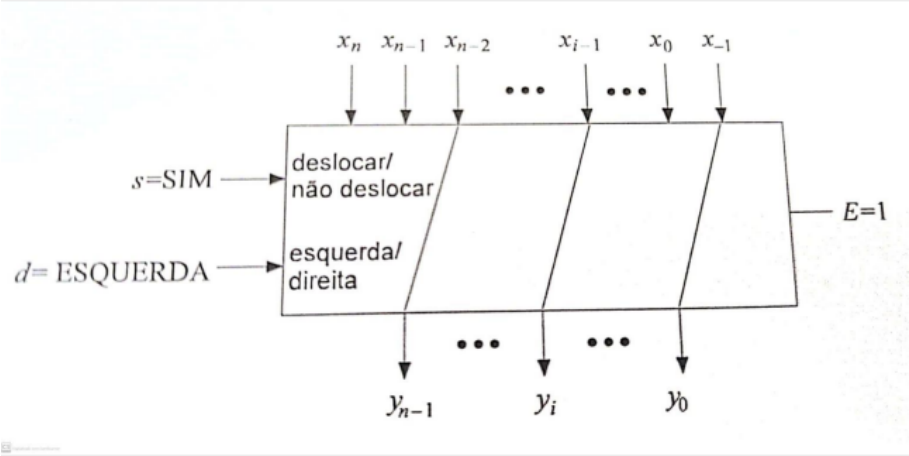

Figura 5. Módulo Deslocador

A implementação do módulo deslocador pode ser feita por meio de portas lógicas ou de multiplexadores, neste caso, escolhemos construir com multiplexadores, aglutinando 4 multiplexadores 4:1 de forma paralela.

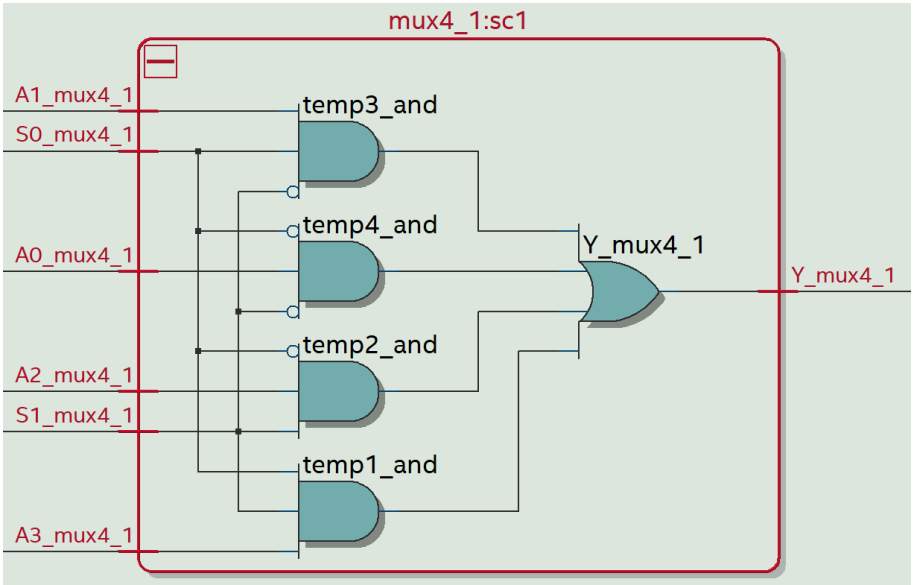

Figura 6. Deslocador

Componentes utilizados: 4 multiplexadores 4:1, sendo eles construídos com 4 portas ANDs e 1 porta OR.

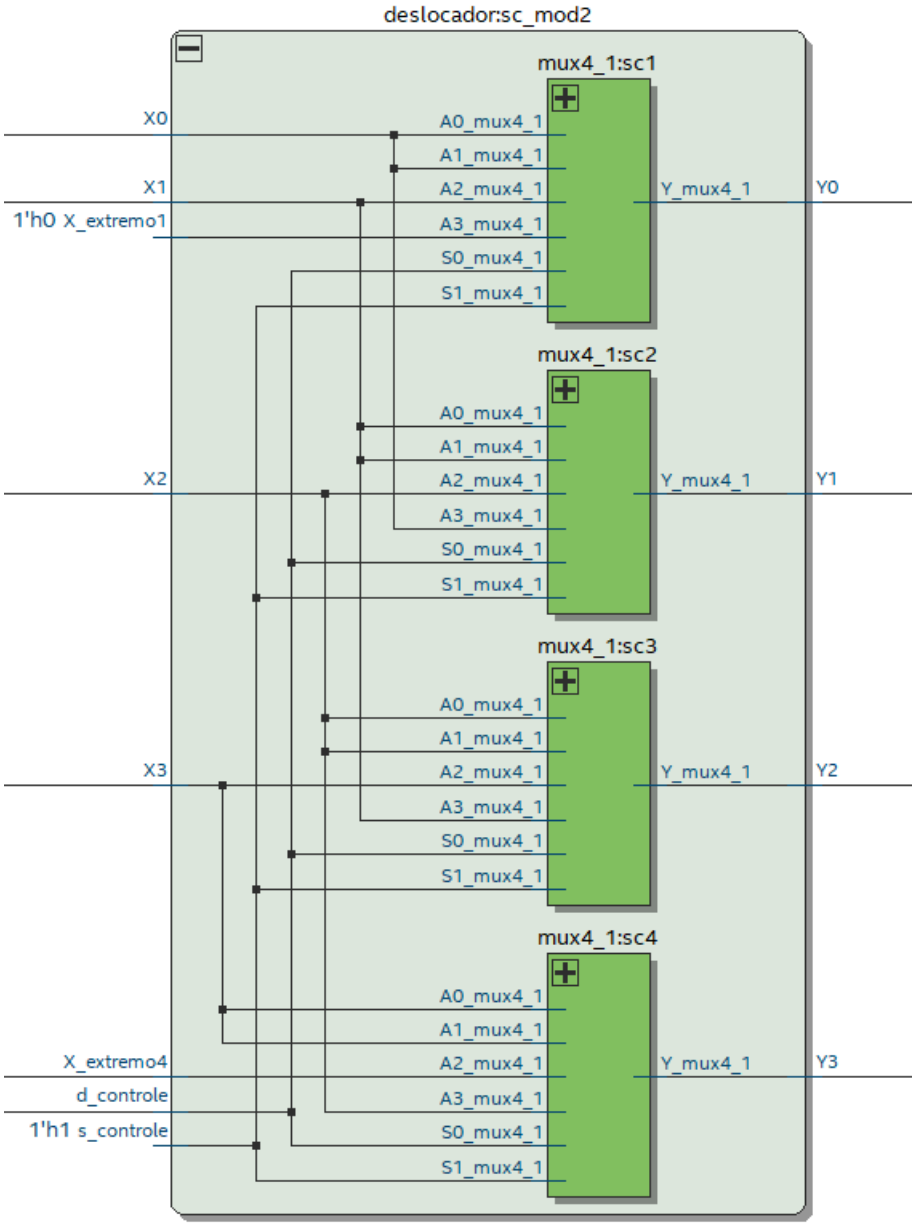

Figura 7. Módulo Deslocador

2.4. Comparador

Este circuito faz a operação de comparação entre números naturais, retornando 3 saídas, sendo uma delas acesa, indicando se o operando A é GREATER(MAIOR), EQUAL(IGUAL) ou SMALLER(MENOR) que o operando B.

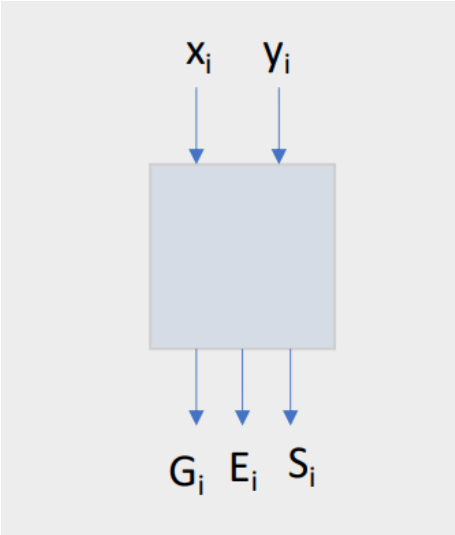

Figura 8. Esquematização do módulo Comparador

Um módulo comparador de n bits é um circuito combinacional com duas entradas de n bits x =(xn-1, ..., x0) e y = (yn-1, ..., y0) e saída z indica a relação entre as magnitude das duas entradas. A saída pode assumir três valores:

- Greater (G) - x maior que y
- Equal (E) - os operandos são iguais
- Smaller (S) - x menor que y

Ainda há uma entrada adicional cin com valores (G, E, S) que é usada para facilitar a implementação.

x_i	y_i	G_i	E_i	S_i
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Figura 9. Tabela verdade - Comparador

Componentes: 12 portas lógicas ANDs, 4 portas XOR, 2 OR e 32 portas NOT

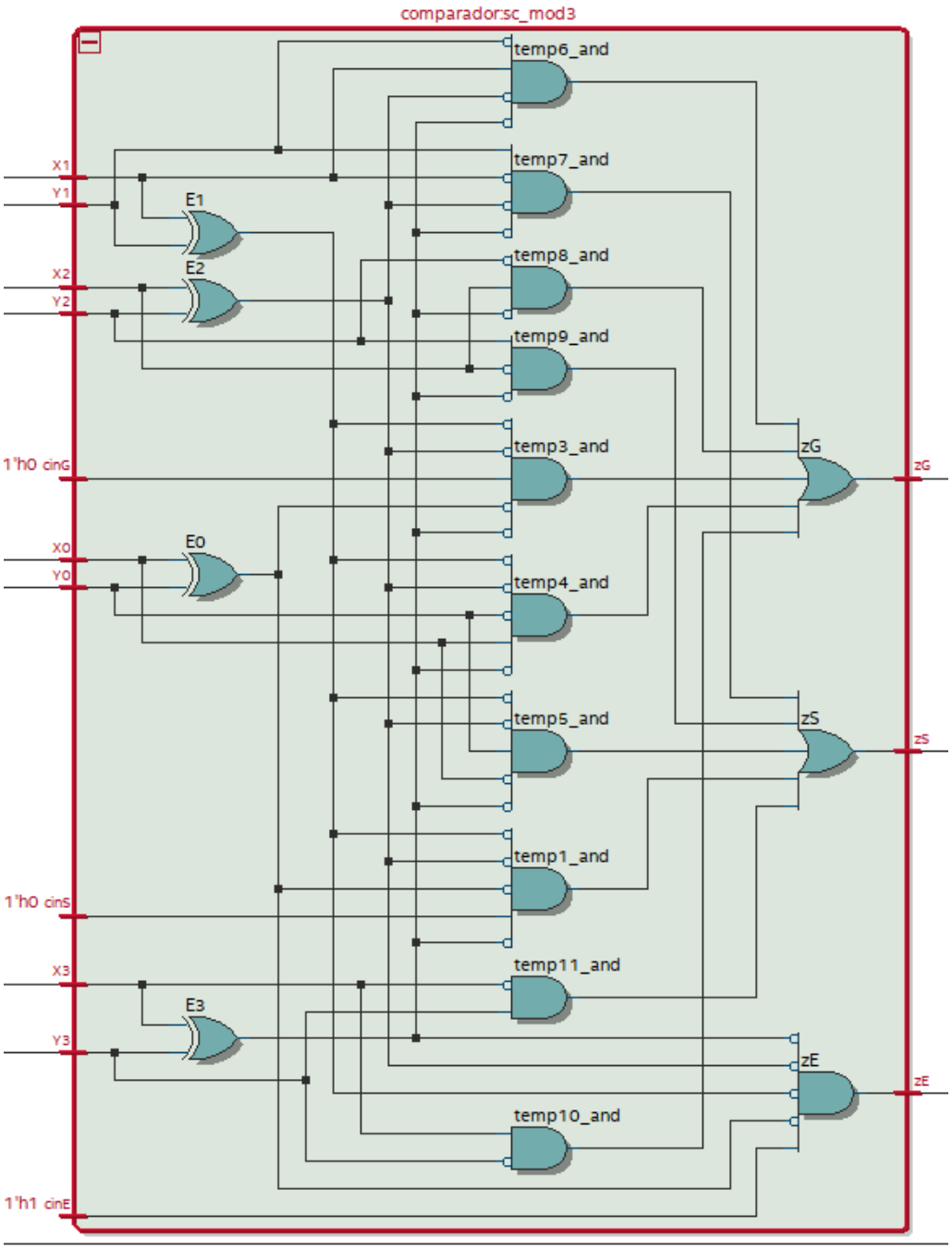

Figura 10. Portas lógicas - circuito comparador

2.5. Multiplexadores

O Multiplexador é um circuito capaz de colocar sequencialmente várias informações paralelas. O multiplexador é utilizado para enviarmos as informações contidas em vários canais a um só canal.

Para o controle de saídas das informacoes da ALU foram utilizados:

- Multiplexador 8:4 com 1 entrada de controle
- Multiplexador 16:4 com 2 entradas de controle

A saída do MUX é igual a entrada selecionada pelas entradas de controle. Dessa forma, o sinal de controle faz com que a saída z corresponda a um determinado sinal da entrada.

O Multiplexador 8:4 foi construído através da junção de 4 Multiplexadores 2:1 de forma paralela.

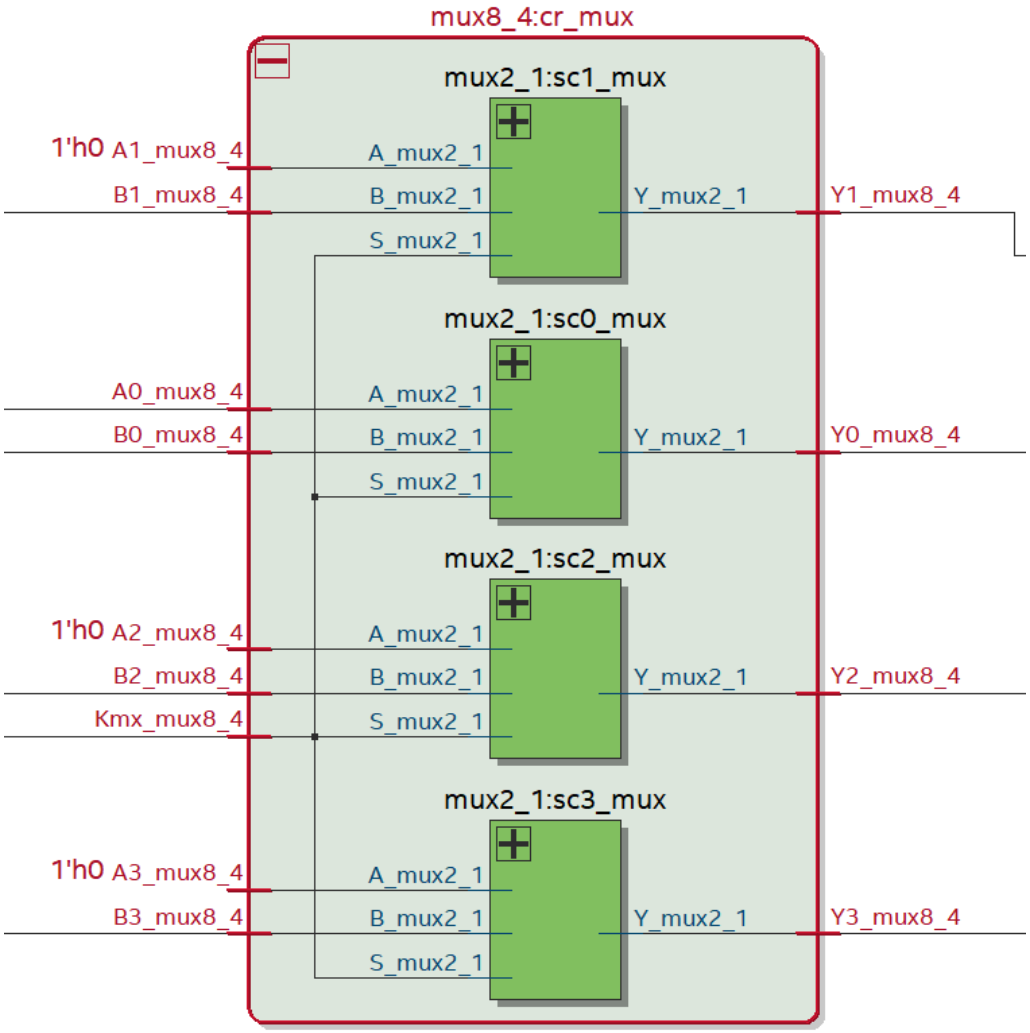

Figura 11. Módulo Multiplexador 8:4

Componentes utilizados: 2 portas lógicas ANDs, 1 porta OR e 1 porta NOT

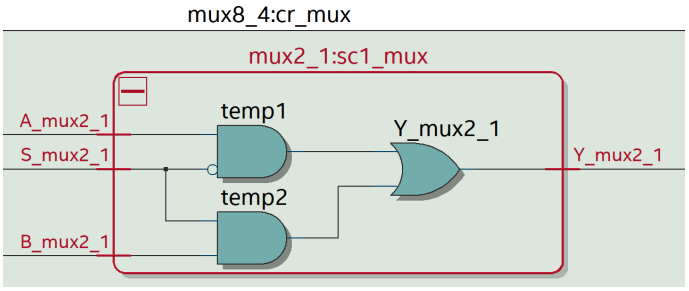

Figura 12. Portas lógicas - Multiplexador 8:4

O mecanismo utilizado para construir este multiplexador 16:4 tem a mesma ideia do anterior. Tem como base o aglutinamento de 4 multiplexadores 4:1 de forma paralela, reaproveitando o circuito criado anteriormente.

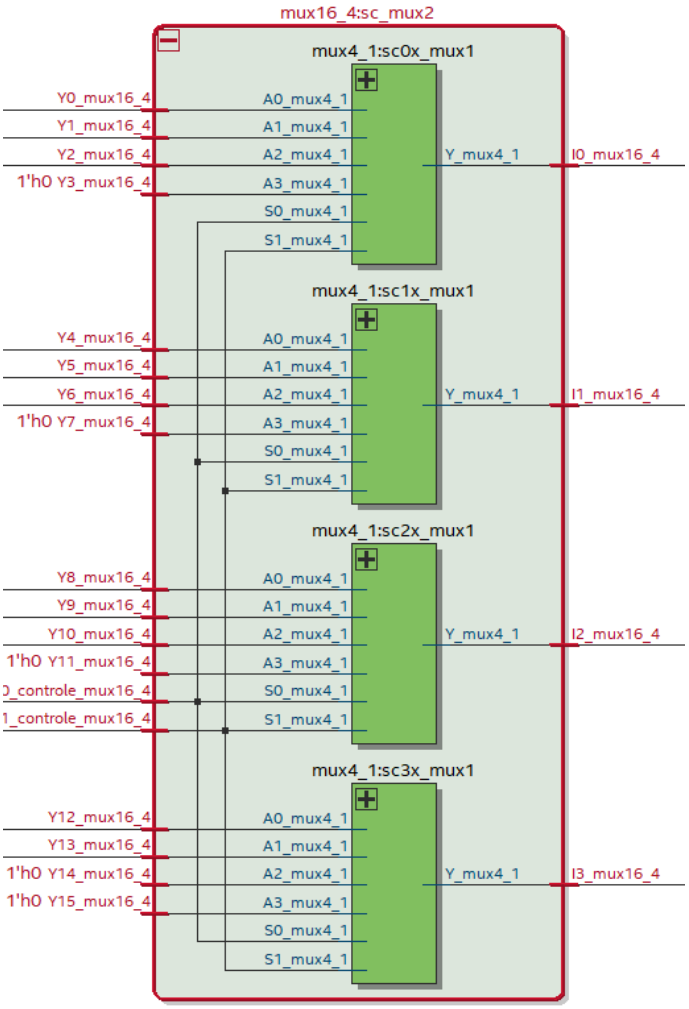

Figura 13. Módulo Multiplexador 16:4

No módulo geral da ALU, foi utilizado 2 multiplexadores para filtrar as saídas dos módulos operacionais, a partir do seletor de operações.

2.6. Demultiplexador

O demultiplexador é um sistema combinacional com n entradas de controle s = (sn-1, ..., s0), apenas uma entrada de dados x e 2n saídas y = (y2n-1, ..., y0).

Este módulo executa a função inversa do MUX. Ele roteia os dados de entrada para a saída selecionada e todas as outras são zero.

Existem 3 tipos de sinais que passam por um demultiplexador: entrada de dados, saída e entrada de controle. O demultiplexador conecta o pino de entrada a um dos pinos de saída. Qual pino vai ser selecionado depende do estado dos pinos de comando.

Dependendo dos sinais enviados ao demultiplexador ele é capaz de conectar um canal a outros, como se fosse uma chave seletora ou um roteador.

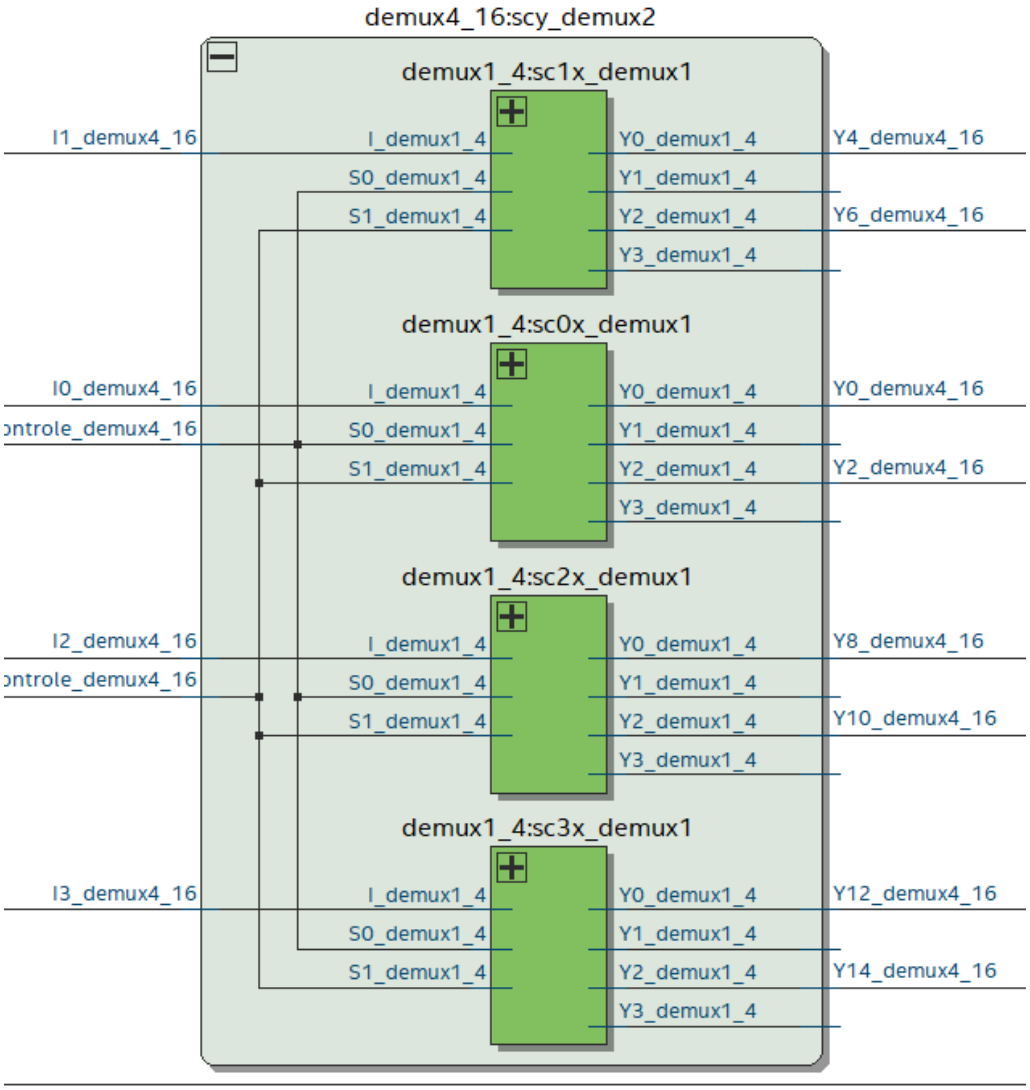

Figura 14. Módulo Demultiplexador

Este módulo foi composto a partir de 4 multiplexadores 1:4 postos em paralelo.

Componentes: 6 portas lógicas ANDs e 3 portas NOT

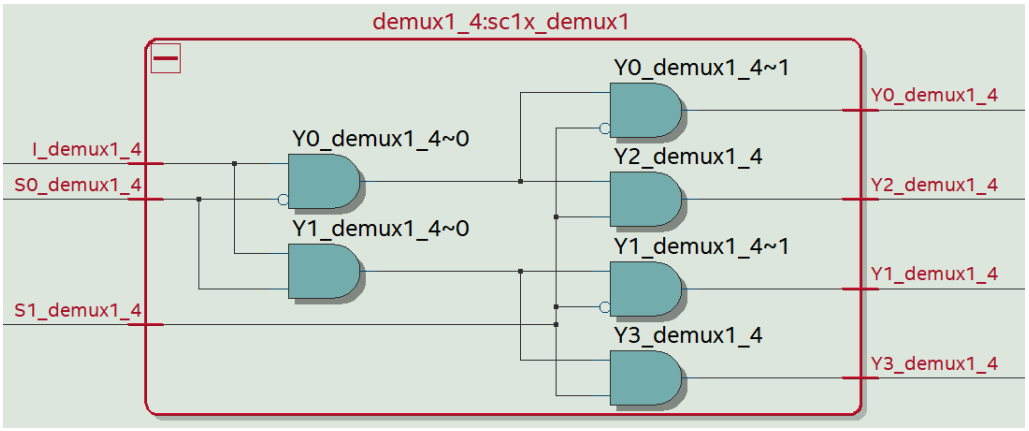

Figura 15. Portas lógicas - Demultiplexador

No módulo geral da ALU, foi utilizado 2 demultiplexadores 4:16 para filtrar as saídas dos módulos operacionais, a partir do seletor de operações.

2.7. Detector de Flags

Para detectar as flags foi criado um módulo que recebe informacoes dos carrys outs de saída do somador paralelo, o resultado da soma e detecta o carry out, overflow, sinal e zero do circuito.

Componentes utilizados: 1 porta OR, 1 porta XOR e 1 port NOT

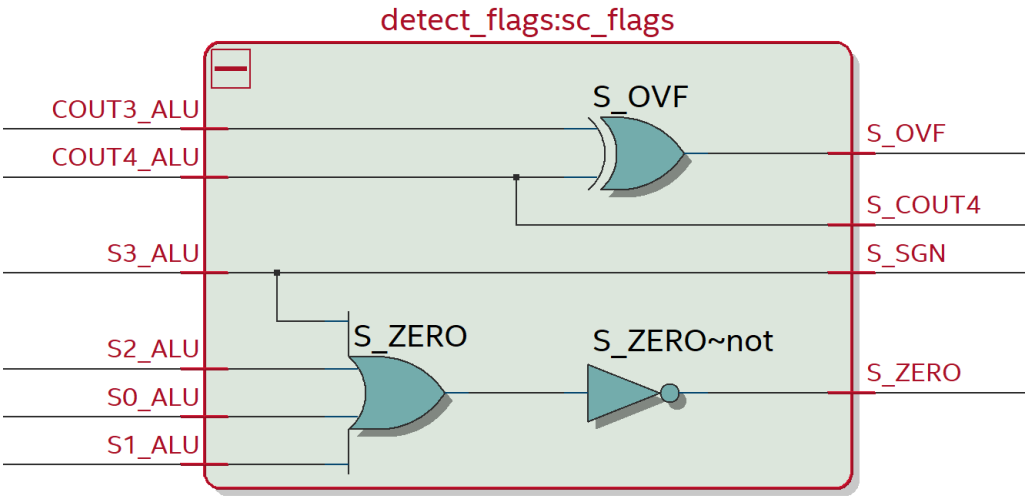

Figura 16. Portas lógicas - Flags

2.8. Rede combinacional

Após desenvolver todos os módulos necessários para realizar as operações previstas, foi necessário definir uma entrada de controle da ALU com o objetivo de criar combinacoes binarias para a seleção das operações. Com isso desenvolvemos nossa rede combinacional.

Primeira etapa consistiu em criar uma tabela verdade para configurar nosso seletor de operacoes da ALU.

F0	F1	F2	Kx	Ky	KMX-mux8:4	Cin0	s1-dmx1-controle	s0-dmx1-controle	d-desloc-controle	Operacao
0	0	0	0	0	1	0	0	0	x	SOMA
0	0	1	0	1	1	1	0	0	x	SUBTRACAO
0	1	0	0	0	0	1	0	0	x	INCREMENTO1
0	1	1	1	0	0	1	0	0	x	TROCA DE SINAL
1	0	0	0	0	0	0	0	1	0	DIV POR 2
1	0	1	0	0	0	x	0	1	1	MULT POR 2
1	1	0	0	1	0	1	0	0	x	DECREMENTO1
1	1	1	0	0	1	0	1	0	x	COMPARACAO
A	B	C								
MAPA DE KARNAUGHT			y = A'BC	y = A'B'C + ABC'	y = A'B' + ABC	y = A'C + BC'	y = ABC	y = AB'	y = C	

Figura 17. Tabela verdade - Rede combinacional

Calculando o mapa de Karnaugh, temos o seguinte circuito:

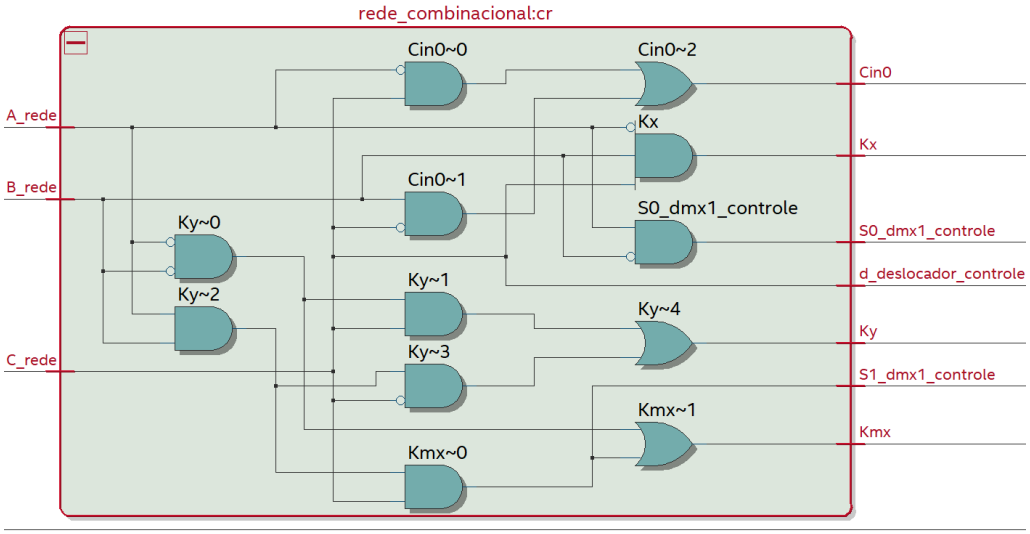

Figura 18. Portas lógicas da rede combinacional

Diagrama geral da ALU

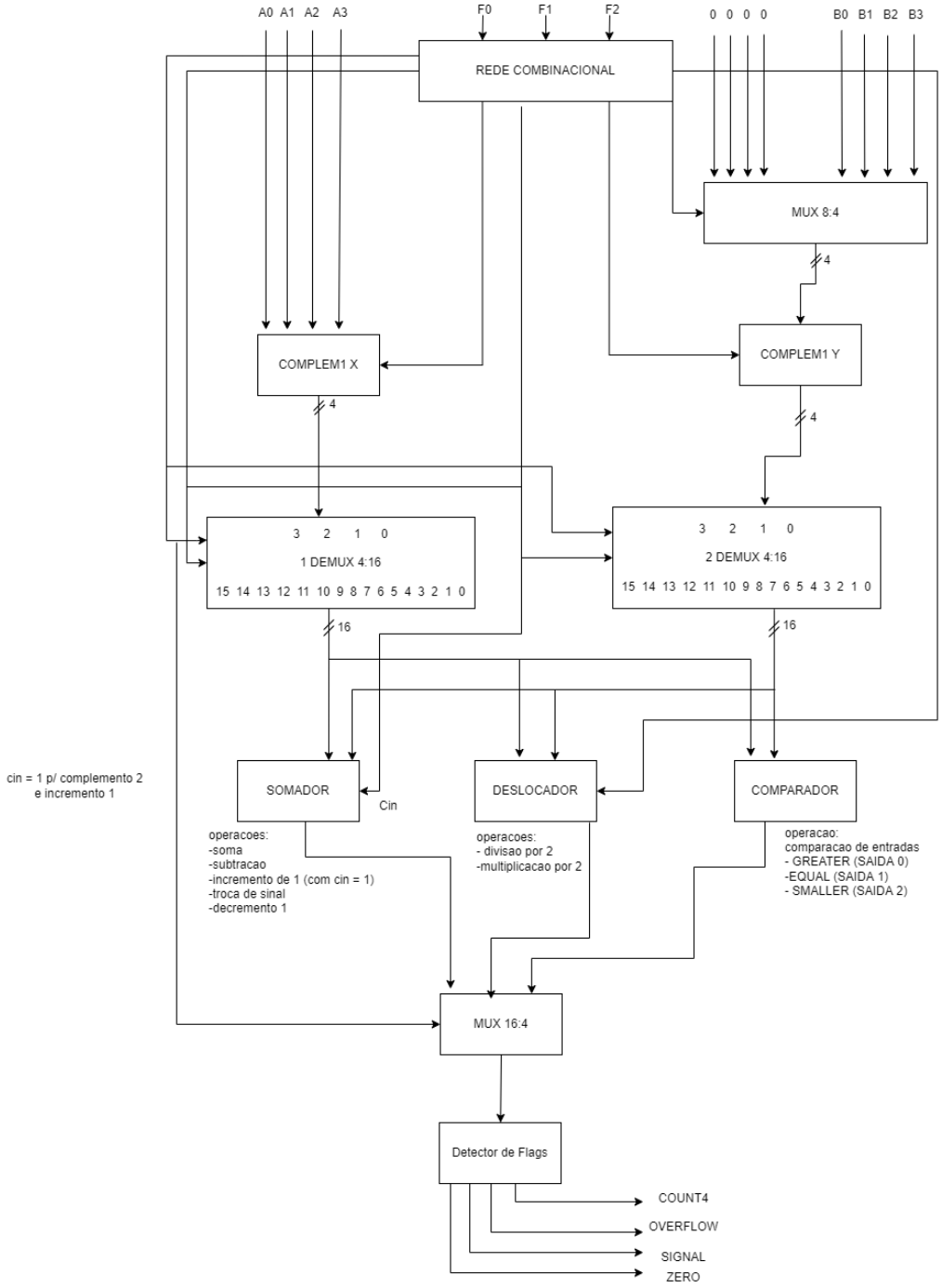

Figura 19. Diagrama ALU produzido em drawio

Explicaremos detalhadamente no próximo tópico cada uma das operações realizadas com os módulos apresentados.

3. Operações

3.1. Soma

Para esta operação utiliza-se o módulo Somador Completo Paralelo de 4 bits, realizando a operação de soma entre dois operandos (A e B).

Considere a entrada da rede combinacional como ’000’, o circuito realizará a adição entre os operandos de entrada da ALU. Onde as entradas de controle dos circuitos complementadores estarão configuradas em ’0’, e com isso não farão o complemento de 1 nos operandos, além disso a entrada de controle do multiplexador estará ativa, passando os bits de entrada para a saída.

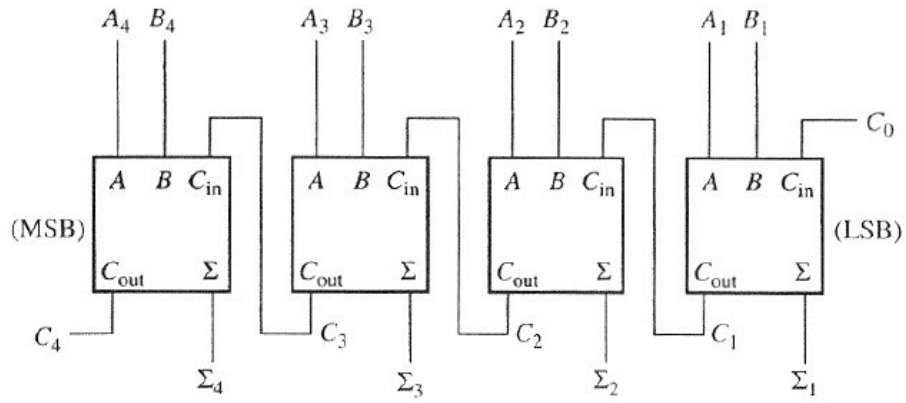

Figura 20. Somador Paralelo de 4 bits

3.2. Subtração em Complemento de 2

Utilizando o método de complemento de 2, a subtração consiste em uma adição, na qual o número a ser subtraído (subtraendo) está na sua em Complemento de 2, e então é somado ao minuendo.

Para representarmos números binários com sinal, é necessário acrescentarmos mais um bit a frente da magnitude do número (módulo do número);

- Quando o bit de sinal é 0, significa que o número é positivo.
- Quando o bit de sinal é 1, significa que o número é negativo.

Nos casos onde o número é negativo, é necessário realizar o Complemento de 2 para saber qual é a magnitude do número.

A conversão de um número binário para sua forma em complemento ocorre por meio de:

- 1. Inversão de cada bit
- 2. Soma-se 1 no LSB

Com isso, concluímos que é possível utilizar o próprio módulo somador para gerar tal operação, com o acréscimo da passagem do minuendo pelo módulo de complemento de 1 invertendo os bits e um carry in de entrada no circuito somador para realizar o complemento de 2.

O funcionamento desse sistema se dá da seguinte forma:
Um módulo Complementador de X(operando1) e outro de Y(operando2) tem 4 entradas de dados cada um, 1 entrada de controle (Ky/Kx) e 4 saídas. Esta entrada de controle permite o configurar a passagem desses dados para a saída do módulo de acordo com a operação a ser realizada. No caso da subtração estará ativa, e com isso aplicará o complemento de 1 nos bits de entrada.

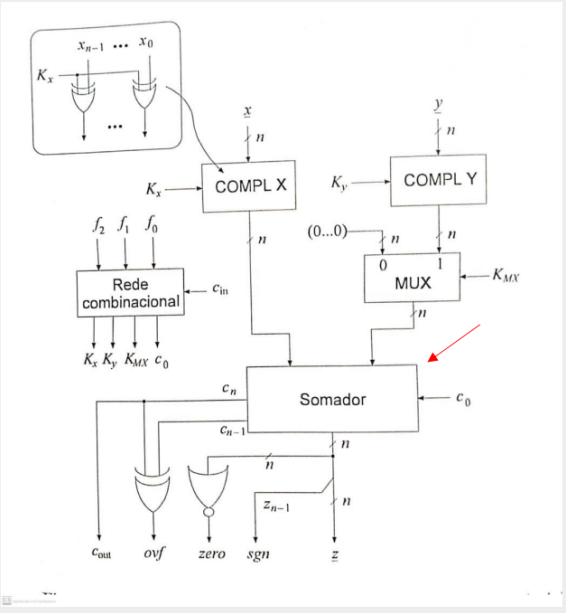

Figura 21. Utilização do circuito Somador para realizar outras operações

3.3. Incremento de 1

Para esta operação apenas o valor do operando A é relevante, portanto através de um multiplexador 8:4 podemos controlar a saída do operando B, que neste caso, sua entrada de controle é configurada para zero pois não importa seu valor para esta operação.

Assim como é possível calcular uma soma entre dois operandos binários, esta operação é realizada utilizando o módulo do somador paralelo de 4 bits, através deste é possível somar mais 1 a entrada de um dos operandos, configurando para 1 o valor do carry de entrada do circuito somador.

Neste tipo de operação o valor do operando B não é relevante, portanto a entrada de controle do multiplexador é posta em '0'.

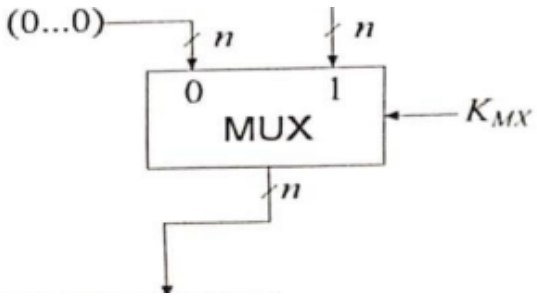

Figura 22. Entradas de controle do Multiplexador 8:4

3.4. Decremento de 1

Esta operação é realizada utilizando o módulo do somador paralelo de 4 bits, com uma configuração diferente da operação apresentada anteriormente, pois assim como é possível realizar a operação de subtração entre dois operandos, também é possível realizar o decremento de 1 de um dos operandos utilizando essa mesma logica, tendo em vista a possibilidade de configuração do multiplexador de entrada do operando B para zero e adicionar 1 ao carry de entrada deste módulo.

3.5. Troca de Sinal

Neste caso, operação também é realizada utilizando o módulo do somador paralelo de 4 bits. A entrada de controle do complementador de X (operando A) é ativa, de tal forma a inverter os bits de entrada, e após isso é adicionado um carry in de entrada ao circuito somador.

3.6. Multiplicação por 2

Para a multiplicação utiliza-se o módulo Deslocador, através do deslocamento de 1 bit. Com isso, é possível definir se o deslocamento ocorrerá para a direita ou para esquerda com as entradas de controle do módulo.

Para realizar a operação de multiplicação de um operando por 2 utilizamos o deslocamento de 1 bit para a esquerda. Este deslocamento joga fora os bits mais significativos (bit mais a esquerda) e acrescenta um bit a direita.

	Controle		Dados					
	s	d	x_4	x_3	x_2	x_1	x_0	x_{-1}
			1	0	0	1	1	0
Não deslocar	NÃO	—		0	0	1	1	
Deslocar à direita	SIM	DIREITA		1	0	0	1	
Deslocar à esquerda	SIM	ESQUERDA		0	1	1	0	
				y_3	y_2	y_1	y_0	

Figura 23. Relação das entradas de controle

Logo, a entrada de controle ’d’ é configurada para a esquerda, e a entrada s permanece ativa.

3.7. Divisão inteira por 2

Esta operação é similar a operação de deslocamento para a esquerda, apenas invertendo a direção de deslocamento. Para realizar a operação de divisão de um operando por 2 utilizamos o deslocamento de 1 bit para a direita, de tal forma que o deslocamento joga fora os bits menos significativos (bit mais a direita) e acrescenta um bit a esquerda.

Logo, a entrada de controle ’d’ é configurada para a direita, e a entrada s permanece ativa.

3.8. Comparação

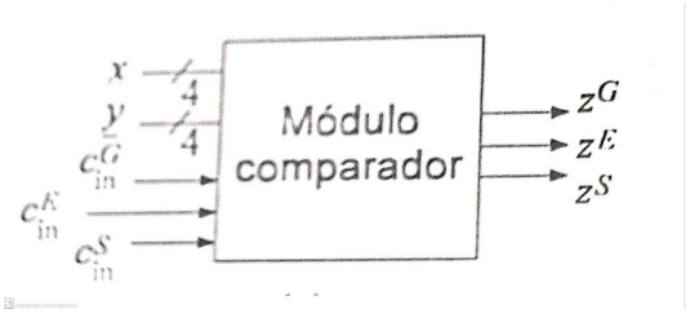

Figura 24. Relação das entradas de controle

Para realizar esta operação, o módulo receberá dois operandos binários equivalente a números naturais de 4 bits, a entrada de controle do multiplexador 8:4 estará ativa, e a entrada de controle do demultiplexador também. Sua saída será zG,zE,zS sendo apenas 1 delas ativa, indicando em qual condição se encontra o operando A em relação ao B. Onde:

- SAÍDA 0 ativa indica o operando A é maior que B
- SAÍDA 1 ativa indica o operando A é igual que B
- SAÍDA 2 ativa indica o operando A é menor que B

4. Resultados Experimentais

Resultados obtidos a partir do test bench produzido no Quartus Prime, onde as variáveis A1_ALU_test, A2_ALU_test,A3_ALU_test,A4_ALU_test são as entradas do primeiro operando da ALU, B1_ALU_test, B2_ALU_test,B3_ALU_test,B4_ALU_test são as entradas do segundo operando.

COUT0_ALU_test se refere ao carry de entrada do circuito somador e COUT4_ALU_test que se refere ao carry de saida do circuito somador.

F0_ALU_test,F1_ALU_test,F2_ALU_test são as entradas da rede combinacional

S0_ALU_test,S1_ALU_test,S2_ALU_test,S3_ALU_test são os bits de resultado, ou seja, da saída da ALU dependendo da operação que for selecionada.

E por fim, OVF_ALU_teste se refere a identificação do overflow, ZERO_ALU_teste indica a identificação do zero, e SGN_ALU_test identifica o sinal do resultado da saída da ALU.

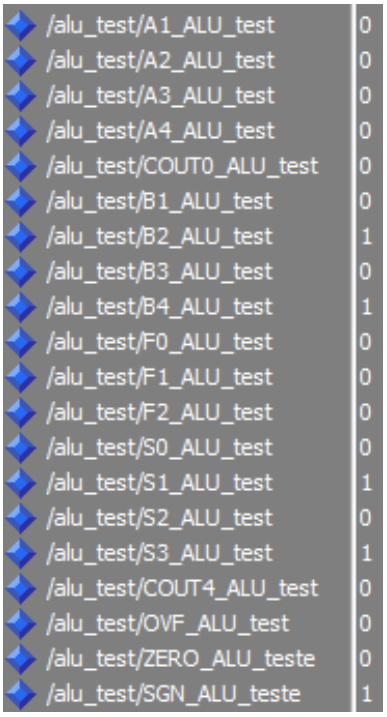

Figura 25. Soma: 0 + (-6) = -6, flag de sinal = 1

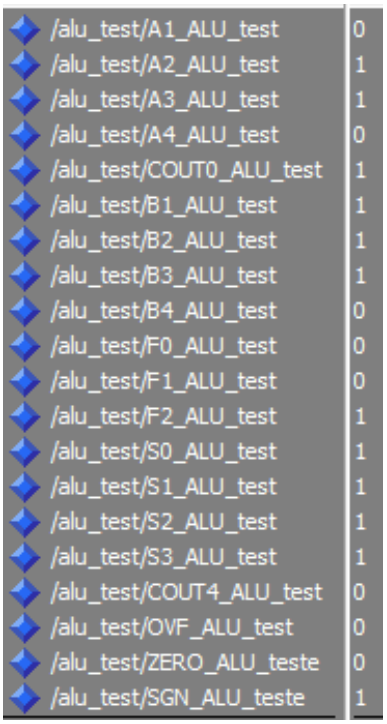

Figura 26. 6 - 7 = -1, flag de sinal = 1

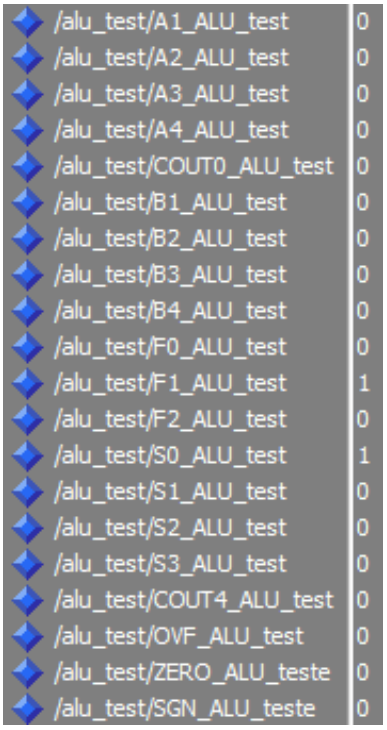

Figura 27. Incremento de 1: 0 + 1 = 1

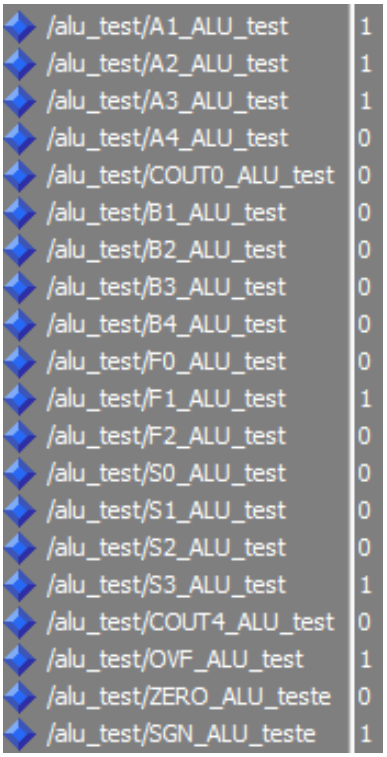

Figura 28. Incremento de 1: 7 + 1 = 8, flag de Overflow = 1)

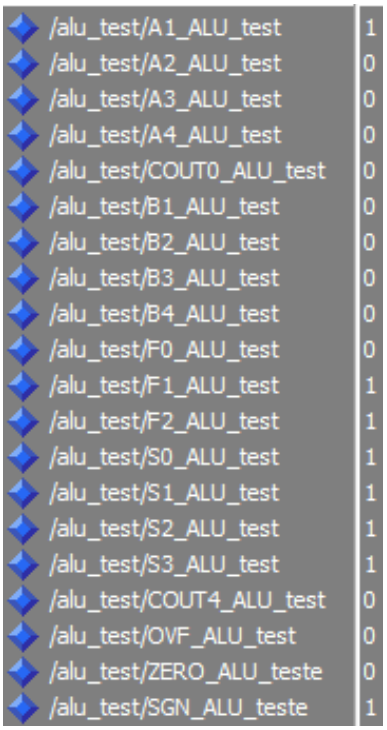

Figura 29. Troca de sinal: 1 para -1, flag de sinal = 1

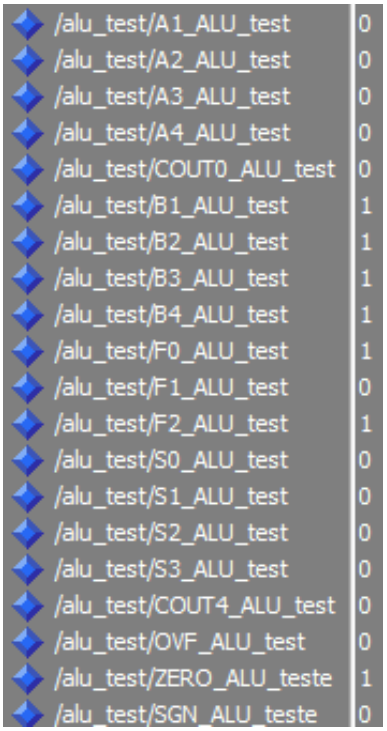

Figura 30. 0 x 2 = 0, flag de zero = 1

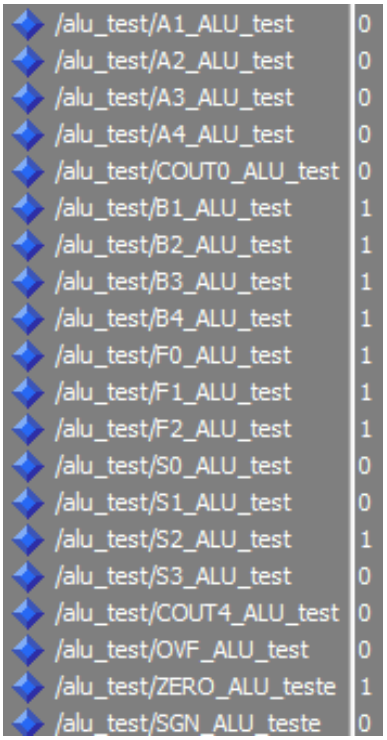

Figura 31. 0 <1

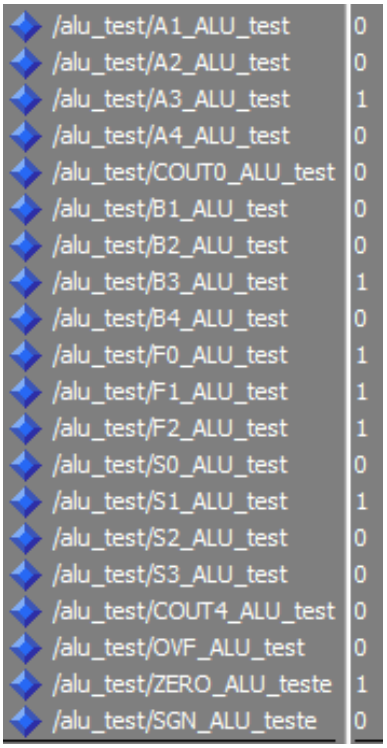

Figura 32. 4 = 4

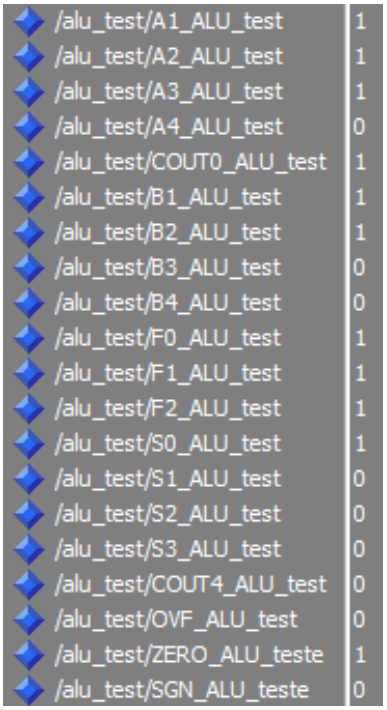

Figura 33. 7 >3

5. Conclusões

Durante a elaboração deste trabalho podemos afirmar que adquirimos bastante conhecimento sobre os desafios de se projetar uma ALU em uma placa através do VHDL, dentre eles: aprender uma nova linguagem no VHDL, o Quartus ter diversos problemas de performance e a dificuldade na utilização a placa pelo labsland por conta de seus avisos de erro serem pouco esclarecedores, esta última inclusive se mostrou inviável, uma vez que não conseguimos rodar nosso código em VHDL na placa por conta de limitações de memória da mesma.

Apesar das dificuldades, encontramos satisfação em conseguir aplicar os conceitos aprendidos nas aulas teóricas no projeto além de finalizarmos o trabalho com satisfação de termos implementado as operações propostas mesmo sendo nossa primeira experiência com as ferramentas utilizadas.

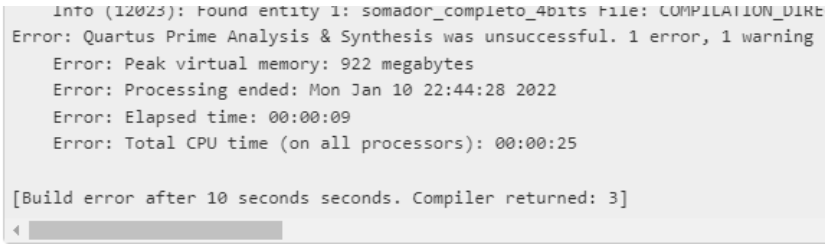

Figura 34. Terminal plataforma labsland

6. Referencias

Volnei A. Pedroni, ”Circuit Design and Simulation with VHDL”, The MIT Press, 2nd Ed.