RELATÓRIO LABORATÓRIO 3: MÁQUINA DE ESTADOS FINITOS E ULAS

Arthur Faria Campos 16/0024242

Programa de Engenharia Eletrônica
Faculdade Gama – U niversidade de Brasília
St. Leste Projeção A - Gama Leste, Brasília - DF, 72444-240
email: arthur-fc@hotmail.com

Felipe Lima Alcântara 16/0027918

Programa de Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
St. Leste Projeção A - Gama Leste, Brasília - DF, 72444-240
email: lipelima0327@gmail.com

RESUMO

O documento apresenta o relatório técnico do terceiro experimento, da matéria Prática de Eletrônica Digital 2. Este experimento denominado "Máquina de Estados finitos e ULAs". Esta atividade teve o objetivo de realizar implementações em VHDL de máquinas de estados finitos, na implementação de uma ULA e realizar uma expansão de Maclaurin com a ULA implementada.

1. INTRODUÇÃO

A Unidade Lógica e Aritmética é um circuito combinacional, responsável pela realização de operações matemáticas e lógicas dentro de um circuito digital.

A ULA mais básica, tem três entradas e duas saídas, duas entradas para inserir os valores a serem operados e uma para selecionar a operação, sendo a saída o resultado final e o status. A complexidade da ULA depende diretamente há necessidade do sistema.

Em VHDL podemos descrever a ULA como uma Máquina de Estados, sendo cada operação um Estado, um estado para registrar cada entrada e um estado para saída.

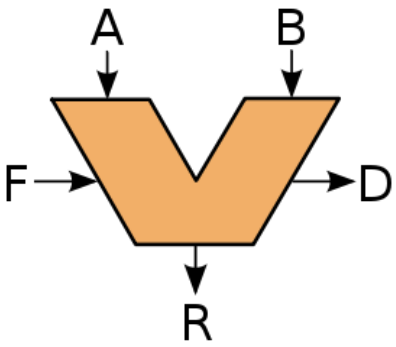

Figura 1: Um símbolo esquemático típico para uma ULA, onde "A" e "B" são operandos, "R" é a saída, "F" é a entrada da unidade de controle e "D" é a saída de status.

2. EXPERIMENTO

Para o Laboratório 3 tivemos que resolver 2 exercícios, o primeiro era adicionar alguns operadores ao projeto da ULA já iniciado em sala junto ao professor e o segundo era com essa ULA realizar a expansão por séries de Maclaurin do operador exponencial.

2.1. Exercício 01

As operações a serem implementadas à ULA eram: soma um (+1), subtrai um (-1), comparação de maior (>) , de menor (<) e de igualdade (=), e negação not (A). Além disso adicionar as saídas de overflow e underflow para as operações de soma e subtração respectivamente.

Para a implementação da ULA, foi necessário, 4 chaves para a entrada "sel", 12 chaves para a entrada"oper", 4 botões: lopA, lopB, start, reset. As saidas serão representadas em leds: uma para overflow, outra para underflow e as outras 12 leds para o resultado das operações.

Inicialmente foi adicionado uma seletora de 4 bits, denominada "sel", para determinar a operação, segue a tabela:

Sel(4bits)	**Saida**	**Operação**
"0000"	saida<= A + B	Soma
"0001"	saida<= A - B	Subtração
"0010"	saida<= A * B	Multiplicação
"0011"	saida<= A / B	Divisão
"0100"	saida<= A + '1'	Soma um
"0101"	saida<= A - '1'	Subtrai um
"0110"	saida<= A< B	A menor que B
"0111"	saida<= A>B	A maior que B
"1000"	saida<= A= B	A igual a B
"1001"	N/A	N/A
"1010"	saida<= A and B	AND
"1011"	saida<= A or B	OR
"1100"	saida<= A xor B	XOR
"1101"	saida<= not(A)	NOT
"1110"	saida<= shiftL A	Anda esquerda
"1111"	saida<= shiftR A	Anda direita

Tabela 1: Operações

Para registrar os operandos foi inserido as entradas “lopA” e “lopB” que acionadas faz com que o registrador do operando seja preenchido com o que está escrito na entrado “oper” naquele momento. Ex: foi colocado o número em binário que representa o número em decimal 2, na entrada oper, logo em seguida foi pressionado o botão lopA, logo a entrada A terá o valor 2.

Com os valores registrados ao apertar o botão “start” a condição para os estados de operação será completada e a operação será realizada e por fim o resultado é mostrado no estado display.

2.2. Exercício 02

O exercício 02 é a implementação de uma FSM que implemente a expansão por séries de Maclaurin do operador exponencial de quarta ordem mostrada na equação a seguir, onde x deve estar na faixa [-1.0 a 1.0].

$$f(x)=e^{x}=\sum_{n=0}^{4}\frac{x^{n}}{n!}=1+x+\frac{x^{2}}{2!}+\frac{x^{3}}{3!}$$

Para a implementação usamos a ULA de 12 bits em ponto fixo desenvolvida em sala de aula.
A FSM tem um botão Start para indicar o início da operação e uma saída Ready indicando o fim da operação.

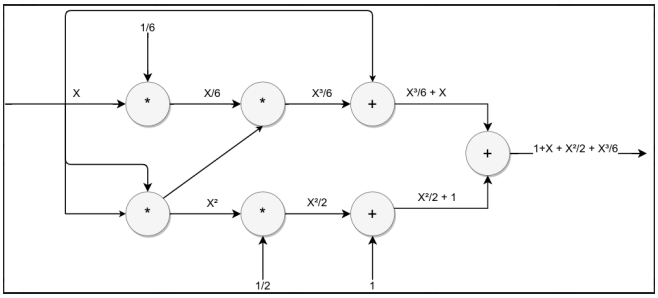

Figura 2: Esquemático da FSM

O funcionamento da FSM é feito pela implementação de 2 Ulas que funcionam em paralelo e realizam as operações simultaneamente.

A máquina de estados também apresenta estados temporários para que os valores sejam salvos nos registradores das Ul.

Também usamos todos os estados em um único process com clock, o que evita Latchs mas, em compensação, aumenta o número de FFs.

3. RESULTADOS

Antes de ser implementado na FPGA, realizamos simulações por meio de Test Bench.

3.1. Simulações exercício 01

A simulação do Exercício 1 busca verificar a exatidão das operações e comprovar que a implementação da ULA foi realizada de forma correta.

A Figura abaixo demonstra a implementação do overflow, no OpA foi inserido o valor 010110010100(1428 em decimal) e no OpB o valor 001101011011(859 em decimal). O resultado da soma seria 100011101111 (2287 em decimal), porém ocorreu overflow pois o carry foi para o bit de sinal.

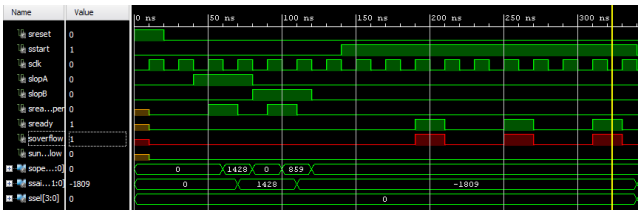

Figura 3: Simulação da soma com overflow.

Agora verificar se o adiciona um(+1) e o resta um (-1) está em funcionamento para isso foi inserido em OpA o valor 010110010100(1428 em decimal). E foi possível verificar o resultado 1429 e 1427.

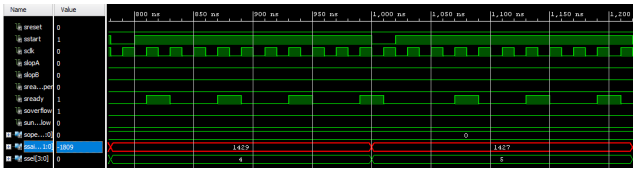

Figura 4: Simulação do +1 e -1.

Para verificar a funcionalidade da igualdade adicionamos tanto ao OpA e ao OpB o valor 111111111111. E se fosse igual todos os bits da saída deveriam receber o valor ‘1’.

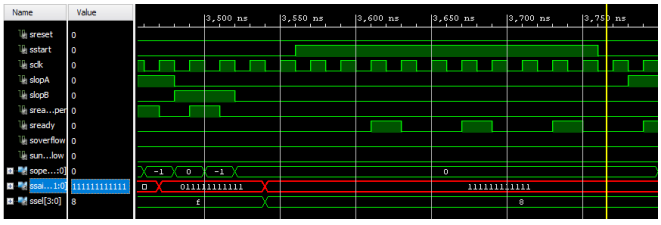

Figura 5: Simulação de igualdade.

Ao verificar a funcionalidade do operador (<) foi necessário fazer o opA ser menor que o OpB, portanto, foi inserido 001101011011(859 em decimal) para opA e 011110110100(1972 em decimal) para opB. Sendo assim A<B.

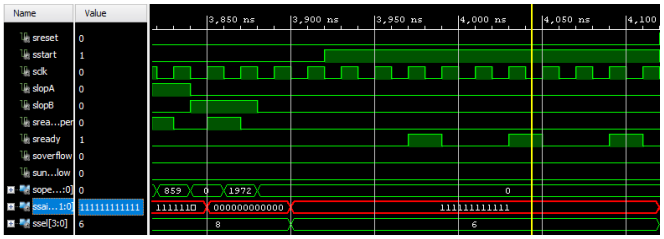

Figura 6: Simulação de menor que

A Figura abaixo demonstra a implementação do overflow, no OpA foi inserido o valor 010110010100(1428 em decimal) e no OpB o valor 001101011011(859 em decimal). Por tanto A é maior que B.

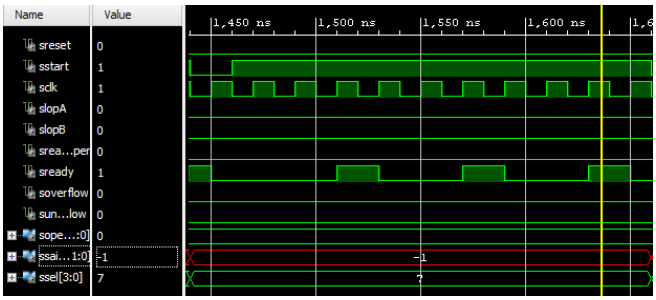

Figura 7: Simulação de maior que.

Seguimos com o OpA sendo 1428 e testamos a operação "not" da ULA com o resultado dando -1429 que é 101001101011 em binário.

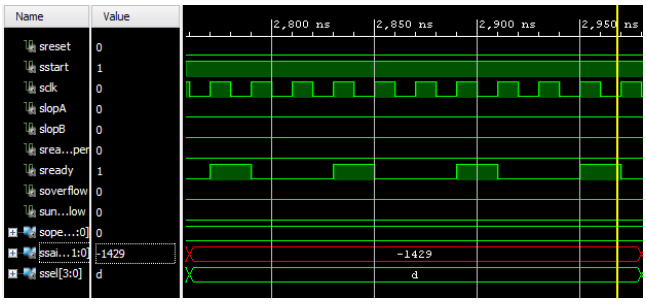

Figura 8: Simulação do not(A)

Os componentes utilizados pela placa são descritos nesta tabela:

Resource	Utilization	Available	Utilization %
LUT	127	20800	0.61
FF	38	41600	0.09
DSP	1	90	1.11
IO	37	106	34.91
BUFG	1	32	3.13

Tabela 2: Componentes da Ula

3.2. Simulações Exercício 02

A simulação do Exercício 02 comprova o funcionamento correto das FSM com as mudanças de estados das Maquinas e a saída fx

Na simulação abaixo x =1 sendo que resultado esperado é de 2,66… ,contudo o valor encontrado começa a se diferenciar já na segunda casa decimal com 2,65.

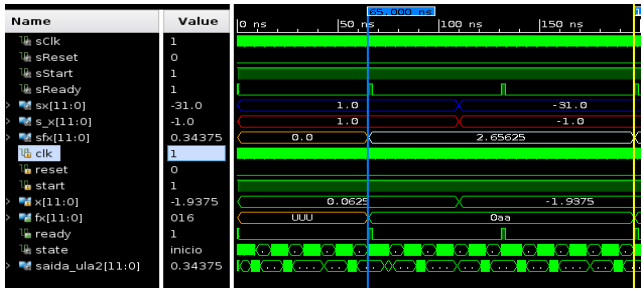

Figura 9: Simulação de Maclaurin (1/2)

Já na simulação abaixo verificamos a saida fx para valores de -1.0 e 1.0.

Para x= 0.2185, tivemos a saida fx = 1.234… , sendo que a resposta correta é de 1.244… .

Para x = -0.40625, tivemos a saida fx = 0.65625 , sendo que a resposta correta é de 0.6650… .

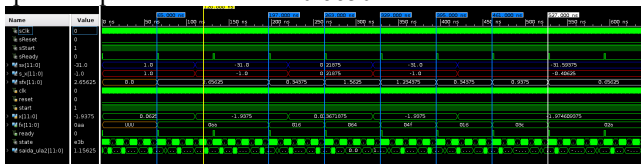

Figura 10: Simulação de Maclaurin(2/2)

Com uma simples análise da Tabela abaixo percebemos que muito pouco da capacidade da FPGA fora utilizada para implementar a expansão de Maclaurin.

Resource	Utilization	Available	Utilization %
LUT	149	20800	0.72
FF	131	41600	0.31
DSP	2	90	2.22
IO	28	106	26.42
BUFG	1	32	3.13

Tabela 3: Componentes de Maclaurin

A Figura abaixo mostra a região da placa Basys 3 em que está implementado a nossa expansão de Maclaurin de 4ª ordem.

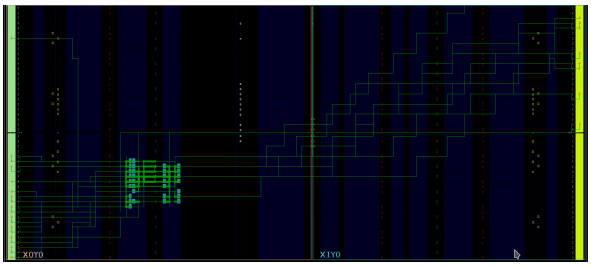

Figura 11: Região da Placa

4. DISCUSSÃO E CONCLUSÕES

A implementação das novas operações na ULA foram simples, pela forma como foi montado o código, bastou adicionar no bloco de referência das saídas as novas opções para a entrada de seleção da operação, a maior complexidade na inserção de operações foram nas comparações, que foi necessário criar uma componente para verificar o maior, menor e igualdade.

Já o overflow e underflow foi necessário criar também uma componente e nós adicionamos uma forma da saída só aparecer quando a operação fosse selecionada.

O exercício 2 foi necessário compreender como seriam os blocos de operação de cada estado para otimizar e diminuir o tempo gasto para se achar o resultado. Uma das dificuldades esteve na hora de se realizar as operações com números negativos, pois o complemento de dois do vivado, mudava o nosso número já com complemento de 2.

Um problema encontrado no exercício 2 é o erro de precisão feito pelo uso de contantes já calculadas. Pois não é possível guardar o valor exato de alguns valores. Assim, quando se vai multiplicar por 1/6 acarreta um erro na saída da função. Pelas simulações o erro ocorre já na segunda casa decimal, a qual é a mesma que possui o erro na constante de 1/6.

Os projetos foram entregues com sucesso e com todas as operações rodando de forma correta.

5. REFERÊNCIAS

[1] Villanova, G. (2016). Uma arquitetura PWM em VHDL. [online] Embarcados. Disponivel em: https://www.embarcados.com.br/uma-arquitetura-pwm-em-vhdl/ [Acessado 7 Sep. 2017].

[2] Wakerly, John F., Digital Design: Principles and Practices, 4th ed., Prentice Hall, 2005.

[3] Chu, Pong P., FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version, Wiley, 2011. [EBRARY]

[4] D'Amore, R., VHDL: Descrição e Síntese de Circuitos Digitais, 2 a . edição, LTC, 2012