UNIVERSIDADE DE BRASÍLIA
DEPARTAMENTO DE CIÊNCIAS DA COMPUTAÇÃO
ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES - 1º/2016

Laboratório 2 - Aritmética Fracionária em Assembly MIPS, ULA e FPULA

Alunos:

André Abreu R. de Almeida - 12/0007100
Arthur de Matos Beggs - 12/0111098
Bruno Takashi Tengan - 12/0167263
Gabriel Pires Iduarte - 13/0142166
Guilherme Caetano - 13/0112925
João Pedro Franch - 12/0060795
Rafael Lima - 10/0131093

Parte A - Aritmética Fracionária em Assembly MIPS

1) Cálculo das raízes da equação de segundo grau usando operações em ponto flutuante precisão dupla.

Dado o polinômio de segundo grau: $ax^2 + bx + c = 0$

a) Escreva um procedimento int bhaskara(double a, double b, double c) que retorne 1 caso as raízes sejam reais e 2 caso as raízes sejam complexas conjugadas, e coloque na pilha os valores das raízes no formato IEEE 754 Precisão Dupla.

O procedimento se encontra no arquivo “Lab2/Lab2a/Pt1_FloatPoint/ bhaskaraFloatPoint.s”.

b) Escreva um procedimento void show(int t) que receba o tipo (t=1 raízes reais, t=2 raízes complexas) e apresente na tela as raízes, conforme os modelos abaixo, que estão na pilha retirando-as da mesma:

Para raízes reais:	**Para raízes complexas:**
R(1)=1234.0000	**R(1)=1234.0000 + 5678.0000 i**
R(2)=5678.0000	**R(2)=1234.0000 – 5678.0000 i**

O procedimento se encontra no arquivo “Lab2/Lab2a/Pt1_FloatPoint/ bhaskaraFloatPoint.s”.

c) Escreva as saídas obtidas para os seguintes polinômios [a, b, c]:

c.1) [1, 0, -9.86960440]

R(1) = -3.1415926534164162
R(2) = 3.1415926534164162

c.2) [1, 0, 0]

R(1) = -0.0
R(2) = 0.0

c.3) [1, 99, 2459]

R(1) = -49.5 + 2.958039891549808i
R(2) = -49.5 - 2.958039891549808i

c.4) [1, -2468, 33762440]

R(1) = 1234.0 + 5678.0i
R(2) = 1234.0 - 5678.0i

c.5) [0, 10, 100]

R(1) = -Infinity
R(2) = NaN

Obs.: A fórmula de Bhaskara não é capaz de calcular as raízes do polinômio se seu coeficiente “a” for zero.

2) Cálculo das raízes da equação de segundo grau usando operações em ponto fixo de 32 bits.

Dado a equação de segundo grau: $ax^2 + bx + c = 0$

a) Escreva um procedimento int bhaskara(int a, int b, int c) que retorne 1 caso as raízes sejam reais e 2 caso as raízes sejam complexas conjugadas, e coloque na pilha os valores das raízes no formato ponto fixo com um Q adequado.

O procedimento se encontra no arquivo "Lab2/Lab2a/Pt2_FixedPoint/ bhaskaraFixedPoint.s".

b) Escreva um procedimento void show(int t) que receba o tipo (t=1 raízes reais, t=2 raízes complexas) e apresente na tela as raízes, conforme os modelos abaixo, que estão na pilha retirando-as da mesma:

Para raízes reais:	**Para raízes complexas:**
R(1)=1234.0000	**R(1)=1234.0000 + 5678.0000 i**
R(2)=5678.0000	**R(2)=1234.0000 – 5678.0000 i**

O procedimento se encontra no arquivo "Lab2/Lab2a/Pt2_FixedPoint/ bhaskaraFixedPoint.s".

c) Escreva as saídas obtidas para os seguintes polinômios [a, b, c]:

c.1) [1, 0, -9.86960440]

R(1) = -3.0
R(2) = 3.0

Obs.: As entradas são truncadas para serem convertidas em números inteiros. Assim, o número -9.86960440 foi truncado para -9. Caso se queira trabalhar com números reais, as linhas não comentadas da macro "_convertToFixedPoint_" devem ser comentadas e vice-versa.

c.2) [1, 0, 0]

R(1) = 0.0
R(2) = 0.0

c.3) [1, 99, 2459]

R(1) = -49.5 + 2.95703125i

R(2) = -49.5 - 2.95703125i

c.4) [0, 10, 100]

Valor nao permitido para o coeficiente! O programa sera encerrado.

Obs.: O coeficiente “a” não pode ser zero, ou uma operação de divisão por zero será realizada. Para impedir overflows, foram estabelecidos os seguintes limites de valor para os coeficientes:

A -> -1 ou 1

B -> entre -127 e 127

C -> entre -4095 e 4095

Qualquer valor de entrada fora das faixas definidas para cada coeficiente resultará em uma mensagem de erro e o programa será abortado.

Parte B - ULA e FPULA

1) Unidade Lógico Aritmética de Inteiros:

a) Para a ULA MIPS32 fornecida, descreva suas funções e levante a tabela verdade de seus códigos de operação.

Operação ULA	Código de Operação	Descrição
AND	5'b00000 //0	Bitwise AND - AND bit a bit entre duas entradas
OR	5'b00001 //1	Bitwise OR - OR bit a bit entre duas entradas
ADD	5'b00010 //2	Adição duas entradas
MFHI	5'b00011 //3	Move from High - Saída = HI

SLL	5'b00100	//4	Shift Left Logico - Desloca a entrada para a esquerda
MFLO	5'b00101	//5	Move from Low - Saída = LO
SUB	5'b00110	//6	Subtração
SLT	5'b00111	//7	Set on Less Than - (A>B; saida=1; saída=0);
SRL	5'b01000	//8	Shift Right Lógico - Desloca sem extensão de sinal
SRA	5'b01001	//9	Shift Right Aritiético - Desloca com exenção de sinal
XOR	5'b01010	//10	Bitwise XOR - XOR bit a bit entre duas entradas
SLTU	5'b01011	//11	Set Less Than Unsigned - comparação sem sinal
NOR	5'b01100	//12	Bitwise NOR - NOR bit a bit entre duas entradas
MULT	5'b01101	//13	Multiplicação - Multiplica as entradas - Saída: HI e LO
DIV	5'b01110	//14	Divisão - Quociente => LO, Resto => HI
LUI	5'b01111	//15	Load Upper Immediate - Concatena os 16 MSBits
SLLV	5'b10000	//16	Shift Left - Quantidade: o conteúdo dos 5 LSBits
SRAV	5'b10001	//17	Quantidade: o conteúdo dos 5 LSBits / Sem sinal
SRLV	5'b10010	//18	Quantidade: o conteúdo dos 5 LSBits/ Com sinal

MULTU	5'b10011	//19	Multiplicação sem sinal
DIVU	5'b10100	//20	Divisão sem sinal
MTHI	5'b10101	//21	Move to High - HI = Entrada
MTLO	5'b10110	//22	Move to Low - LO = Entrada

b) Verifique cada operação por simulação de forma de onda.

Analiza-se aqui a forma de onda funcional das operações da ULA. O sinal de entrada foi disposto com um offset do posedge do clock de propósito para demonstrar o funcionamento das funções síncronas e assíncronas.
Para as assíncronas, que são a maioria, como a simulação funcional simula um circuito ideal, assim que a mudança é feita na entrada ou no iControlSignal, a saída é alterada.
Para as síncronas, assim que o posedge do clock é atingido.

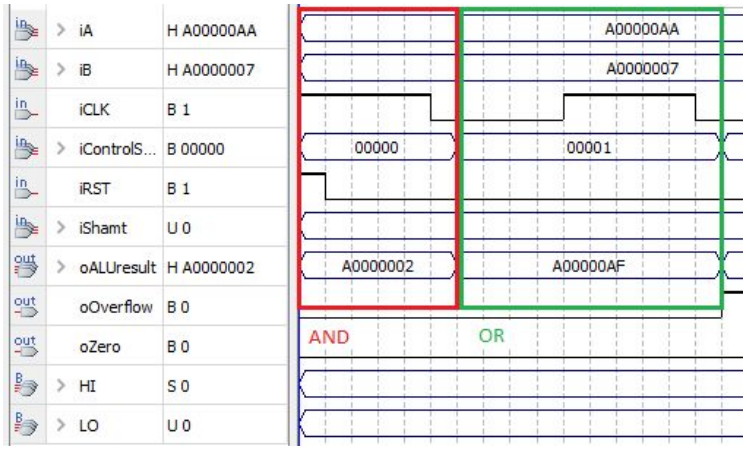

Fig. 1 - Operações AND e OR

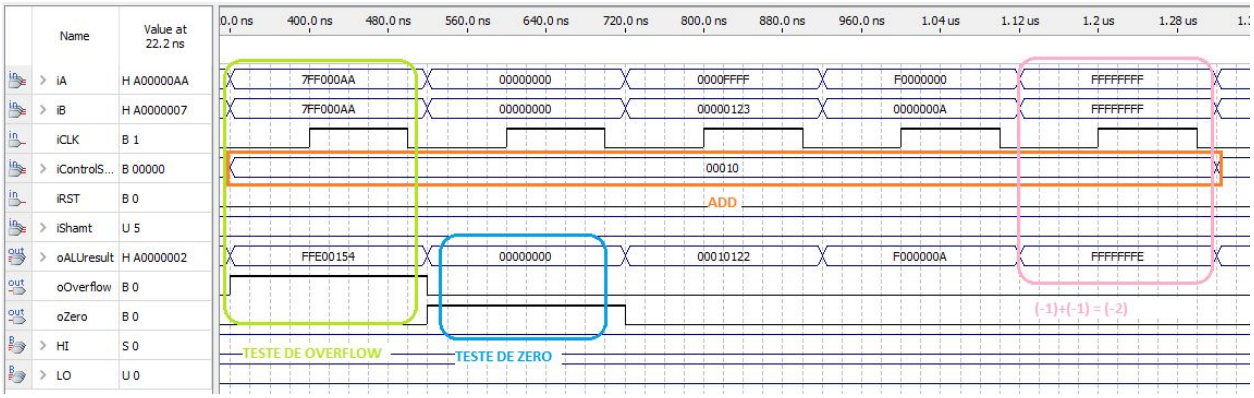

Fig. 2 - ADD - overflow; zero; pos+pos; neg+pos; neg+neg

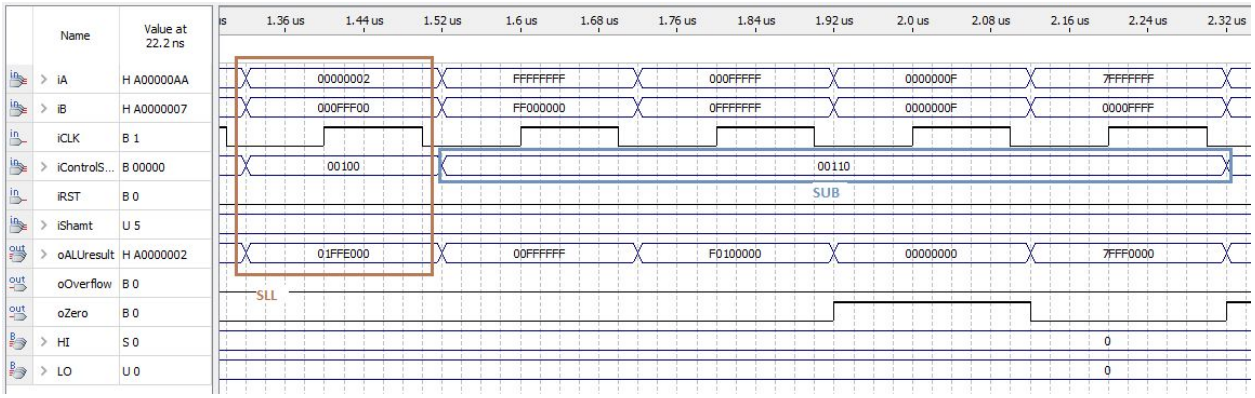

Fig. 3 - SLL (shamt 5) e SUB (neg-neg; pos menor-maior; iguais; pos maior-menor)

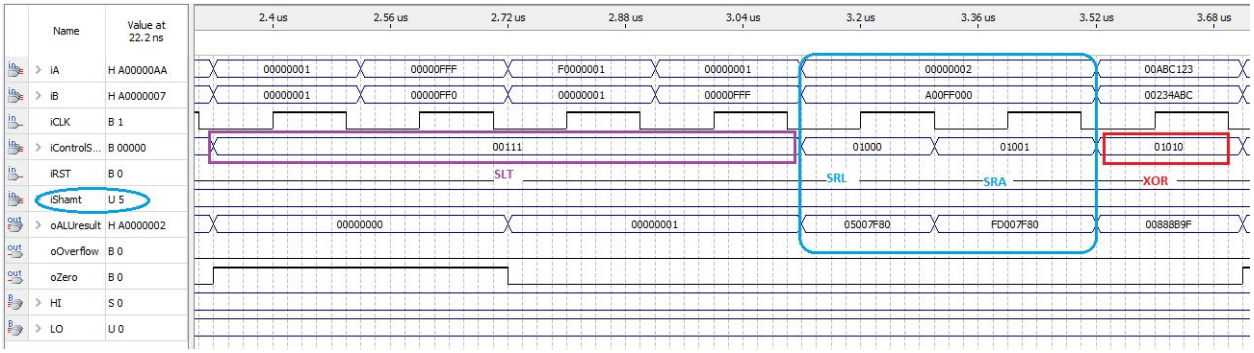

Fig. 4 - SLT (iguais; maior; menor-neg; menor-pos), SRL, SRA e XOR

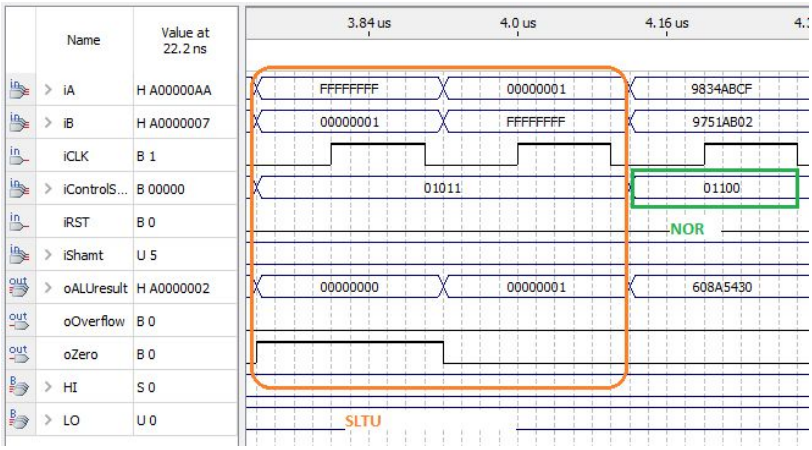

Fig. 5 - SLTU (extremos) e NOR

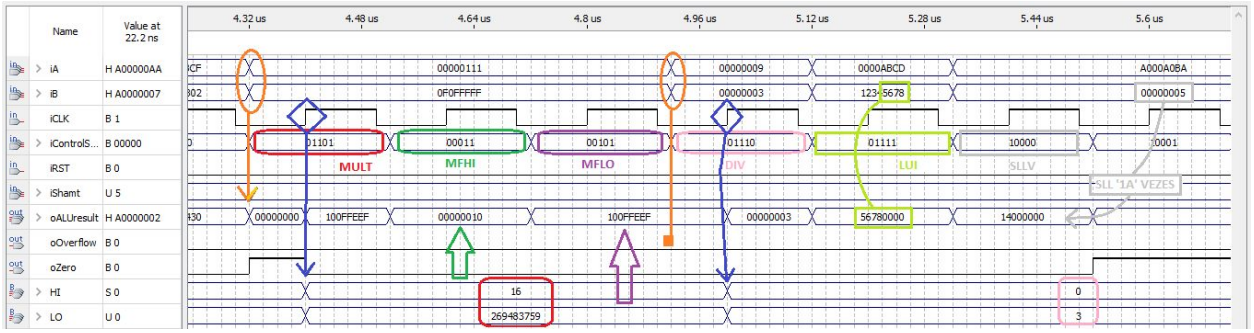

Fig. 6 - MULT, MFHI, MFLO, DIV, LUI e SLLV

Destaca-se aqui que as operações MULT e DIV são síncronas.

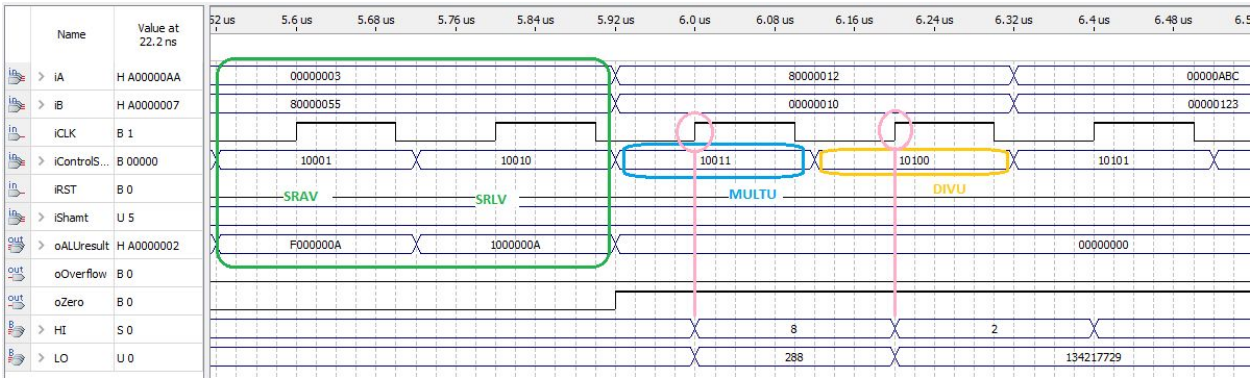

Fig. 7 - SRAV, SRLV, MULTU e DIVU

Destaca-se aqui também que MULTU e DIVU também são síncronas.

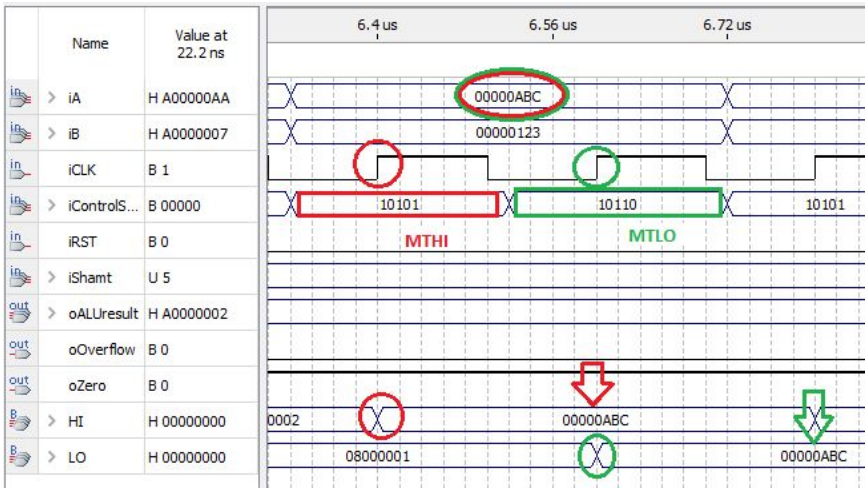

Fig. 8 - MTHI e MTLO

Destaca-se aqui que as operações MTHI e MTLO são síncronas.

Após a análise confirma-se o funcionamento correto de todas as funções, em aspecto funcional pelo menos.

Há uma cópia dos Print Screens das waveforms em anexo na pasta “Lab2/Lab2b/ALU/Parte 1b”.

c) Sintetize esta ULA na placa DE2-70 e verifique, em tempo real, cada operação implementada usando a ferramenta SignalTap Analyser. Filme o experimento e coloque no YouTube.

Link do YouTube: https://youtu.be/aQR1IQXui88

d) Levante os requisitos físicos necessários, isto é, número de elementos lógicos, tempos envolvidos e máxima frequência de clock utilizável para cada operação da ULA.

Operação ULA	Elementos lógicos	Tempos envolvidos	Máx frequência de clock (Mhz)
AND	50	19.42 (TPD)	51.49331
OR	50	18.904 (TPD)	52.89886
ADD	81	18.083 (TPD)	55.30056
MFHI	2 comb	9.887 (TPD)	101.1429
SLL	183 comb	20.194 (TPD)	49.51966
MFLO	2 comb	9.887 (TPD)	101.1429
SUB	78 comb	17.792 (TPD)	56.20504
SLT	36 comb	15.008 (TPD)	66.63113
SRL	181 comb	16.764 (TPD)	59.65163
SRA	189 comb	18.627 (TPD)	53.68551
XOR	50 comb	19.553 (TPD)	51.14305
SLTU	36 comb	15.008 (TPD)	66.63113
NOR	50 comb	24.654 (TPD)	40.56137
MULT	96 comb, 32 reg	14.753 (TPD)	67.78282
DIV	1302 comb, 32 reg	162.99 (SETUP)	6.135496
LUI	25 comb	13.338 (TPD)	74.97376
SLLV	177 comb	18.331 (TPD)	54.5524
SRAV	189 comb	18.018 (TPD)	55.50006
SRLV	187 comb	17.509 (TPD)	57.11348

MULTU	2 comb	9.887 (TPD)	101.1429
DIVU	2 comb	9.887 (TPD)	101.1429
MTHI	2 comb	9.887 (TPD)	101.1429
MTLO	2 comb	9.887 (TPD)	101.1429

Os recursos foram obtidos compilando o código verilog para cada estrutura isoladamente. Aparentemente para as operações MFHI, MFLO, MULTU, DIVU, MTHI, MTLO, pois todas ficaram com os mesmos recursos segundo o quartus, mesmo sendo diferentes. Há muito poucas portas lógicas e operações síncronas não possuem registradores em sua composição. Provavelmente em todos esses casos o Quartus considerou apenas o caso de excessão, que é atingido quando nenhum circuito é selecionado e a saída é zero.

Observa-se que a operação de divisão é muito mais lenta que as outras.

Os tempos envolvidos foram retirados do TimeQuest analyzer. A quantidade de elementos lógicos foi tirada do Analysis & Synthesis. Os Print Screens que mostram de onde cada numero foi tirado encontram-se na pasta “Lab2/Lab2b/ALU/Parte 1d”.

e) Retire as operações de divisão e módulo. Sintetize novamente e levante os novos requisitos físicos da ULA. Comente.

	Elementos Lógicos	SETUP TIME	TCO	TPD	Max frequência de clock
Completa	6238 (64 reg)	161.298	18.721	30.616	638.98 MHz
Sem DIV	3752 (64 reg)	140.675	18.712	28.512	604.96 MHz

A frequencia máxima foi tirada do TimeQuest analyzer, assim como os outros tempos envolvidos. A quantidade de elementos lógicos foi tirada do Analysis & Synthesis. Os Print Screens que mostram de onde cada numero foi tirado encontram-se na pasta “Lab2/Lab2b/ALU/Parte 1e”.

2) Unidade Aritmética de Ponto Flutuante:

a) Para a FPULA MIPS fornecida, descreva suas funções (MegaWizard Plug-In Manager Edit) e levante a tabela verdade de seus códigos de operação.

Operação FPULA	Código de operação	Descrição
ADDS	4'b0001	A + B
SUBS	4'b0010	A - B
MULS	4'b0011	A * B
DIVS	4'b0100	A / B
SQRT	4'b0101	A^(1/2)
ABS	4'b0110	Pega o módulo da entrada A.
NEG	4'b0111	-A
CEQ	4'b1000	(A = B)? 1:0
CLT	4'b1001	(A < B)? 1:0
CLE	4'b1010	(A <= B)? 1:0
CVTSW	4'b1011	Converte a entrada A de lógica de inteiros para a lógica de precisão simples.
CVTWS	4'b1100	Converte a entrada A de lógica de precisão simples para lógica de inteiros.

b) Verifique cada operação por simulação de forma de onda.

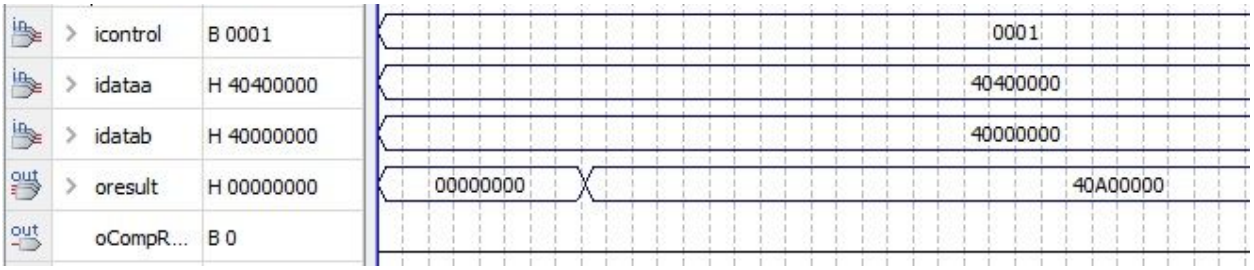

Fig. 9 - Operação ADDS, realizando 3 + 2 = 5.

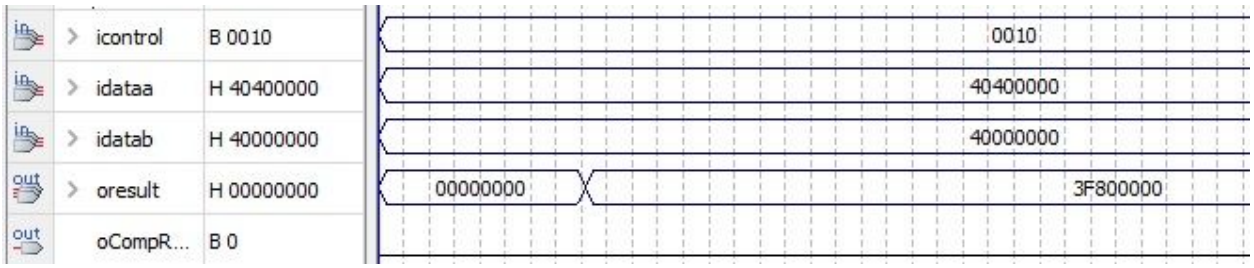

Fig. 10 - Operação SUBS, realizando 3 - 2 = 1.

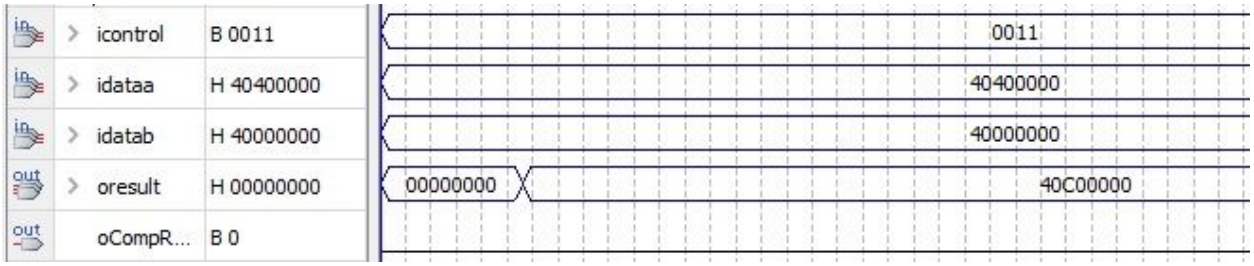

Fig. 11 - Operação MULS, realizando 3 * 2 = 6.

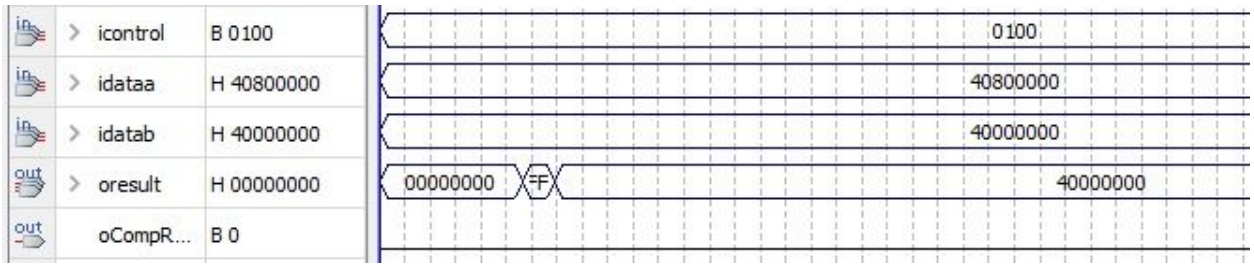

Fig. 12 - Operação DIVS, realizando 4 / 2 = 2.

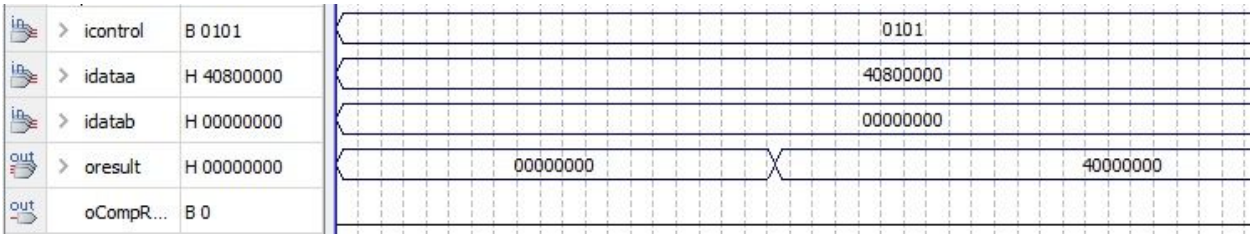

Fig. 13 - Operação SQRT, realizando 4^(1/2) = 2.

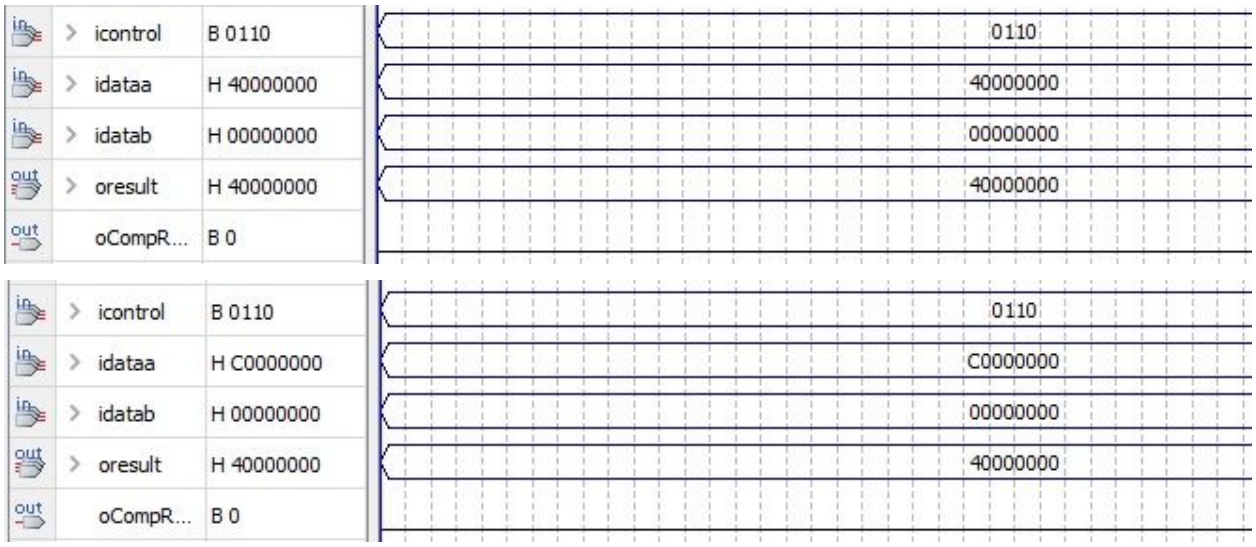

Fig. 14 - Operação ABS, realizando ABS(2) = 2 e ABS(-2) = 2.

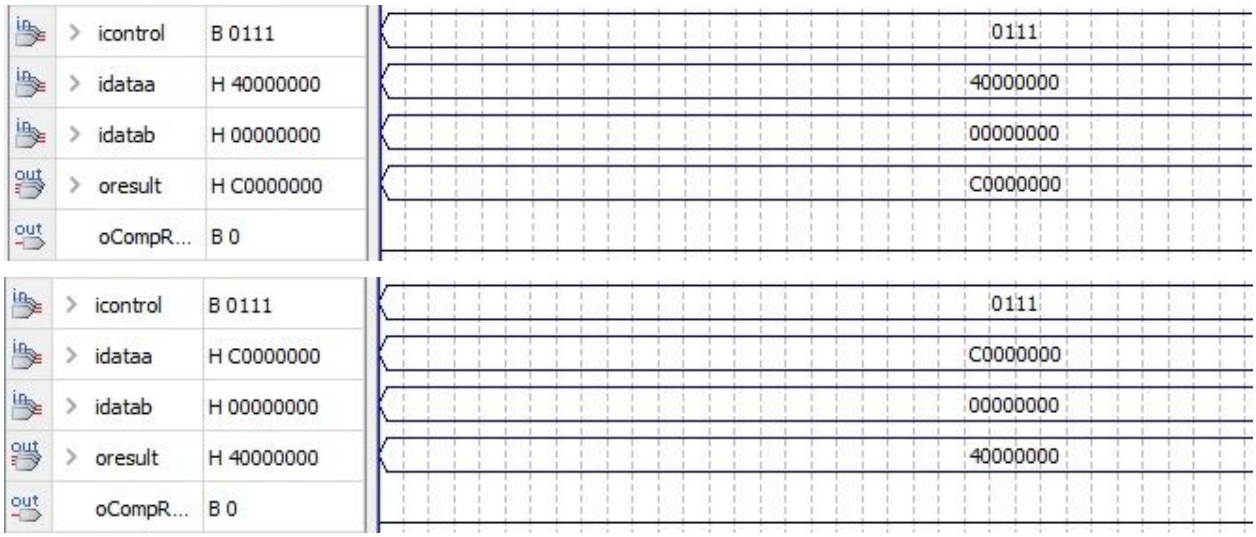

Fig. 15 - Operação NEG, realizando -(2) = -2 e -(-2) = 2.

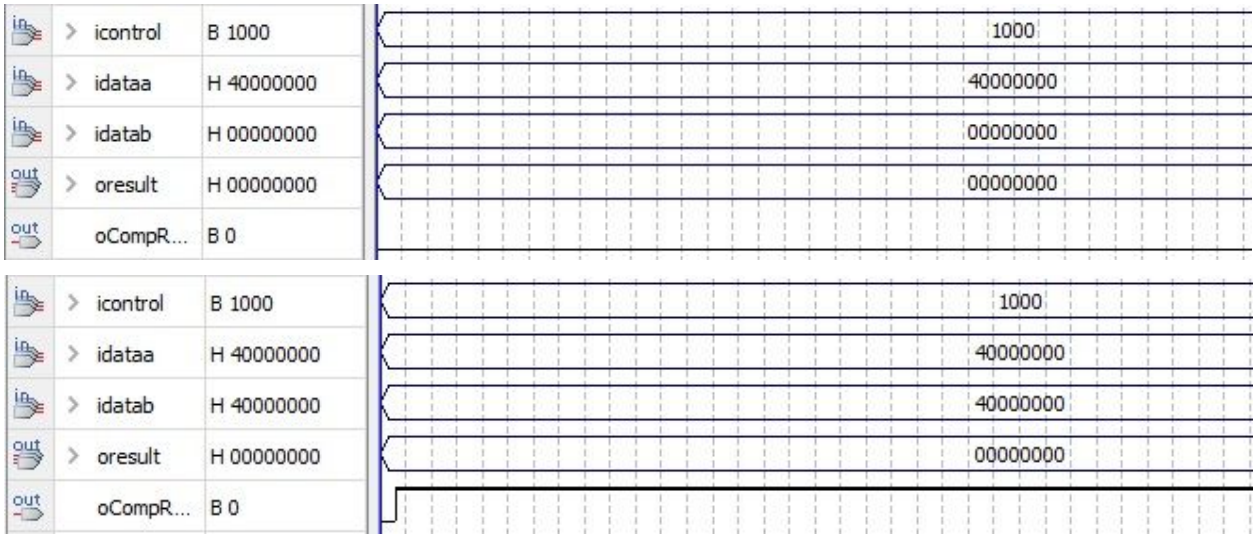

Fig. 16 - Operação CEQ, realizando (2 = 0)? = false e (2 = 2)? = true.

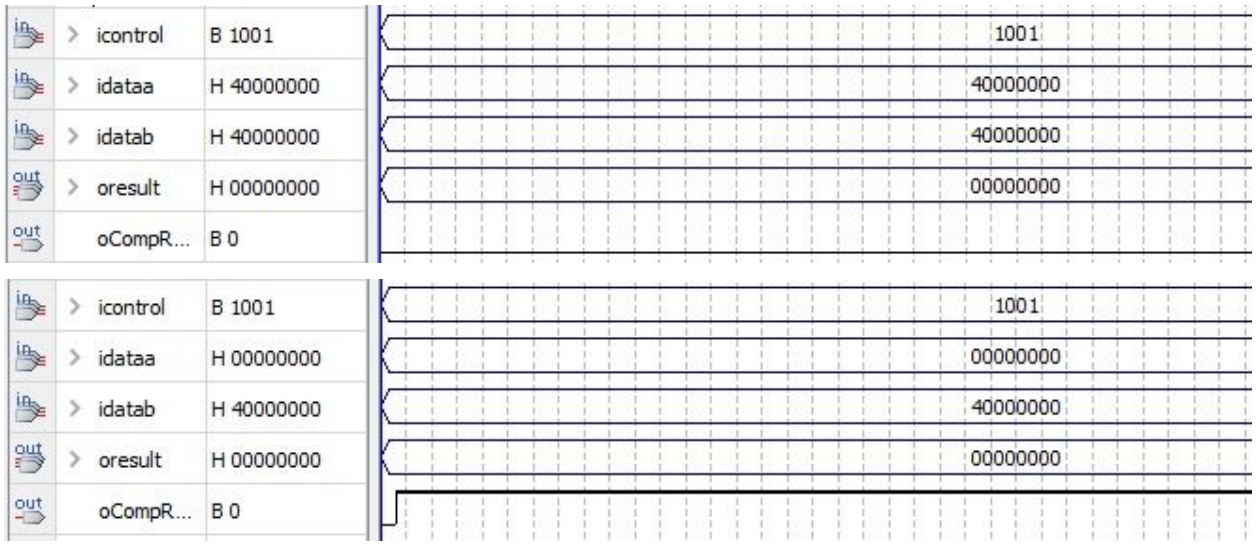

Fig. 17 - Operação CLT, realizando (2 < 2)? = false e (0 < 2)? = true.

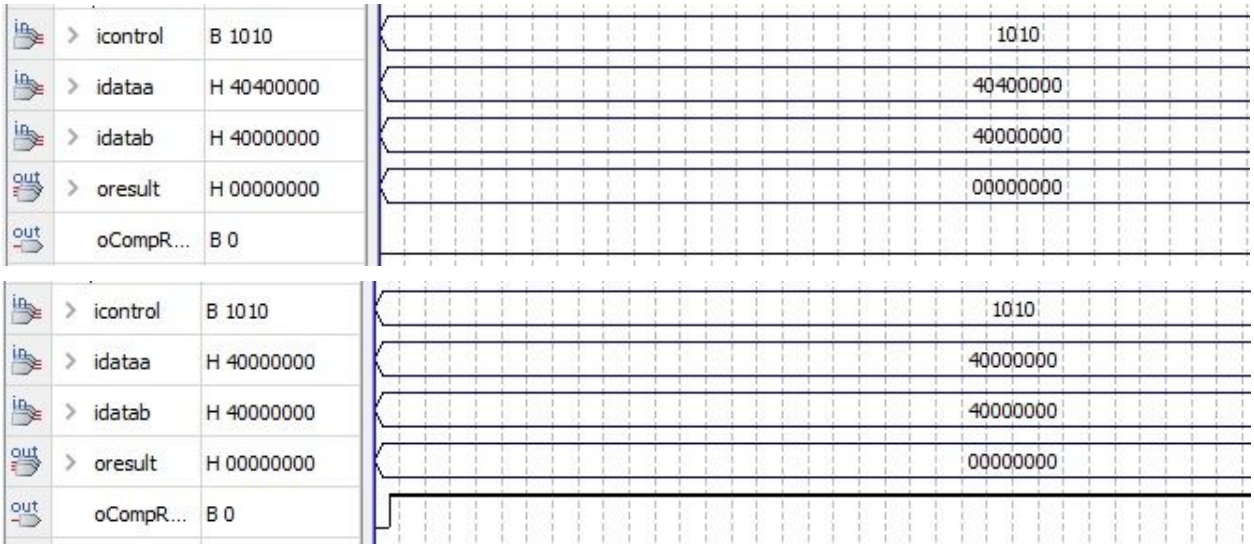

Fig. 18 - Operação CLE, realizando (3 <= 2)? = false e (2 <= 2)? = true.

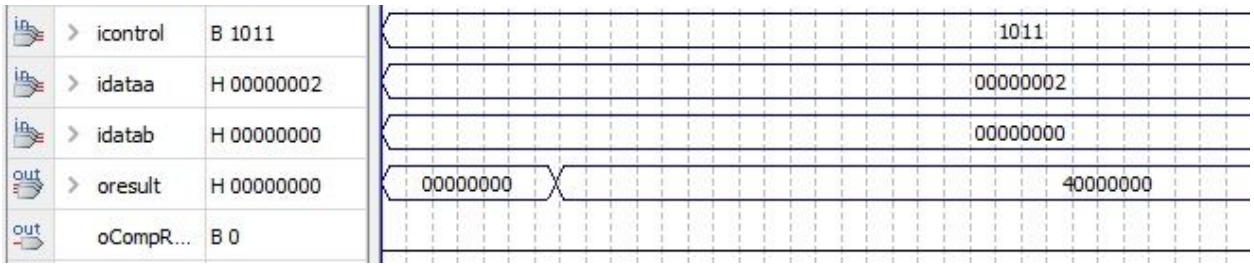

Fig. 19 - Operação CVTSW, convertendo 2 de lógica de inteiros para lógica de precisão simples.

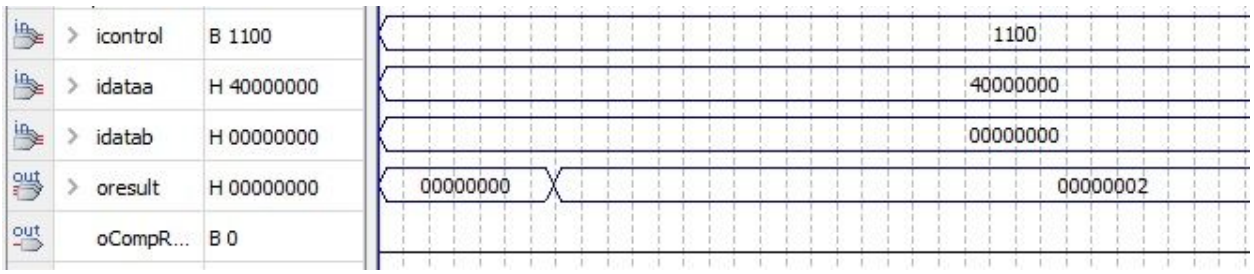

Fig. 20 - Operação CVTWS, convertendo 2 de lógica de precisão simples para lógica de inteiros.

c) Sintetize esta FPULA na placa DE2-70 e verifique, em tempo real, cada operação implementada usando a ferramenta SignalTap Analyser. Filme o experimento e coloque no YouTube.

Link do YouTube: https://youtu.be/oWLI_t1dwQU

d) Levante os requisitos físicos, isto é, número de elementos lógicos necessários, número de ciclos, tempos envolvidos e máxima frequência de clock utilizável para cada operação da FPULA.

Operação FPULA	Elementos lógicos	Número de ciclos	Máxima frequência de clock (Tempo máximo de delay)
ADDS	170 lut + 382 reg.	7	156.84 MHz (6.376 ns)
SUBS	170 lut + 382 reg.	7	156.84 MHz (6.376 ns)
MULS	4 lpm_add_sub + 1lpm_mult + 140 reg.	5	192.72 MHz (5.188 ns)
DIVS	16 dsp_9bit + 194 lut + 1 M4K + 74 mux21 + 351 reg.	6	113.66 MHz (8.798 ns)
SQRT	370 lut + 764 reg.	16	137.34 MHz (7.281 ns)
ABS	-	0	-
NEG	-	0	-
CEQ	51 lut + 1 reg.	1	450.04 MHz (2.222 ns)
CLT	51 lut + 1 reg.	1	450.04 MHz (2.222 ns)
CLE	51 lut + 1 reg.	1	450.04 MHz (2.222 ns)
CVTSW	5 lpm_add_sub + 1 lpm_compare + 247 reg.	6	284.50 MHz (3.515 ns)
CVTWS	5 lpm_add_sub + 4 lpm_compare + 284 reg.	6	205.17 MHz (4.874 ns)

Obs: os elementos lógicos foram obtidos atravéz do “MegaWizard Plug-In Manager”, os ciclos de clocks necessários através do “MegaWizard Plug-In Manager” e da análise das waveforms e as máximas frequências e os tempos máximos de delay a partir do “TimeQuest Timing Analyzer” do “Quartus II” considerando o “Slow Model”.

e) Qual a operação que mais impacta em cada requisito físico?

Impacto	Operação FPULA
Maior quantidade de elementos lógicos.	SQRT
Maior número de ciclos de clock.	SQRT
Menor frequência máxima de clock.	DIVS
Maior tempo de execução considerando sua própia frequência máxima.	SQRT