Universidade de Brasília
Campus Universitário Darcy Ribeiro

Disciplina: Organização e Arquitetura de Computadores
Turma: A
Prof.: Marcus Vinicius Lamar

Laboratório 2
ULA e FPULA

Abdullah Zaiter - 15/0089392
Daniel Bauchspiess - 15/0078901
Danielle Almeida Lima - 14/0135740
José Reinaldo da Cunha - 14/0169148
Lucas dos Santos Schiavini - 14/0150749

Brasília, 7 de maio de 2018.

1) Multiplexadores

Após implementarmos os multiplexadores, encontramos os requerimentos físicos e temporais, assim como é mostrado na Tabela 3. Para os casos de 32 bits de 1x8, 1x16 e 1x32 (em cinza) foram usados pinos virtuais, uma vez que a quantidade de pinos da fpga utilizada não eram suficientes. Os códigos dos multiplexadores utilizados se encontram na pasta Grupo4/newMultiplex do diretório do trabalho.

	Número de ALMs	Tpd(largest time) ns
1bit		
1x2	1 / 32,070 (< 1 %)	6.042
1x4	2 / 32,070 (< 1 %)	6.972
5bits		
1x2	3 / 32,070 (< 1 %)	7.378
1x4	6 / 32,070 (< 1 %)	8.010
1x8	13 / 32,070 (< 1 %)	11.843
12 bits		
1x2	7 / 32,070 (< 1 %)	11.231
1x4	14 / 32,070 (< 1 %)	11.568
1x8	31 / 32,070 (< 1 %)	13.265
1x16	61 / 32,070 (< 1 %)	20.455
32 bits		
1x2	17 / 32,070 (< 1 %)	10.703
1x4	33 / 32,070 (< 1 %)	11.050
1x8	226 / 32,070 (< 1 %)	3.505
1x16	435 / 32,070 (1 %)	4.594
1x32	876 / 32,070 (3 %)	8.406

Tabela 3 - Tabela comparativa de requerimentos físicos e temporais.

2) Unidade Lógico Aritmética de Inteiros

a) A Tabela 4 apresenta a descrição das funções, bem como a tabela verdade de seus códigos para cada operação. E a figura 5 mostra o diagrama de blocos da ULA. Realizamos algumas alterações no código da ula de inteiros, uma vez que era necessário a adaptação de algumas

instruções do MIPS para RISC-V, as instruções modificadas estão especificadas na aba de comentários da Tabela 4. Ambas as versões encontram-se na pasta Grupo4_Lab2>Codigos_ULAint, sendo ALU_arrumada_pra_riscv.v e ALU_versaoComErros.v os arquivos com as versões com as correções e original respectivamente.

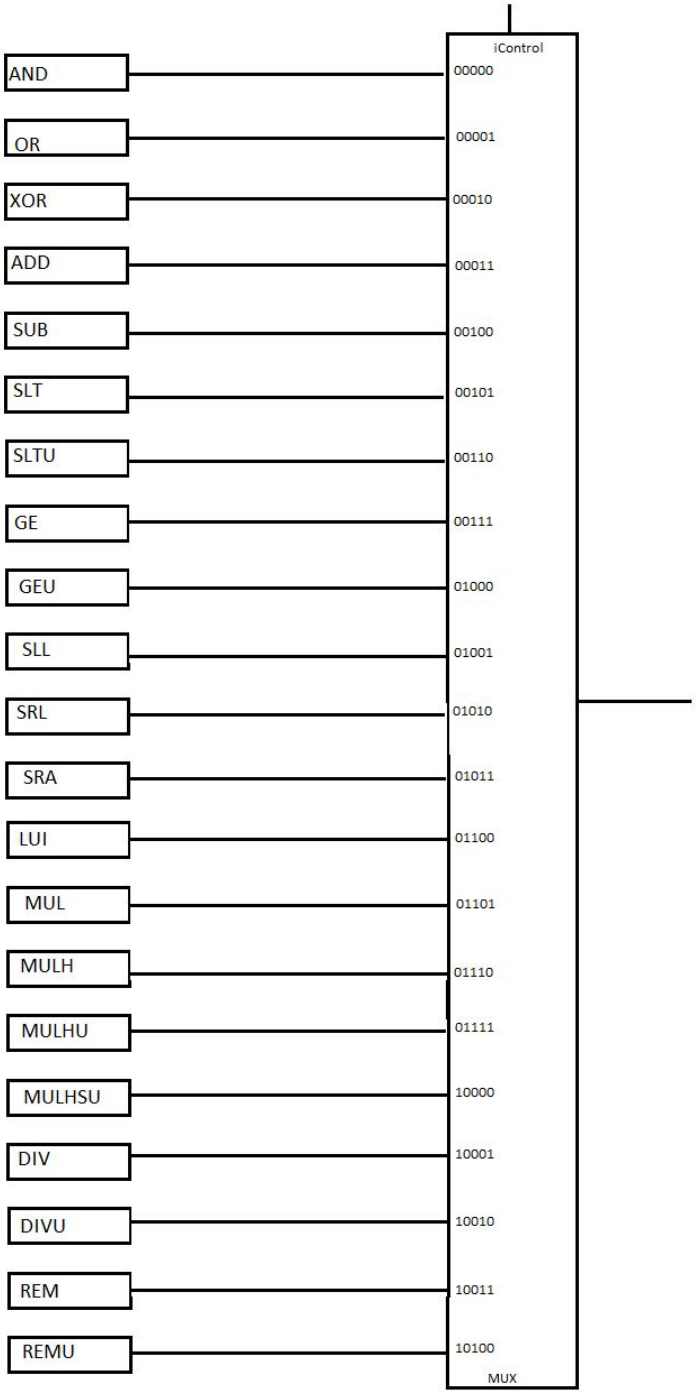

Figura 5 - Diagrama de blocos da ULA de inteiros assíncrona.

IControl	Operação	Saída	Explicação	Comentários
0	**AND**	iA & iB;	operação de AND bit a bit para 32 bits entre iA e iB	
1	**OR**	iA | iB;	operação de OR bit a bit para 32 bits entre iA e iB	
2	**XOR**	iA ^ iB;	operação de XOR bit a bit para 32 bits entre iA e iB	
3	**ADD**	iA + iB;	soma de iA e iB	
4	**SUB**	iA - iB;	subtração de iA e iB de 32 bits	
5	**SLT**	iA < iB;	se iA é menor que iB o resultado é 1	
6	**SLTU**	$unsigned(iA) < $unsigned(iB)	se o módulo de iA é menor que o módulo iB o resultado é 1	
7	**GE**	iA >= iB	se iA é maior ou igual que iB o resultado é 1	
8	**GEU**	$unsigned(iA) >= $unsigned(iB)	se o módulo de iA é maior ou igual que o módulo iB o resultado é 1	
9	**SLL**	iA << iB[4:0]	Deslocar o iB pra esquerda a quantidade de bits do iA estendendo com 0	Arrumamos a ordem trocada dos operandos
10	**SRL**	iA >> iB[4:0]	Deslocar o iB pra direita a quantidade de bits do iA estendendo com 1	Arrumamos a ordem trocada dos operandos
11	**SRA**	iA >>> iB[4:0]	Deslocar o iB pra direita a quantidade de bits do iA estendendo com o bit mais significativo	Arrumamos a ordem trocada dos operandos
12	**LUI**	{iA[31:12],12'b0}	Considerar os 20 bits mais significativos e completa o resultado com 12 bits	Arrumamos para pegar os 20 mais significativos
13	**MUL**	mul[31:0]	multiplicar iA e iB retornando os 32 bits menos significativos	
14	**MULH**	mul[63:32]	multiplicar iA e iB retornando os 32 mais significativos	
15	**MULHU**	mulu[63:32]	multiplicar o módulo de iA e o modulo de iB retornando os 32 mais significativos	
16	**MULHSU**	mulsu[63:32]	multiplicar o módulo de iA de iB retornando os 32 mais significativos	
17	**DIV**	iA / iB	Dividir o iA pelo iB retornando valor inteiro sem resto	
18	**DIVU**	$unsigned(iA) / $unsigned(iB)	Dividir o módulo do iA pelo o modulo do iB retornando valor inteiro sem resto	
19	**REM**	iA % iB	retorna resto de divisão do iA por iB	
20	**REMU**	$unsigned(iA) % $unsigned(iB)	retorna o resto de divisão do módulo de iA pelo módulo de iB	

Tabela 4 - Tabela verdade ULA com comentários.

b) O código utilizado como testbench (ALU_tb.v) encontra-se na pasta Grupo4>questao3_testbenchPrintsECodigo, juntamente com as fotos das entradas e resultados obtidos do modelsim, assim como é mostrado na Figura 6.

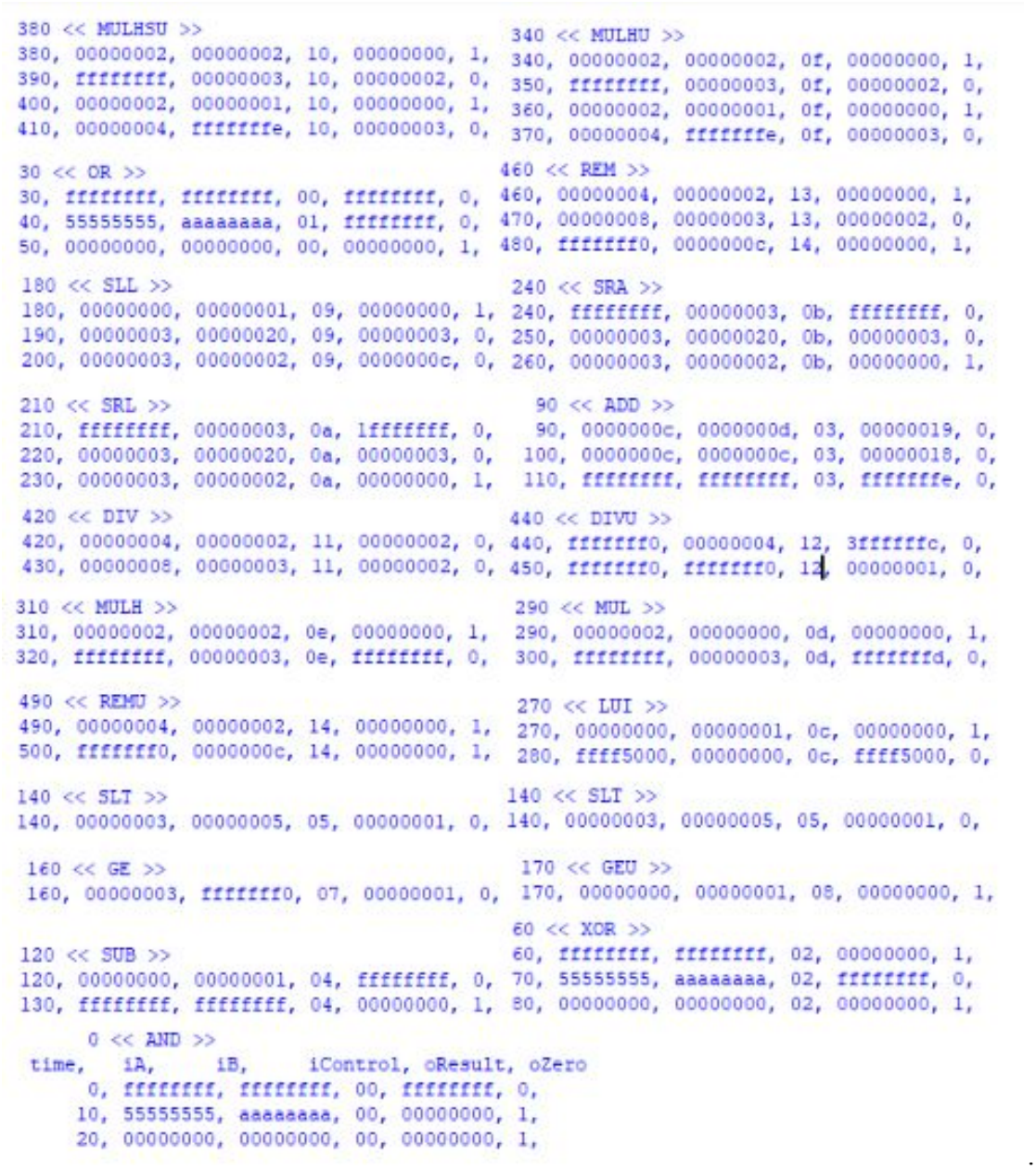

Figura 6 - Resultados obtidos no ModelSim.

c, d) A Tabela 5 apresenta os resultados encontrados para os requisitos físicos (item c) e requerimentos temporais (item d) da implementação da ULA total e para cada operação separadamente.

Operação	ALM	Registradores	Quantidade de bits de memória	Quantidade de dsps	O caminho de maior atraso	Maior tempo de atraso tpd(ns)	Se há algum requerimento não atendido.
AND	28 / 32,070 (< 1 %)	0	0	0	RF de iA[7] para oZero	12.192	
OR	28 / 32,070 (< 1 %)	0	0	0	RF de iA[7] para oZero	12.192	
XOR	28 / 32,070 (< 1 %)	0	0	0	FF de iA[7] para oZerp	12.611	
ADD	23 / 32,070 (< 1 %)	0	0	0	FR de iA[8] para oZero	14.696	
SUB	23 / 32,070 (< 1 %)	0	0	0	FF de iB[17] para oZero	16.004	
SLT	31 / 32,070 (< 1 %)	0	0	0	FF de iB[3] para oResult[0]	14.107	
SLTU	31 / 32,070 (< 1 %)	0	0	0	FF de iB[3] para oResult[0]	14.107	
GE	31 / 32,070 (< 1 %)	0	0	0	FR de iB[3] para oResult[0]	14.107	
GEU	31 / 32,070 (< 1 %)	0	0	0	FR de iB[3] para oResult[0]	14.107	
SLL	83 / 32,070 (< 1 %)	0	0	0	FF de iA[5] para oResult[31]	18.292	
SRL	82 / 32,070 (< 1 %)	0	0	0	RF de iA[18] para oZero	21.008	Não cabe num clock de 50Mhz
SRA	85 / 32,070 (< 1 %)	0	0	0	FF de iA[16] para oResult[8]	19.788	
LUI	5 / 32,070 (< 1 %)	0	0	0	RF de iA[12] para oZero	10.553	
MUL	15 / 32,070 (< 1 %)	0	0	2	FF de iB[13] para oResult[25]	18.883	
MULH	32 / 32,070 (< 1 %)	0	0	3	FF de iA[30] para oResult[12]	20.638	Não cabe num clock de 50Mhz

MULHU	32 / 32,070 (< 1 %)	0	0	3	FF de iB[15] para oResult[3]	21.824	Não cabe num clock de 50Mhz
MULHSU	32 / 32,070 (< 1 %)	0	0	3	FF de iB[15] para oResult[3]	21.824	Não cabe num clock de 50Mhz
DIV	675 / 32,070 (2 %)	0	0	0	FF de iB[8] para oZero	103.604	Não cabe num clock de 50Mhz
DIVU	595 / 32,070 (2 %)	0	0	0	RF de iB[28] para oZero	96.986	Não cabe num clock de 50Mhz
REM	704 / 32,070 (2 %)	0	0	0	FF de iB[15] para oZero	104.601	Não cabe num clock de 50Mhz
REMU	623 / 32,070 (2 %)	0	0	0	FF de iB[27] para oResult[7]	96.890	Não cabe num clock de 50Mhz

Tabela 5 - Requisitos físicos e temporais da ULA total.

e) Vídeo do funcionamento: https://youtu.be/O5kNL6tztnE - "Laboratório 2 - OAC - ULA de Inteiros e Float".

f) Otimizamos o circuito cortando variáveis de multiplicação globais e aplicando a multiplicação dentro de cada instrução usando variável auxiliar como wire. O código antigo e o código otimizado da ula encontram-se na pasta Grupo4>questao3_otimizacao como ULA_OTIMIZADA.v e ULA_SEM-OTIMIZACAO.v.

	ALM	DSPS
ULA SEM OTIMIZAÇÃO	2,982 / 32,070 (9 %)	9 / 87 (10 %)
ULA COM OTIMIZAÇÃO	**2,972** / 32,070 (9 %)	**6** / 87 (7 %)

Tabela 6 - Comparativo entre a ula com e sem otimização.

3) Unidade Aritmética de Ponto Flutuante

a) O diagrama apresentado na figura 7 foi obtido a partir dos arquivos disponibilizados pelo professor no arquivo Lab2, na pasta FPALU. A tabela 7 contém as informações quanto à descrição das funções utilizadas na ULA de ponto flutuante 32 bits.

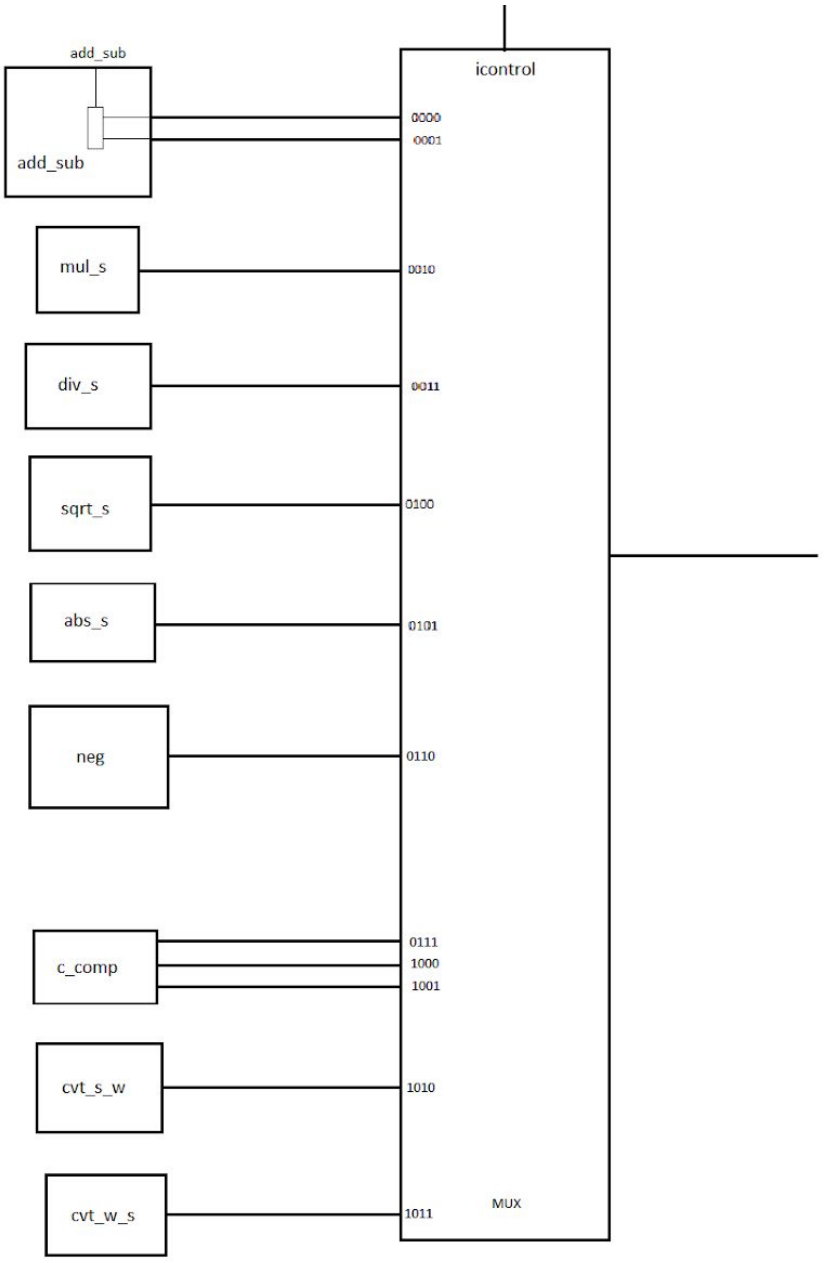

Figura 7 - Diagrama de blocos da ULA de ponto flutuante síncrona

IControl	Operação	Saída	Explicação	Comentários
0	**ADD**	iA + iB	Soma de iA+iB em ponto flutuante(PF)	
1	**SUB**	iA - iB	Subtração de iA-iB em PF	
2	**MUL**	iA * iB	Multiplicação de iA*iB em PF	
3	**DIV**	iA / iB	Divisão de iA/iB em PF	
4	**SQRT**	sqrt(iA)	Raiz quadrada de iA em PF	
5	**ABS**	abs(iA)	Valor absoluto de iA em PF	
6	**NEG**	~(iA)	Negação de iA em PF	
7	**CEQ**	iA == iB	Se iA == iB, CompResult = 1	

8	**CLT**	iA < iB	Se iA < iB, CompResult = 1	
9	**CLE**	iA <= iB	Se iA <= iB, CompResult = 1	
10	**CVTSW**	(float)iA <- (int)iA	Converte um inteiro 32 bits para ponto flutuante de precisão simples	
11	**CVTWS**	(int)iA <- (float)iA	Converte um número em ponto flutuante de precisão simples em inteiro 32 bits	

Tabela 7 - Tabela verdade FP.ULA com comentários.

b) A Figura 8 apresenta o resultado de simulação do testbench dado no arquivo FPALU_tb.v. Nele são testadas as instruções da ULA de ponto flutuante 32 bits e algumas instruções específicas que geram resultados singulares como overflow, underflow e divisão por zero.

```
#                  0<< Operacoes com resultados comuns >>
# << ADD >>
#                time, iA,       iB,       iControl, oResult,      oZero, oNaN, oVerflow, oUnderflow, oCompResult
#                  0,  3f800001, 40000000,  0,       00000000,      1,    0,    0,         0,          0
#                 65,  3f800001, 40000000,  0,       40400000,      0,    0,    0,         0,          0
# << SUB >>
#                200,  40000000, 40800000,  1,       40400000,      0,    0,    0,         0,          0
#                265,  40000000, 40800000,  1,       c0000000,      0,    0,    0,         0,          0
# << MUL >>
#                400,  40400000, 3fc00000,  2,       41000000,      0,    0,    0,         0,          0
#                445,  40400000, 3fc00000,  2,       40900000,      0,    0,    0,         0,          0
# << DIV >>
#                600,  40400000, 40000000,  3,       40000000,      0,    0,    0,         0,          0
#                655,  40400000, 40000000,  3,       3fc00000,      0,    0,    0,         0,          0
# << SQRT >>
#                800,  41500000, 00000000,  4,       3fddb3d7,      0,    0,    0,         0,          0
#                955,  41500000, 00000000,  4,       4066c15a,      0,    0,    0,         0,          0
# << ABS >>
#               1000,  bf800000, 00000000,  5,       3f800000,      0,    0,    0,         0,          0
# << NEG >>
#               1200,  3f800000, 00000000,  6,       bf800000,      0,    0,    0,         0,          0
# << CEQ >>
#               1400,  3f800000, c0800000,  7,       00000000,      0,    0,    0,         0,          0
# << CLT >>
#               1600,  3f800000, 40000000,  8,       00000000,      0,    0,    0,         0,          0
#               1605,  3f800000, 40000000,  8,       00000000,      0,    0,    0,         0,          1
# << CLE >>
#               1800,  3f800000, 40000000,  9,       00000000,      0,    0,    0,         0,          1
# << CVTSW >>
#               2000,  00000004, 00000000, 10,       4e7e0000,      0,    0,    0,         0,          0
#               2055,  00000004, 00000000, 10,       40800000,      0,    0,    0,         0,          0
# << CVTWS >>
#               2200,  40800000, 00000000, 11,       00000000,      1,    0,    0,         0,          0
#               2255,  40800000, 00000000, 11,       00000004,      0,    0,    0,         0,          0
#               2400<< Operacoes com resultados singulares >>
# << ADD entre um numero muito grande e um muito pequeno >>
#               2400,  7f000000, 00800000,  0,       40800000,      0,    0,    0,         0,          0
#               2465,  7f000000, 00800000,  0,       7f000000,      0,    0,    0,         0,          0
# << Divisao por zero >>
#               2600,  40400000, 00000000,  3,       7f800000,      0,    0,    1,         0,          0
#               2655,  40400000, 00000000,  3,       7f800000,      0,    0,    0,         0,          0
# << MUL de (2^127)*(2^127) resulta em Overflow >>
#               2800,  7f000000, 7f000000,  2,       00000000,      1,    0,    0,         0,          0
#               2845,  7f000000, 7f000000,  2,       7f800000,      0,    0,    1,         0,          0
# << Divisao de 2^(-126) por 2^8 resulta em Underflow >>
#               3000,  00800000, 43800000,  3,       3f800000,      0,    0,    0,         0,          0
#               3055,  00800000, 43800000,  3,       00000000,      1,    0,    0,         1,          0
# << Resultado de 5-5 = 0 >>
#               3200,  40a00000, c0a00000,  0,       c3800000,      0,    0,    0,         0,          0
#               3265,  40a00000, c0a00000,  0,       00000000,      1,    0,    0,         0,          0
#               3400<< Final da Simulacao >>
```

Figura 8 - Resultados obtidos no ModelSim da simulação da FPULA.

c, d) A Tabela 8 e Tabela 9 apresentam os resultados encontrados para os requisitos físicos (item c) e requerimentos temporais (item d) da implementação da ULA total e para cada operação separadamente. Note que um "x" nas tabelas indica um item não existente para aquela instrução (elementos síncronos não são caracterizados por time propagation delay, por exemplo).

As instruções ABS e NEG (valor absoluto e negação, respectivamente) em ponto flutuante não tem tempos representativos na tabela 9. Isto acontece, pois os circuitos destas instruções são puramente combinacionais. Logo os tempos de hold, clock to output, setup e slacks não caracterizam estes circuitos. Ao invés disto, utilizamos o time propagation delay para caracterizar tais instruções.

Operação	ALM	Registradores	Quantidade de bits de memória	Quantidade de dsps	O caminho de maior atraso	Maior tempo de atraso tpd(ns)	Se há algum requerimento não atendido.
ADD	364	387	0	0	x	x	x
SUB	364	387	0	0	x	x	x
MUL	96	196	0	1	x	x	x
DIV	110	233	4640	6	x	x	x
SQRT	240	532	143	0	x	x	x
ABS	1	0	0	0	FF de idataa[14] para oresult[14]	9.891	
NEG	1	0	0	0	FF de idataa[15] para oresult[15]	9.817	
CEQ	31	1	0	0	x	x	x
CLT	39	1	0	0	x	x	x
CLE	40	1	0	0	x	x	x
CVTSW	127	249	0	0	x	x	x
CVTWS	146	282	0	0	x	x	x
ULA completa	1132	1605	4935	7	FF de icontrol[1] para oresult[26]	16.898	

Tabela 8 - Requisitos físicos e temporais da ULA de ponto flutuante.

Operação	th (ns)	tco (ns)	tsu (ns)	slack setup (ns)	slack hold (ns)	slack minimum pulse width	Máxima frequência de clock utilizável (MHz)
ADD	1.391	11.568	13.808	14.955	0.228	9.387	198.22
SUB	1.391	11.568	13.808	14.955	0.228	9.387	198.22
MUL	1.372	10.989	4.253	14.237	0.392	9.380	173.52
DIV	1.582	17.266	6.323	12.771	0.221	8.880	138.33
SQRT	1.560	11.103	2.736	14.167	0.244	8.882	171.44
ABS	x	x	x	x	x	x	x
NEG	x	x	x	x	x	x	x
CEQ	-1.528	7.473	3.612	x	x	9.330	x
CLT	0.286	7.901	5.758	6.092	x	4.273	x
CLE	0.063	7.650	9.993	x	x	4.279	x
CVTSW	1.347	9.783	4.742	4.892	0.382	4.394	195.77
CVTWS	1.579	11.946	3.311	6.087	0.276	4.380	255.56
ULA completa	1.509	16.453	11.225	2.695	0.234	3.878	136.89

Tabela 9 - Requerimentos temporais da ULA de ponto flutuante.

e) Vídeo do funcionamento é o mesmo da questão 3)e), nele tem operações da ULA de inteiros e da de Float.

f) Foram feitas várias tentativas e não obtivemos resultados otimizados mudando o código da ula de floats.