Relatório - Laboratório 2

Eduardo Ferreira de Assis, 17/0102289
Thiaggo Ferreira Bispo de Souza, 17/0157024
Emanoel Johannes Cardim Lazaro, 17/0140997
Alexandre Souza Costa Oliveira, 17/0098168
Gabriel Pinheiro dos Santos, 170103579
Maurílio de Jesus Silveira , 17/0152294

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CiC 116394 - Organização e Arquitetura de Computadores - Turma A

Questões

2.

a. A ULA de inteiros fornecida é uma unidade capaz de fazer operações: são elas AND, OR, XOR, ADD(soma), SUB(subtração), SLT(set on less than), SLTU(set on less than unsigned), SLL(shift left logical), SRL(shift right logical), SRA(shift right arithmetic), LUI(load upper immediate), MUL(multiplicação), MULH(multiplicação (high)), MULHU(MULH Unsigned), MULHSU(MULH Signed Unsigned), DIV(Divisão), DIVU(Divisão Unsigned), REM(Resto da divisão), REMU(Resto da divisão Unsigned) e OPNULL(não faz nada). Para executar essas funções, a ULA tem duas entradas de dados sendo elas iA e iB e uma entrada de controle, para escolher as operações, iControl e uma saída chamada oResult que disponibiliza o resultado da operação executada. No verilog da ULA, cada operação possui o prefiro "OP" antes. A imagem abaixo mostra os códigos de cada operação e a sua descrição de acordo com o que foi implementado em verilog.

Operação	Código	Descrição
OPAND	00000	Realiza a operação AND entre iA e iB e coloca o reusltado em oResult
OPOR	00001	Realiza a operação OR entre iA e iB e coloca or resultado em oResult
OPXOR	00010	Realiza a operação XOR entre iA e iB
OPADD	00011	Realiza a operação de soma de iA e iB e coloca no oResult o resultado
OPSUB	00100	Realiza a subtração iA-iB e coloca no oResult o resultado
OPSLT	00101	Verifica se iA é menor que iB, se sim oResult = 1 se não, 0
OPSLTU	00110	Verifica se iA é menor que iB, ambos sem sinal, se sim oResult = 1 se não, 0
OPSLL	00111	oResult recebe o deslocamento lógico de iA para esquerda, iB vezes
OPSRL	01000	oResult recebe o deslocamento lógico de iA para a direita, iB vezes
OPSRA	01001	oResult recebe o deslocamento aritmético de iA para a direita, iB vezes
OPLUI	01010	oResult recebe iB
OPMUL	01011	oResult recebe os 32 bits menos significativos de iA multiplicado por iB
OPMULH	01100	oResult recebe os 32 bits mais significativos de iA multiplicado por iB
OPMULHU	01101	oResult recebe os 32 bits mais significativos de iA multiplicado por iB, ambos sem sinal
OPMULHSU	01110	oResult recebe os 32 bits mais significativos de iA multiplicado por iB, sendo iA com sinal e iB sem sinal
OPDIV	01111	oResult recebe iA dividido por iB
OPDIVU	10000	oResult recebe iA dividido por iB, ambos sem sinal
OPREM	10001	oResult recebe o resto da divisão de iA por iB
OPREMU	10010	oResult recebe o resto da divisão de iA por iB, ambos sem sinal
OPNULL	11111	oResult recebe zero (o processador não faz nada)

Figura 1. Códigos e descrição das funções

b. No Waveform as operações foram feitas na ordem em que foram definidas nos parâmetros do processador, de 1 a 18, excluindo a OPNULL pois a ideia é que esta não faça nenhuma operação.

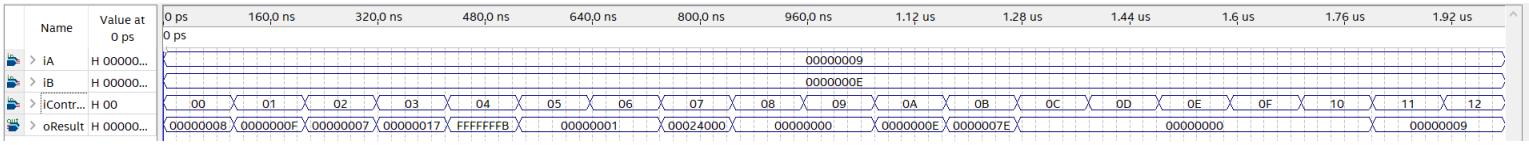

Figura 2. Waveform com entradas representativas

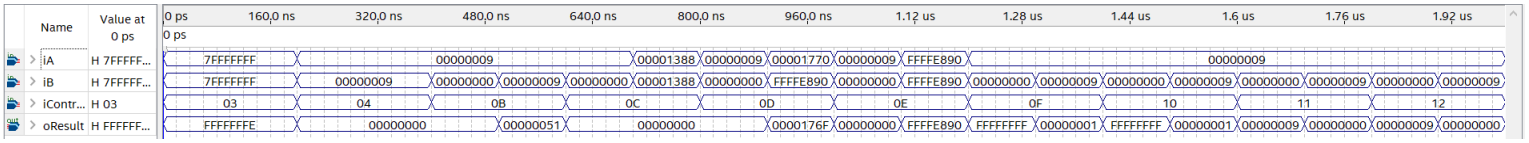

Figura 3. Waveform com entradas que geram resultados singulares

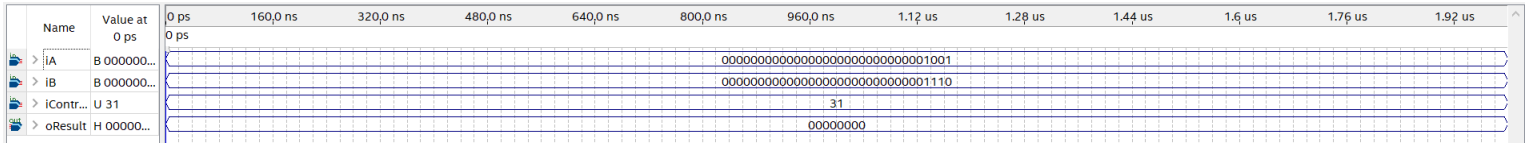

Figura 4. Waveform da função OPNULL. A saída sempre será zero.

No RTL Simulation algumas saídas divergem das saídas das Waveforms pois no Waveform temos uma visão mais funcional do circuito, assim, ainda que haja divisão por zero ou qualquer tipo de incoerência matemática, haverá um output.

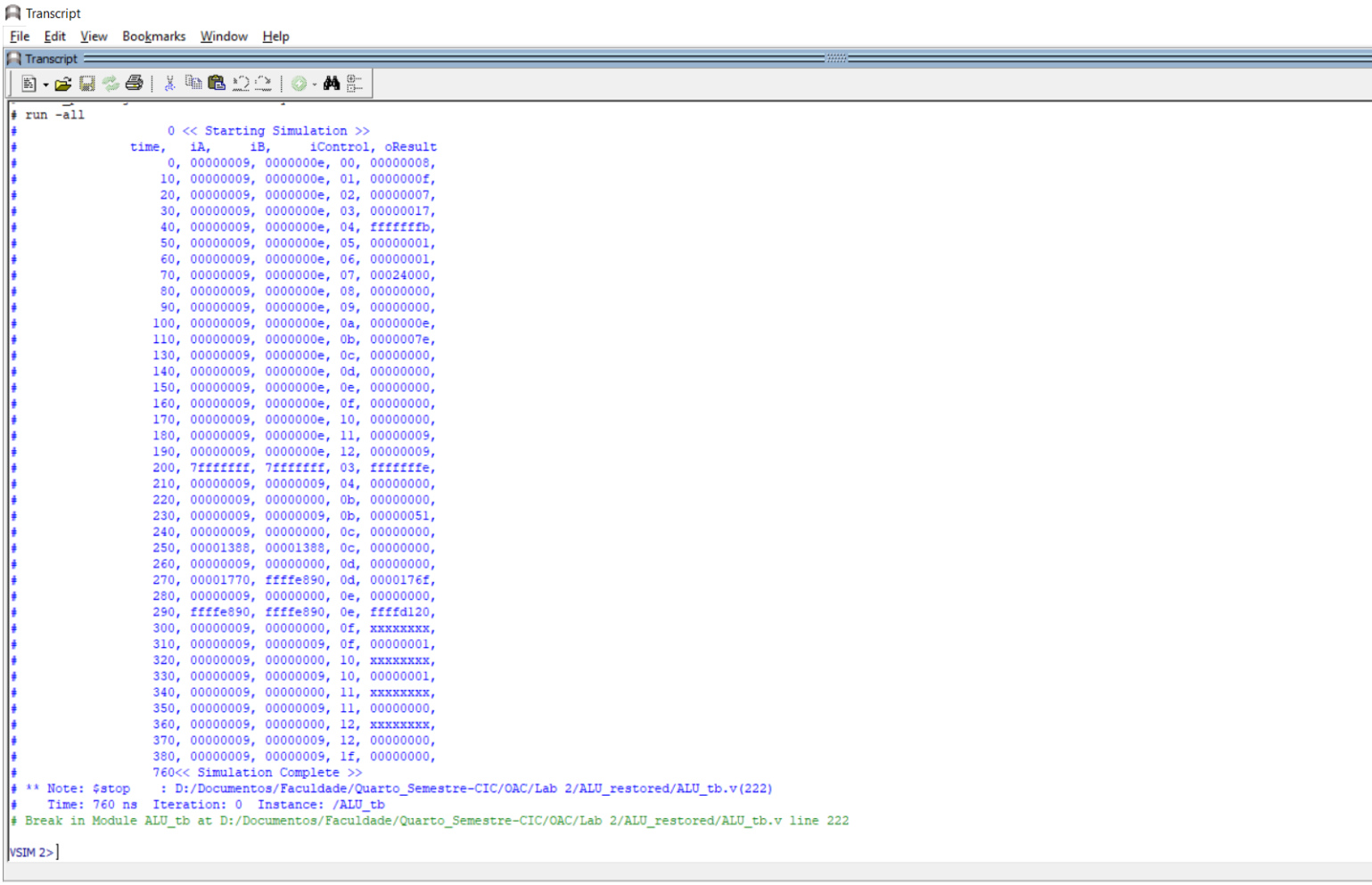

Figura 5. RTL Simulation

c.

	ALMs	Nº de Registradores	Bits de memória	Número de blocos DSP
OPAND	65	0	0	0
OPOR	65	0	0	0
OPXOR	65	0	0	0
OPADD	65	0	0	0
OPSUB	65	0	0	0
OPSLT	79	0	0	0
OPSLTU	79	0	0	0
OPSLL	79	0	0	0
OPSRL	121	0	0	0
OPSRA	118	0	0	0
OPLUI	49	0	0	0
OPMUL	56	0	0	2
OPMULH	72	0	0	3
OPMULHU	72	0	0	3
OPMULHSU	88	0	0	6
OPDIV	704	0	0	0
OPDIVU	633	0	0	0
OPREM	731	0	0	0
OPREMU	652	0	0	0
OPNULL	49	0	0	0
Soma	3907	0	0	14
ULA total	3029	0	0	12

Figura 6. Tabela com os requisitos das operações, com a soma dos requisitos e com os requisitos da ULA total

v) A soma do número de ALMs de cada operação é maior do que o número de ALMs da ULA total pois é possível aproveitar alguns circuitos para executar diferentes operações e é exatamente isso que o Quartus Prime faz. Dessa forma, reduz-se consideravelmente o número de ALMs necessários para fazer a ULA.

vi) As funções cujo circuito ocupa uma maior parte no tamanho da ULA são a OPDIV (divisão), OPDIVU(divisão com entradas unsigned), OPREM(resto da divisão) e OPREMU(resto da divisão com entradas unsigned), pois utilizam um grande número de ALMs e, ainda que sejam parecidos, são circuitos muito complexos e ocupam uma grande área de chip.

d. i) O caminho de maior atraso é o FF (Fall to Fall)

ii)O maior tempo é 106.096

iii)

Operação	tpd	Caminho de maior atraso
OPAND	2.421	FF
OPOR	2.421	FF
OPXOR	2.421	FF
OPADD	3.127	FF
OPSUB	3.282	FF
OPSLT	4.490	FF
OPSLTU	4.490	FF
OPSLL	4.150	FF
OPSRL	4.412	FF
OPSRA	4.639	FF
OPLUI	0.516	FF
OPMUL	7.734	FF
OPMULH	8.156	FF
OPMULHU	8.684	FF
OPMULHSU	10.803	FF
OPDIV	94.892	FF
OPDIVU	83.135	FF
OPREM	90.438	FF
OPREMU	88.790	FF
OPNULL	---------	--
ULA total	106.096	FF

Figura 7. Tabela com os tpds de cada operação e da ula total, assim como o seu caminho de maior atraso.

As operações que mais influenciam no tpd da ULA total são OPDIV, OPDIVU, OPREM e OPREMU. A OPDIV é a que mais influencia, sendo 89,43% do tpd da ULA total, o que nos mostra, que o circuito para esta operação é complexo e requer não só uma maior área de chip, como uma maior quantidade de tempo para ser executada.

e. Este foi o vídeo gravado para a questão: `https://youtu.be/9xJJv2vyvBs` Nem todos os testes feitos foram contemplados pois display não nos permite ver os valores por inteiro e o número de switches não permite inserir grandes valores de entrada.

f. Foi verificado que o grupo não foi capaz de otimizar o circuito uma vez que seu código está simplificado e as operações realizadas pela ULA são diretamente utilizadas em sua forma de Verilog.

3.

a.

Funções	Códigos
FOPADD	0
FOPSUB	1
FOPMUL	2
FOPDIV	3
FOPSQRT	4
FOPABS	5
FOPCEQ	6
FOPCLT	7
FOPCLE	8
FOPCVTSW	9
FOPCVTWS	10
FOPMV	11
FOPSGNJ	12
FOPSGNJN	13
FOPSGNJX	14
FOPMAX	15
FOPMIN	16
FOPCVTSWU	17
FOPCVTWUS	18
FOPNULL	31

Tabela 1. Tabela das operações da FPULA.

A FPULA é a unidade lógica aritmética responsável por realizar as operações de ponto flutuante no processador. Como podemos ver em FPULA o módulo possui 20 operações possíveis, sendo **FOPNULL** uma operação default (i.e., não faz nenhum procedimento).

Analisando o verilog, o **módulo FPALU** (i.e. FPULA) recebe 2 inputs que serão processados, *idataa* e *idatab*. A operação a ser realizada é definida pelo controle do processador, externo a FPULA, e é informada através de *icontrol*. Há também o *iclock* responsável por receber o CLOCK de operação para os circuitos sequenciais da FPALU.

Cada operação necessita de um tempo (ciclos de clock) diferente para ser computada, esse tempo é definido dentro da FPALU por *ciclos*. Quando o resultado estiver pronto *oresult* conterá a resposta e *oreadye* estará em **1**, indicando que a FPALU terminou de operar e a resposta contida em *oresult* é o valor final do procedimento.

Cabe a observação de que o módulo FPALU possuí submódulos responsáveis que desempenham funções específicas, isto é, para a operação de adição por exemplo, existe o submódulo *add1* responsável por efetivamente realizar a soma/subtração das 2 entradas. E assim consequentemente para cada operação: *mul1, div1, sqrt1, comp_s1, cvt_s_w1, cvt_w_s1, cvt_s_wu1, fmax_s1, fmin_s1*.

Funções	Descrição
FOPADD	Realiza a soma de 2 números em ponto flutuante.
FOPSUB	Realiza a subtração de 2 números em ponto flutuante.
FOPMUL	Realiza a multiplicação de 2 números em ponto flutuante.
FOPDIV	Realiza a divisão de 2 números em ponto flutuante.
FOPSQRT	Efetua a raiz quadrada de 1 número em ponto flutante.
FOPABS	Calcula o valor absoluto (módulo) de 1 número em ponto flutuante.
FOPCEQ	Compara se 2 números em fp são iguais (1) ou não (0).
FOPCLT	Compara se num1* <num2* em fp.
FOPCLE	Compara se num1 <= num2 em fp.
FOPCVTSW	Converte de word para simple.
FOPCVTWS	Converte de simple para word.
FOPMV	Move de simple para word (e vice-versa).
FOPSGNJ	Define o sinal (sinal(num2), valor(num1)).
FOPSGNJN	Define o sinal (∼sinal(num2), valor(num1)).
FOPSGNJX	Define o sinal (sinal(sinal2 ˆnum1), valor(num1)).
FOPMAX	Retorna o maior valor entre num1 e num2.
FOPMIN	Retorna o menor valor entre num1 e num2.
FOPCVTSWU	Converte de unsigned word para simple.
FOPCVTWUS	Converte de simple para unsigned word.
FOPNULL	Coloca EEEEEEEE na saída (default).

Tabela 2. Tabela das operações e suas definições da FPULA.

(*) num1: número em ponto flutuante.
(*) num2: número em ponto flutuante.

b.

Operação	Resultado
FOPADD	0x7F7FFFFF + 0x7F7FFFFF (número muito grande) = Infinito
FOPADD	0xFF7FFFFF + 0xFF7FFFFF (número ngeativo muito pequeno) = -Infinito
FOPSUB	0xFF7FFFFF (número negativo muito pequeno) - 0x7F7FFFFF (número muito grande) = - Infinito
FOPSUB	0x7F7FFFFF - 0xFF7FFFFF = Infinito
FOPMUL	2 números muito grandes = Infinito
FOPMUL	2 números muito pequenos = Zero (Underflow)
FOPDIV	0x3F800000 (1) / 0x7F7FFFFF (número muito grande) = Zero (Underflow)
FOPDIV	0x7F7FFFFF / 0x00800001 (número fracionário muito pequeno) = Zerp (Underflow)
FOPDIV	0x7F7FFFFF / 0x000000 = Infinito
FOPSQRT	sqrt(0xFF7FFFFF) (número negativo muito pequeno) = NaN
FOPABS	Representação normal para todos os casos.
FOPCEQ	Representação normal para todos os casos.
FOPCLT	Representação normal para todos os casos.
FOPCLE	Representação normal para todos os casos.
FOPCVTSW	Representação normal para todos os casos.
FOPCVTWS	Representação normal para todos os casos.
FOPMV	Representação normal para todos os casos.
FOPSGNJ	Representação normal para todos os casos.
FOPSGNJN	Representação normal para todos os casos.
FOPSGNJX	Representação normal para todos os casos.
FOPMAX	Representação normal para todos os casos.
FOPMIN	Representação normal para todos os casos.
FOPCVTSWU	Representação normal para todos os casos.
FOPCVTWUS	Representação normal para todos os casos.
FOPNULL	Coloca EEEEEEEE na saída (default).

Tabela 3. Operações e saídas da FPALU.

A tabela 3 foi obtida através da análise e manipulação da *waveform* **FPALU.vwf** e da análise do *testbench* executado pelo **RTL simulator** apresentado a seguir:

```
1  #                  0 << Início da Simulação >>
2  #                time, iA,       iB,       iControl, ISTARTE, oResult,      oReady
3  #                150,  3f800000, 40000000,  0, 1,       40400000,      1
4  #                370,  40000000, 40800000,  1, 1,       c0000000,      1
5  #                530,  40400000, 3fc00000,  2, 1,       40900000,      1
6  #                870,  40000000, 40400000,  3, 1,       3f2aaaab,      1
7  #               1030,  41500000, 00000000,  4, 1,       4066c15a,      1
8  #               1150,  bf800000, 00000000,  5, 1,       3f800000,      1
9  #               1410,  3f800000, 3f800000,  6, 1,       00000001,      1
10 #               1630,  3f800000, 40000000,  7, 1,       00000001,      1
11 #               1850,  bf800000, 3f800000,  8, 1,       00000001,      1
12 #               2090,  00000004, 00000000,  9, 1,       40800000,      1
13 #               2270,  40800000, 00000000, 10, 1,       00000004,      1
14 #               2420<< Final da Simulação >>
```

Figura 8. Saída resumida do RTL simulator.

c.

Função	ALMs	REGs	MEM (bits)	DSPs
Total	**1318**	**1099**	**47616**	**6**
FOPADD	386	284	0	0
FOPSUB	383	289	0	0
FOPMUL	116	76	0	1
FOPDIV	206	285	31744	3
FOPSQRT	126	129	15872	2
FOPABS	49	2	0	0
FOPCEQ	77	25	0	0
FOPCLT	90	31	0	0
FOPCLE	88	31	0	0
FOPCVTSW	186	157	0	0
FOPCVTWS	202	77	0	0
FOPMV	49	2	0	0
FOPSGNJ	49	2	0	0
FOPSGNJN	49	2	0	0
FOPSGNJX	50	2	0	0
FOPMAX	95	2	0	0
FOPMIN	95	2	0	0
FOPCVTSWU	95	2	0	0
FOPCVTWUS	160	110	0	0
FOPNULL	49	2	0	0

Tabela 4. Tabela contendo os recursos utilizados para cada operação da FPULA.

Na tabela 4 podemos observar que o módulo FPULA total aloca 1318 ALMs para o seu funcionamento, isso equivale a cerca de 4% da capacidade da FPGA DE-1. Além disso, vemos que dos 47616 *bits* utilizados, 31744 são utilizados para a operação de divisão. Somando a isso, vemos que o uso de ALMs e registradores pela divisão são consideráveis. Resumindo, dos recursos alocados só para a divisão são utilizados: 15,6% de ALMs, 25,9% dos registradores, 33,3% da memória e 50% das DSPs. Visto isso, vemos que o custo para a implementação da divisão é bem alto, tanto pelo tipo de recursos utilizados, predominantemente memória, quanto pelo espaço ocupado no circuito.

As outras funções têm um gasto bem mais moderado, sendo a segunda de maior gasto de recursos a operação de raiz quadrada (FOPSQRT) seguida da adição (FOPADD).

Somando os recursos utilizados por cada função isoladamente, temos: 2600 ALMs, 1512 REGs, 47616 bits de memória, 6 DSPs. Comparando esses resultados com os recursos alocados na compilacão completa da FPULA, podemos observar que o número de ALMs e REGs do circuito completo (Tabela 4) é menor do que se cada circuito fosse sintetizado isoladamente um dos outros, isso é devido ao foto do Quartus realizar diversas simplificações lógicas, diminuindo a complexidade dos circuitos sempre que possível. Por outro lado, notamos que tanto a memória quanto as DSP não sofreram alterações da compilação individual para a compilação total da FPULA, podemos deduzir que esses são recursos individuais para cada submódulo da FPULA.

d.

Função	Requisitos temporais						
	Ciclos	**TH**	**TCO**	**TSU**	**SLACKS**	**MAX FREQ**	**Requeriments**
Total	**10**	**3.740**	**16.052**	**6.845**	**13.665/-0.015/17.218/0.423**	**157.85 MHz**	**No (slack hold)**
FOPADD	6	3.543	7.905	5.423	14.756/0.378/18.135/0.588	190.69 MHz	Yes
FOPSUB	6	3.463	8.644	3.296	14.919/0.255/18.139/0.567	196.81 MHz	Yes
FOPMUL	3	4.040	8.389	0.499	4.179/0.221/8.095/0.460	171.79 MHz	Yes
FOPDIV	9	3.732	7.570	-0.108	3.492/0.255/7.630/0.495	153.66 MHz	Yes
FOPSQRT	6	3.757	11.602	0.560	15.928/0.173/17.874/0.386	245.58 MHz	Yes
FOPABS	1	1.658	7.420	-0.505	19.130/0.307/-/-	717.36 MHz	Yes
FOPCEQ	3	3.415	8.291	0.380	18.729/0.287/18.819/0.567	717.36 MHz	Yes
FOPCLT	3	3.686	8.710	1.500	8.192/0.255/8.632/0.575	553.1 MHz	Yes
FOPCLE	3	3.439	4.656	0.743	18.235/0.325/18.812/0.570	566.57 MHz	Yes
FOPCVTSW	4	3.526	9.225	4.535	15.813/0.292/18.377/0.571	238.83 MHz	Yes
FOPCVTWS	2	3.400	8.842	2.657	14.295/0.360/18.324/0.559	175.28 MHz	Yes
FOPMV	1	1.627	7.300	-0.456	19.130/0.307/-/-	717.36 MHz	Yes
FOPSGNJ	1	1.627	7.300	-0.456	19.130/0.307/-/-	717.36 MHz	Yes
FOPSGNJN	1	1.627	7.300	-0.456	19.130/0.307/-/-	717.36 MHz	Yes
FOPSGNJX	1	1.627	7.300	-0.456	19.130/0.307/-/-	717.36 MHz	Yes
FOPMAX	1	1.594	7.400	-0.434	19.131/0.308/-/-	717.36 MHz	Yes
FOPMIN	1	1.594	7.400	-0.434	19.131/0.308/-/-	717.36 MHz	Yes
FOPCVTSWU	4	3.209	10.356	3.209	15.329/0.324/18.336/0.770	214.09 MHz	Yes
FOPCVTWUS	2	3.255	10.692	4.296	18.915/0.310/18.635/0.570	717.36 MHz	Yes
FOPNULL	1	1.627	7.300	-0.456	19.130/0.307/-/-	717.36 MHz	Yes

Tabela 5. Requisitos temporais da FPULA para clock de 50 MHz.

Aumentando o clock para 200 MHz através do Timing Analyzer foi possível perceber que dos 4 slacks monitorados (*setup, hold, recovery, removal*) 2 diminuíram: *hold* foi de 13.665 para -1.335, *recovery* foi de 18.135 para 2.218 (na FPULA completa). Além disso, o tempo para *tsu* também sofreu alteração, como podemos ver na tabela 6.

Apesar do aumento significativo do CLOCK (Tabela 5 e Tabela 6, os requisitos temporais do circuito *tsu, th e tco* praticamente não mudaram.

Função	Requisitos temporais						
	Ciclos	**TH**	**TCO**	**TSU**	**SLACKS**	**MAX FREQ**	**Requeriments**
Todas	**10**	**3.740**	**16.052**	**6.845**	**-1.335/-0.015/2.218/0.423**	**157.85 MHz**	**No (slack setup e hold)**
FOPADD	6	3.543	7.905	5.423	-0.244/0.378/3.135/0.588	190.69 MHz	No (slack setup)
FOPSUB	6	3.463	8.644	3.296	-0.081/0.255/3.139/0.567	196.81 MHz	No (slack setup)
FOPMUL	3	4.040	8.389	0.499	-0.821/0.221/3.095/0.460	171.79 MHz	No (slack setup)
FOPDIV	9	3.732	7.706	-0.205	-1.508/0.255/2.630/0.495	153.66 MHz	No (slack setup)
FOPSQRT	6	3.757	11.602	0.560	0.928/0.173/2.874/0.386	245.58 MHz	Yes
FOPABS	1	-0.505	1.658	-0.615	4.130/0.307/-/-	717.36 MHz	Yes
FOPCEQ	3	3.415	8.291	0.380	3.729/0.287/3.819/0.567	717.36 MHz	Yes
FOPCLT	3	3.686	8.710	1.500	3.192/0.255/3.632/0.575	553.1 MHz	Yes
FOPCLE	3	3.439	8.773	0.743	3.235/0.325/3.812/0.570	566.57 MHz	Yes
FOPCVTSW	4	3.526	9.225	4.535	0.813/0.292/3.377/0.571	238.83 MHz	Yes
FOPCVTWS	2	3.400	8.842	2.657	-0.705/0.360/3.324/0.559	175.28 MHz	Yes
FOPMV	1	1.627	7.300	-0.456	4.130/0.307/-/-	717.36 MHz	Yes
FOPSGNJ	1	1.627	7.300	-0.456	4.130/0.307/-/-	717.36 MHz	Yes
FOPSGNJN	1	1.627	7.300	-0.456	4.130/0.307/-/-	717.36 MHz	Yes
FOPSGNJX	1	1.627	7.300	-0.456	4.130/0.307/-/-	717.36 MHz	Yes
FOPMAX	1	1.594	7.400	-0.434	4.131/0.308/-/-	717.36 MHz	Yes
FOPMIN	1	1.594	7.400	-0.434	4.131/0.308/-/-	717.36 MHz	Yes
FOPCVTSWU	4	3.493	10.356	3.209	-3.671/0.324/-0.664/0.770	214.09 MHz	No (slack setup e recovery)
FOPCVTWUS	2	3.298	10.692	4.296	3.915/0.310/3.635/0.570	717.36 MHz	Yes
FOPNULL	1	1.615	7.310	-0.446	4.129/0.306/-/-	717.36 MHz	Yes

Tabela 6. Requisitos temporais da FPULA para clock de 200 MHz.

Função	Requisitos temporais						
	Ciclos	**TH**	**TCO**	**TSU**	**SLACKS**	**MAX FREQ**	**Requeriments**
Todas	**10**	**3.740**	**16.052**	**6.845**	**-0.035/-0.015/3.518/0.423**	**157.85 MHz**	**No (slack setup e hold)**

Tabela 7. Requisitos temporais da FPULA para clock de 157.85 MHz.

Função	Requisitos temporais						
	Ciclos	**TH**	**TCO**	**TSU**	**SLACKS**	**MAX FREQ**	**Requeriments**
Total	**10**	**3.740**	**16.052**	**6.845**	**43.665/-0.015/47.218/0.423**	**157.85 MHz**	**No (slack hold)**

Tabela 8. Requisitos temporais da FPULA para clock de 20 MHz.

Apesar da frequência máxima ter sido definida para 157.85 MHz (tabela 7) no circuito completo da FPULA, mesmo a 50 MHz o circuito falhou nas checagens do requisito temporal (slack hold), e mesmo diminuindo o clock para 20 Mhz, ele continuou não antendendo aos requisitos.

Acrescentando, através da análise da tabela 5 observamos que a divisão é o recurso que mais limita a frequência da FPULA, devido a complexidade do seu circuito e, provavelmente, o uso considerável de memória, sendo esta mais lenta que o resto dos componentes, formados em sua maioria por circuitos combinacionais.

É importante observar que no *Tutorial_Quartus_Prime.pdf* o professor cita: "*O tsu e o tco devem ser sempre positivos, o th deve ser sempre negativo para que o circuito esteja OK em termos de temporização.*" Porém, mesmo infligindo essa condição, nas simulações realizadas no Quartus, o mesmo acusava como OK esses campos no *Timing Analyzer* e o circuito funcionou corretamente. Dessa forma, foi considerado somente os slacks para definir o campo *Requiriments* nas tabelas.

e. Link para o video desmonstrando o funcionamento da FPULA na FPGA: `https://youtu.be/c4LyySe_Y-s`

É importante observar que devido ao fato da limitação dos display de 7 segmentados da FPGA, não foi possível averiguar o funcionamento das seguintes funções: FOPCEQ, FOPCLT, FOPCLE. Além disso, no video não foi checado as funções: FIOSGNJ, FOPSGNJN, FOPSGNJX. É possível conferir essas funções através do *waveform* a seguir:

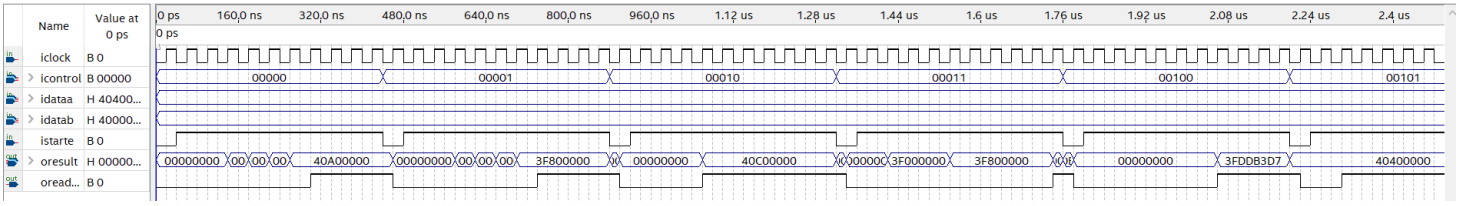

Figura 9. Waveform da FPULA, com os valores 0x40400000 (3) e 0x40000000 (2) como entradas.

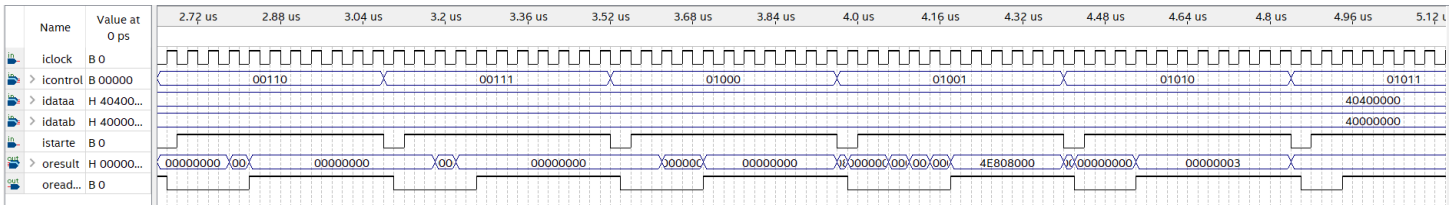

Figura 10. Waveform da FPULA, com os valores 0x40400000 (3) e 0x40000000 (2) como entradas.

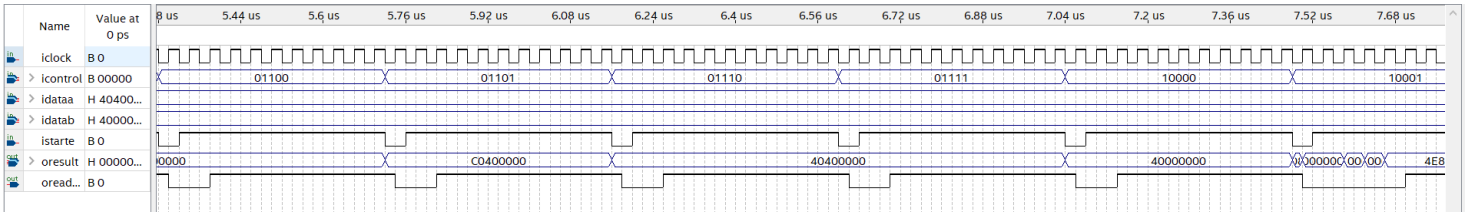

Figura 11. Waveform da FPULA, com os valores 0x40400000 (3) e 0x40000000 (2) como entradas.

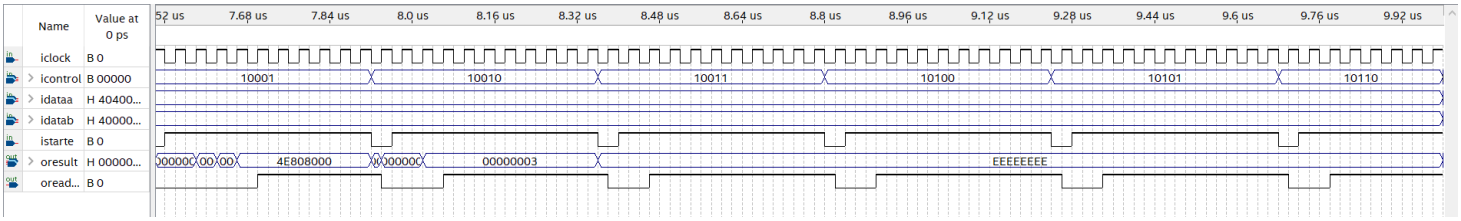

Figura 12. Waveform da FPULA, com os valores 0x40400000 (3) e 0x40000000 (2) como entradas.

f. Foi verificado que o grupo não foi capaz de otimizar o circuito uma vez que seu código está simplificado e as operações realizadas pela ULA são diretamente utilizadas em sua forma de Verilog.