Laboratório 2
- ULA e FPULA –

GRUPO 6
Dayanne Fernandes da Cunha, 13/0107191
Lucas Mafra Chagas, 12/0126443
Marcelo Giordano Martins Costa de Oliveira, 12/0037301
Lucas Junior Ribas, 16/0052289
Caio Nunes de Alencar Osório, 16/0115132
Diego Vaz Fernandes, 16/0117925

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

1. Objetivos

- Introduzir ao aluno a Linguagem de Descrição de *Hardware Verilog*;
- Familiarizar o aluno com a plataforma de desenvolvimento *FPGA DE2* da *Altera* e o *software QUARTUS-II*;
- Desenvolver a capacidade de análise e síntese de sistemas digitais usando *HDL*.

2. Ferramentas

- FPGA DE2 da Altera
- QUARTUS-II
- Verilog HDL

3. Exercícios

Todos os códigos escritos neste laboratório podem ser encontrados no repositório `https://github.com/Dayof/OAC172` do *GitHub*.

3.1. Exercício 1. Implementação de um *driver* para *display* de 7 segmentos

Conforme descrito no arquivo *QuartusIIv3.txt* e *Set.txt*, um novo projeto foi criado no diretório *Lab2*, denominado *Display*.

Para as versões síncrona e assíncrona foram geradas as simulações temporais (Figura 1 e Figura 3) e funcionais (Figura 2 e Figura 4).

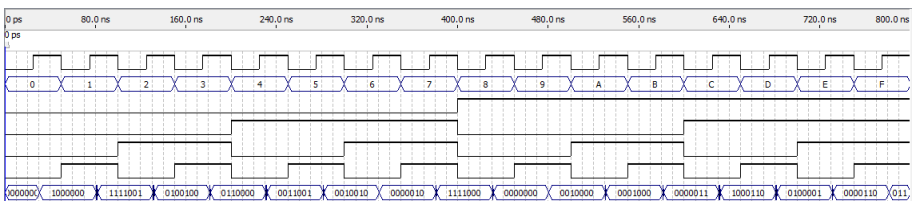

Figure 1. Simulação síncrona temporal do *decoder7*.

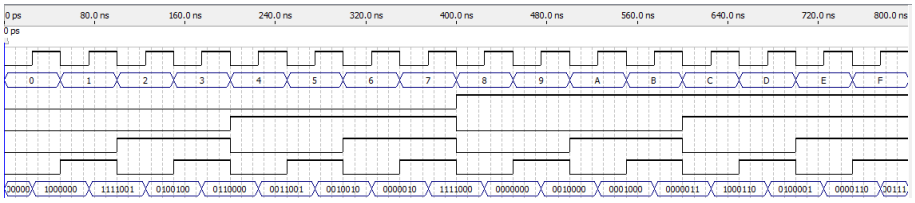

Figure 2. Simulação síncrona funcional do *decoder7*.

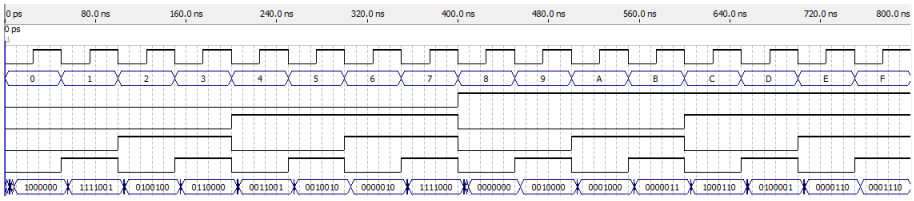

Figure 3. Simulação assíncrona temporal do *decoder7*.

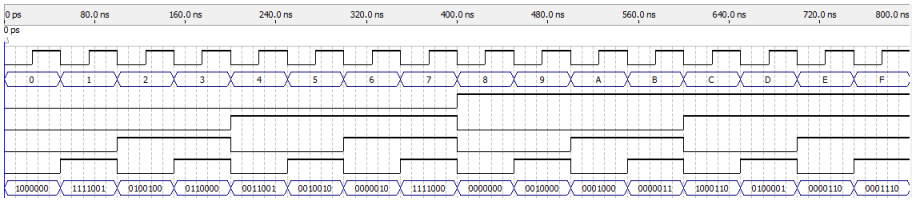

Figure 4. Simulação assíncrona funcional do *decoder7*.

O arquivo de interface *TopDE.v* foi incluso no projeto, sintetizado e testado como é mostrado no link `https://youtu.be/wGKjze5PkcU`.

3.2. Exercício 2. Unidade Lógica Aritmética de Inteiros

3.2.1. Operações

Operação AND bit a bit com os registradores iA e iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

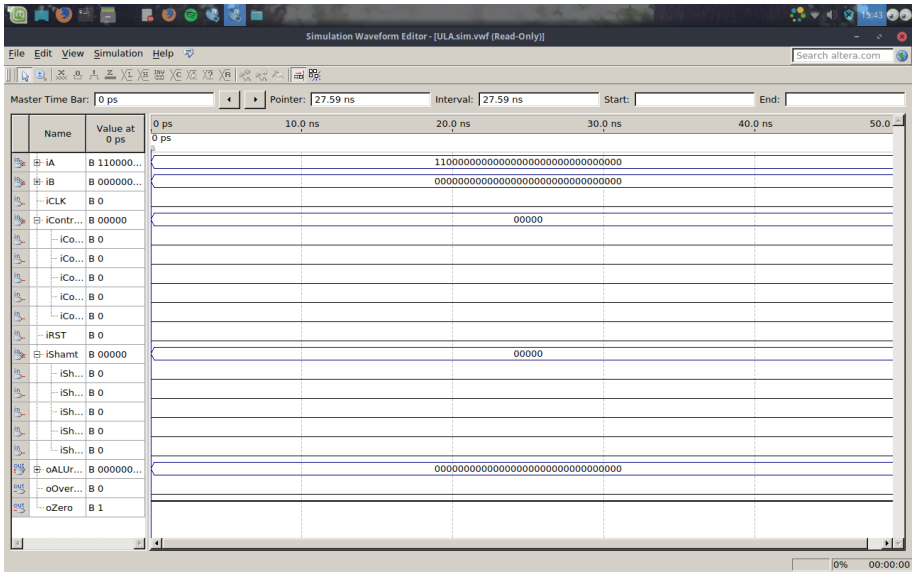

Figure 5. 00000-and-zero

Operação OR bit a bit com os registradores iA e iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

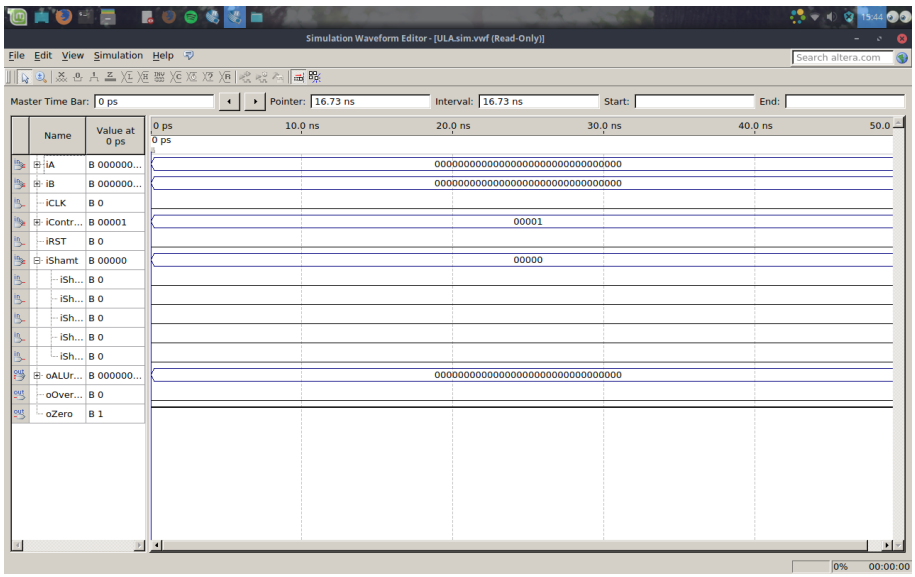

Figure 6. 00001-or-zero

Operação ADD com os registradores iA e iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0. E a saida do registrador oOver que indica overflow com saida 1 se ha e 0 se não.

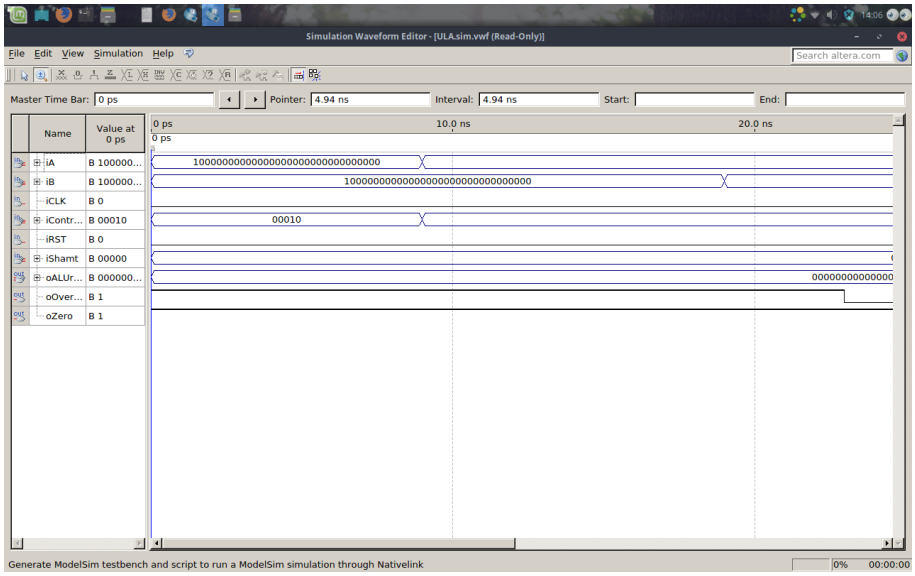

Figure 7. 00010-add-over-zero

Operação MFHI do HI para o oALUresult. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

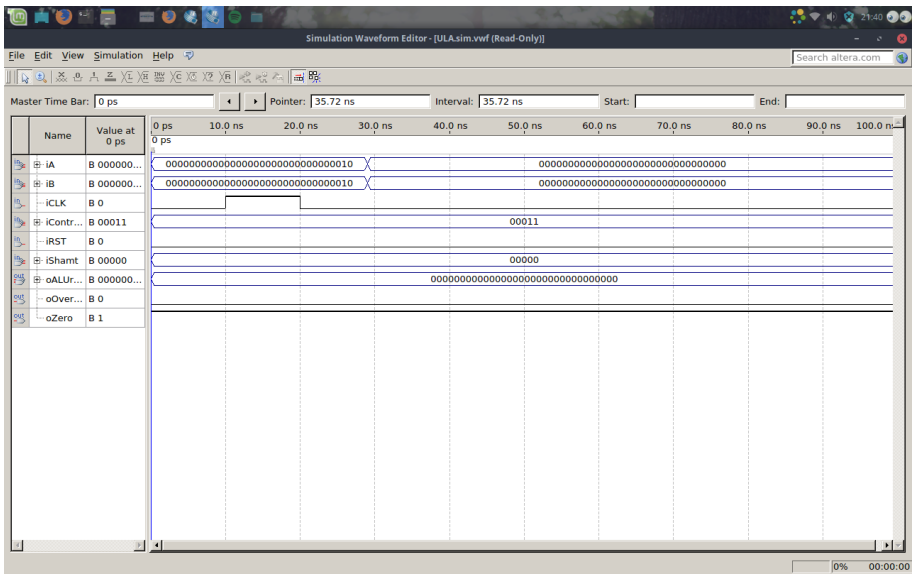

Figure 8. 00011-mfhi-zero

Operação SLL bit a bit com os registradores iA, shifitando o valor no shant em bits no iA. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

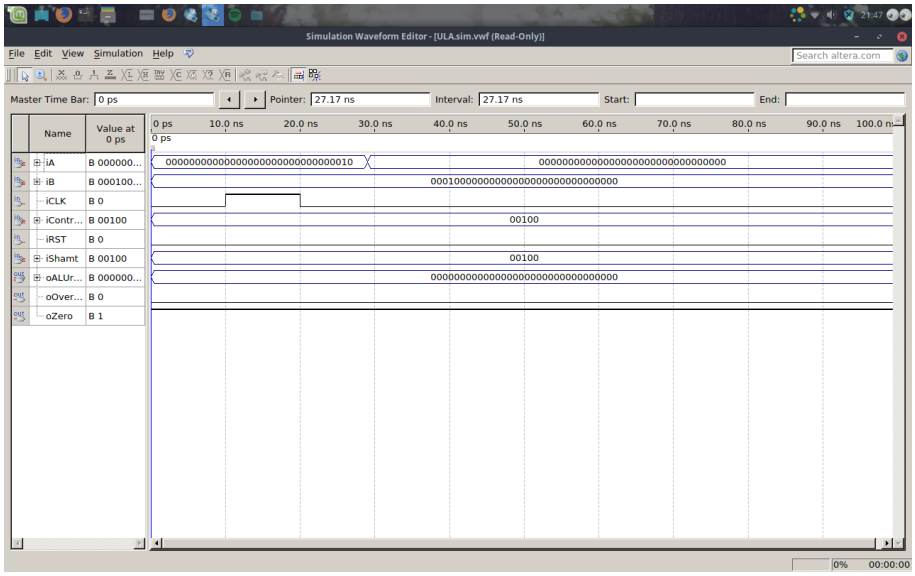

Figure 9. 00100-sll-zero

Operação MFLO do LO para o oALUresult. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

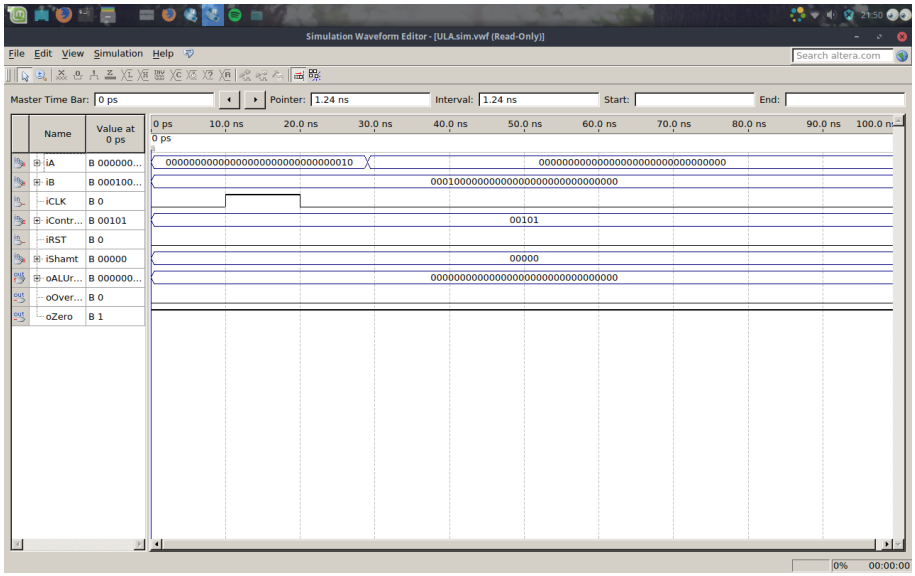

Figure 10. 00101-mflo-zero

Operação SUB com os registradores iA e iB. A imagem a baixo demonstra um dos valores singulares possíveis, como a saida do registrador oOver que indica overflow com saida 1 se ha e 0 se não.

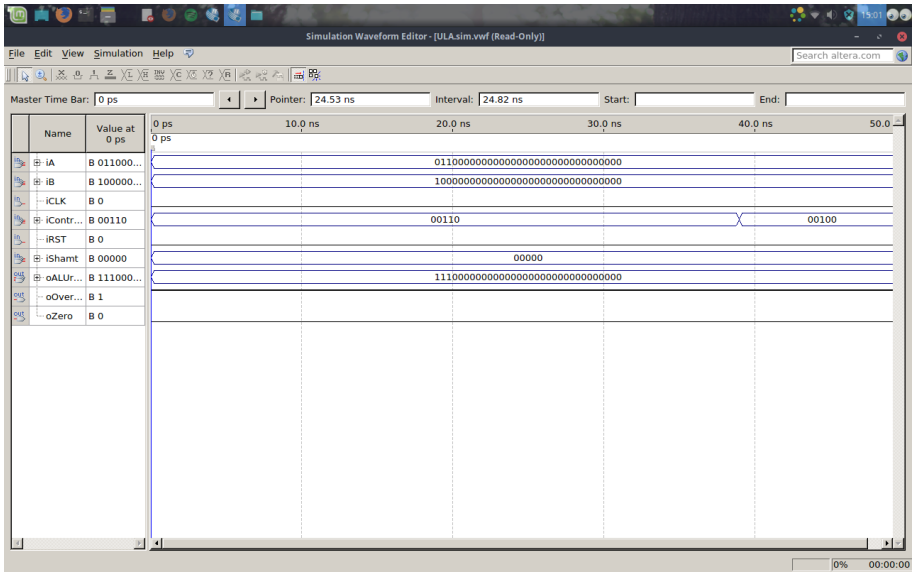

Figure 11. 00110-sub-over

Operação SUB com os registradores iA e iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

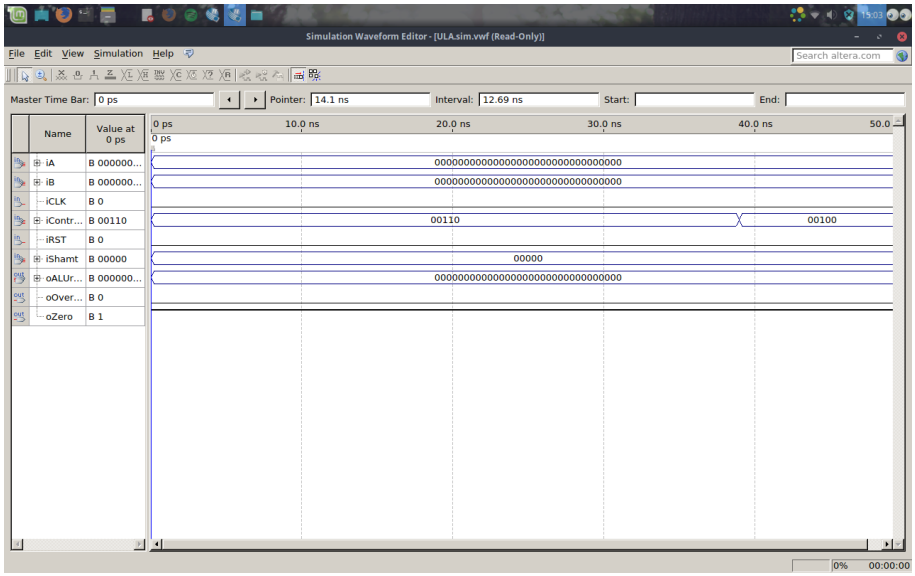

Figure 12. 00110-sub-zero

Operação SLT seta 1 em oALUresult se iA for menor que iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

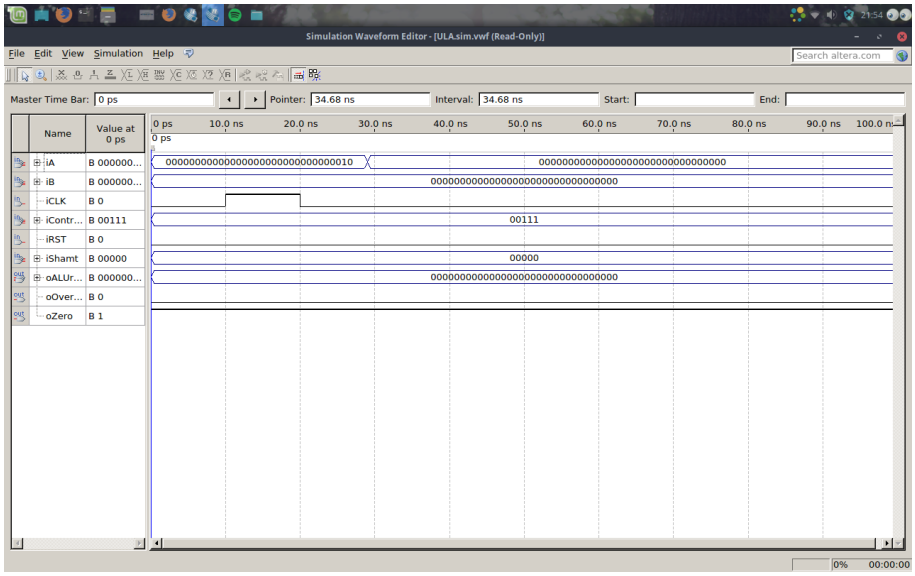

Figure 13. 00111-slt-zero

Operação SRL shitfita o valor em bits no shamt para a direita. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

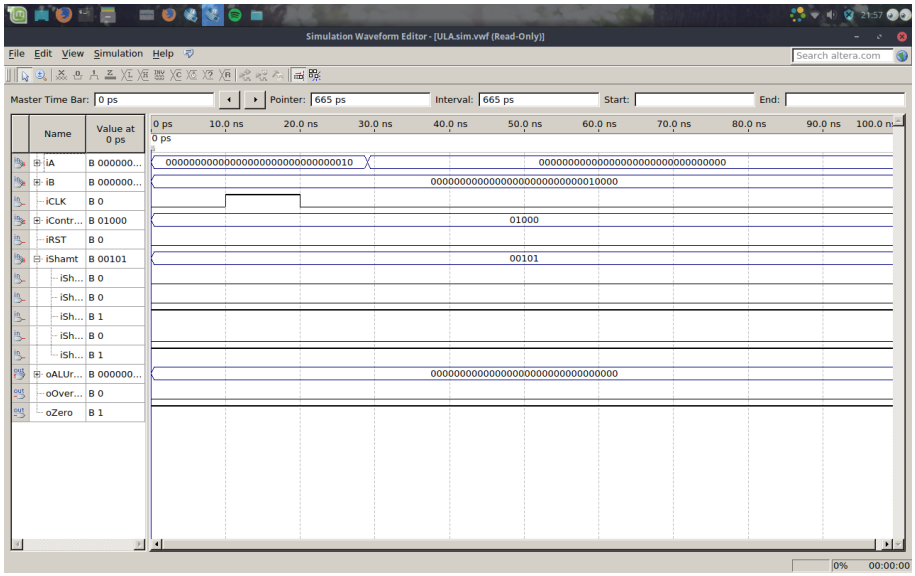

Figure 14. 01000-srl-zero

Operação SRA shitfita o valor em bits no shamt para a direita. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

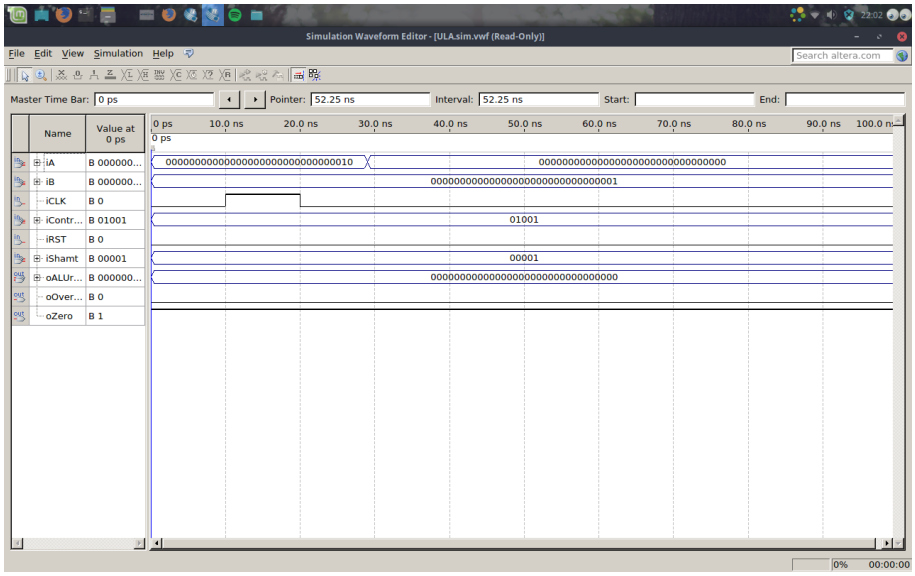

Figure 15. 01001-sra-zero

Operação XOR bit a bit. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

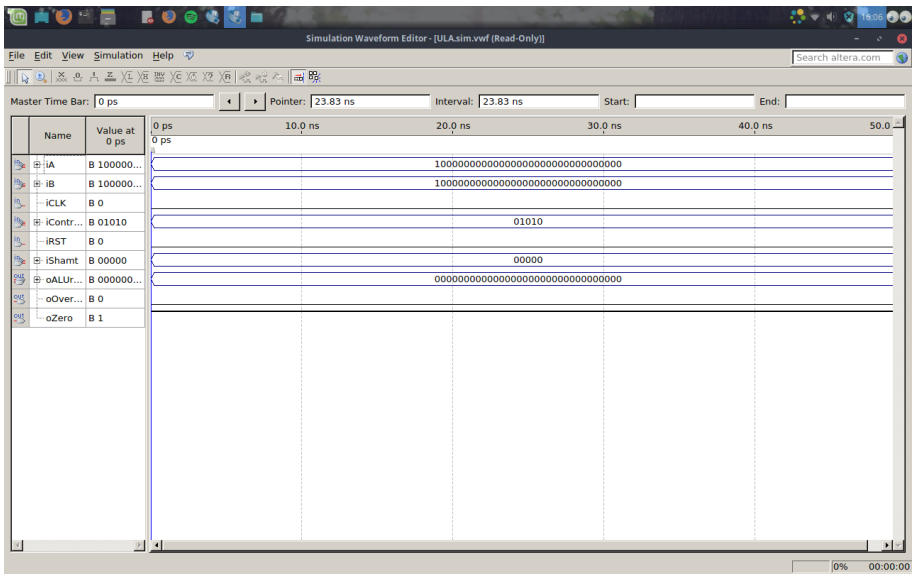

Figure 16. 01010-xor-zero

Operação SLTu seta 1 em oALUresult se iA(sem sinal) for menor que iB(sem sinal). A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

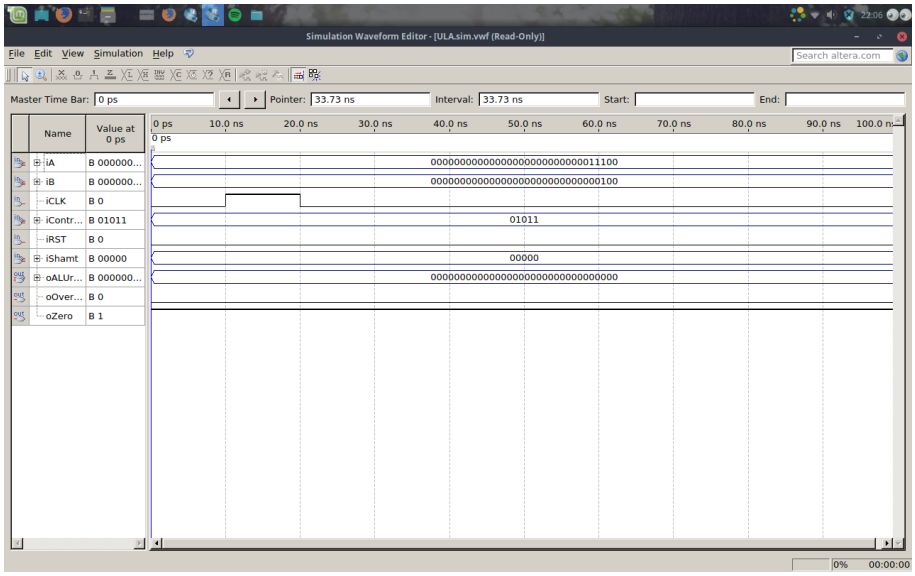

Figure 17. 01011-sltu-zero

Operação NOR bit a bit. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

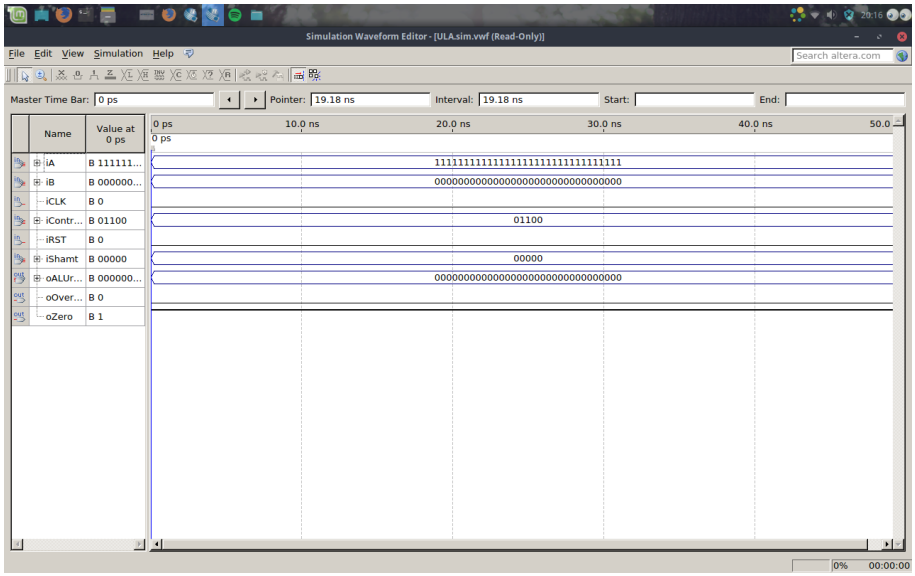

Figure 18. 01100-nor-zero

Operação MULT entre iA e iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

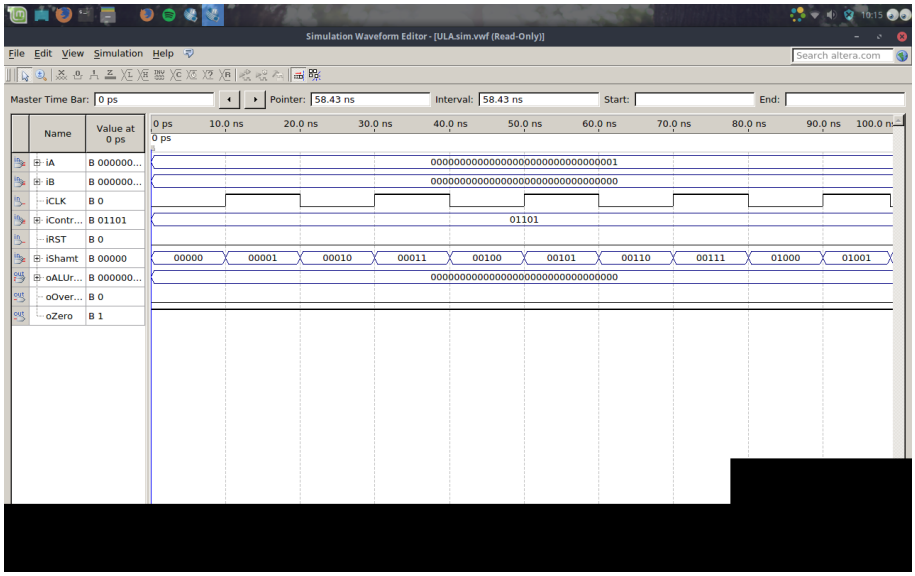

Figure 19. 01101-mult-zero

Operação DIV iA por iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

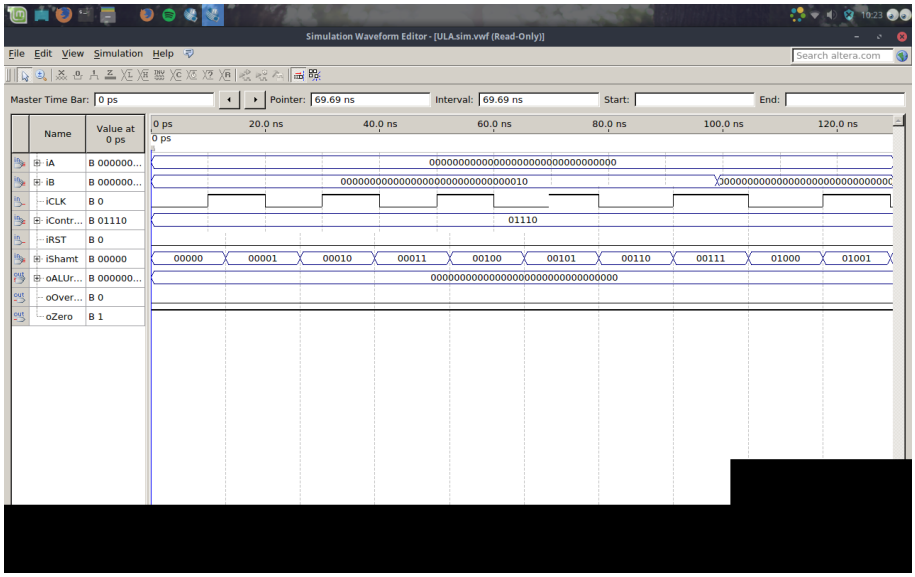

Figure 20. 01110-div-zero

Operação LUI carrega os 16 bits mais significativos do iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

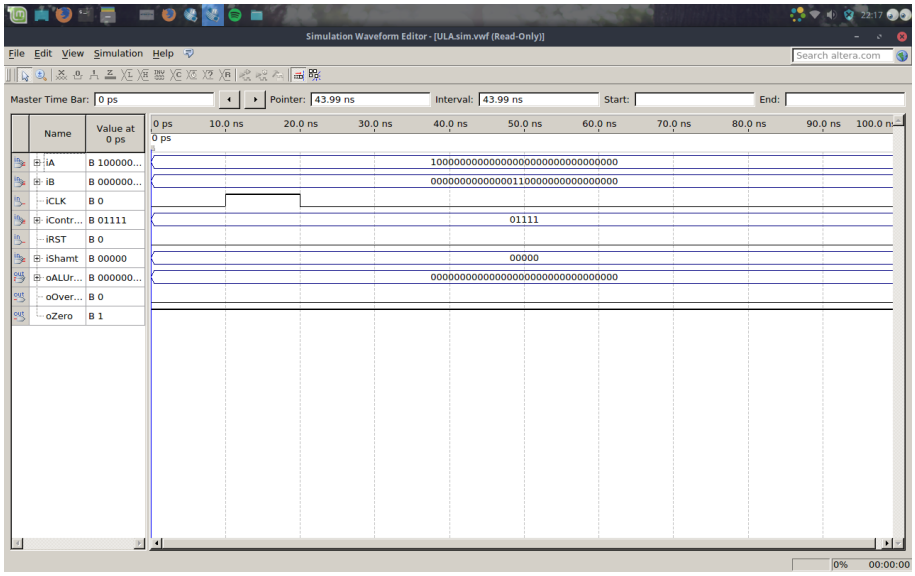

Figure 21. 01111-lui-zero

Operação SLLV shitfita o valor em bits do iA para a esquerda no iB e coloca no oALUresult. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

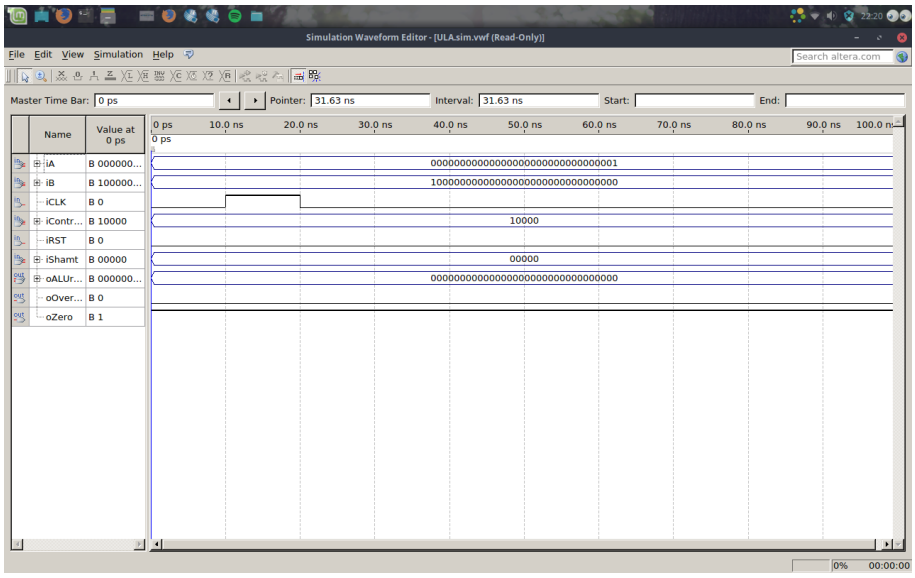

Figure 22. 10000-sllv-zero

Operação SRAV shitfita o valor em bits no iA para a direita no iB e coloca no oALUresult. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

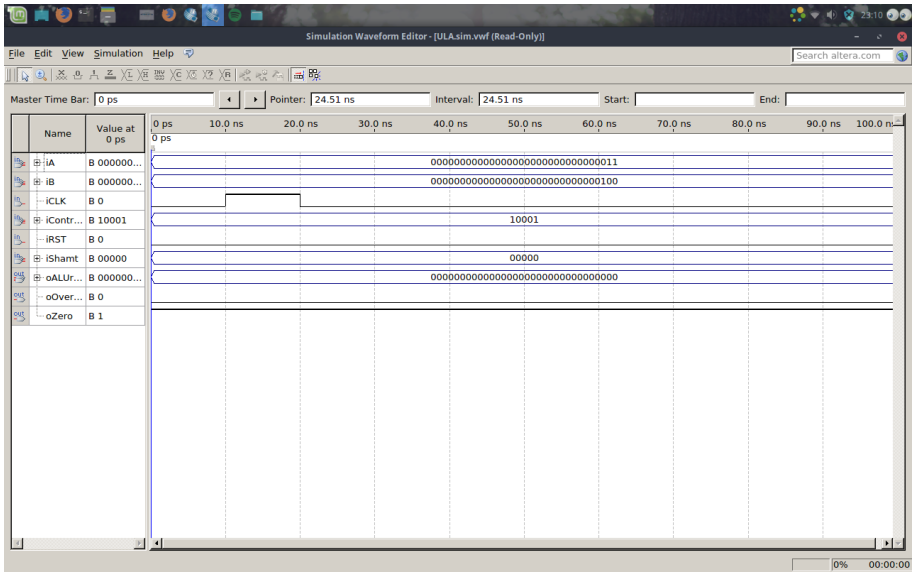

Figure 23. 10001-srav-zero

Operação SRLV shitfita o valor em bits no iA para a direita no iB e coloca no oALUresult. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

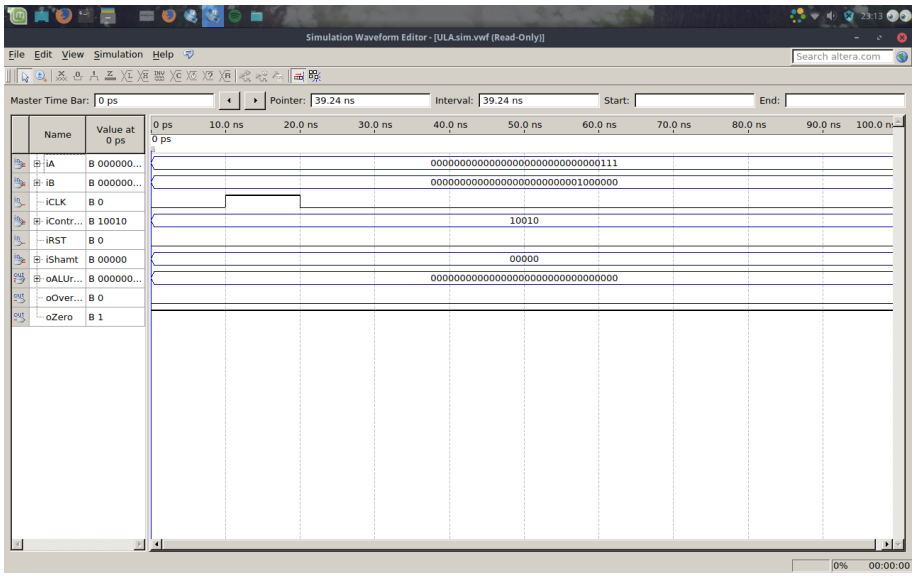

Figure 24. 10010-srlv-zero

Operação MULTU entre iA e iB ambos sem sinal. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

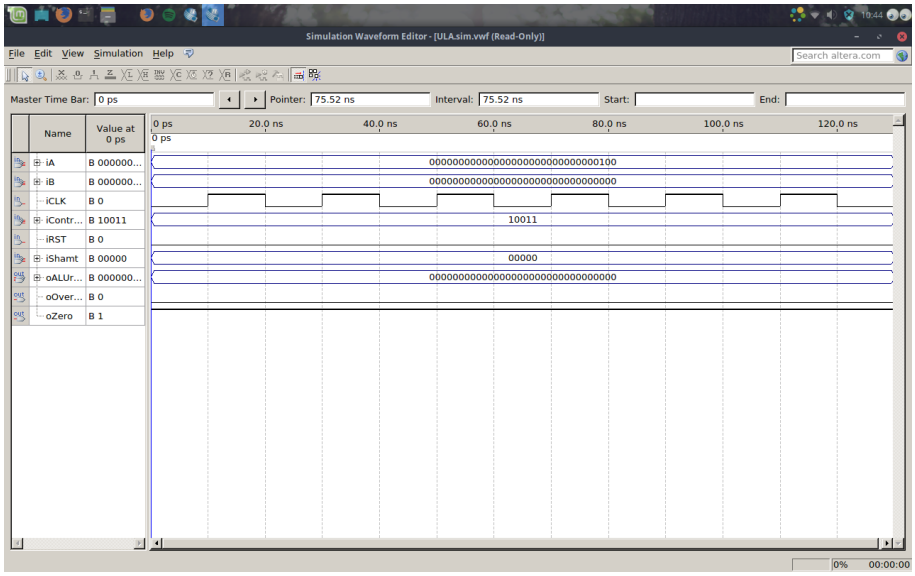

Figure 25. 10011-multu-zero

Operação DIVu iA por iB ambos sem sinal. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

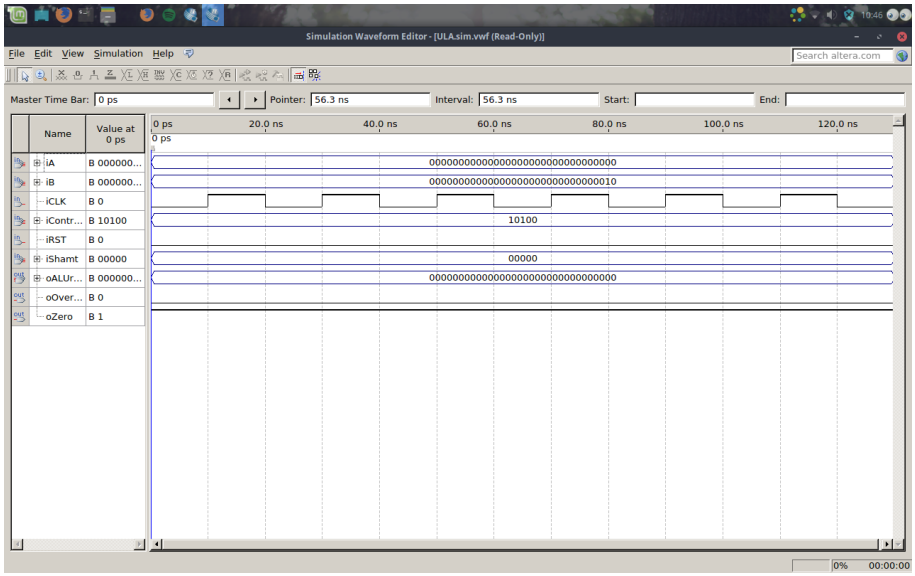

Figure 26. 10100-divu-zero

Operação MTHI move do iA para o HI. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

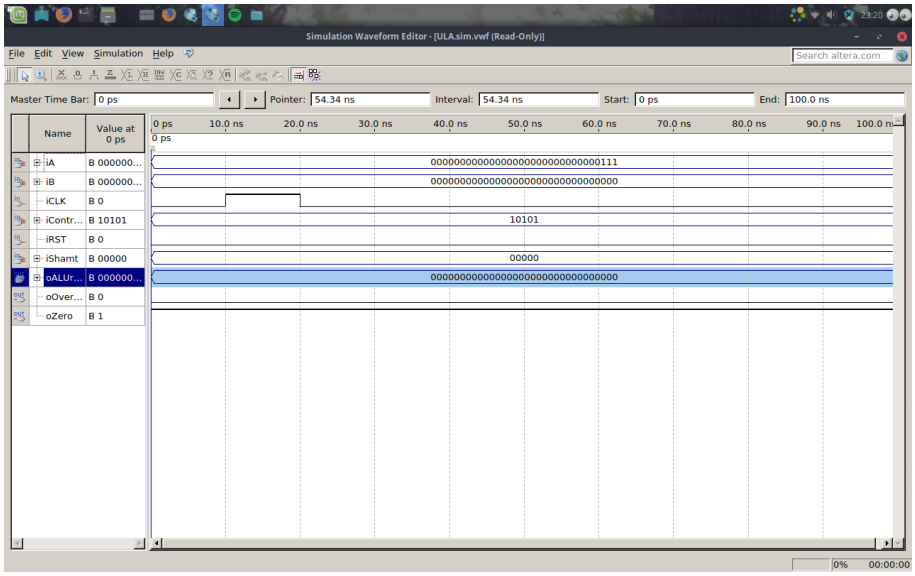

Figure 27. 10101-mthi-zero

Operação MTLO move do iA para o LO. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

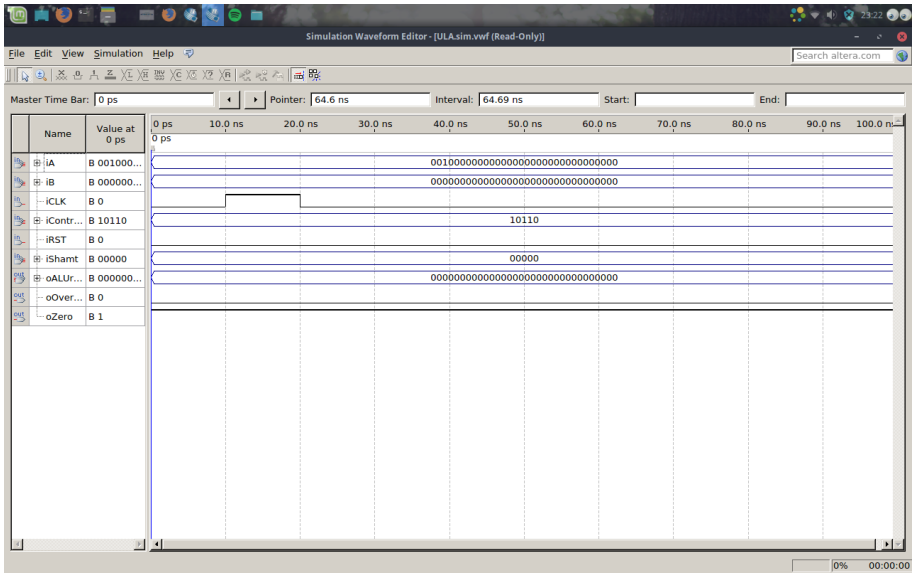

Figure 28. 10110-mtlo-zero

Operação SGT seta 1 em oALUresult se iA for maior que iB. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

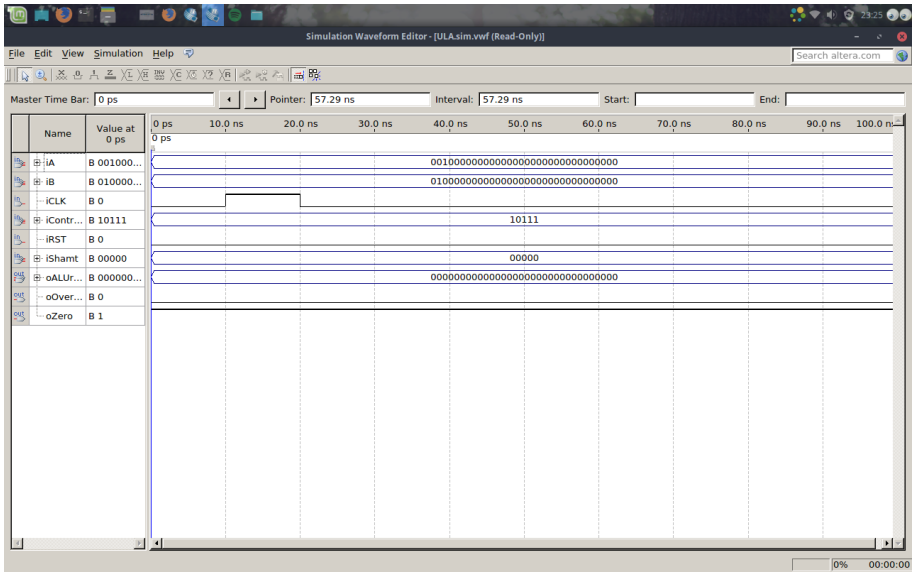

Figure 29. 10111-sgt-zero

Operação MADD multplica iA por iB e soma no HI. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

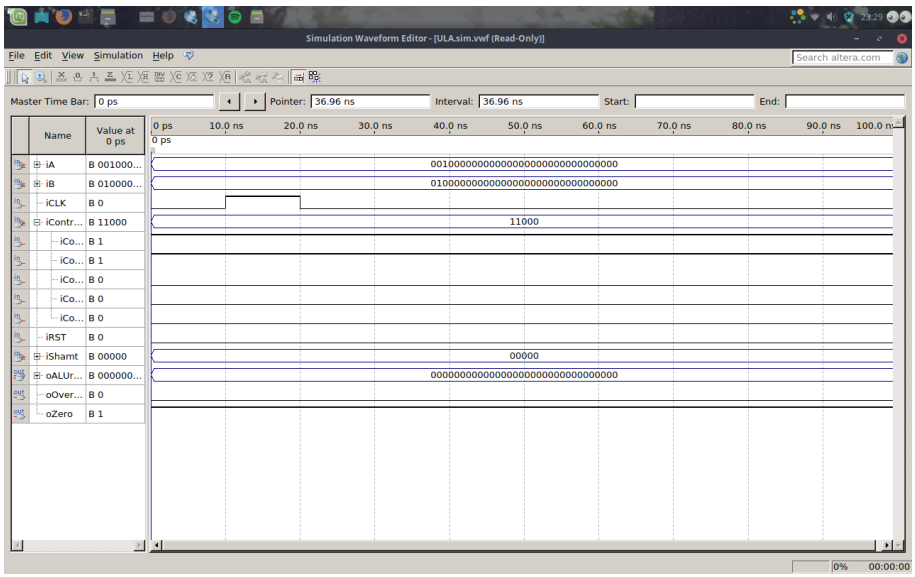

Figure 30. 11000-madd-zero

Operação MADD multplica iA por iB(AMBOS EM SINAL) e soma no HI. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

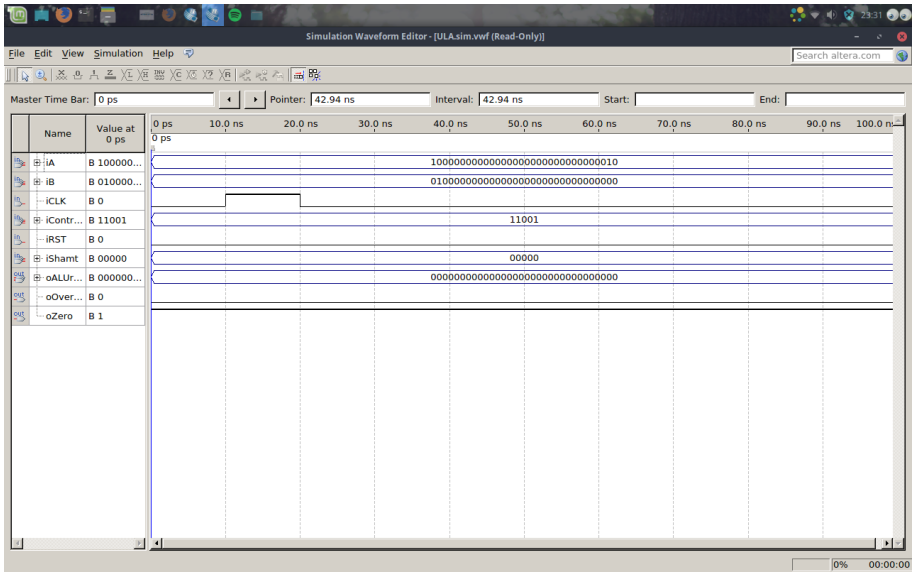

Figure 31. 11001-maddu-zero

Operação MSUB multplica iA por iB e subtrai no HI. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

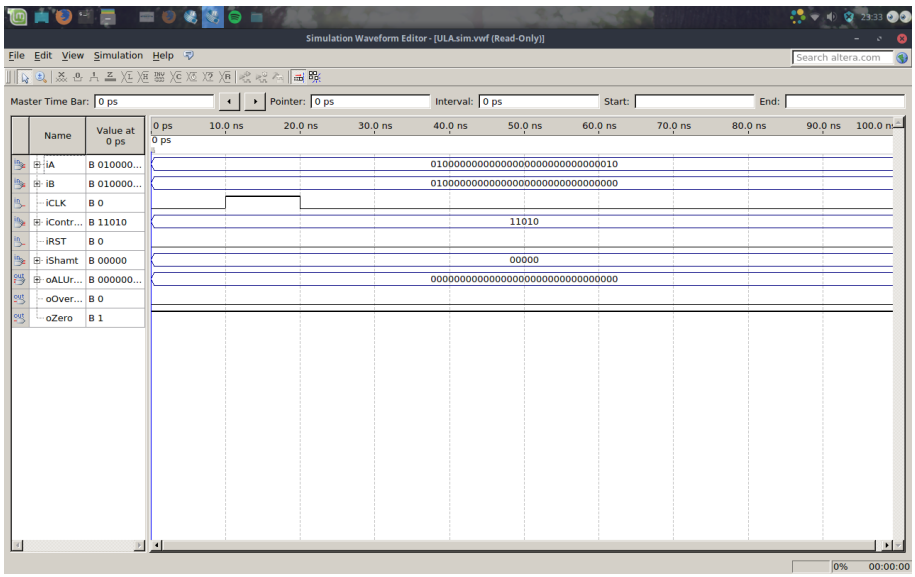

Figure 32. 11010-msub-zero

Operação MSUBu multplica iA por iB(ambos sem sinal) e subtrai no HI. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.

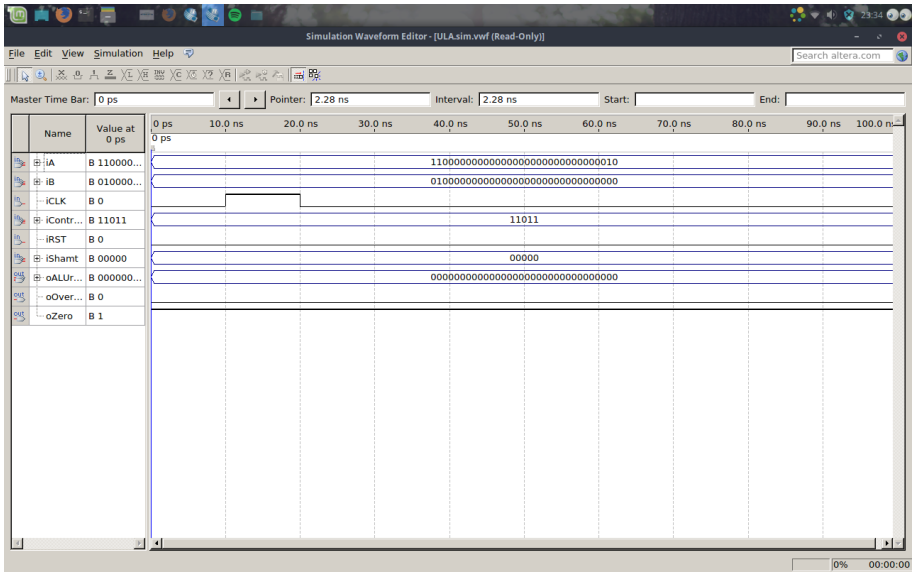

Figure 33. 11011-msubu-zero

3.3. Exercício 3. Unidade Lógica Aritmética fracionária

3.3.1. Operações

Operação ADDS , adição de registradores em precisão simples com overflow. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0. E o indicador de overflow, se a saida for 1 ocorreu, se for 0 não.

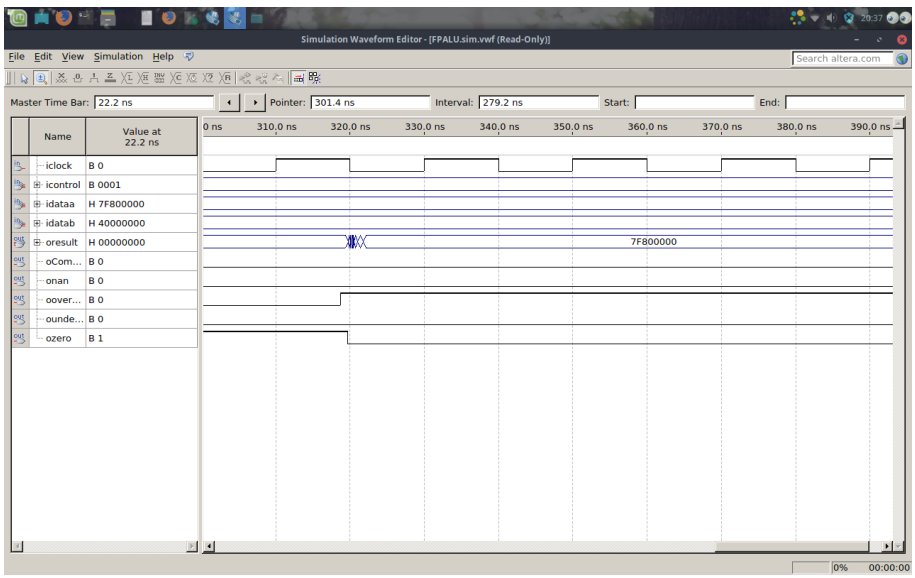

Figure 34. 0001-adds-over-zero

Operação ADDS , adição de registradores em precisão simples. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que

indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0.A mesma logica para o ”not a number”.

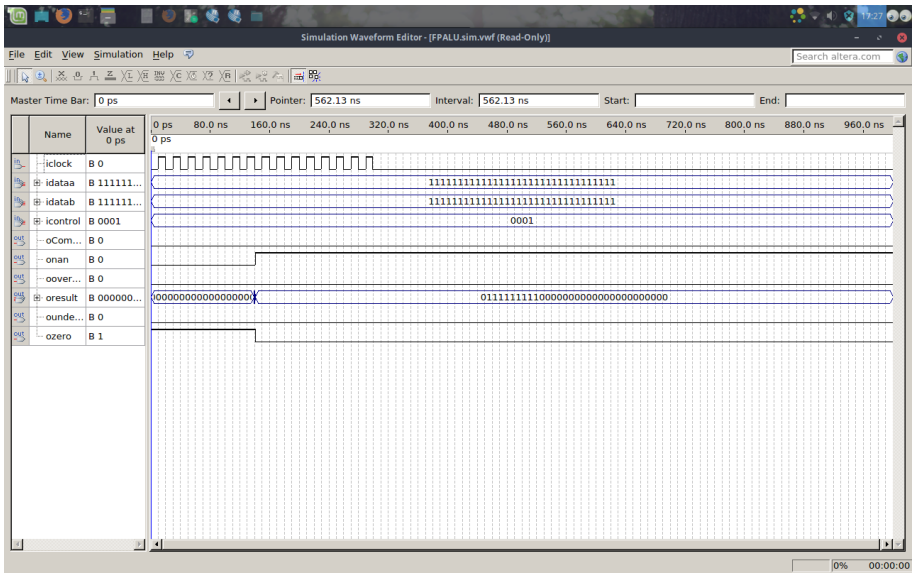

Figure 35. 0001-adds-zero-nan

Operação MULTS , multiplicação de registradores em precisão simples com overflow. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0. E o indicador de overflow, se a saida for 1 ocorreu, se for 0 não.

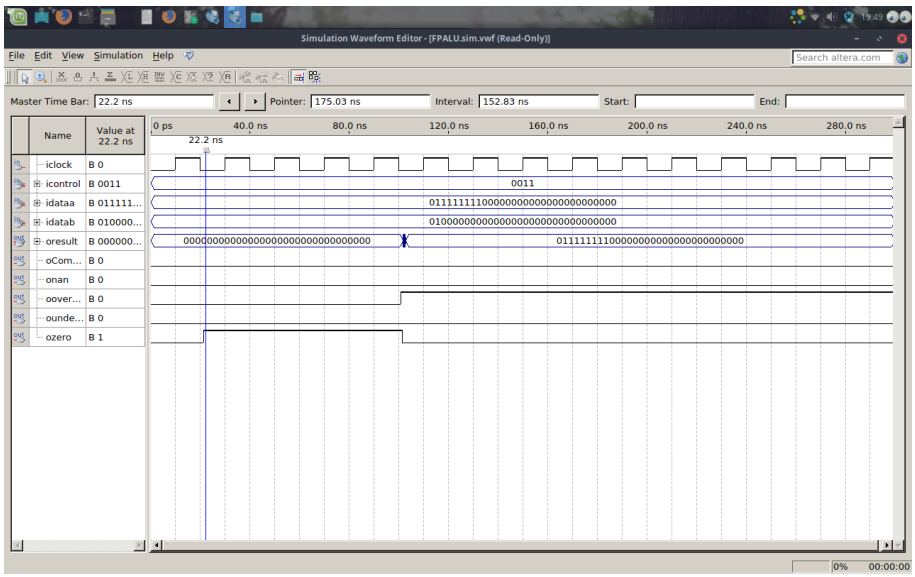

Figure 36. 0011-muls-over-zero

Operação DIVS , divisão de registradores em precisão simples com overflow. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é

0. E o indicador de overflow, se a saida for 1 ocorreu, se for 0 não. A mesma logica para o ”not a number”.

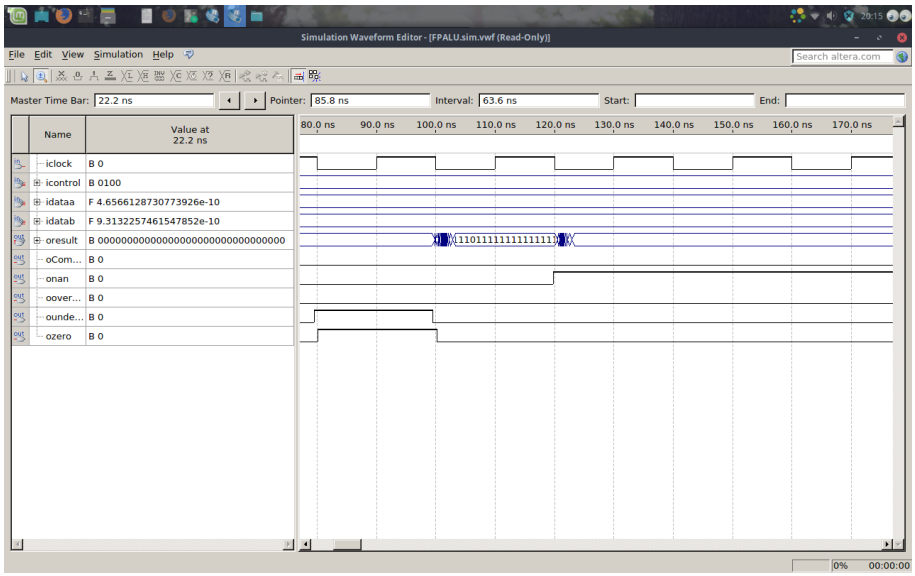

Figure 37. 0100-divs-nan-under-zero

Operação SQRT , raiz quadrada do registrador iA em precisão simples com overflow. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0. A mesma logica para o ”not a number”.

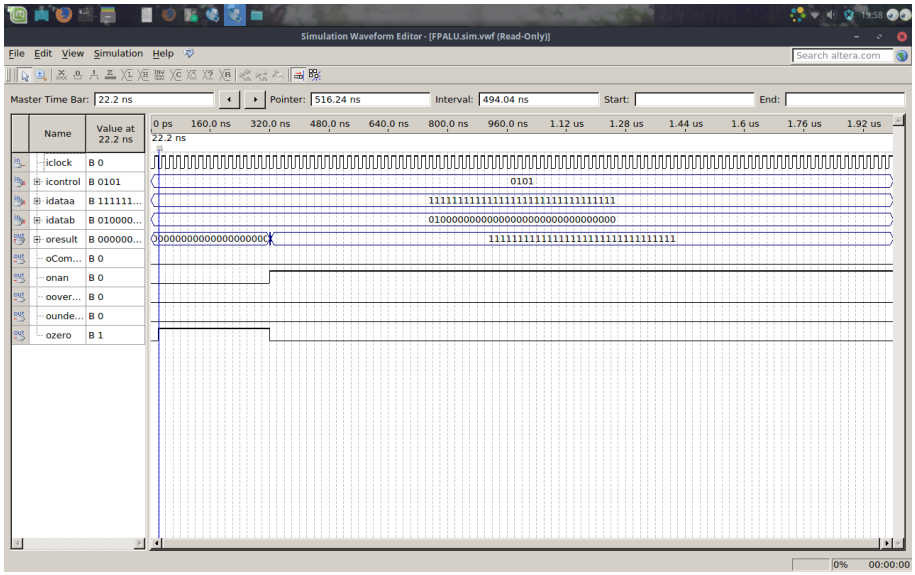

Figure 38. 0101-sqrt-nan-zero

Operação CVTWS , converte de word para precisão simples. A imagem a baixo demonstra um dos valores singulares possíveis, como o registrador oZero de um bit que

indinca se todos os bits do oALUresult forem 0, se sim a saída é 1, se não é 0. A mesma logica para o ”not a number”.

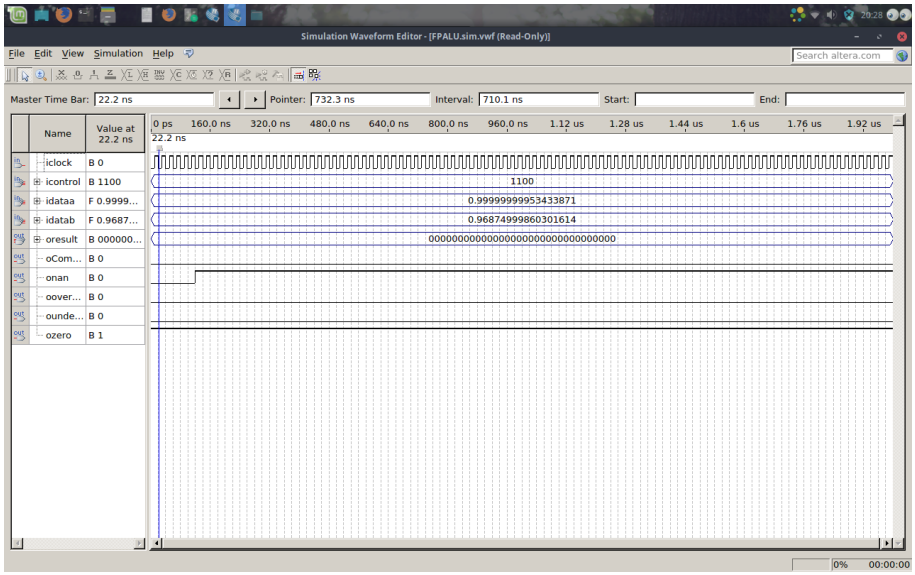

Figure 39. 1100-cvtws-nan-zero

3.3.2. Requisitos físicos

Para a ULA de inteiros foi levantado os requisitos físicos de cada operação e da ULA total como podemos ver na Tabela 1 e Tabela 2. Todos estes dados foram encontrados utilizando o seguinte procedimento:

- Foi aberto o projeto da *ULA* no *Quartus II 64-Bit*;
- No arquivo *ALU.v* para testar a ULA completa foi preciso comentar as linhas ”*wire [4:0] iControlSignal*” e ”*assign iControlSignal=OPMSUB*” e descomentar a linha ”*input [4:0] iControlSignal*”. No arquivo *ALU.v* para testar cada operação foi preciso descomentar as linhas ”*wire [4:0] iControlSignal*” e ”*assign iControlSignal=OPMSUB*” e comentar a linha ”*input [4:0] iControlSignal*”. A troca de operação avaliada foi feita substituindo o nome da operação na variável *iControlSignal* (e.g. assign iControlSignal=OPSLL);
- Ao trocar a operação desejada foi compilado o projeto;
- Com a nova aba (*Compilation Report - ULA*) aberta, no menu *Flow Summary* foi possível achar informações da quantidade total de elementos lógicos usado naquela operação;
- No menu *TimeQuest Timing Analyzer* > *Multicorner Datasheet Report Summary* foram encontrados valores dos maiores / menores tempos de atraso para concluir a operação. Estes tempos são medidos desde o ato de inserir o dado na entrada (*iA* e/ou *iB*) e resultar em algo na saída (*oALUresult*). Alguns resultados assíncronos eram aparentes na aba *RR* (medição ao subir a borda inicial até a subida da borda final), outros na *RF* (medição ao subir a borda inicial até a descida da borda final) [**?**]. Para operações síncronas era possível captar os resultados na aba *Rise*;

  - Para operações puramente assíncronas o maior tempo foi encontrado no menu *Propagation Delay*. Para operações também síncronas tiveram estes dados aparentes no menu *Clock to Output Times*;

  - Para operações puramente assíncronas o menor tempo foi encontrado no menu *Minimum Propagation Delay*. Para operações também síncronas tiveram estes dados aparentes no menu *Minimum Clock to Output Times*.

- A frequência máxima de *clock* utilizável foi gerada a partir do cálculo $F_{MAX} = 1/T$, sendo T o maior tempo de atraso da operação. Esse T tem que ser o pior caso de tempo ocorrido pois precisa ser suficiente para concluir toda a operação em qualquer caso.

	Elementos lógicos	**Menor atraso (ns)**	**Maior atraso (ns)**	**Frequência máxima de *clock* utilizável (MHz)**
ULA	6686	4,788	26,648	37,526
OPAND	43	5,495	9,810	101,937
OPOR	43	5,490	9,811	101,926
OPADD	44	5,081	14,134	70,751
OPMFHI	0	0	0	0
OPSLL	170	6,000	15,048	66,454
OPMFLO	0	0	0	0
OPSUB	44	5,119	13,909	71,896
OPSLT	32	5,341	13,470	74,239
OPSGT	32	5,341	13,470	74,239
OPSRL	170	6,065	16,137	61,969
OPSRA	174	4,908	15,736	63,549
OPXOR	43	4,802	8,511	117,495
OPSLTU	32	5,341	13,470	74,239
OPNOR	43	4,822	9,811	101,926
OPLUI	5	4,546	8,330	120,048
OPSLLV	170	5,397	15,048	66,454
OPSRAV	174	4,908	15,736	63,549
OPSRLV	170	5,445	16,137	61,969

Table 1. Requisitos físicos da *ULA* total e de cada operação. Informações das operações assíncronas.

	Elementos lógicos	Menor atraso (ns)	Maior atraso (ns)	Frequência máxima de *clock* utilizável (MHz)
ULA	6686	6,804	14,672	68,157
OPMULT	53	3,914	8,885	112,549
OPDIV	1266	4,051	9,446	105,865
OPMULTU	40	5,602	22,197	45.051
OPDIVU	1127	4,419	11,187	84.624
OPMTHI	11	4,743	10,741	93.101
OPMTLO	11	4,743	10,741	93.101
OPMADD	40	5,607	21,340	46.860
OPMADDU	40	5,602	22,197	45.051
OPMSUB	72	5,746	23,337	42.850
OPMSUBU	72	6,002	24,519	40.785

Table 2. Requisitos físicos da *ULA* total e de cada operação. Informações das operações síncronas.

3.3.3. Funcionamento

O projeto da *ULA* de inteiros foi sintetizado utilizando a interface *TopDE.v* na placa *DE2-70*.É possível observar todos os experimentos no seguinte canal: Canal do Laboratório de OAC

As funções testadas foram:

Operação	Vídeo	Operação	Vídeo
AND	AND	**OR**	OR
ADD	ADD	**MFHI**	MFHI
SLL	SLL	**MFLO**	MFLO
SUB	SUB	**SLT**	SLT
SRL	SRL	**SRA**	SRA
XOR	XOR	**SLTU**	SLTU
NOR	NOR	**MULT**	MULT
DIV	DIV	**LUI**	LUI
SLLV	SLLV	**SRAV**	SRAV
SRLV	SRLV	**MULTU**	MULTU
DIVU	DIVU	**MTHI**	MTHI
MLTO	MLTO	**SGT**	SGT
MADD	MADD	**MADDU**	MADDU
MSUB	MSUB	**MSUBU**	MSUBU

Table 3. Funcionamento das operações da ULA.

3.4. Exercício 3. Unidade Aritmética de Ponto Flutuante

3.4.1. Operações

3.4.2. Requisitos físicos

Para a *ULA* de ponto flutuante foi levantado os requisitos físicos de cada operação e da *FPULA* total como podemos ver na Tabela 4. Todos estes dados foram encontrados utilizando o seguinte procedimento:

- Foi aberto o projeto da *FPULA* no *Quartus II 64-Bit*;
- No arquivo *FPALU.v* para testar a *FPULA* completa foi preciso comentar as linhas "*wire [3:0] icontrol*" e "*assign icontrol=OPSQRT*" e descomentar a linha "*input [3:0] icontrol*". No arquivo *FPALU.v* para testar cada operação foi preciso descomentar as linhas "*wire [3:0] icontrol*" e "*assign icontrol=OPSQRT*" e comentar a linha "*input [3:0] icontrol*". A troca de operação avaliada foi feita substituindo o nome da operação na variável *icontrol* (e.g. assign icontrol=OPSQRT);
- Ao trocar a operação desejada foi compilado o projeto;
- Com a nova aba (*Compilation Report - FPULA*) aberta, no menu *Flow Summary* foi possível achar informações da quantidade total de elementos lógicos usado naquela operação;
- No menu *TimeQuest Timing Analyzer* $>$ *Multicorner Datasheet Report Summary* $>$ *Clock to Output Times* foram encontrados os períodos de clock máximos da operação avaliada. Os resultados eram aparentes na aba *Rise* e na *FALL* [**?**];
- No menu *TimeQuest Timing Analyzer* $>$ *Multicorner Datasheet Report Summary* $>$ *Mínimum Clock to Output Times* foram encontrados os períodos de clock mínimos da operação avaliada. Os resultados eram aparentes na aba *Rise* e na *FALL* [**?**];
- A frequência máxima de *clock* utilizável foi gerada a partir do cálculo $F_{MAX} = 1/T$, sendo T o período Máximo de clock da operação.

	Elementos lógicos	**T de clk mínimo (ns)**	**T de clk máximo (ns)**	**Frequência máxima de *clock* utilizável (MHz)**
FPULA	2992	6,864	19,449	51.417
OPADDS	824	3,807	8,471	118.050
OPSUBS	830	3,979	8,684	115.154
OPMULS	280	3,808	9,064	110.327
OPDIVS	325	3,498	9,748	102.585
OPSQRT	807	3,749	7,773	128.650
OPABS	0	4,966	8,932	111.957
OPNEG	0	4,522	8,934	111.932
OPCEQ	49	3,432	6,446	155.135
OPCLT	88	3,198	5,963	167.701
OPCLE	88	3,341	6,215	160.901
OPCVTSW	311	3,782	7,542	132.591
OPCVTWS	453	3,984	10,358	96.544

Table 4. Requisitos físicos da *FPULA* total e de cada operação.

3.4.3. Funcionamento

O projeto da *ULA* de pontos flutuantes foi sintetizado utilizando a interface *TopDE.v* na placa *DE2-70*. É possível observar todos os experimentos no seguinte canal: Canal do Laboratório de OAC

As funções testadas foram:

Operação	**Vídeo**	**Operação**	**Vídeo**
ADDS	ADDS	**SUBS**	SUBS
MULS	MULS	**DIVS**	DIVS
SQRT	SQRT	**ABS**	ABS
NEG	NEG	**CEQ**	CEQ
CLT	CLT	**CLE**	CLE
CVTSW	CVTSW	**CVTWS**	CVTWS

Table 5. Funcionamento das operações da ULA.