

Aluno : TARLISON SANDER LIMA BRITO
MATRICULA: 2017013008

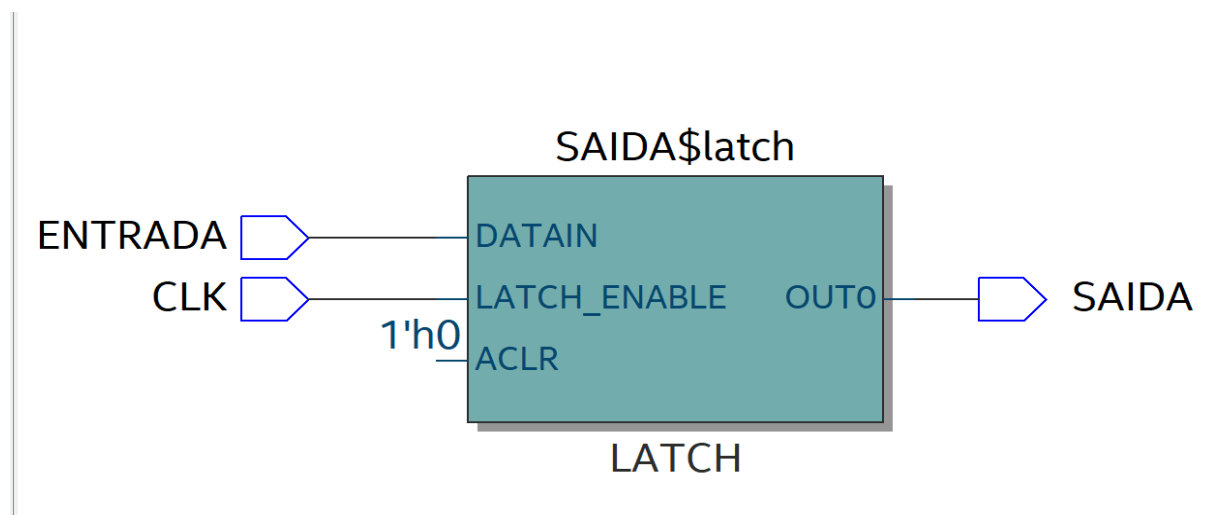
Questão 1 - FlipFlopD e FlipFlopJK

```
FlipFlopD.vhd
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY FlipFlopD IS
5  |
6  |   PORT(
7  |       CLK, ENTRADA : IN STD_LOGIC;
8  |       SAIDA : OUT STD_LOGIC
9  |   );
10 |
11 |   END FlipFlopD;
12 |
13 |   ARCHITECTURE BEHAVIOR OF FlipFlopD IS
14 |   |
15 |   |   BEGIN
16 |   |       SAIDA <= ENTRADA WHEN CLK = '1';
17 |   |
18 |   |   END BEHAVIOR;
19 |
20 |
```

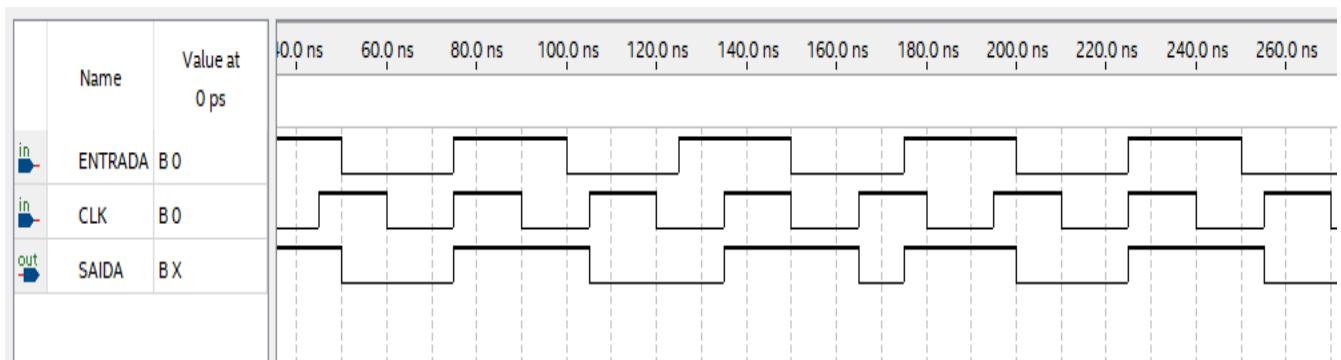
A porta ENTRADA recebe o bit e SAIDA a saída desse bit o valor dentrada dependendo do CLK se estiver ativo ou não.

O FlipFlop do tipo D transfere o bit da entrada para a saída somente quando há um pulso de clock então o WHEN CLK = '1'.

RTL Viewer:



WaveForm:



Como foi dito o dado de entrada é salvo para saída apenas se houver o CLK igual a '1' quando a borda está baixa o valor não será 'salvo' podemos observar isso quanto tempo borda baixa na ENTRADA e borda alta na CLK e temos a saída borda baixa assim como da entrada.

FlipFlopJK -----

Código do programa:

```

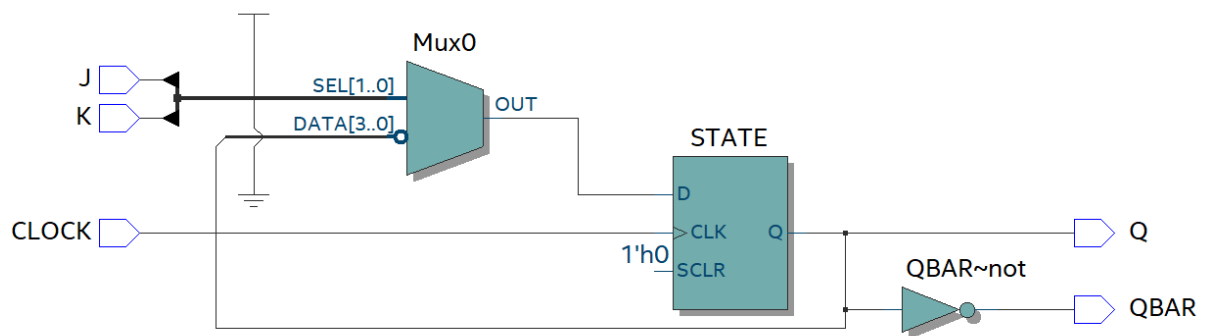
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY FlipFlopJK IS
5
6  PORT ( CLOCK: IN STD_LOGIC;
7        J, K: IN STD_LOGIC;
8        Q, QBAR: OUT STD_LOGIC
9        );
10
11  END FlipFlopJK;
12
13  ARCHITECTURE BEHV OF FlipFlopJK IS
14
15      SIGNAL STATE: STD_LOGIC;
16      SIGNAL INPUT: STD_LOGIC_VECTOR(1 DOWNTO 0);
17
18  BEGIN
19
20      INPUT <= J & K;
21
22      PROCESS(CLOCK) IS
23      BEGIN
24          IF (RISING_EDGE(CLOCK)) THEN
25
26              CASE (INPUT) IS
27                  WHEN "11" => STATE <= NOT (STATE);
28                  WHEN "10" => STATE <= '1';
29                  WHEN "01" => STATE <= '0';
30                  WHEN OTHERS => NULL;
31              END CASE;
32          END IF;
33
34      END PROCESS;
35
36      Q <= STATE;
37      QBAR <= NOT STATE;
38
39  END BEHV;
40

```

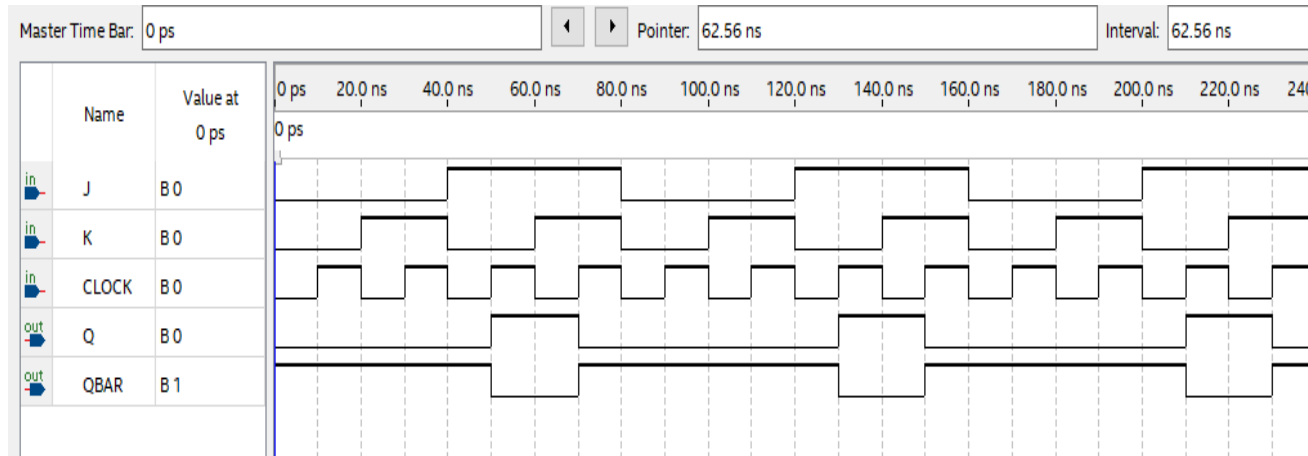
Temos o J e o K como entradas para o FlipFlop do tipo JK que serão analisados juntos e dependendo do valor que possuem a saída será Q ou QBAR. Seguindo a ‘tabela verdade’ desse tipo de registrador(mostrada abaixo) é que é feita a análise.

J	K	Q
1	1	Q'
1	0	1
0	1	0
0	0	Q

RTL Viewer:



WaveForm:



Seguindo a tabela de verdade do registrador podemos observar que está condizente já que por exemplo com J e K iguais a 1 e o CLOCK igual a 1 temos Q igual a 0 e QBAR igual a 1.

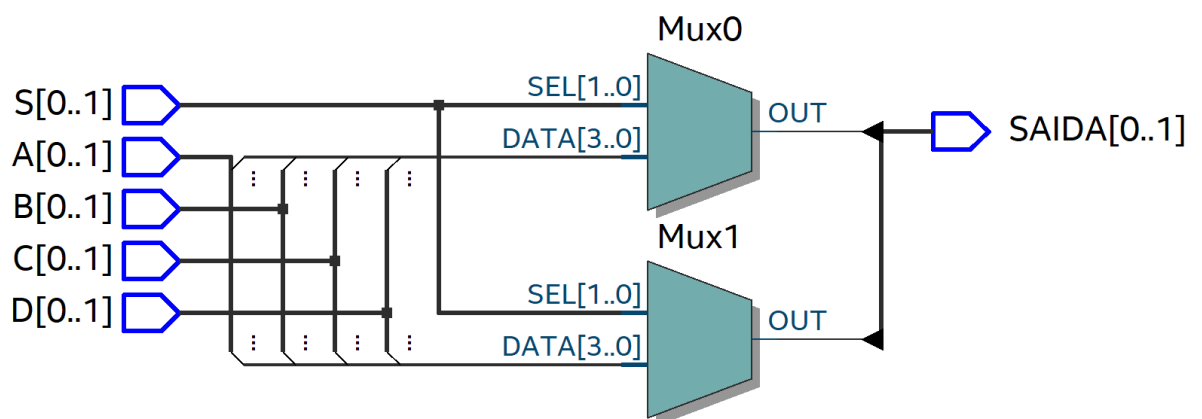
Questão 2 – Multiplexador de 4 Entradas

Código do programa:

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_arith.all;
4
5  ENTITY Multiplexador4x1 IS
6  PORT(
7      SIGNAL A,B,C,D,S: IN BIT_VECTOR(0 TO 1);
8
9      SIGNAL SAIDA: OUT BIT_VECTOR(0 TO 1)
10 );
11
12 END Multiplexador4x1;
13
14 ARCHITECTURE Multiplexador4x1 OF Multiplexador4x1 IS
15 BEGIN
16     PROCESS (A,B,C,D,S)
17     BEGIN
18         CASE S IS
19         WHEN "00" => SAIDA <= A;
20         WHEN "01" => SAIDA <= B;
21         WHEN "10" => SAIDA <= C;
22         WHEN "11" => SAIDA <= D;
23
24         END CASE;
25
26     END PROCESS;
27
28 END Multiplexador4x1;
```

As portas de entrada A,B,C,D são os valores que entraram no multiplexador e a entrada S será o seletor para decidir se será A,B,C, ou D que irá sair em SAIDA. O código funciona com um CASE, usando o S (seletor) para definir que valor sera a SAIDA, então caso o valor de S seja '00' a saída será o valor de A, caso S seja '01' a saída será o valor de B, caso S seja '10' a saída será o valor de C, caso S seja '11' a saída será o valor de D, assim fazendo com que o multiplexador funcione.

RTL Viewer:



WaveForm:

Como podemos ver a WaveForm comprova o funcionamento do multiplexador, quando o S (seletor) assume o valor de '01' a SAIDA tem que ser o valor de B e B tem o

	Name	Value at 0 ps	340.0 ns	350.0 ns	
in	> A	B 01	00	10	1
in	> B	B 01	10	11	00
in	> C	B 10	00	10	11
in	> D	B 11	00	01	00
in	> S	B 01	01	11	00
out	> SAIDA	B 01	10	01	11

valor de '10' e a SAIDA também, assim como na linha ao lado em que S possui o valor de '11' a SAIDA foi o valor de D que é '01'.

Questão 3 - Porta Lógica XOR:

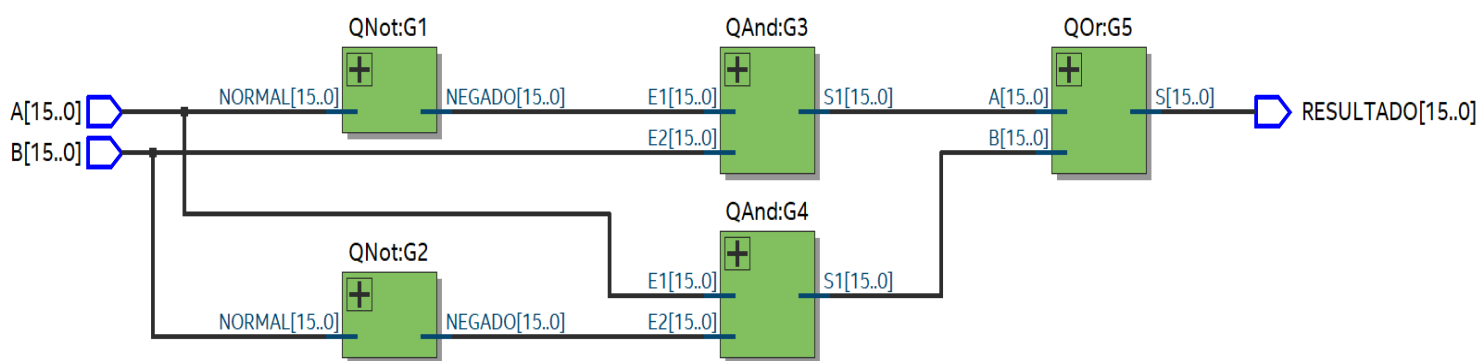
Código do programa:

QXor.vhd

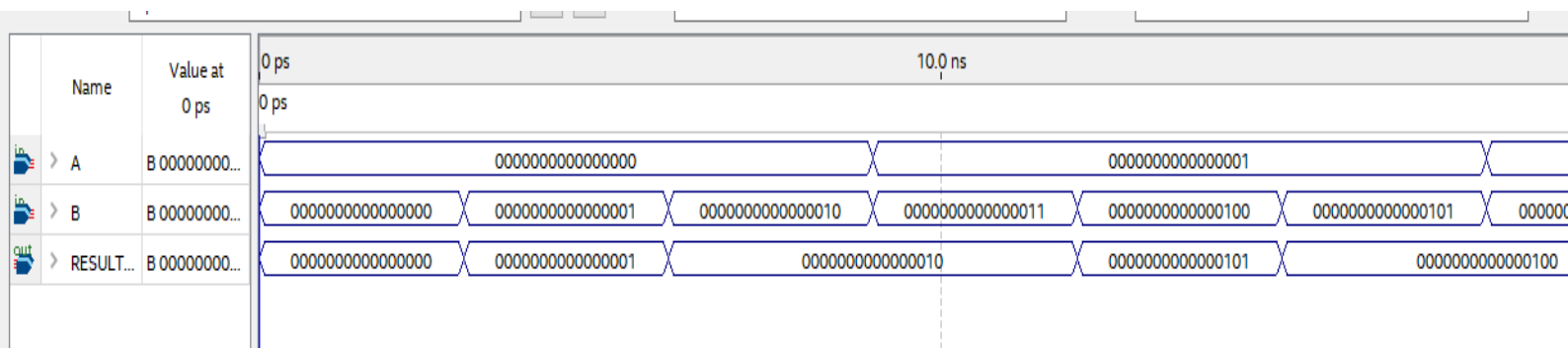
```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY QXor IS
5  PORT (
6      A, B : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
7      RESULTADO : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
8  );
9  END QXor;
10
11 ARCHITECTURE QXor OF QXor IS
12
13     COMPONENT QAnd IS
14     PORT(
15         E1, E2 : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
16         S1 : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
17     END COMPONENT;
18
19     COMPONENT QOr IS
20     PORT(
21
22         A, B : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
23         S : OUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
24     END COMPONENT;
25
26
27
28     COMPONENT QNot IS
29     PORT(
30         NORMAL : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
31         NEGADO : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
32     );
33     END COMPONENT;
34
35     SIGNAL A_NEGADO, B_NEGADO, R1, R2 : STD_LOGIC_VECTOR(15 DOWNTO 0);
36
37 BEGIN
38     G1 : QNot PORT MAP(A, A_NEGADO);
39     G2 : QNot PORT MAP(B, B_NEGADO);
40     G3 : QAnd PORT MAP(A_NEGADO, B, R1);
41     G4 : QAnd PORT MAP(A, B_NEGADO, R2);
42     G5 : QOr PORT MAP(R1, R2, RESULTADO);
43
44 END QXOR;
```

As entradas A e B são os valores em que serão realizadas a operação XOR e a RESULTADO será a saída dessa operação, uso os componentes QAnd, QOr e QNot que serão usados com o port map para efetuar as operações, em que primeiro é feito a negação do valor de entrada A com o componente QNot e esse resultado é mandado para o Signal criado chamado A_NEGADO e então B é negado usando a mesma operação e mandando o seu resultado para o Signal B_NEGADO e assim continuando é feito a operação AND com o A_NEGADO e o valor de B e depois A com o valor de B_NEGADO e isso será salvo, respectivamente, em R1,R2 e esses resultados serão usados para efetuar o OR com o componente QOr e mandando esse resultado para RESULTADO. Tudo isso foi feito para que a operação do XOR acontecesse que é $(\neg A * B) + (A * \neg B)$.

RTL Viewer:



WaveForm:



Como podemos ver o resultado sai como esperado seguindo a lógica de tabela verdade da porta XOR, já que o que é 'igual a saída é 0 e o que é diferente a saída é 1' e podemos ver isso na WaveForm apesar de ser um vetor de 16 bits é comparado bit com bit e assim temos 0 com 0 e 1 com 1 saindo 1 e 1 com 1 e 0 com 0 saindo 0.

Questão 4: Somador + 4:

Código do programa:

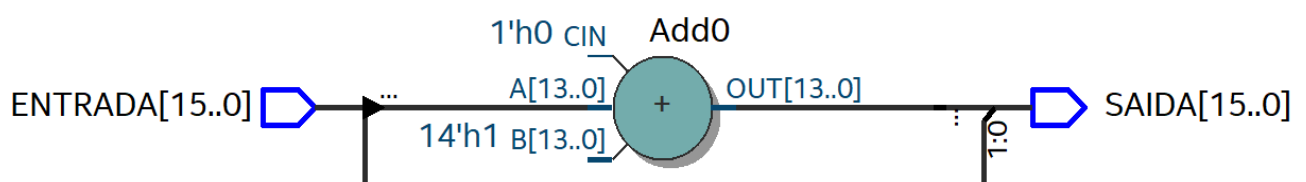
```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_unsigned.all;
4  USE ieee.std_logic_arith.all;
5
6  ENTITY SomadorMais4 IS
7  PORT
8  (
9      ENTRADA : IN STD_LOGIC_VECTOR (15 DOWNT0 0);
10     SAIDA : OUT STD_LOGIC_VECTOR(15 DOWNT0 0)
11 );
12 END SomadorMais4;
13
14 ARCHITECTURE SomadorMais4 OF SomadorMais4 IS
15
16 BEGIN
17     SAIDA <= ENTRADA + "0000000000000100";
18
19
20
21 END SomadorMais4;

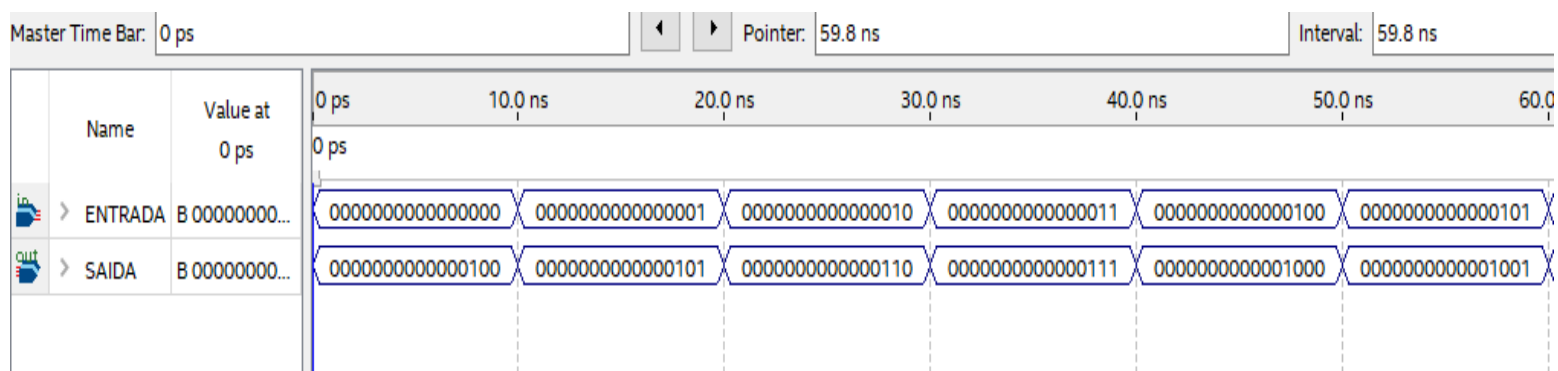
```

No código temos a ENTRADA que será o valor que será somado mais 4 e SAIDA que será o resultado da entrada mais o valor 4. No código é recebido o valor e somado mais 4 (em binário de 16 bits já que a entrada é um vetor de 16 bits) e esse resultado é mandado para a porta SAIDA.

RTL Viewer:



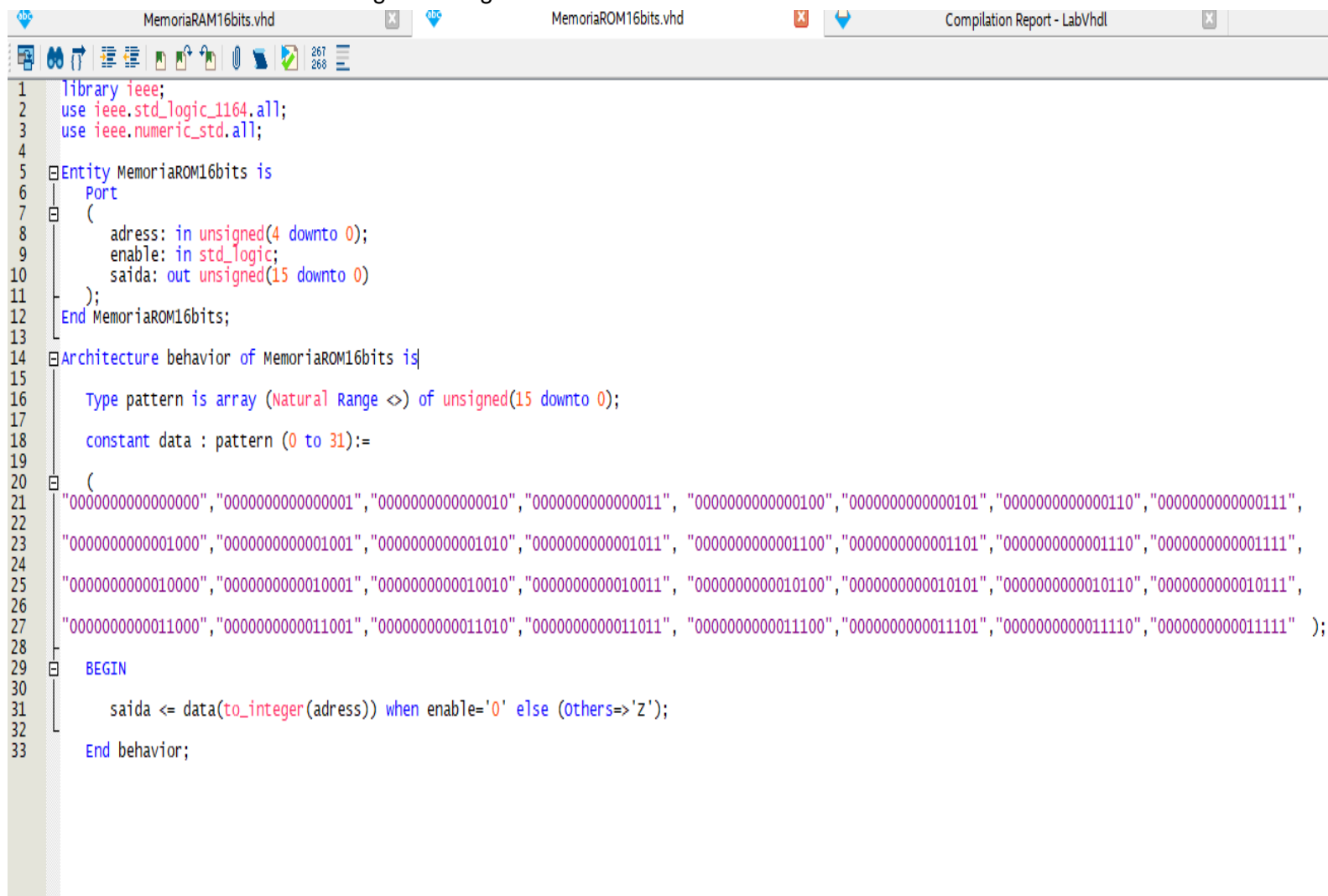
WaveForm:



Como Podemos ver na WaveForm acontece a soma de mais 4 em cada valor em que a Entrada assume, como 0000000000000000 gera a saída 0000000000000100 e 0000000000000001 gera a saída 0000000000000101 e assim sucessivamente.

Questão 5 - Memória ROM

Código do Programa:



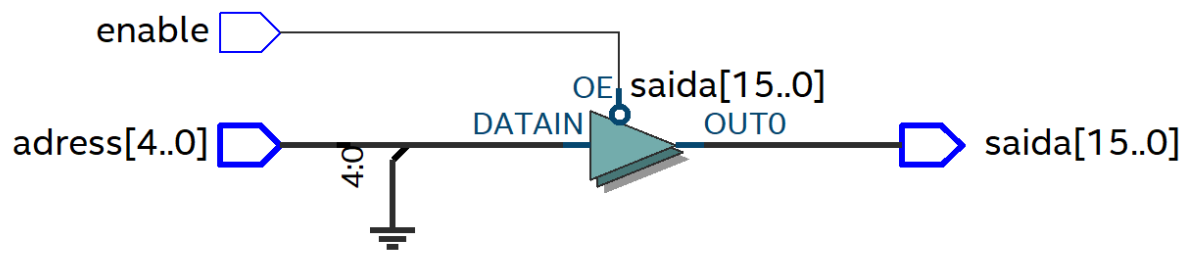
```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  Entity MemoriaROM16bits is
6  Port
7  (
8      address: in unsigned(4 downto 0);
9      enable: in std_logic;
10     saida: out unsigned(15 downto 0)
11 );
12 End MemoriaROM16bits;
13
14 Architecture behavior of MemoriaROM16bits is
15
16     Type pattern is array (Natural Range <>) of unsigned(15 downto 0);
17
18     constant data : pattern (0 to 31) :=
19
20     (
21         "0000000000000000", "0000000000000001", "0000000000000010", "0000000000000011", "0000000000000100", "0000000000000101", "0000000000000110", "0000000000000111",
22         "0000000000001000", "0000000000001001", "0000000000001010", "0000000000001011", "0000000000001100", "0000000000001101", "0000000000001110", "0000000000001111",
23         "0000000000010000", "0000000000010001", "0000000000010010", "0000000000010011", "0000000000010100", "0000000000010101", "0000000000010110", "0000000000010111",
24         "0000000000011000", "0000000000011001", "0000000000011010", "0000000000011011", "0000000000011100", "0000000000011101", "0000000000011110", "0000000000011111" );
25
26 BEGIN
27
28     saida <= data(to_integer(address)) when enable='0' else (others=>'Z');
29
30 End behavior;
31
32
33

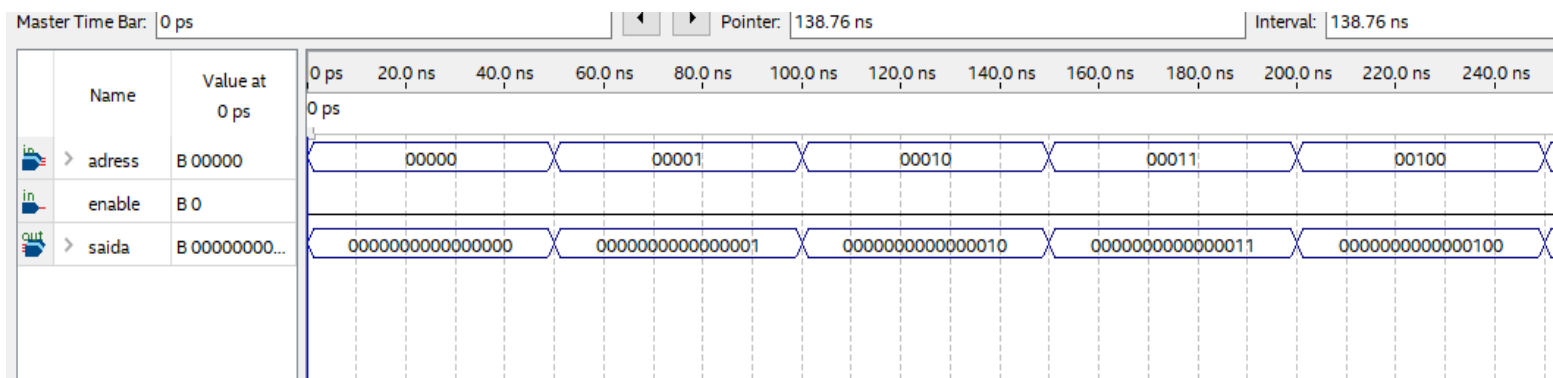
```

A memória ROM é uma memória não volátil em que podemos armazenar os dados e manter salvo mesmo quando sem energia. Temos as portas de entrada que serão o endereço do arquivo que será lido a porta ADDRESS e temos o ENABLE que irá efetuar a leitura quando ativado e a SAIDA que será a saída do dado que foi pedido para ser lido pelo endereço, o código está assim funcionando apenas para um array de 32 espaços mas para $(2^{16}-1)$ seria o mesmo tamanho.

RTL VIEWER:



WaveForm:



Como podemos ver o valor do endereço retorna o valor já , lido, acessado na memória ROM em 00000,00001 e até o fim cada endereço acessa seu respectivo valor na memória.

Questão 6 - Memória RAM

'Código do programa:

```
MemoriaRAM16bits.vhd

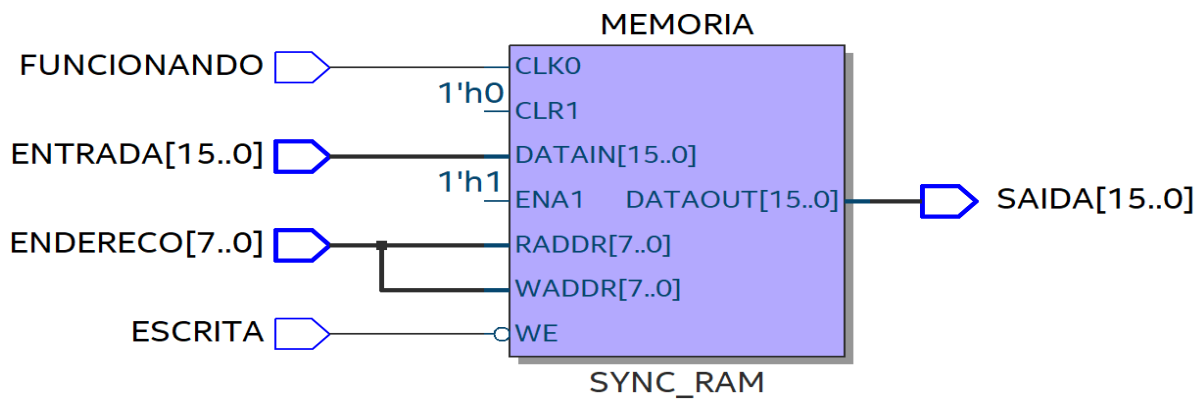
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY MemoriaRAM16bits IS
6  PORT
7  (
8    ENTRADA : IN UNSIGNED (15 DOWNTO 0);
9    SAIDA : OUT UNSIGNED (15 DOWNTO 0);
10   ENDERECO : IN UNSIGNED (7 DOWNTO 0);
11   ESCRITA, FUNCIONANDO : IN STD_LOGIC);
12 END MemoriaRAM16bits;
13
14 ARCHITECTURE BEHAVIOR OF MemoriaRAM16bits IS
15     TYPE ARRANJO IS ARRAY (0 TO 65535) OF UNSIGNED (15 DOWNTO 0);
16     SIGNAL MEMORIA:ARRANJO;
17
18 BEGIN
19     PROCESS(FUNCIONANDO, ENDERECO)
20     BEGIN
21
22         IF RISING_EDGE(FUNCIONANDO) THEN
23
24             IF ESCRITA = '0' THEN MEMORIA(TO_INTEGER(ENDERECO)) <= ENTRADA;
25
26             END IF;
27
28             END IF;
29
30             END PROCESS;
31
32     SAIDA <= MEMORIA(TO_INTEGER(ENDERECO));
33
34 END BEHAVIOR;
35
```

A memória RAM é um hardware de armazenamento , ela é volátil, quando possui energia ela armazena, uma vez que a energia cessa, ela elimina os dados ali presentes, uma das funções é quando usada no computador para guardar os dados das aplicações abertas no sistema operacional como o navegador, o word, etc. Neste código a memória RAM possui 65536((2¹⁶)-1) espaços de memória por ser de 16 bits, quando ela for ativada para leitura e escrita a saída será alterada, pois quando se escrever não vai ter saída.

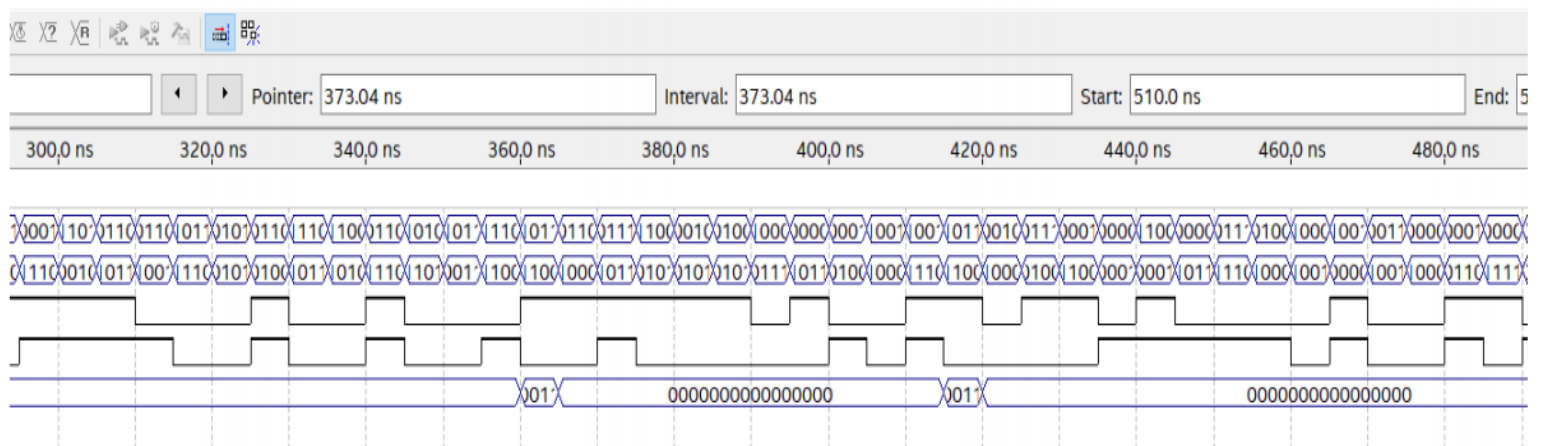
As portas de ENDEREÇO, ENTRADA são usadas para indicar o endereço do dado na memória e a entrada é o valor que será colocado na memória.

As portas de escrita e funcionando são flags para mostrar que está funcionando a escrita na memória RAM. O vetor da memória é criado e se a flag da escrita estiver funcionando ou seja = 1 dentro do rising_edge da flag funcionando então a MEMÓRIA na posição dada pelo ENDEREÇO recebe o dado de ENTRADA e no fim o dado de SAIDA recebe o valor que está na MEMORIA na posição do ENDEREÇO.

RTL Viewer:



WaveForm:



Como na waveform representa só vai ter saída quando se for fazer leitura de arquivos mostrando isso temos a ENTRADA :0000000000000000, LEITURA = 1, ESCRITA = 0, SAIDA = 0000011100000000

Questão 7 Banco de Registradores 16 bits:

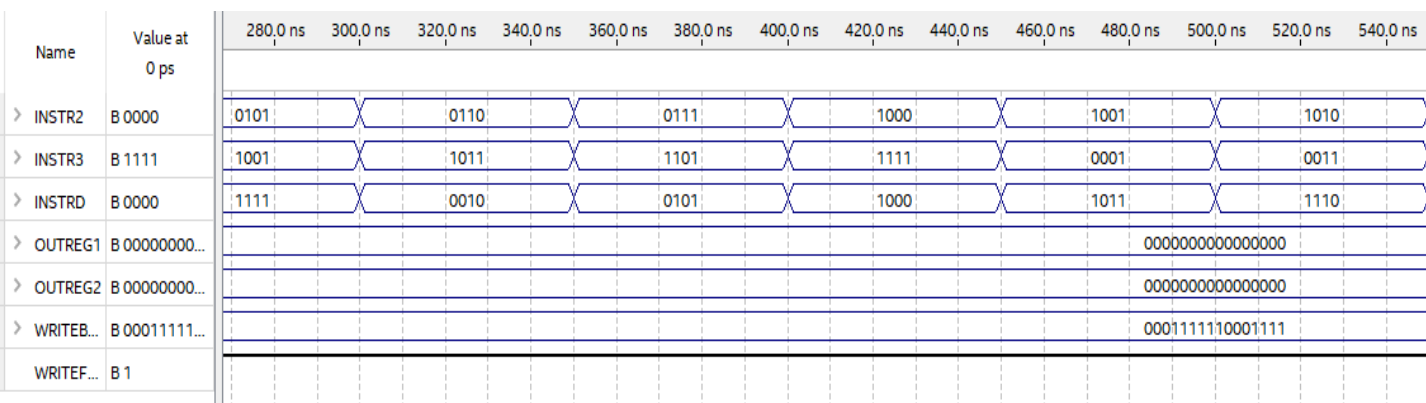
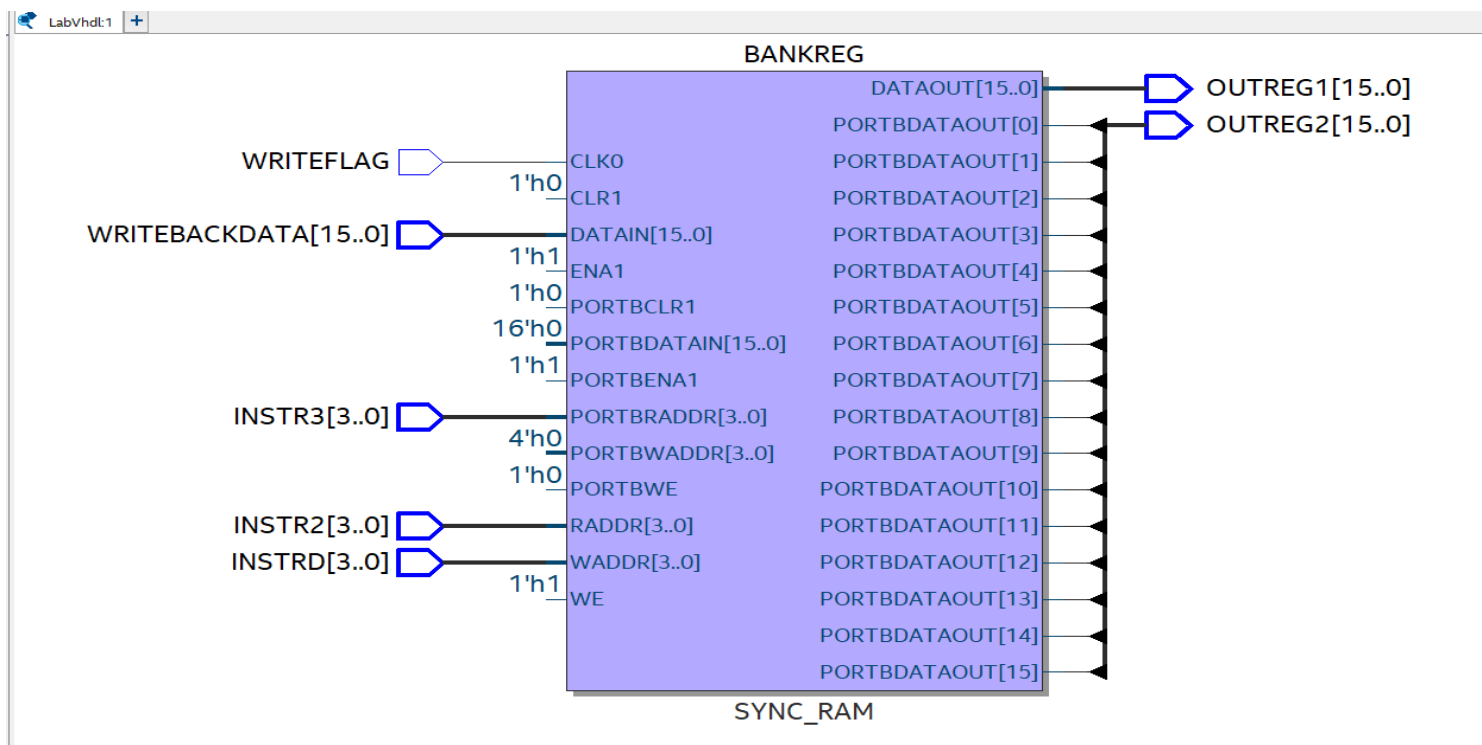
Código do programa:

banco
de
registra
dores
recebe
3
endere
ços
indican
do o
endere
ço dos
3
registra
dores
que
serão
usado,
e as
portas
para
isso
são

```
BancodeRegistadores.vhd
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4  use IEEE.NUMERIC_STD.ALL;
5
6  ENTITY BancodeRegistadores IS
7  PORT(
8      INSTR2, INSTR3, INSTRD : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
9      WRITEFLAG: IN STD_LOGIC;
10     WRITEBACKDATA: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
11     OUTREG1, OUTREG2 : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
12 );
13 END BancodeRegistadores;
14
15 ARCHITECTURE BEHAVIOR OF BancodeRegistadores IS
16
17     TYPE BANK IS ARRAY(0 TO 15) OF STD_LOGIC_VECTOR(15 DOWNTO 0);
18     SIGNAL BANKREG : BANK;
19
20 BEGIN
21     OUTREG1 <= BANKREG( TO_INTEGER(UNSIGNED(INSTR2)));
22     OUTREG2 <= BANKREG( TO_INTEGER(UNSIGNED(INSTR3)));
23     PROCESS(WRITEFLAG)
24     BEGIN
25         IF WRITEFLAG = '1' THEN
26             BANKREG(TO_INTEGER( UNSIGNED(INSTRD))) <= WRITEBACKDATA;
27         END IF;
28     END PROCESS;
29
30 END BEHAVIOR;
```

INSTR2, INSTR3 E INSTRD que irão ser esses endereços. a WRITEFLAG será a flag ativada pela unidade de controle para dizer se é preciso escrever no registrador de destino ou não, WRITEBACKDATA será os dados que vem da memória de dados para escrever no registrador de destino e OUTREG1 E OUTREG2 serão os valores dos registradores que sairão para a unidade de controle que serão identificados pelo INSTR2 e INSTR3. Seguindo a ordem do código primeiro temos o BANK que será onde estará os registradores e o Signal BANK que será usado para acessar o BANK (uma 'instaciação') os valores OUTREG1 e OUTREG2 recebem o os valores respectivos do INSTR2 e INSTR3 acessando o vetor do BANKREG com esses endereços, depois é feito uma análise se a flag WRITEFLAG está ativada para caso esteja ativada o dado que veio no WRITEBACK é mandado para onde está o INSTRD no vetor BANK salvando assim o valor nele.

RTL Viewer:



WaveForm:

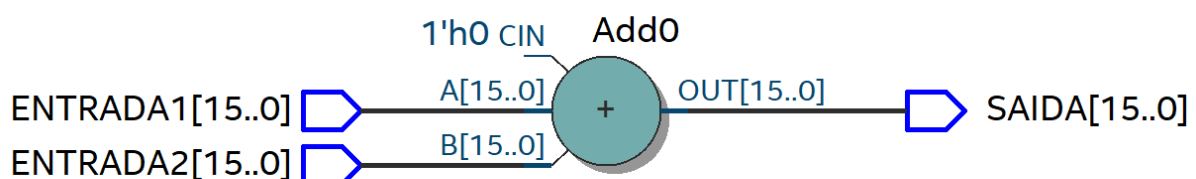
Como podemos ver na WaveForm o retorno dos dois registradores é 0000000000 isso acontece porque ainda não possuem valores dentro dos registradores (o que é certo pois ainda não foi setado nenhum valor para eles e o retorno tem que ser realmente 'vazio').

Questão 8 - Somador de 16 bits

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.numeric_std.all;
4  USE ieee.std_logic_unsigned.all;
5
6  ENTITY Somadorde16bits IS
7  PORT(
8      ENTRADA1, ENTRADA2 : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
9      SAIDA : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
10 );
11 END Somadorde16bits;
12
13 ARCHITECTURE BEHAVIOR OF Somadorde16bits IS
14 BEGIN
15     SAIDA <= ENTRADA1 + ENTRADA2;
16 END ARCHITECTURE;
```

O somador de 16 bits funciona para somar dois valores de 16 bits na ULA. O código funciona com duas portas de entrada que são ENTRADA1, ENTRADA2 e SAIDA (porta de saída) é simplesmente feito uma soma com ENTRADA1 + ENTRADA2 e esse resultado é recebido por SAIDA.

RTL Viewer:



WaveForm:

Como está sendo mostrado na WaveForm os dois valores de ENTRADA1 e ENTRADA2 são somados e passados para SAIDA.

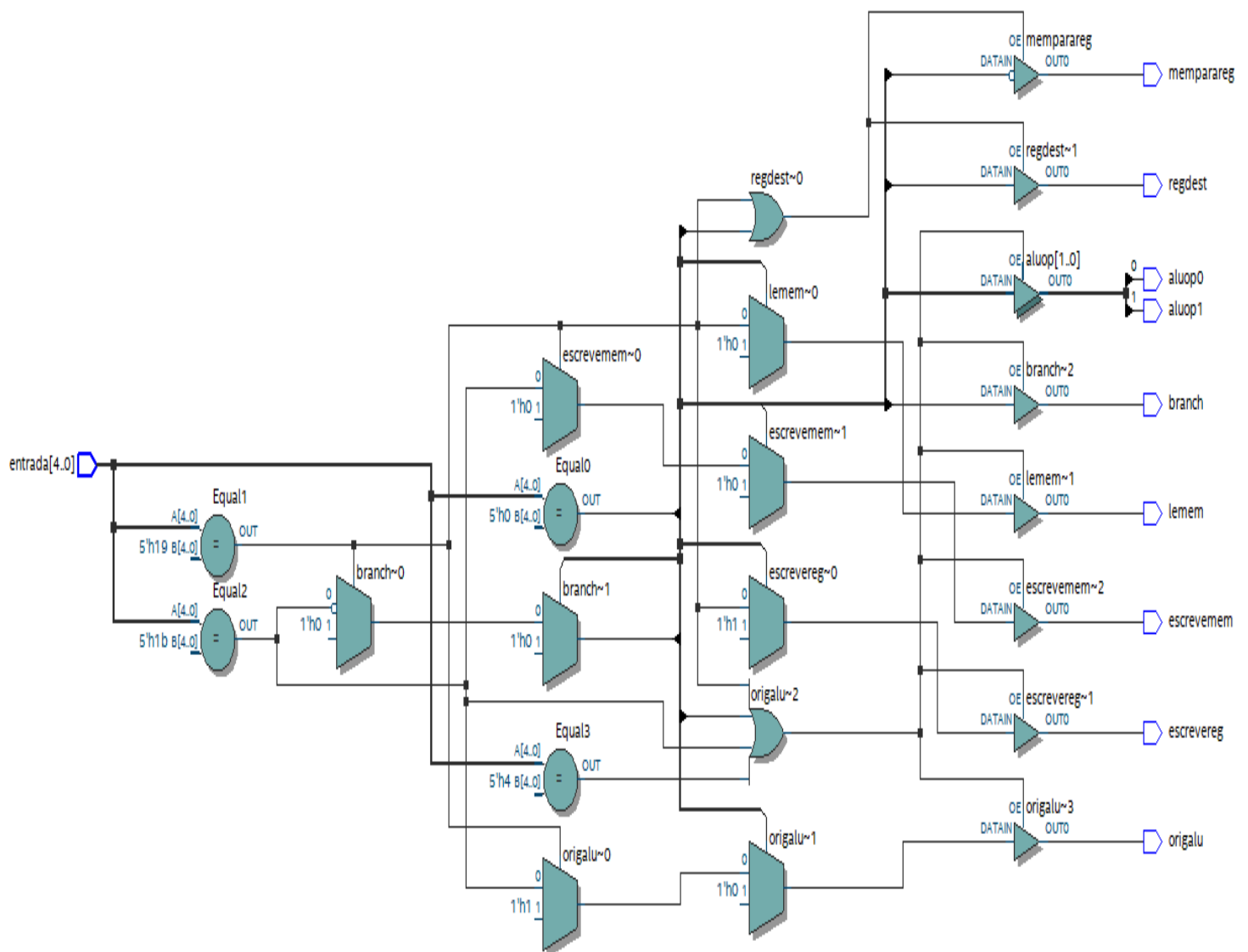
Name	Value at 0 ps	0 ps				
> ENTRAD...	B 00000000...	0000000000000000	000000000100000	000000001000000	000000001100000	
> ENTRAD...	B 11111111...	1111111111111111	000000000111111	000000001111111	000000001011111	
> SAIDA	B 11111111...	1111111111111111	000000001011111	000000001011111	000000010001111	

```

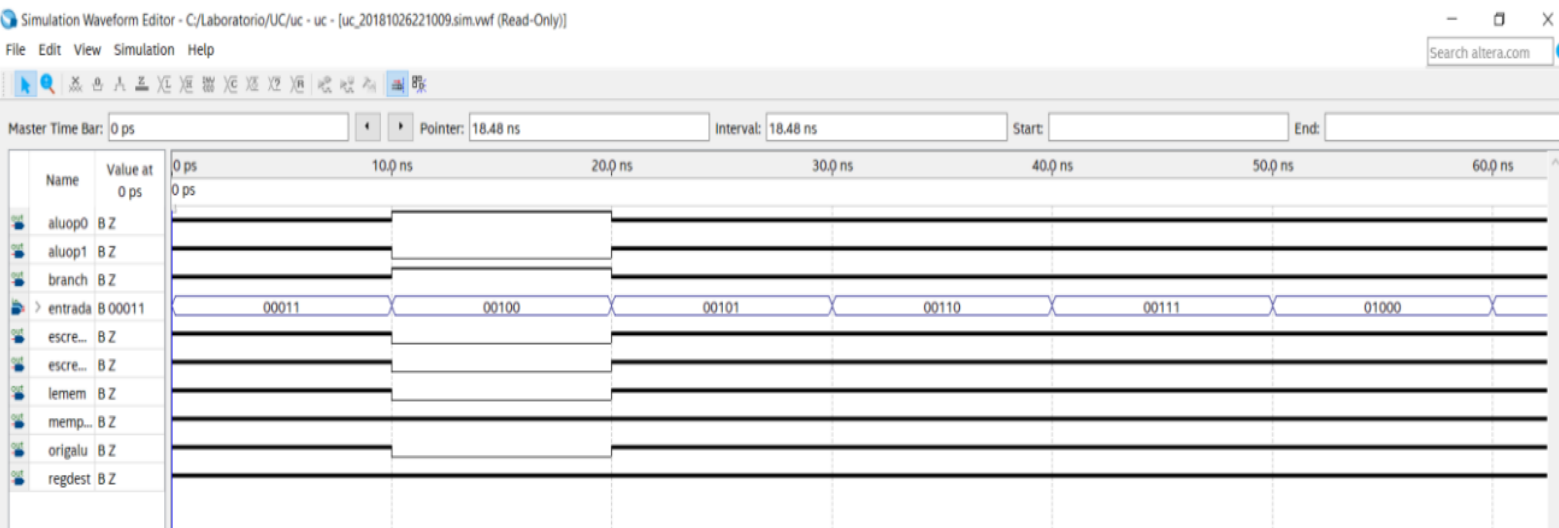
1  LIBRARY ieee; USE ieee.std_logic_1164.all;
2  ENTITY unidadeDeControle IS
3  PORT
4  (
5      entrada : in std_logic_vector (4 DOWNTO 0);
6      regdest : out std_logic;
7      origalu : out std_logic;
8      memparareg : out std_logic;
9      escrevereg : out std_logic;
10     lememe : out std_logic;
11     escrevemem : out std_logic;
12     branch : out std_logic;
13     aluop1 : out std_logic;
14     aluop0 : out std_logic);
15 END unidadeDeControle;
16 ARCHITECTURE behavior OF unidadeDeControle IS
17 BEGIN
18     abc: PROCESS (entrada)
19     BEGIN
20         IF entrada = "00000" then --instrução tipo r
21             regdest <= '1';
22             origalu <= '0';
23             memparareg <= '0';
24             escrevereg <= '1';
25             lememe <= '0';
26             escrevemem <= '0';
27             branch <= '0';
28             aluop1 <= '1';
29             aluop0 <= '0';
30         ELSIF entrada = "11001" then --instrução load
31             regdest <= '0';
32             origalu <= '1';
33             memparareg <= '1';
34             escrevereg <= '1';
35             lememe <= '1';
36             escrevemem <= '0';
37             branch <= '0';
38             aluop1 <= '0';
39             aluop0 <= '0';
40         ELSIF entrada = "11011" then --instrução store
41             regdest <= 'Z';
42             origalu <= '1';
43             memparareg <= 'Z';
44             escrevereg <= '0';
45             lememe <= '0';
46             escrevemem <= '1';
47             branch <= '0';

```

Questão 9 -
 Unidade de
 Controle
 Código do
 programa:



WaveForm:

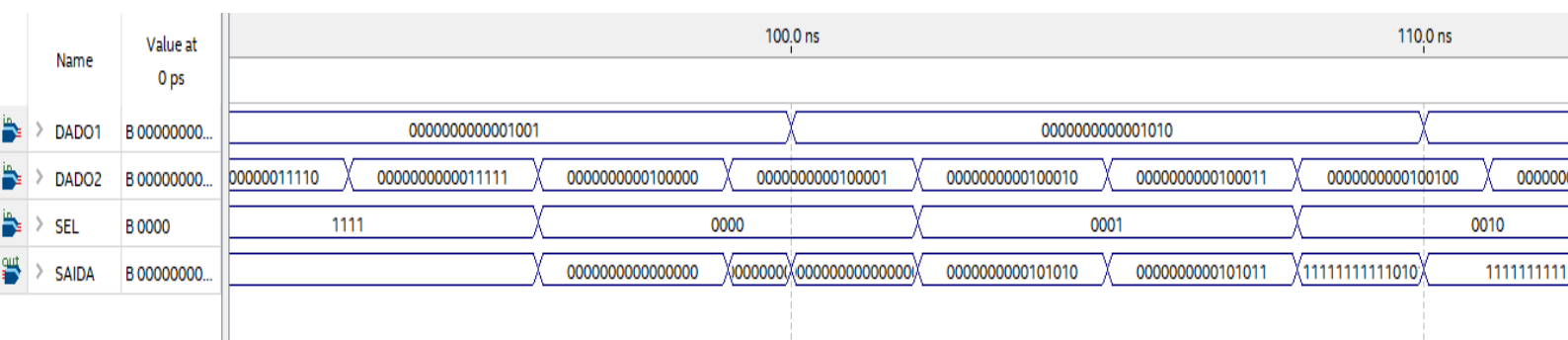
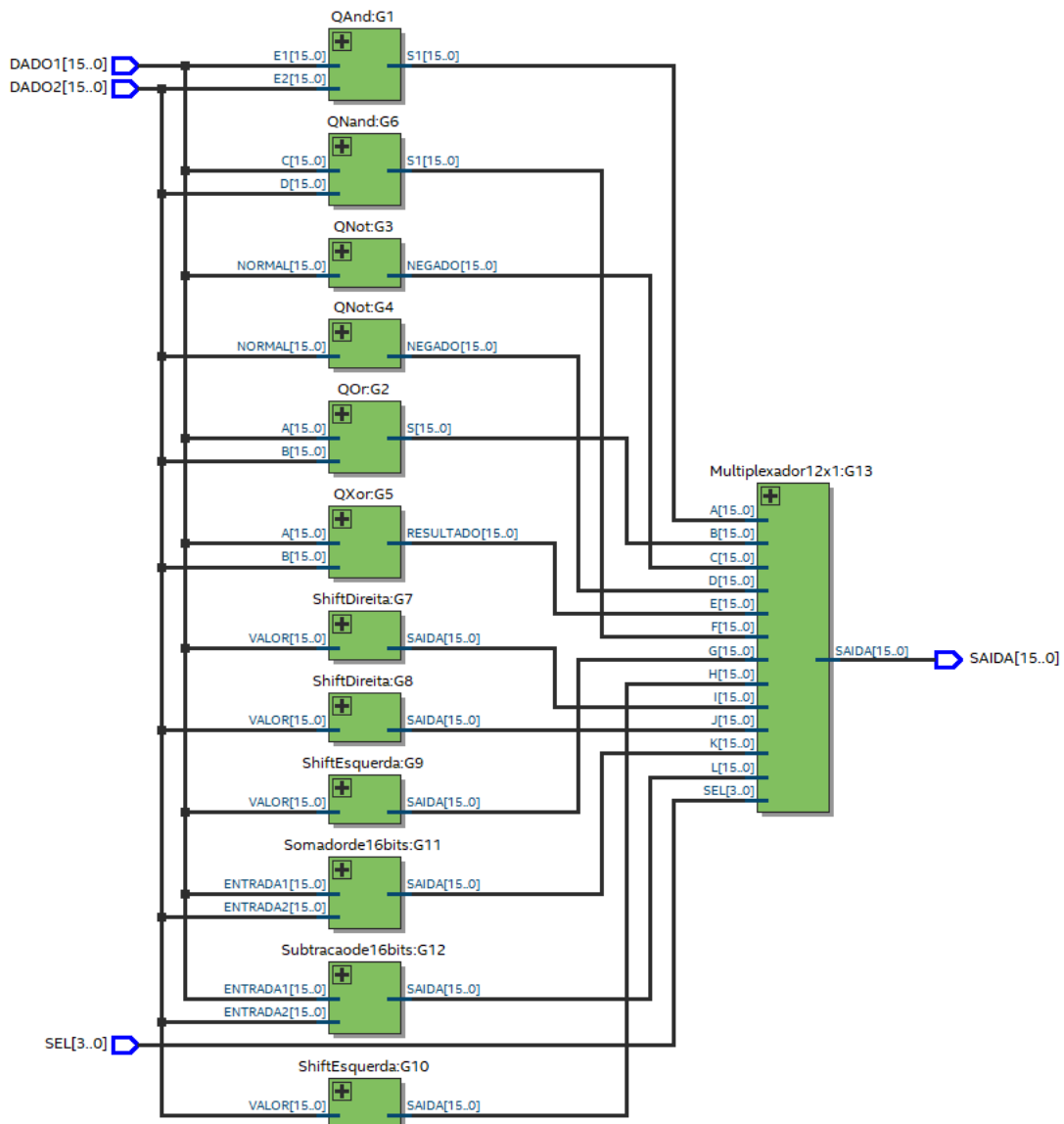


Como podemos ver temos o teste da wave for comprovando o funcionamento da unidade de controle temos entao ENTRADA : 00000, o REGDEST = '1'; ORIGALU <= '0'; MEMPARAREG <= '0'; ESCRIVEREG <= '1'; LEMEM <= '0'; ESCRIVEMEM <= '0';branch <= '0';ALUOP1 <= '1'; ALUOP0 <= '0';

Questão 10 - ULA de 16 bits Código do programa:

```
1  LIBRARY ieee; USE ieee.std_logic_1164.all; USE ieee.numeric_std.all; USE ieee.std_logic_unsigned.all;
2
3  ENTITY ULA16bits IS
11 ARCHITECTURE BEHAVIOR OF ULA16bits IS
12  |
13  COMPONENT QAnd IS
20  COMPONENT QOr IS
27  COMPONENT QNot IS
34  COMPONENT QXor IS
41  COMPONENT QNand IS
48  COMPONENT ShiftDireita IS
54  COMPONENT ShiftEsquerda IS
60  COMPONENT Somadorde16bits IS
66  COMPONENT Subtracao16bits IS
73  COMPONENT Multiplexador12x1 IS
74  PORT(
75  SIGNAL A,B,C,D,E,F,G,H,I,J,K,L: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
76  SIGNAL SEL : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
77  SIGNAL SAIDA: OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
78  );
79  END COMPONENT;
80  SIGNAL SAIDAAND, SAIDAOR, SAIDANOT, SAIDANOT2, SAIDAXOR: STD_LOGIC_VECTOR(15 DOWNTO 0);
81  SIGNAL SAIDANAND, SAIDASLL, SAIDASLL2, SAIDASRL, SAIDASRL2: STD_LOGIC_VECTOR(15 DOWNTO 0);
82  SIGNAL SAIDASOMA, SAIDASUBTRACAO: STD_LOGIC_VECTOR(15 DOWNTO 0);
83  BEGIN
84  G1: QAnd PORT MAP(DADO1,DADO2, SAIDAAND);
85  G2: QOr PORT MAP(DADO1,DADO2,SAIDAOR);
86  G3: QNot PORT MAP(DADO1,SAIDANOT);
87  G4: QNot PORT MAP(DADO2,SAIDANOT2);
88  G5: QXor PORT MAP(DADO1,DADO2,SAIDAXOR);
89  G6: QNand PORT MAP(DADO1,DADO2,SAIDANAND);
90  G7: ShiftDireita PORT MAP(DADO1,SAIDASRL);
91  G8: ShiftDireita PORT MAP(DADO2,SAIDASRL2);
92  G9: ShiftEsquerda PORT MAP(DADO1,SAIDASLL);
93  G10: ShiftEsquerda PORT MAP(DADO2,SAIDASLL2);
94  G11: Somadorde16bits PORT MAP(DADO1,DADO2,SAIDASOMA);
95  G12: Subtracao16bits PORT MAP (DADO1,DADO2,SAIDASUBTRACAO);
96  G13: Multiplexador12x1 PORT MAP(SAIDAAND, SAIDAOR, SAIDANOT, SAIDANOT2, SAIDAXOR,SAIDANAND, SAIDASLL, SAIDASLL2, SAIDASRL, SAIDASRL2,SAIDASOMA, SAIDASUBTRACAO,SEL,SAIDA);
97
98  END BEHAVIOR;
```

A ULA importa os componentes AND, OR, NOT, XOR, NAND, SHIFTDIREITA, SHIFTESQUERDA, SOMADORDE16BITS, SUBTRACAODE16BITS, MULTIPLEXADOR12X1, para as operações e então são usadas com PORTMAP e todas as suas saídas (do portmap) são jogadas no multiplexador de 12 x 1 e então o seletor decide qual operação será jogada para a saída.



WaveForm:

como podemos ver o resultado depende do SEL do multiplexador para decidir qual a operação foi realizada e a manda para a para SAIDA na primera parte em que o SEL é 0000 a operação AND é realizada por isso o final 0000000000000000.

Questão 11 - Extensor de sinal de 8 bits para 16 bits

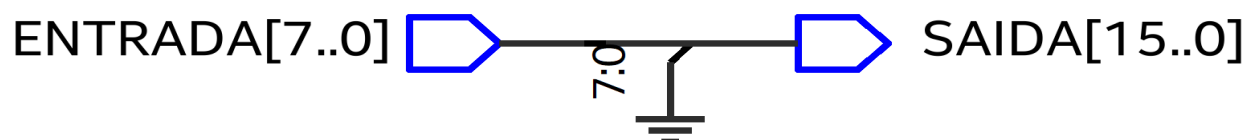
Código do Programa:

```
ExtensordeSinal8To16bits.vhd
Compilation Report - LabVhdl

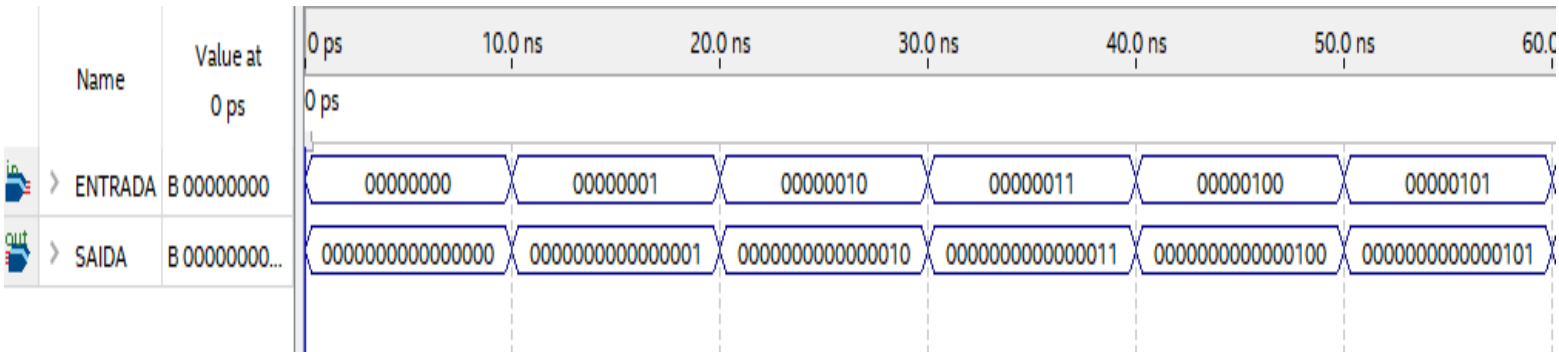
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.numeric_std.all;
4  USE ieee.std_logic_unsigned.all;
5
6  ENTITY ExtensordeSinal8To16bits IS
7  PORT(
8      ENTRADA : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
9      SAIDA   : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
10 );
11
12 END ExtensordeSinal8To16bits;
13
14 ARCHITECTURE BEHAVIOR OF ExtensordeSinal8To16bits IS
15
16 BEGIN
17
18     SAIDA <= ("00000000") & ENTRADA;
19
20 END BEHAVIOR;
```

A ENTRADA de 8 bits é o valor que será estendido e SAIDA é o valor já estendendo o código funciona basicamente pegando o valor de 8 bits e concatenando com oito bits 00000000 e o resultado é recebido pela SAIDA, assim os valores passam a ter 16 bits.

RTL Viewer:



WaveForm:



A WaveForm mostra a extensão acontecendo já que a entrada de oito bits em todos os casos da WaveForm saem com 16 bits apos a concatenação dos 00000000.



Project Navigator Files

- ShiftDireita.vhd
- ShiftDireitaWaveForm.vwf
- Subtracao16bits.vhd
- Multiplexador12x1.vhd
- ULA16bitsWaveForm.vwf
- ExtensordeSinal8To16bits.vhd
- ExtensordeSinal8To16bitsWaveForm.vwf
- BancodeRegistadores.vhd
- FlipFlopD.vhd
- FlipFlopJK.vhd
- BancodeRegistadoresWaveForm.vwf
- FlipFlopD.vwf
- FlipFlopJKWaveForm.vwf
- UnidadedeControle.vhd
- Maquina_estados.vhd

Tasks Compilation

Task	
▼	Compile Design
✓	> Analysis & Synthesis
	> Fitter (Place & Route)
	> Assembler (Generate programming fi
	> Timing Analysis
	> EDA Netlist Writer
	Edit Settings
	Program Device (Open Programmer)

MemoriaRAM16bits.vhd MemoriaROM16bits.vhd Unidadedec

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5  ENTITY Maquina_estados IS
6  PORT(
7      Clock, X, reset: IN STD_LOGIC;
8      Saida: OUT STD_LOGIC_VECTOR (1 downto 0)
9  );
10 END Maquina_estados;
11 ARCHITECTURE RTL OF Maquina_estados IS
12     TYPE tipo_estado IS (A, B, C, D);
13     signal estado: tipo_estado;
14 BEGIN
15     process(Clock, reset) begin
16
17         if reset = '1' then
18             estado <= A;
19
20         ELSIF rising_edge(Clock) THEN
21             case estado is
22                 when A =>
23                     if X = '1' then
24                         estado <= B;
25                         -- Saida <= "01";
26                     end if;
27
28                 when B =>
29                     if X = '1' then
30                         estado <= C;
31                         -- Saida <= "10";
32                     end if;
33
34                 when C =>
35                     if X = '1' then
36                         estado <= D;
37                         -- Saida <= "11";
38                     end if;
39
40                 when D =>
41                     if X = '1' then
42                         estado <= B;
43                         -- Saida <= "01";
44                     else
45                         estado <= A;
46                         -- Saida <= "00"
```


Quartus Prime Lite Edition - C:/Lab_VHDL/LabVhdl - LabVhdl

File Edit View Project Assignments Processing Tools Window Help

LabVhdl

Project Navigator

- ShiftDireita.vhd
- ShiftDireitaWaveForm.vwf
- Subtracao16bits.vhd
- Multiplexador12x1.vhd
- ULA16bitsWaveForm.vwf
- ExtensordeSinal8To16bits.vhd
- ExtensordeSinal8To16bitsWaveForm.vwf
- BancodeRegistadores.vhd
- FlipFlopD.vhd
- FlipFlopJK.vhd
- BancodeRegistadoresWaveForm.vwf
- FlipFlopD.vwf
- FlipFlopJKWaveForm.vwf
- UnidadeControle.vhd
- Maquina_estados.vhd

Tasks

Compilation

- Task
- Compile Design
- Analysis & Synthesis
- Fitter (Place & Route)
- Assembler (Generate programming file)
- Timing Analysis
- EDA Netlist Writer
- Edit Settings
- Program Device (Open Programmer)

MemoriaRAM16bits.vhd

```

19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

```

```

ELSIF rising_edge(clock) THEN
  case estado is
    when A =>
      if X = '1' then
        estado <= B;
        -- Saida <= "01";
      end if;
    when B =>
      if X = '1' then
        estado <= C;
        -- Saida <= "10";
      end if;
    when C =>
      if X = '1' then
        estado <= D;
        -- Saida <= "11";
      end if;
    when D =>
      if X = '1' then
        estado <= B;
        -- Saida <= "01";
      else
        estado <= A;
        -- Saida <= "00";
      end if;
    when others =>
      estado <= A;
  end case;
end if;
end process;
WITH estado SELECT
  Saida <= "00" when A,
           "01" when B,
           "10" when C,
           "11" when D;
END RTL;

```

MemoriaROM16bits.vhd

UnidadeControle.vhd

RTL VIEWER:

