Prática de Eletrônica Digital 1 - (119466)

Turma E (Unb - Gama)

Pré-Relatório Experimento 3

Circuitos somatórios e subtratores

Agosto 26, 2016

Nome	Matrícula	Assinatura
Arthur Temporim	140016759	
Eduardo Nunes	140056149	

1 Pesquisa bibliográfica

Três tipos de somadores são utilizados com maior frequência. O meio somador, somador completo e somador completo paralelo.

O circuito meio somador é constituído por duas entradas, que são os dois bits a serem somados, uma saída, que é a resposta da soma, e o carry para a próxima posição. Para construí-lo, utilizamos a tabela verdade com essas quatro variáveis.

A	B	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Podemos obsevar que a saída S é exatamente idêntica a uma por XOR entre A e B. e Cout é uma porta AND entre A e B. Assim,

$$\mathbf{S = A \oplus B}$$

$$\mathbf{Ts = AB}$$

O circuito meio somado recebe esse nome pois não consegue fazer uma soma com números de dois bits com apenas um circuito. São necessários dois circuitos somadores por bit. Pois, esse circuito tem a saída para o carry, porém, não tem a entrada para ele. É necessário acoplar outro circuito para fazer a soma do carry. Visto esse problema, foi desenvolvido o somador completo, que tem uma entrada para o carry e consegue fazer somas com um circuito por bit. Conseguimos encontrar o circuito do somador completo através da tabela verdade de uma soma.

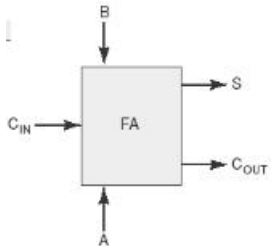
Entradas de bits da primeira parcela	Entradas de bits da segunda parcela	Entradas de bits do carry	Saída de bits da soma	Saída de bits do carry
A	B	C_{IN}	S	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Utilizando as simplificações, podemos perceber que as saídas S e Cout tem o seguinte resultado:

$$S = A \oplus [B \oplus C_{IN}]$$

$$C_{OUT} = BC_{IN}(\overline{A} + A) + AC_{IN}(\overline{B} + B) + AB(\overline{C}_{IN} + C_{IN}) =$$
$$= BC_{IN} + AC_{IN} + AB \quad (6\text{-}3)$$

Assim, o circuito somador completo é representado pela simbologia a seguir.



Dessa maneira, é possível calcular a soma das entradas e do carry para um bit. Para somar vários bits, são colocados vários SC em paralelo, um circuito por bit. A ligação pode ser observada a seguir.

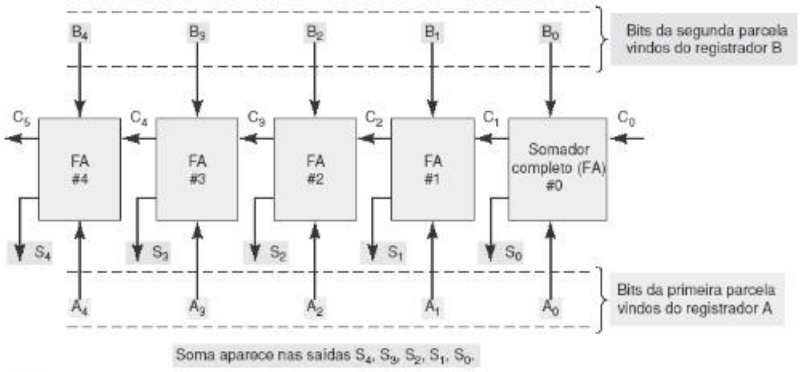


FIGURA 6.6
Diagrama em bloco de um circuito somador paralelo usando somadores completos.

Para executar a subtração dos números, seria necessário o mesmo procedimento que foi feito com os somadores, o que dobraria o número de circuitos. Para evitar esse excesso de portas, foi desenvolvido um sistema, chamado complemento de 2, que faz uma transformação no número, tornando-o diferente. Esse número modificado, sempre que for somado a outro número, será tido como negativo. Ou seja, seu quisermos fazer a operação A - B, devemos aplicar o complemento de 2 no número B, tornando-o -B e somando-o a A. O resultado estará correto. A transformação do número em complemento de 2(CP2) é feita em duas etapas. Na primeira é feito o complemento de 1(CP1) , onde todos os bits do número são invertidos, por exemplo o 100. Ao aplicarmos CP1 temos 001. O resultado do CP1 é somado ao número 1, resultando assim o CP2. Portanto, temos que o CP2 de 100 é 010.

2 Projetos e Simulações

2.1 Projeto1 - Complemento de 1

2.1.1 Diagramas

Diagrama alto nível:

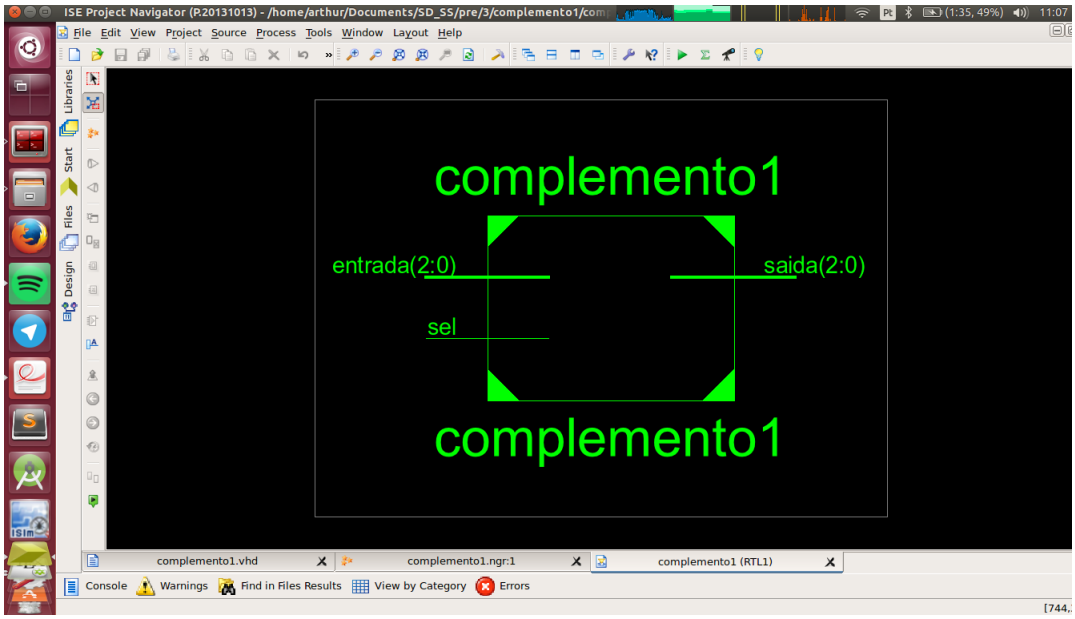


Figure 1: Diagrama 1 - Ise Design Suit 14.7

Diagrama 2

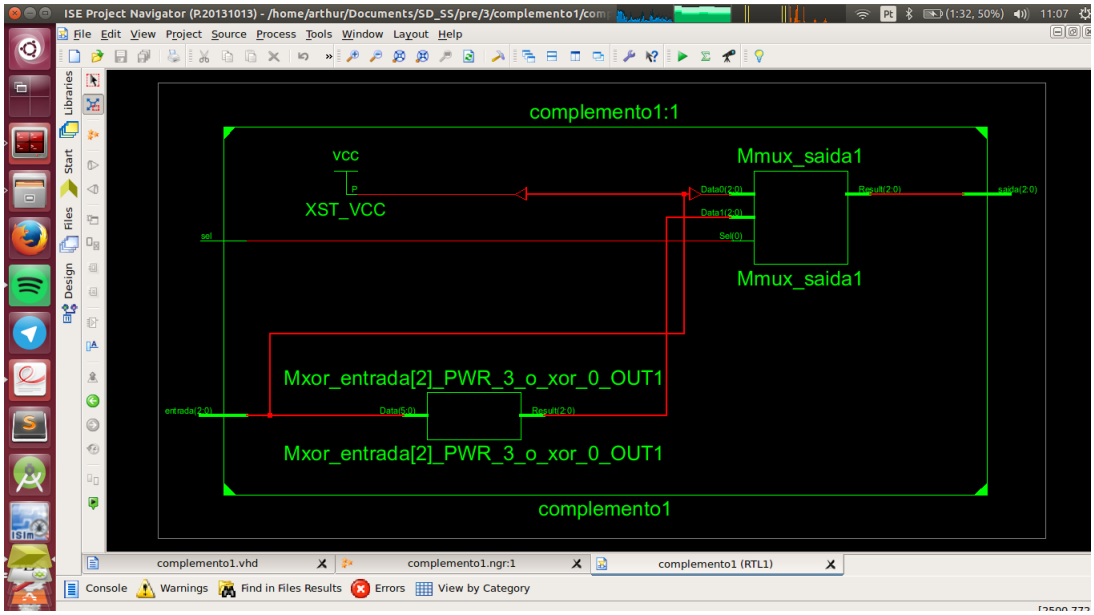


Figure 2: Diagrama 2 - Ise Design Suit 14.7

2.1.2 Plano de validação

Para Assegurar que o circuito está com o comportamento correto foram feitos os seguinstes testes:

Entrada	sel	saida
000	1	111
000	0	000

2.1.3 Código VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity complemento1 is
5      Port ( entrada : in  STD_LOGIC_VECTOR (2 downto 0) := "000";
6             sel     : in  STD_LOGIC := '1';
7           saida    : out STD_LOGIC_VECTOR (2 downto 0)
8     );
9  end complemento1;
10
11 architecture Behavioral of complemento1 is
12
13 signal aux : STD_LOGIC_VECTOR (2 downto 0);
14
15 begin
16
17 process (entrada,sel,aux)
18   begin
19     if (sel = '1') then
20       aux <= entrada xor "111";
21     else
22       aux <= entrada;
23     end if;
24 end process;
25   saida <= aux;
26 end Behavioral;
```

2.1.4 Forma de onda

Forma de onda gerada pelo circuito.

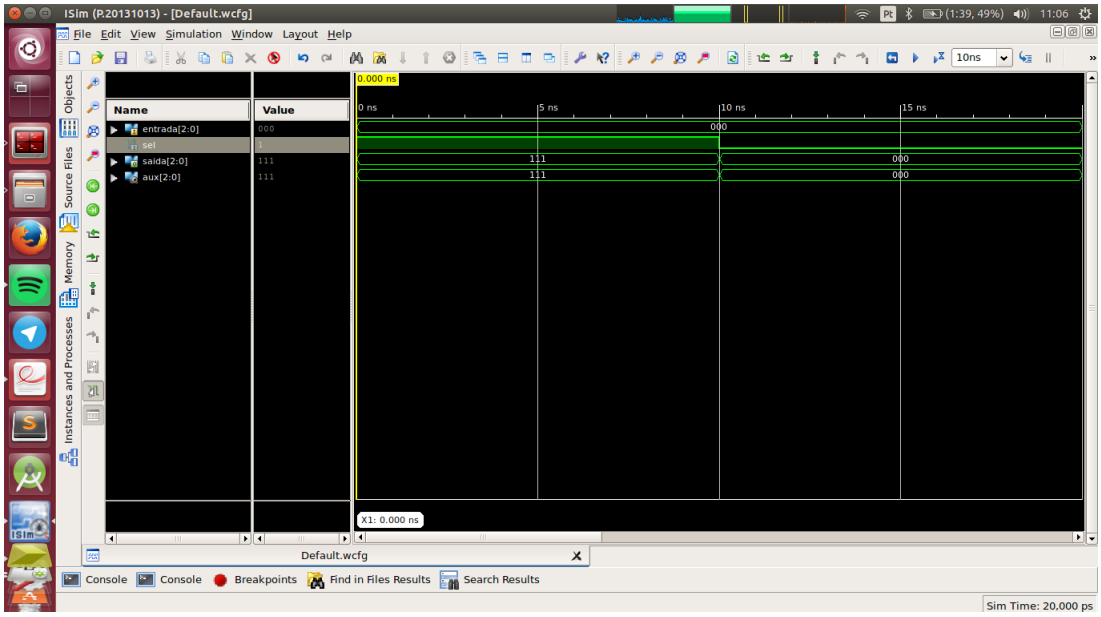


Figure 3: Forma de onda - Ise Design Suit 14.7

2.2 Projeto3 - Somador/Subtrator 3 bits

2.2.1 Diagramas

Diagrama alto nível:

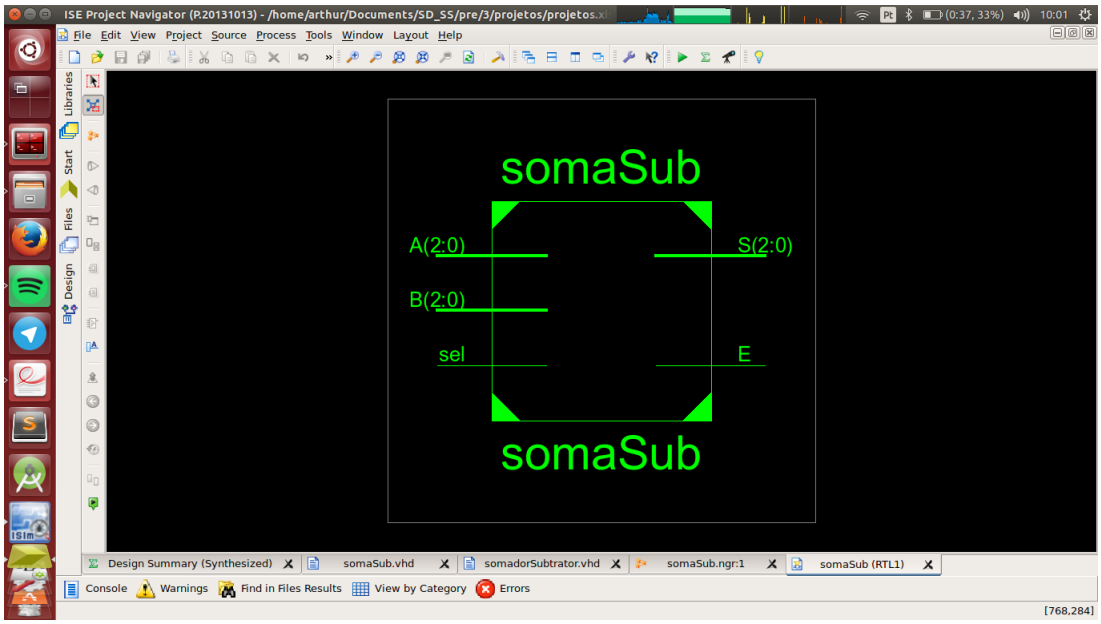


Figure 4: Diagrama 1 - Ise Design Suit 14.7

Diagrama 2

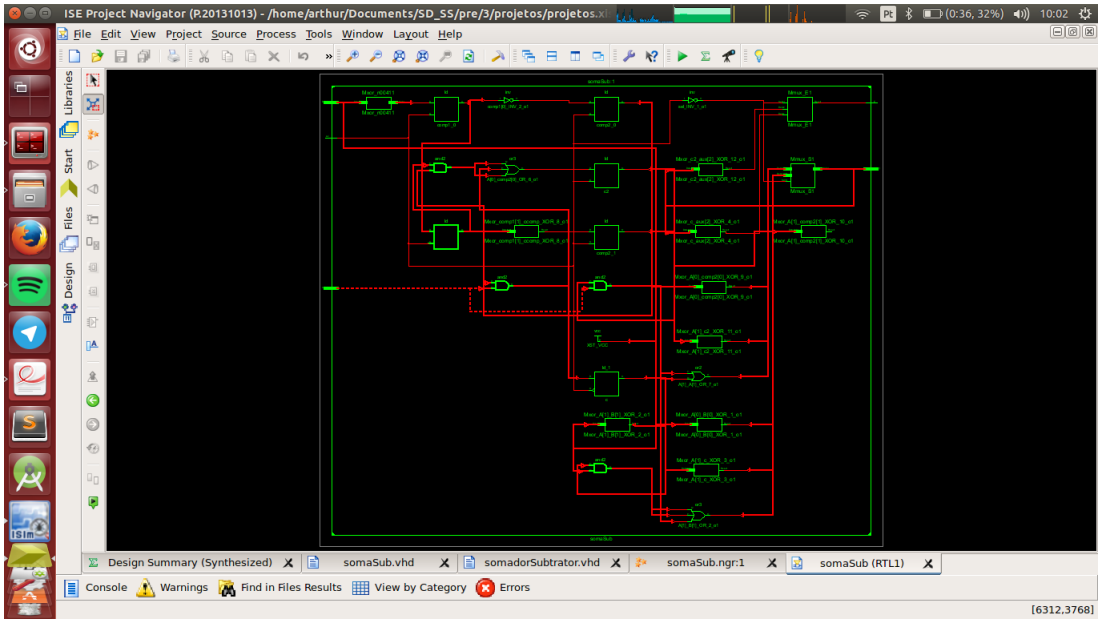


Figure 5: Diagrama 2 - Ise Design Suit 14.7

2.2.2 Plano de validação

Para assegurar que o circuito está com o comportamento correto foram feitos os seguinstes testes:

Entrada1	Entrada2	Sel	Saida	Overflow
001	001	0	010	0
010	001	1	001	0
011	001	0	100	1

2.2.3 Código VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity somaSub is
5    Port (
6      A   : in std_logic_vector (2 downto 0) := "011";
7      B   : in std_logic_vector (2 downto 0) := "001";
8      sel : in std_logic := '0';
9      S   : out std_logic_vector (2 downto 0);
10     E   : out std_logic
11   );
12 end somaSub;
13
14 architecture Behavioral of somaSub is
15
16 signal aux   : std_logic_vector (2 downto 0);
17 signal c     : std_logic;
18 signal c2    : std_logic;
19 signal ccomp : std_logic;
20 signal over  : std_logic;
21 signal igua  : std_logic;
22 signal comp1 : std_logic_vector (2 downto 0);
23 signal comp2 : std_logic_vector (2 downto 0);
24
25 begin
26
27   process (a,b,sel,c,c2,comp1,comp2,ccomp,aux, igua)
28   begin
29     -- Soma
30     if (sel = '0') then
31       aux(0) <= a(0) xor b(0);
32       c <= a(0) and b(0);
33       aux(1) <= a(1) xor b(1) xor c;
34       aux(2) <= (a(1) and b(1)) or (a(1) and c) or (b(1) and c);
35
36       igua <= not(a(0) xor b(0));
37       over <= c and igua;
38     --subtrai
39     else
40     -- Aplica complemento de 1 no B
41       comp1 <= b xor "111";
42
43     -- Aplica complemento de 2 no B
44       comp2(0) <= comp1(0) xor '1';
```

```
45        ccomp <= comp1(0) and '1';
46        comp2(1) <= comp1(1) xor ccomp;
47        comp2(2) <= (comp1(1) and '1') or (comp1(1) and ccomp) or ('1' and ccomp);
48
49      -- Faz a soma
50        aux(0) <= a(0) xor comp2(0);
51        c2 <= (a(0) and comp2(0)) or (a(0) and ccomp) or (comp2(0) and ccomp);
52        aux(1) <= a(1) xor comp2(1) xor c2;
53        aux(2) <= (a(1) and comp2(1)) or (a(1) and c2);
54
55        igua <= not(a(0) xor comp2(0));
56        over <= c2 and igua;
57      end if;
58    end process;
59
60    e <= over;
61    s <= aux;
62  end Behavioral;
```

2.2.4 Forma de onda

Forma de onda gerada pelo circuito. SOMA:

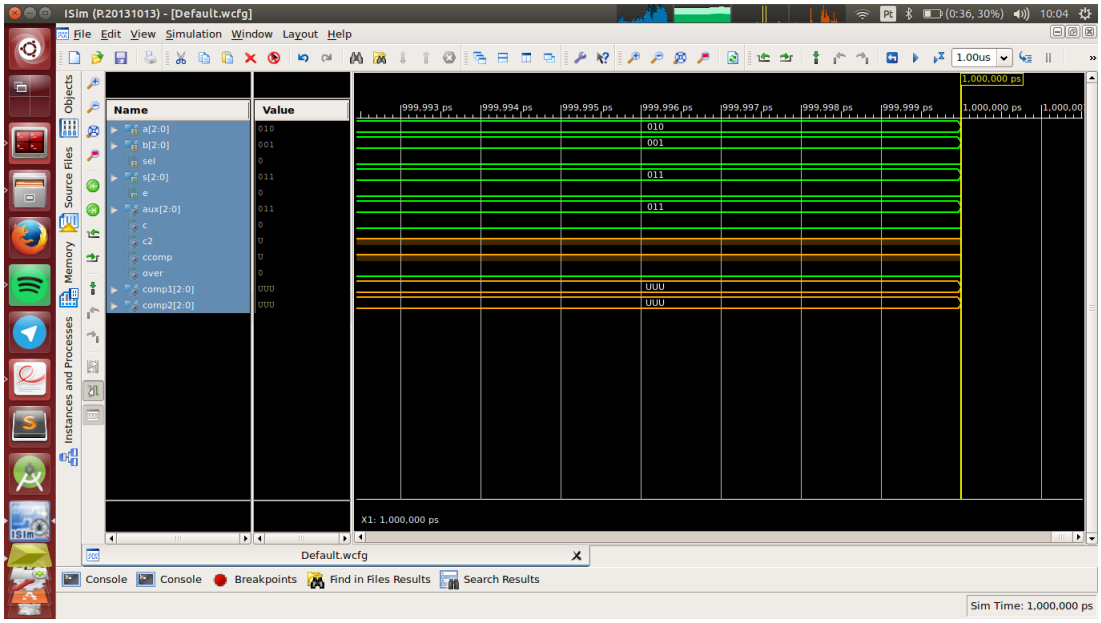


Figure 6: Forma de onda - Ise Design Suit 14.7

SUBTRAÇÃO:

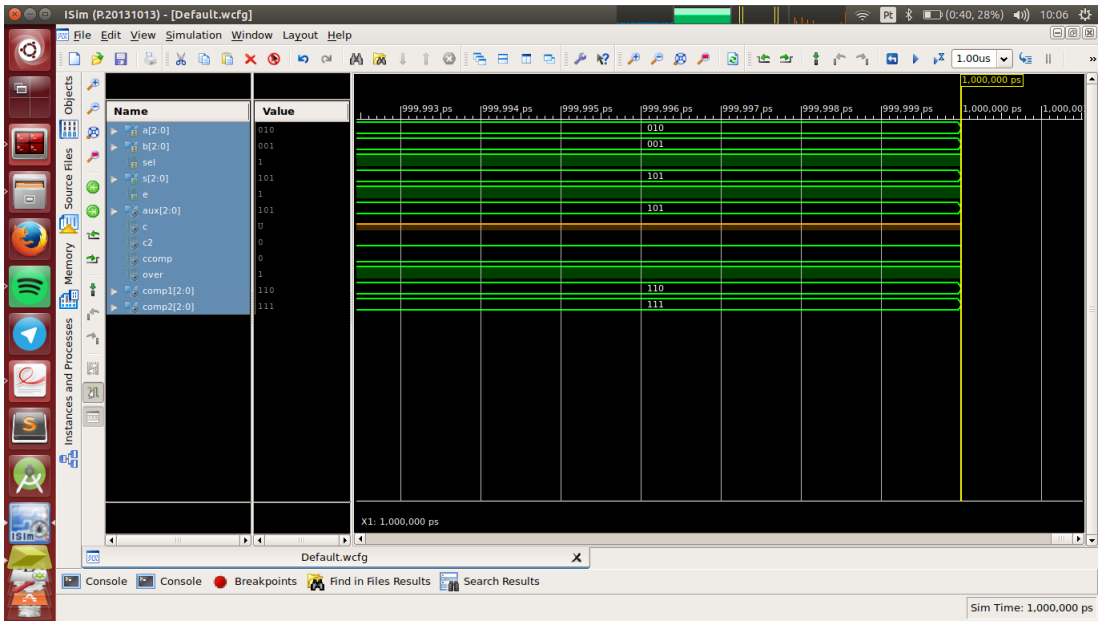


Figure 7: Forma de onda - Ise Design Suit 14.7

OVERFLOW:

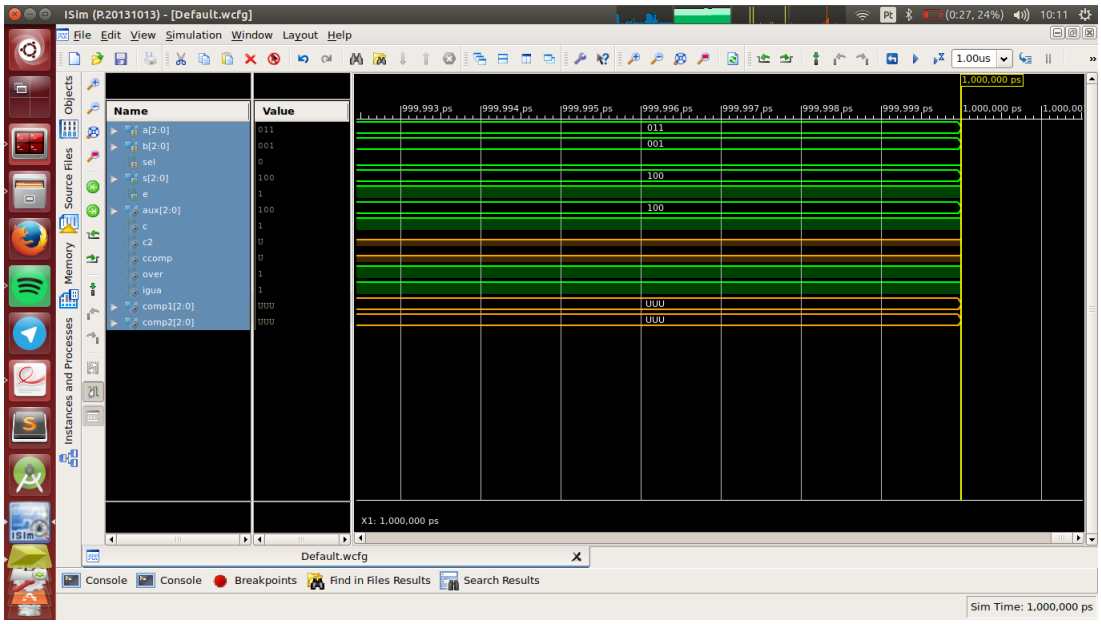


Figure 8: Forma de onda - Ise Design Suit 14.7