

### Código para o comparador de 1 bit:

```

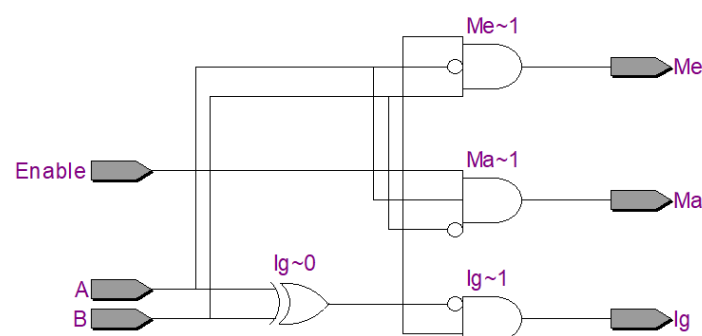
1  /*
2  Aluno: Alexandre Roque Silva de Paula
3  Professora: Mara Coelho
4  Disciplina: Sistemas Digitais */
5
6  module comparador1bit(A , B , Enable, Ma, Me, Ig);
7
8      input A, B, Enable;
9      output Ma, Me, Ig;
10
11     assign Ma = (A & ~B) & Enable;
12
13     assign Me = (~A & B) & Enable;
14
15     assign Ig = (A ^ B) & Enable;
16
17 endmodule
18

```

### Tabela verdade do circuito comparador de 1 bit:

| A | B | Enable | Maior | Menor | Igual |
|---|---|--------|-------|-------|-------|
| 0 | 0 | 0      | 0     | 0     | 0     |
| 0 | 0 | 1      | 0     | 0     | 1     |
| 0 | 1 | 0      | 0     | 0     | 0     |
| 0 | 1 | 1      | 0     | 1     | 0     |
| 1 | 0 | 0      | 0     | 0     | 0     |
| 1 | 0 | 1      | 1     | 0     | 0     |
| 1 | 1 | 0      | 0     | 0     | 0     |
| 1 | 1 | 1      | 0     | 0     | 1     |

### RTL Viewer do circuito comparador de 1 bit:



## Obtenção das expressões booleanas:

Saída: **Maior**  
Formato: Soma de produtos

**B, Enable**

|   |   |    |    |    |    |
|---|---|----|----|----|----|
|   |   | 00 | 01 | 11 | 10 |
| A | 0 | 0  | 0  | 0  | 0  |
|   | 1 | 0  | 1  | 0  | 0  |

$A \bar{B} \text{ Enable}$

Saída: **Menor**  
Formato: Soma de produtos

**B, Enable**

|   |   |    |    |    |    |
|---|---|----|----|----|----|
|   |   | 00 | 01 | 11 | 10 |
| A | 0 | 0  | 0  | 1  | 0  |
|   | 1 | 0  | 0  | 0  | 0  |

$\bar{A} B \text{ Enable}$

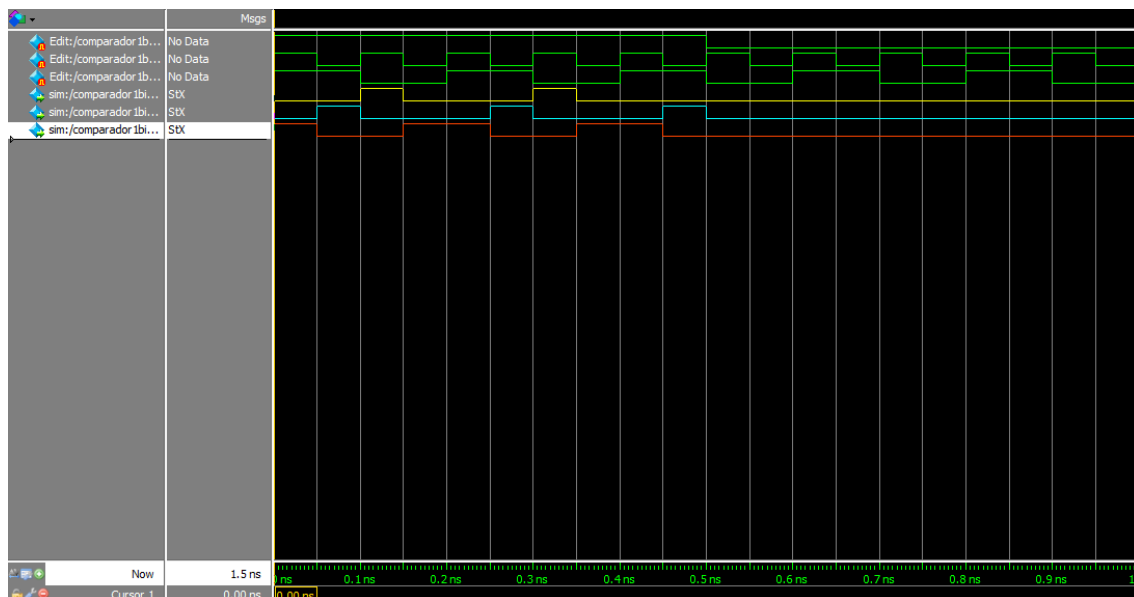
Saída: **Igual**  
Formato: Soma de produtos

**B, Enable**

|   |   |    |    |    |    |
|---|---|----|----|----|----|
|   |   | 00 | 01 | 11 | 10 |
| A | 0 | 0  | 1  | 0  | 0  |
|   | 1 | 0  | 0  | 1  | 0  |

$\bar{A} \bar{B} \text{ Enable} + A B \text{ Enable}$

## Simulador ModelSim do circuito comparador de 1 bit:



## Especificações:

Enable: Frequência de 1000 ps ; Duty cycle de 50%.

A: Frequência de 100 ps ; Duty cycle de 50%.

B Frequência de 200 ps ; Duty cycle de 50%.

Ma: Amarelo.

Me: Ciano.

Ig: Laranja.

Observa-se que durante toda a parte em que o Enable está desativado todas as outras saídas são “0”, e quando o Enable está em “1”, elas variam de acordo com a entrada. Quando A e B estão ambas ativadas ou ambas desativadas, o sinal de Ig é “High”, pois isso representa que os bits são iguais. Ademais, quando A é “1” e B é “0”, o sinal de Ma fica “High”, denotando que A é maior que B, contrariamente de Me, que se torna “High” quando o “A” está em “Low” e B em “High”.

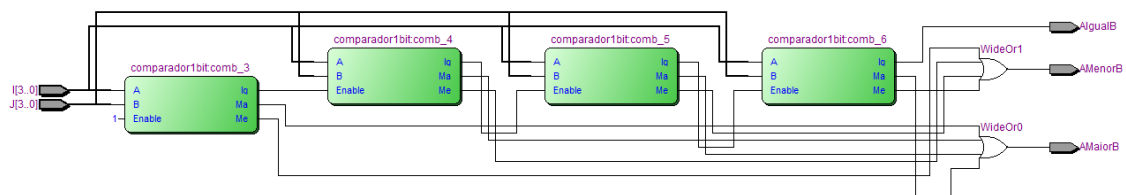
Código do circuito comparador de 4 bits:

```

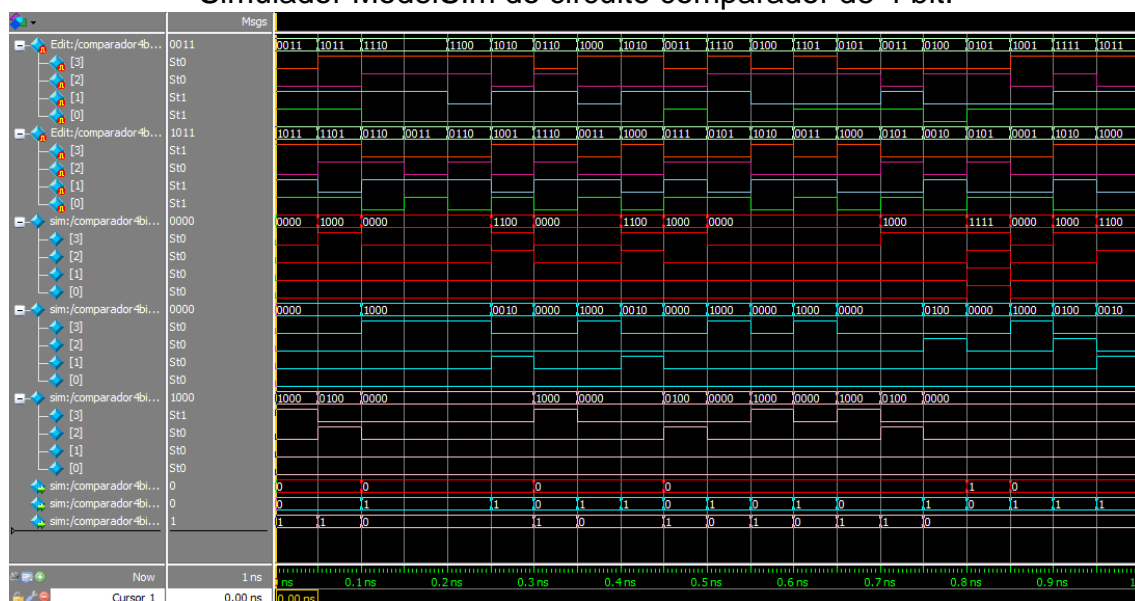
5
6 module comparador4bit(I, J, AIGualB, AMaiorB, AMenorB);
7
8     input [3:0]I;
9     input [3:0]J;
10
11     output AIGualB, AMaiorB, AMenorB;
12
13     wire[3:0] auxIguar;
14     wire[3:0] auxMaior;
15     wire[3:0] auxMenor;
16
17     /* Parâmetros:
18     comparador1bit(A , B , Enable, Ma, Me, Ig)*/
19     comparador1bit(I[3], J[3], 1, auxMaior[3], auxMenor[3], auxIguar[3]);
20     comparador1bit(I[2], J[2], auxIguar[3], auxMaior[2], auxMenor[2], auxIguar[2]);
21     comparador1bit(I[1], J[1], auxIguar[2], auxMaior[1], auxMenor[1], auxIguar[1]);
22     comparador1bit(I[0], J[0], auxIguar[1], auxMaior[0], auxMenor[0], auxIguar[0]);
23
24
25     assign AIGualB = auxIguar[0];
26     assign AMaiorB = |auxMaior;
27     assign AMenorB = |auxMenor;
28
29
30 endmodule
31

```

RTL Viewer circuito comparador de 4 bits:



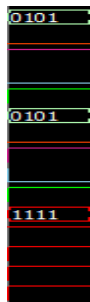
Simulador ModelSim do circuito comparador de 4 bit:



## Especificações:

Sendo I, o vetor de números A e J o vetor de números B, a terceira posição (3) de ambos está representada em “Orange Red”, a segunda posição (2) em “Violet Red”, a primeira posição (1) em “Sky Blue” e a posição zero (0) em “Green”. A wave foi aplicada de maneira aleatória, em I com o seed de 5 e em J com o seed de 10, formando assim valores distintos para ambos.

O bloco auxIguar representa quando os valores do bloco de números da mesma posição nos vetores I e J são iguais, ou seja, torna-se 1 somente quando ambos os números são 0 ou quando ambos são 1. Ele está com a cor “Red” no gráfico.



| Posição | Valor |
|---------|-------|
| 0       | 1     |
| 1       | 1     |
| 2       | 1     |
| 3       | 1     |
| 4       | 0     |

A tabela ao lado representa os números I: 0101 e J: 0101, e mostra que o bloco auxIguar se torna 1 em todos os valores (1111), revelando que todas as posições dos vetores I e J são iguais, fazendo com que os números sejam iguais.

O bloco auxMaior representa a situação em que o valor A do vetor I é maior do que o valor B do vetor J, na mesma posição, tornando-se 1 quando isso acontece e 0, caso contrário. Ele está com a cor “Cyan” no gráfico.



| Posição | Valor |
|---------|-------|
| 0       | 1     |
| 1       | 1     |
| 2       | 1     |
| 3       | 1     |
| 4       | 0     |

A tabela ao lado representa os números I: 1001 e J: 0001, e mostra que o bloco auxMaior, na posição 3, se torna 1, revelando que nesta posição, o número A é maior que B.

O bloco auxMenor funciona da mesma forma que o bloco auxMaior, só que ao contrário, ou seja, será 1 quando B for maior que A. Ele está com a cor “Pink” no gráfico.



| Posição | Valor |
|---------|-------|
| 0       | 1     |
| 1       | 1     |
| 2       | 1     |
| 3       | 1     |
| 4       | 0     |

A tabela ao lado representa os números I: 0100 e J: 1100, e mostra que o bloco auxMenor, na posição 3, se torna 1, revelando que nesta posição, o número A é menor que B.

Ao final do gráfico, podemos notar as saídas `AigualB`, `AmaiorB`, e `AmenorB`, que exibem a comparação final dos números, após a comparação dos blocos de bits. Eles estão com radix: "Boolean" e format "Literal". Vale salientar que a comparação está sendo feita em passos, comparando bloco por bloco e depois passando para a próxima etapa, para obter o resultado total e final, descobrindo se A é igual, maior ou menor que B.