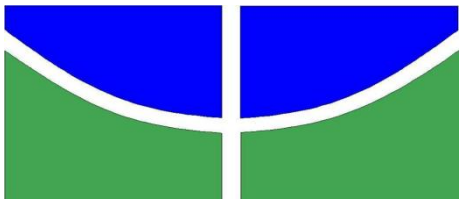


Universidade de Brasília

Campus Universitário Darcy Ribeiro

Disciplina: Circuitos Digitais (116351)

Semestre: 2/ 2013

Horário: Diurno – Turma A - Sexta 10:00 11:50

Local: LINF

4º EXPERIMENTO

Filipe Cunha Oliveira – 12/0011395

Samuel Sousa Almeida – 12/0062003

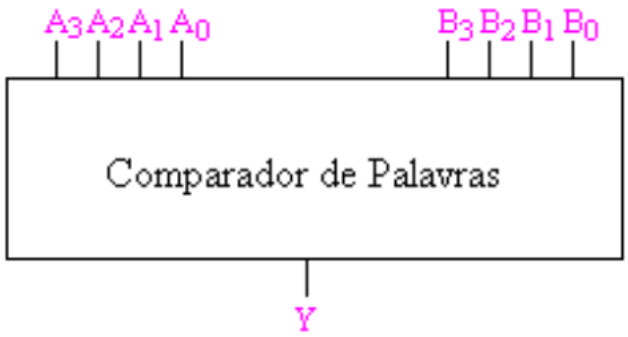
Brasília, 20 de Setembro de 2013.

1. OBJETIVOS

Os objetivos do presente relatório são de usar o sistema Quartus II para a implementação de circuitos com comparadores de palavras binárias com as técnicas utilizadas em relatórios passados de simplificação e montagem.

2. INTRODUÇÃO

Um comparador é um circuito combinatório operativo.Ele permite comparar grandezas de dois números binarios.Um comprimento de uma palavra binária é o número de bits que a compõem



Um comparador tem como saída 1 se os comprimentos forem iguais,caso contrário a saída sera 0.Ele também pode possuir três saídas:

se A=B ou

se A>B ou

se A<B

Para uma boa realização de um circuito combinacional é necessário seguir alguns passos,como os mostrados abaixo:

a) Descrever sistema;

b) Elaborar tabela da verdade;

c) Obter funções booleanas a partir da tabela da verdade;

d) Simplificar funções booleanas obtidas ;

e) Elaborar diagrama lógico.

Dessa maneira ,fica mais intuitivo a realização da montagem e análise do circuito

3. MATERIAL UTILIZADO

- Software computacional Quartus II,em sua versão 9.1;

☐ FPGA Altera Cyclone II.

3. PROCEDIMENTOS

PARTE 1

3.1. Usando apenas portas NAND de DUAS entradas,projete um comparador de palavras de *3 bits* e complete a tabela da verdade para A_i e B_i e obtenha a equação para Z_i;

3.2. Minimize a função obtida anteriormente;

3.3. Faça um diagrama lógico parcial. Implemente-o e verifique se o resultado combina com o resultado da tabela da verdade;

3.4. Faça um diagrama lógico total.Implemente-o;

3.5. Comente os resultados obtidos.

PARTE 2

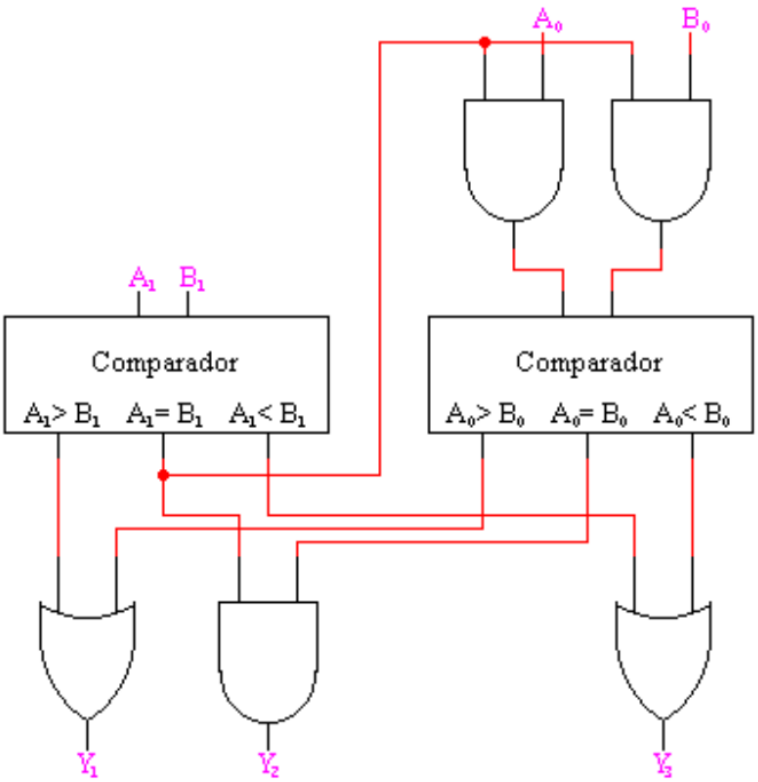
3.6. Y_3,sendo que Y_1 =1 para A>B ,Y_2=1 para A=B e Y_3=1 para A<B (A e B são A_0 e A_1 e B são B_0 e B_1);

3.7. Faça uma tabela da verdade parcial e obtenha funções parciais;

3.8. Minimize a função obtida anteriormente;

3.9. Faça o diagrama lógico parcial. Implemente-o e verifique se o resultado combina com o resultado da tabela da verdade;

3.10. Faça o diagrama lógico total de acordo com a figura abaixo.Implemente-o;



3.11. Comente os resultados obtidos.

4. DADOS E ANÁLISE

PARTE 1

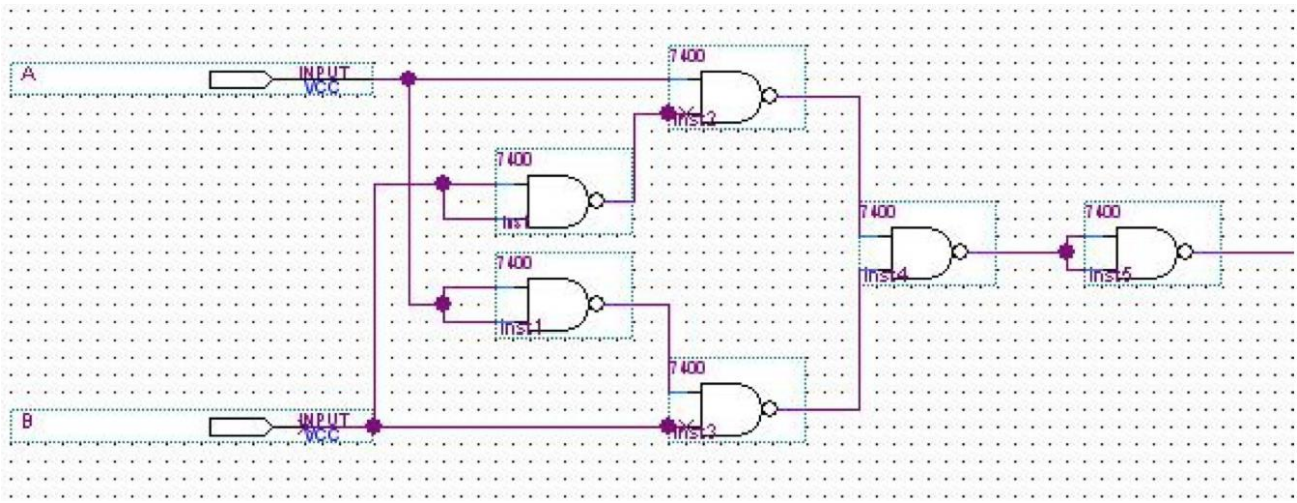
O circuito da figura foi montado graças ao programa Quartus II .Aqui embaixo se encontra a tabela da verdade também.

Entradas		Saída
A	B	Y_1
0	0	1
0	1	0
1	0	0
1	1	1

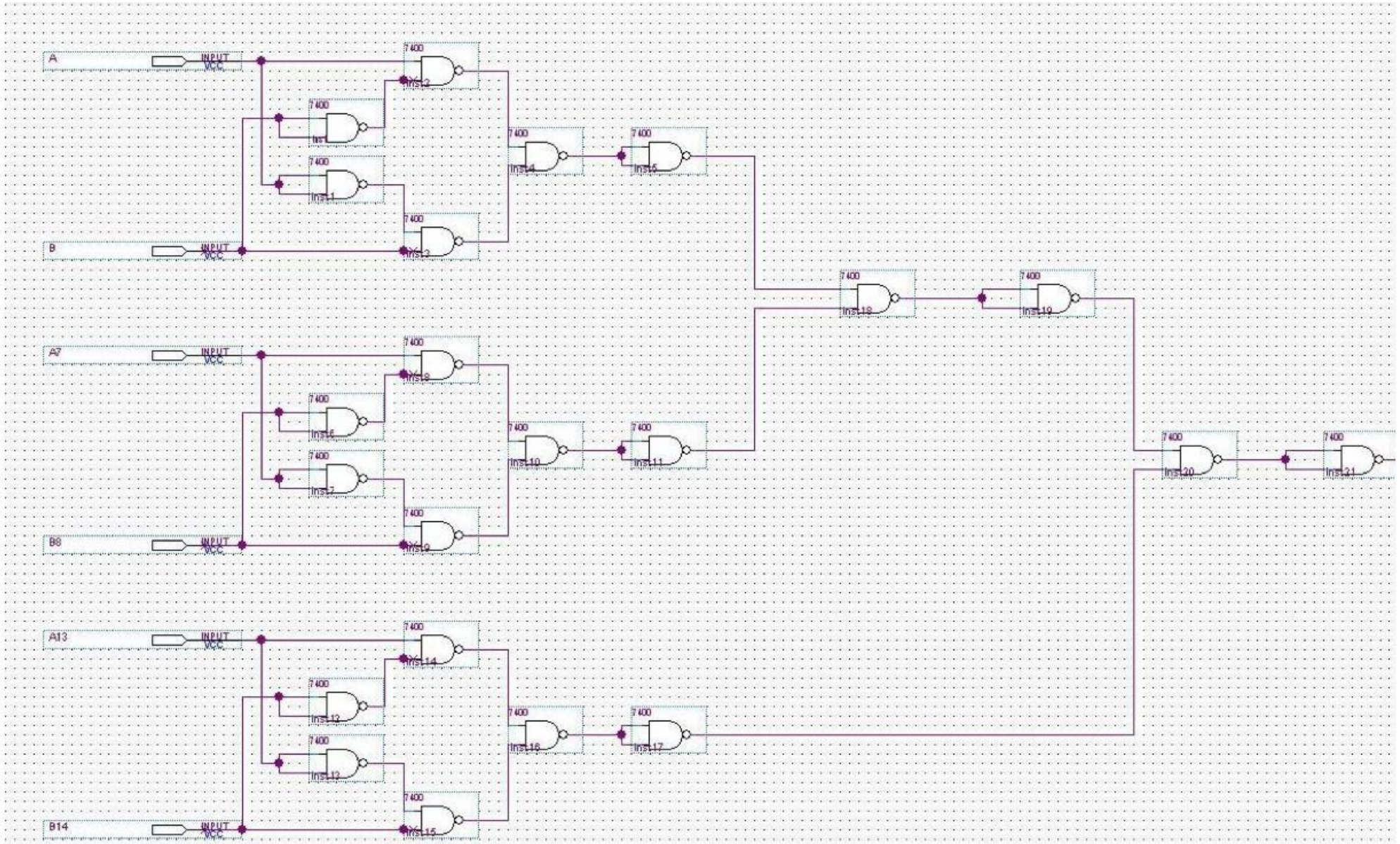
A expressão booleana ficou então como $\overline{(Ai.(\overline{Bi}) + (\overline{Ai}).Bi)}$

Aplicou-se De Morgan e foi visto que ela ficou $(\overline{(Ai)} + Bi).(Ai + \overline{Bi})$

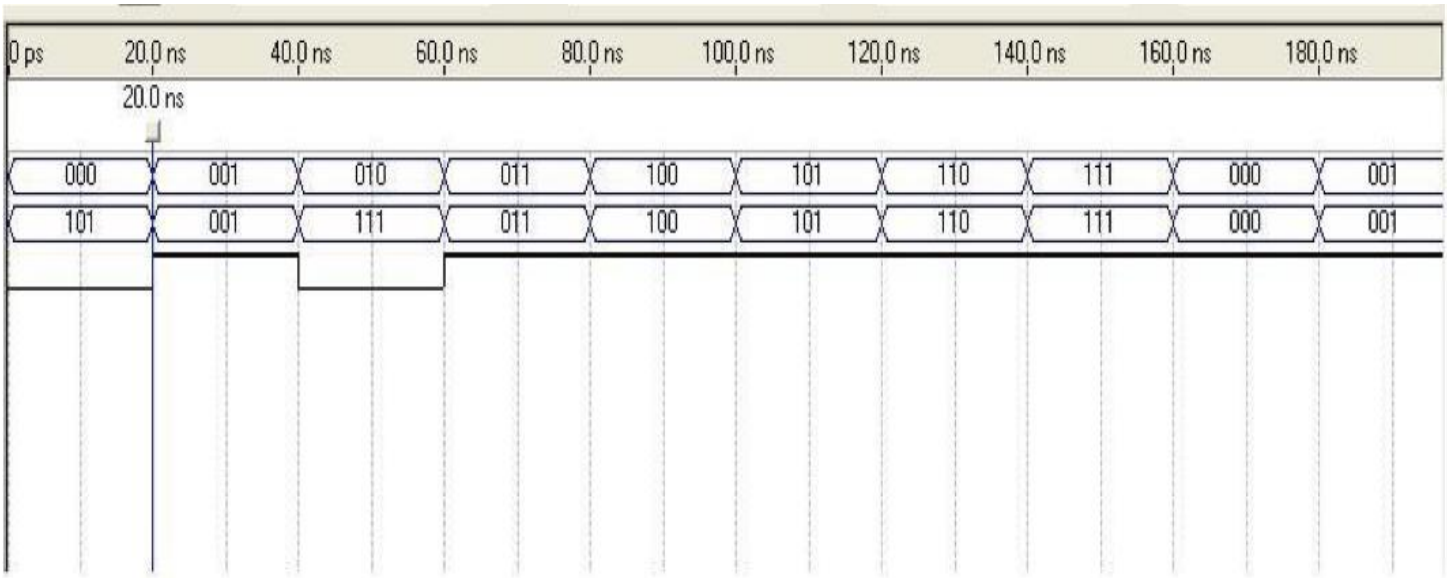
Logo abaixo se encontra o diagrama lógico parcial(comparação do par de bits(A_i,B_i));



Logo abaixo se encontra o diagrama lógico total implementado


A
INPUT
VCC
B
INPUT
VCC
A7
INPUT
VCC
B8
INPUT
VCC
A13
INPUT
VCC
B14
INPUT
VCC
7400
Inst2
Inst1
Inst3
Inst4
Inst5
Inst8
Inst6
Inst7
Inst9
Inst10
Inst11
Inst14
Inst12
Inst13
Inst15
Inst16
Inst17
Inst18
Inst19
Inst20
Inst21

Usando as funcionalidades do Quartus II é possível de comparar os resultados da tabela da verdade mais acima com a implementação do diagrama no programa.



É possível perceber a similaridade da tabela verdade feita com os resultados do programa Quartus II,obtendo sucesso na implementação do comparador de *3 bits*

PARTE 2

As entradas foram testadas e a tabela da verdade foi preenchida

Entradas				Saída		
A1	B_1	A_0	B_0	Y_1	Y_2	Y_3
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	1	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0

1	0	1	1	1	0	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	1	0	0
1	1	1	1	0	1	0

As expressões encontradas estão logo abaixo(possui a original e a minimizada pelo mapa de Karnaugh):

$$Y1 = \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD + A\bar{B}C\bar{D}$$

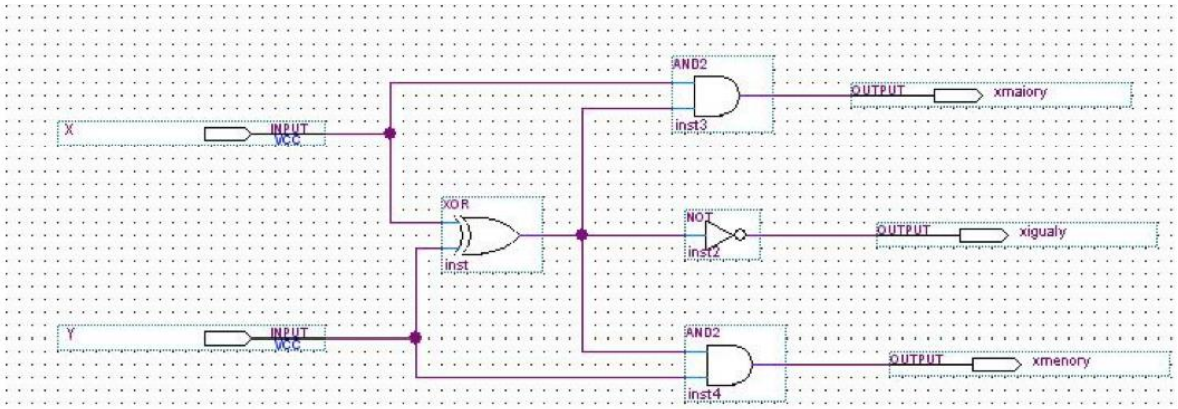
$$Y1 = AC\bar{D} + A\bar{B} + \bar{B}C\bar{D}$$

$$Y2 = \bar{A}\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}\bar{B}CD + ABCD$$

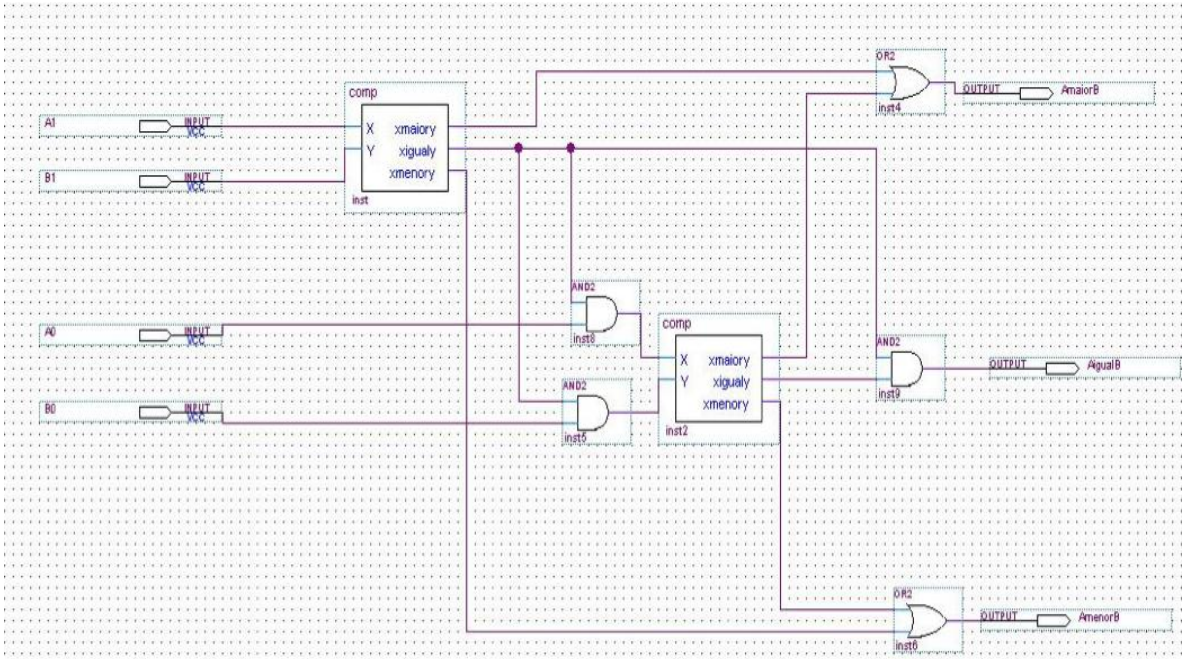
$$Y3 = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + \bar{A}BC\bar{D} + AB\bar{C}D$$

$$Y3 = \bar{A}\bar{C}D + \bar{B}\bar{C}D + \bar{A}B$$

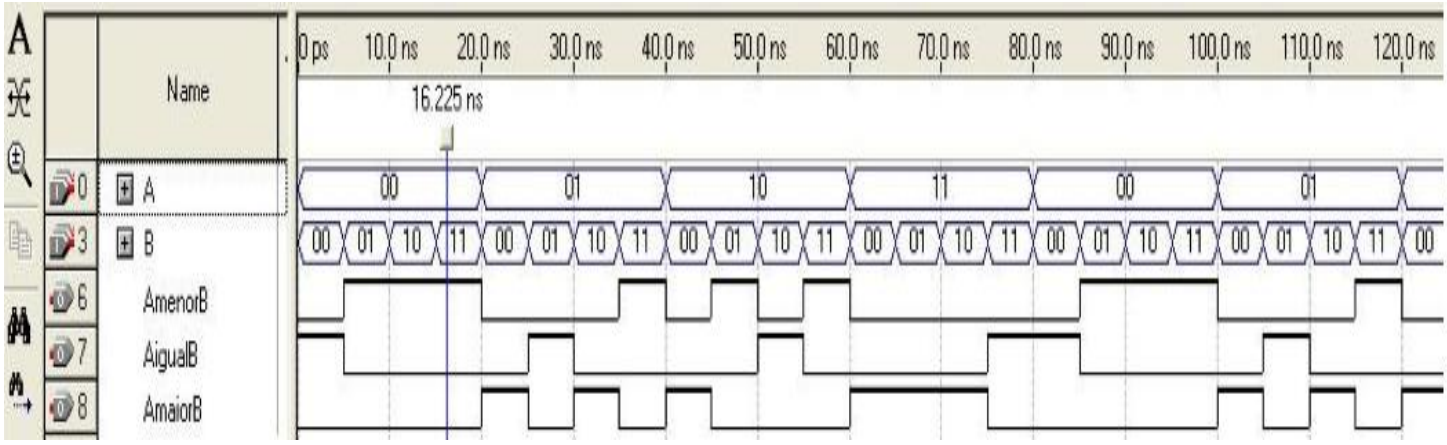
Logo abaixo se encontra o diagrama lógico parcial:



Logo abaixo se encontra o diagrama lógico total implementado



Usando as funcionalidades do Quartus II é possível de comparar os resultados da tabela da verdade mais acima com a implementação do diagrama no programa.



O mesmo ocorre na semelhança de resultados entre a tabela verdade e a do programa.Mais uma vez,foi implementado corretamente o comparador de *2 bits*

CONCLUSÃO

Foi estudado nesse relatório o uso de comparadores de *bits*, bem como o uso da ferramenta Quartus II para a implementação de circuitos. Foram usados métodos conhecidos da síntese de circuitos combinacionais já vistos em aulas teóricas. Pode-se afirmar que, com a semelhança dos resultados nas tabelas verdades com as tabelas do Quartus II que foi um experimento de bastante sucesso

5. TESTE DE AUTO-AVALIAÇÃO

(RESPOSTAS CORRETAS ESTARÃO EM NEGRITO)

1. Implemente um comparador de palavras de 4 bits em que a saída seja 1 somente quando A = B. Use 4 portas XOR e obtenha um diagrama com um total de 5 portas, como o da Figura 2. A porta de saída será uma:

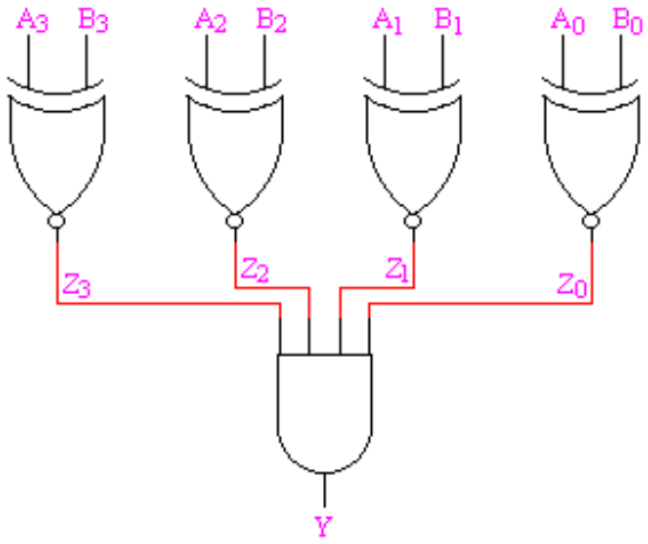


Figura 2

a) NAND de 4 entradas.

b) NOR de 4 entradas.

c) AND de 4 entradas.

d) OR de 4 entradas.

2. Implemente um comparador de palavras de 4 bits em que a saída seja 1 somente quando A ≠ B. Use 4 portas XOR e obtenha um diagrama com um total de 5 portas, como o da Figura 2. A porta de saída será uma:

a) NAND de 4 entradas.

b) NOR de 4 entradas.

c) AND de 4 entradas.

d) OR de 4 entradas.

3. Implemente um comparador de palavras de 4 bits em que a saída seja 1 somente quando A = B. Use apenas portas XOR de 2 entradas. O diagrama terá no mínimo:

a) 6 portas XOR.

b) 7 portas XOR.

c) 8 portas XOR.

d) NDA

4. No comparador de palavras de 2 bits do item 2.2 da parte experimental, se usássemos apenas portas AND de duas entradas, portas OR de duas entradas e NOT, teríamos um circuito com (suponha que as entradas possuam seus complementos disponíveis):

a) 7 portas AND, 4 portas OR e 4 NOT.

b) 9 portas AND, 3 portas OR e 2 NOT.

c) 10 portas AND, 4 portas OR e 2 NOT.

d) 11 portas AND, 4 portas OR e 2 NOT.

5. Utilizando-se somente portas XOR, podemos implementar portas:

a) NOT.

b) OR e NOT.

c) AND e NOT.

d) NDA