Josue Llamas,

Bibek Bhattarai

ECE 304

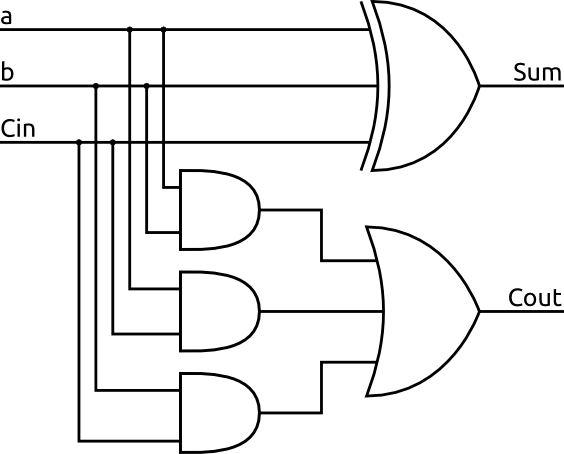
Dr. Chandra

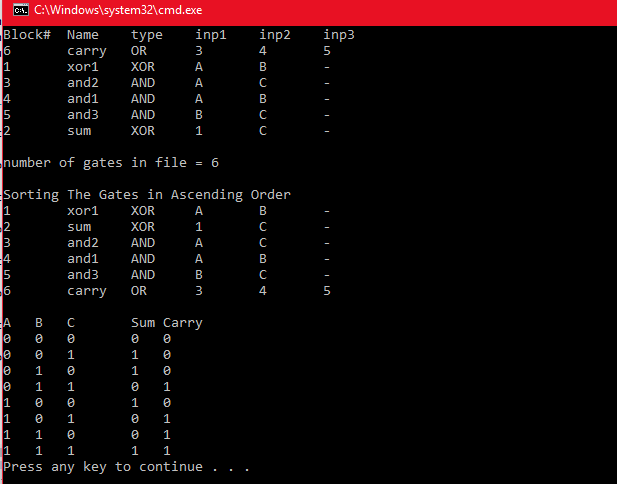
Logic Simulator

Question:

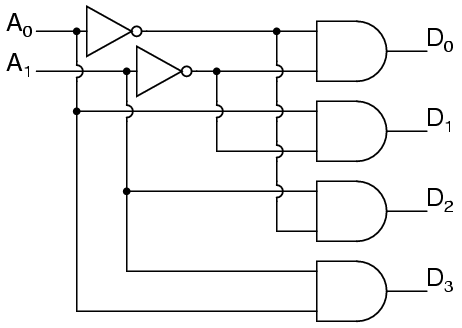
Output:

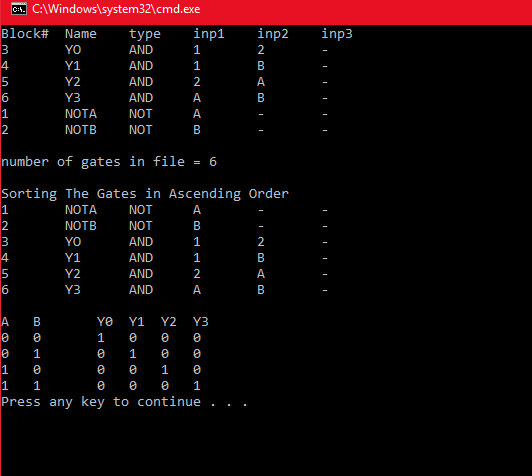
Full Adder





2 to 4 Decoder





Conclusion:

Code:

// LogicSim.cpp : Defines the entry point for the console application.

//

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

#include <cstdlib>

#include <algorithm>

#include <vector>

#include<stdlib.h>

using namespace std;

// declarations of constants

const int True = 1;

const int False = 0;

// Gate class

class lgate

{ public:

int blockx; // number of the block

string namex; // name of the block

string typex; // type of gate

char inp1x, inp2x,inp3x; // input names

int inp1, inp2, inp3; // input values

int out; // output of the gate

};

// funtion that does the logical operations

int LogOp(lgate &A)

{

if (A.typex == "XOR")

{

A.out = A.inp1 ^ A.inp2;

}

if (A.typex == "AND")

{

A.out = A.inp1 & A.inp2;

}

if (A.typex == "OR")

{

A.out = A.inp1 | A.inp2 |A.inp3;

}

if (A.typex == "NOT")

{

A.out = ~( A.inp1) ;

}

return A.out;

}

int main()

{

// ifstream constructor that opens a file

ifstream logicf; // insert logic file

logicf.open("FADDER.txt");

// if reading failes

if (logicf.fail())

{

cerr << "File could not open" << endl;

exit(1);

}

int i = 0;

int numGates = 0;

lgate Gate[6];

//cout whats written

cout << left << setw(8) << "Block#" << setw(8) << "Name" << setw(8) << "type" << setw(8) << "inp1" << setw(8) << "inp2" << setw(8) << "inp3" << setw(8)<< endl;

//Cout the contents of the file

while (logicf >> Gate[i].blockx >> Gate[i].namex >> Gate[i].typex >> Gate[i].inp1x >> Gate[i].inp2x >> Gate[i].inp3x)

{

cout << Gate[i].blockx << setw(8) << Gate[i].namex << setw(8) << Gate[i].typex << setw(8) << Gate[i].inp1x << setw(8) << Gate[i].inp2x << setw(8) << Gate[i].inp3x << setw(8) <<endl;

i++;

numGates++;

}

cout << "\nnumber of gates in file = "<< numGates << "\n" << endl;

// sorting the gates using an array

int Array[6];

int temp[6] ;

for (int i = 0; i < 6; i++)

{

Array[i] = Gate[i].blockx;

temp[i] = Gate[i].blockx;

}

sort(temp, temp + 6);

// Circuit Listing

cout << "Sorting The Gates in Ascending Order" << endl;

lgate Block[6];

for (i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

if (temp[i] == Array[j])

{

Block[i].blockx = Gate[j].blockx;

Block[i].namex = Gate[j].namex;

Block[i].typex = Gate[j].typex;

Block[i].inp1x = Gate[j].inp1x;

Block[i].inp2x = Gate[j].inp2x;

Block[i].inp3x = Gate[j].inp3x;

cout << setw(8);

cout << Block[i].blockx << setw(8) << Block[i].namex << setw(8) << Block[i].typex << setw(8) << Block[i].inp1x << setw(8) << Block[i].inp2x << setw(8) << Block[i].inp3x << setw(8) << endl;

}

}

}

// Solving the combinational circuit

const int in = 8;

const int in2 = 4;

//Decoder

//int b[in2] = { 0 ,1, 0, 1}; // input values

//int a[in2] = { 0, 0, 1, 1};

//int YO[in2];

//int Y1[in2], Y2[in2], Y3[in2];

//Full Adder

int c[in] = { 0 ,1, 0, 1, 0, 1, 0, 1 }; // input values

int b[in] = { 0, 0, 1, 1, 0, 0, 1, 1 };

int a[in] = { 0, 0, 0, 0, 1, 1, 1, 1 };

int Sum[in];

int Carry[in];

for (int k = 0; k < in; k++)

{

for (int i = 0; i < 6; i++)

{

if (Block[i].inp1x == \*"A")

{

Block[i].inp1 = a[k];

}

if (Block[i].inp2x == \*"A")

{

Block[i].inp2 = a[k];

}

if (Block[i].inp1x == \*"B")

{

Block[i].inp1 = b[k];

}

if (Block[i].inp2x == \*"B")

{

Block[i].inp2 = b[k];

}

// uncomment if third input

if (Block[i].inp1x == \*"C")

{

Block[i].inp1 = c[k];

}

if (Block[i].inp2x == \*"C")

{

Block[i].inp2 = c[k];

}

// for input one

if (Block[i].inp1x == \*"1")

{

Block[i].inp1 = LogOp(Block[0]);

}

if (Block[i].inp1x == \*"2")

{

Block[i].inp1 = LogOp(Block[1]);

}

if (Block[i].inp1x == \*"3")

{

Block[i].inp1 = LogOp(Block[2]);

}

if (Block[i].inp1x == \*"4")

{

Block[i].inp1 = LogOp(Block[3]);

}

if (Block[i].inp1x == \*"5")

{

Block[i].inp1 = LogOp(Block[4]);

}

// for input 2

if (Block[i].inp2x == \*"1")

{

Block[i].inp2 = LogOp(Block[0]);

}

if (Block[i].inp2x == \*"2")

{

Block[i].inp2 = LogOp(Block[1]);

}

if (Block[i].inp2x == \*"3")

{

Block[i].inp2 = LogOp(Block[2]);

}

if (Block[i].inp2x == \*"4")

{

Block[i].inp2 = LogOp(Block[3]);

}

if (Block[i].inp2x == \*"5")

{

Block[i].inp2 = LogOp(Block[4]);

}

// for input 3

if (Block[i].inp3x == \*"5")

{

Block[i].inp3 = LogOp(Block[4]);

}

}

//Full Adder

Sum[k] = LogOp(Block[1]);

Carry[k] = LogOp(Block[5]);

// Decoder

//YO[k] = LogOp(Block[2]);

//Y1[k] = LogOp(Block[3]);

//Y2[k] = LogOp(Block[4]);

//Y3[k] = LogOp(Block[5]);

}

// output the Truth Table

cout << endl;

cout << setw(4);

cout << "A" << setw(4) << "B" << setw(8) << "C" << setw(4) << "Sum" << setw(4) << "Carry" << setw(4) << endl; //Full Adder

//cout << "A" << setw(8) << "B" << setw(4) << "Y0" << setw(4) << "Y1" << setw(4) << "Y2" << setw(4) << "Y3"<< setw(4)<< endl; // Decoder

for (int m = 0; m < in; m++) // change in2 to in to 3inputs

{

cout << setw(4);

cout << a[m] << setw(4) << b[m] << setw(8) << c[m] << setw(4) << Sum[m] << setw(4) << Carry[m] << setw(4) << endl; //Full Adder

//cout << a[m] << setw(8) << b[m] << setw(4) << YO[m] << setw(4) << Y1[m] << setw(4) << Y2[m] << setw(4) << Y3[m] << setw(4) << endl; //Decoder

}

return 0;

}