Project One

Simulation of Carry Completion Adder

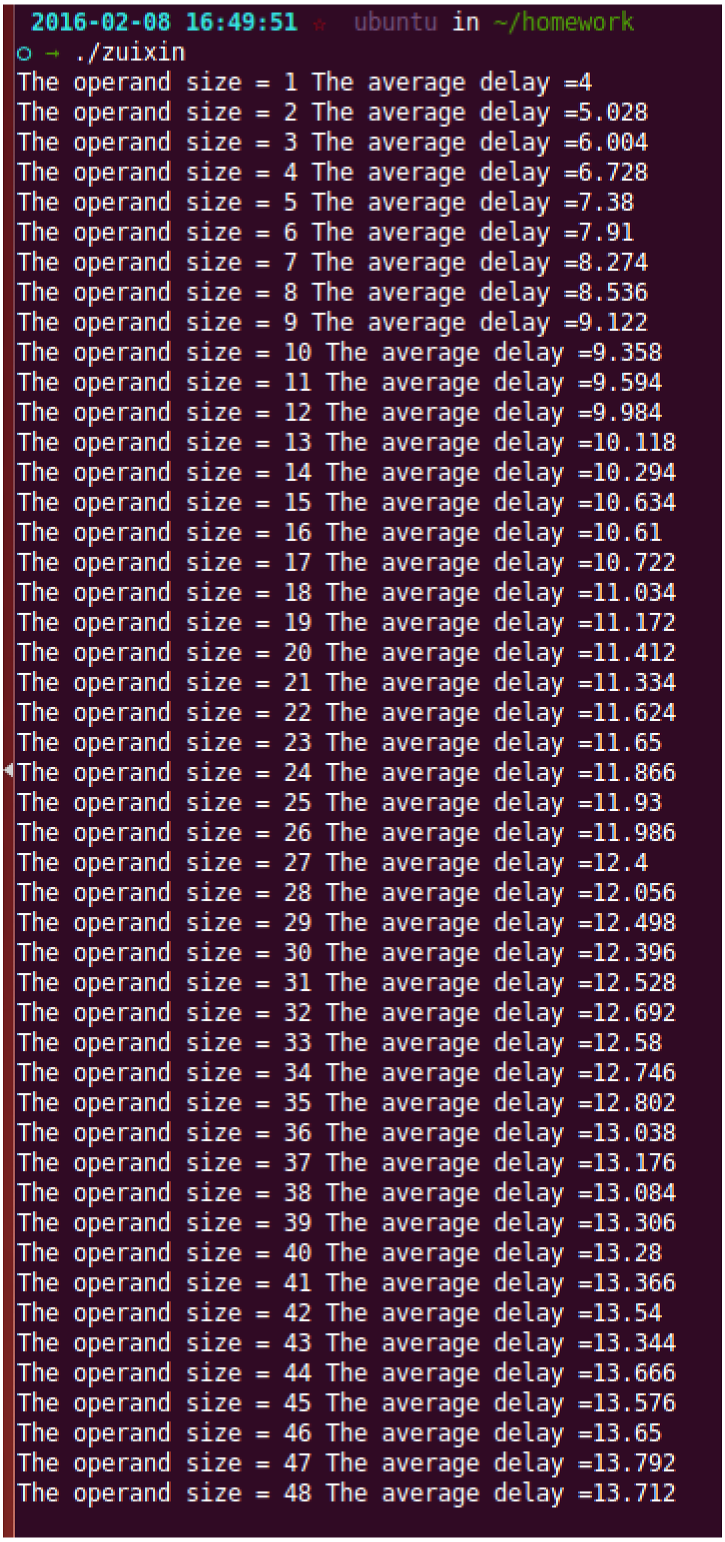
Name:Zhexin Zhang

Student ID:ipk693

1. The code for the simulation of CCA:

```
#include<stdio.h>
#include<cmath>
#include<time.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
#define TESTSET 1000
int main()
{
    //initialize the parameters
    int numA[48], numB[48], operandSize, carryIn1[48], carryIn0[48], carryOut0[48],
carryOut1[48], sum[48];
    int i, j, cycle, temp,temp1;
    bool done[48], CC;
    float delay[48];
    srand((unsigned)time(NULL));
    for (i = 1; i <= 48; i++) delay[i] = 0.0;//initialize delay for every operand size
    for (operandSize = 1; operandSize <= 48; operandSize++)//the first loop that changes
operand size
    {
        for (i = 0; i < TESTSET; i++)//the second loop to test 1000 times for each operand size
        {
            cycle = 0;//at the beginning of every test set,reset the clock cycle to 0;
            CC = false;
            for (j = 0; j < operandSize; j++)/*generate the two operands and initialize other
outputs to zero*/
            {
                numA[j] = rand() % 2;//generate a random binary number A
                numB[j] = rand() % 2;//generate a random binary number B
                carryIn0[j] = 0;
                carryIn1[j] = 0;
                carryOut0[j] = 0;
                carryOut1[j] = 0;
                sum[j] = 0;
            }
/*at the beginning of each test set,carryin0 on the first bit is always 1,that means the first bit
has no carry in.*/
            carryIn0[0] = 1;
            carryIn1[0] = 0;
            while (CC == false)//start the simulation
            {
                if (cycle != 0)
                {
                    CC = true;
                    temp1 = 1;
                    while ((CC == true)&&(temp1<operandSize))//every 2d delay,check if the
carries has been completed
                    {
                        for (j = 0; j<operandSize; j++)//set all the done for each bits in this
cycle
                        {
                            done[j] = ((carryOut0[j] || carryOut1[j]) ? true : false);
                        }
                        for (j = 0; j<operandSize; j++)//if one bit has not been finalized, set
CC to false, and break the loop
                        {
                            if (done[j] == false)
                            {
                                CC = false;
                                break;
                            }
                            temp1++;
                        }
                    }
                }
                if (CC == true) break;
                /*generate new sum,carryout0,carryout1 in this cycle*/
                for (j = 0; j<operandSize; j++)
                {
                    carryOut1[j] = ((numA[j] && numB[j]) || (carryIn1[j] && (numA[j] ^
numB[j])));
                    carryOut0[j] = ((!numA[j] && !numB[j]) || (carryIn0[j] && (numA[j] ^
numB[j])));
                    sum[j] = numA[j] ^ numB[j] ^ carryIn1[j];
                }
                for (j = 1; j<operandSize; j++)
                {
                    carryIn0[j] = carryOut0[j - 1];
                    carryIn1[j] = carryOut1[j - 1];
                }
                cycle++;//need 1 more cycle
            }
            temp = cycle * 2 + 2;
            delay[operandSize] += temp;
        }
        cout << "The operand size = " << operandSize;
        cout << "The average delay =" << (delay[operandSize] / TESTSET) << endl;
    }
}
```

2. The result of the simulation is shown as below:



```
 2016-02-08 16:49:51 ☆  ubuntu in ~/homework
○ → ./zuixin
The operand size = 1 The average delay =4
The operand size = 2 The average delay =5.028
The operand size = 3 The average delay =6.004
The operand size = 4 The average delay =6.728
The operand size = 5 The average delay =7.38
The operand size = 6 The average delay =7.91
The operand size = 7 The average delay =8.274
The operand size = 8 The average delay =8.536
The operand size = 9 The average delay =9.122
The operand size = 10 The average delay =9.358
The operand size = 11 The average delay =9.594
The operand size = 12 The average delay =9.984
The operand size = 13 The average delay =10.118
The operand size = 14 The average delay =10.294
The operand size = 15 The average delay =10.634
The operand size = 16 The average delay =10.61
The operand size = 17 The average delay =10.722
The operand size = 18 The average delay =11.034
The operand size = 19 The average delay =11.172
The operand size = 20 The average delay =11.412
The operand size = 21 The average delay =11.334
The operand size = 22 The average delay =11.624
The operand size = 23 The average delay =11.65
The operand size = 24 The average delay =11.866
The operand size = 25 The average delay =11.93
The operand size = 26 The average delay =11.986
The operand size = 27 The average delay =12.4
The operand size = 28 The average delay =12.056
The operand size = 29 The average delay =12.498
The operand size = 30 The average delay =12.396
The operand size = 31 The average delay =12.528
The operand size = 32 The average delay =12.692
The operand size = 33 The average delay =12.58
The operand size = 34 The average delay =12.746
The operand size = 35 The average delay =12.802
The operand size = 36 The average delay =13.038
The operand size = 37 The average delay =13.176
The operand size = 38 The average delay =13.084
The operand size = 39 The average delay =13.306
The operand size = 40 The average delay =13.28
The operand size = 41 The average delay =13.366
The operand size = 42 The average delay =13.54
The operand size = 43 The average delay =13.344
The operand size = 44 The average delay =13.666
The operand size = 45 The average delay =13.576
The operand size = 46 The average delay =13.65
The operand size = 47 The average delay =13.792
The operand size = 48 The average delay =13.712
```

Fig 1. Simulation output by C

3. The plot of the final results in d (gate delay) versus the operand size (n):

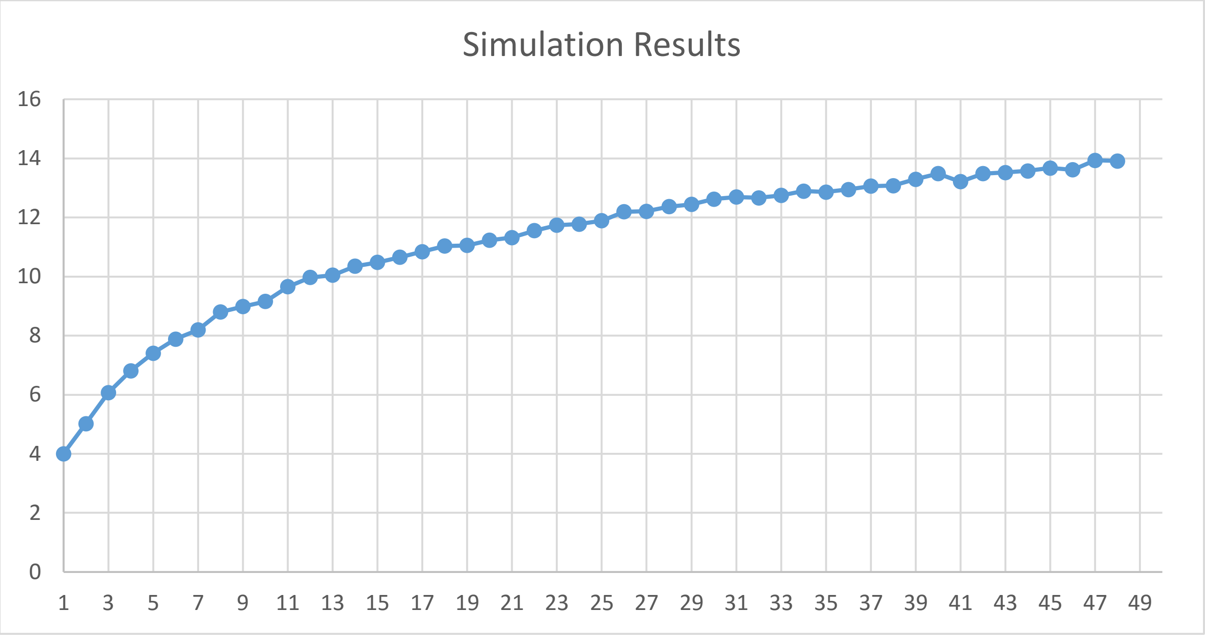
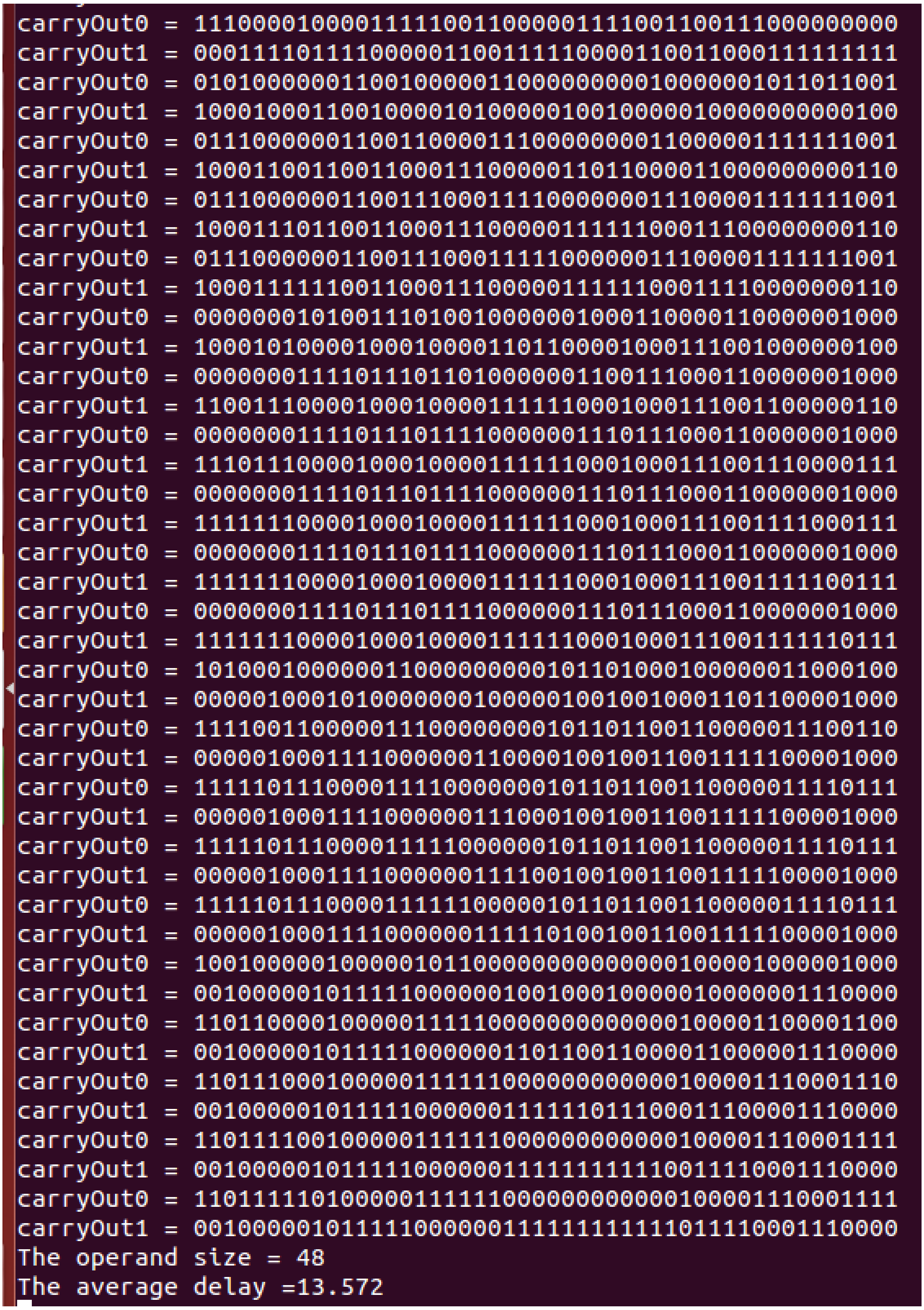


Fig 2. Graphic representation of Operand Size vs Gate Delay

4. The simulation results with a snapshot of all carryin 0 and carryin 1 at the end of each cycle is shown as bolow:



```
carryOut0 = 111000010000111110011000000111100110011100000000
carryOut1 = 000111101111000001100111110000011001100011111111
carryOut0 = 010100000011001000000110000000000100000001011011001
carryOut1 = 100010001100100001010000001001000000100000000000100
carryOut0 = 011100000011001100001110000000001100000111111001
carryOut1 = 100011001100110001110000001101100001100000000110
carryOut0 = 011100000011001110001111000000001110000111111001
carryOut1 = 100011101100110001110000001111110001110000000110
carryOut0 = 011100000011001110001111100000001110000111111001
carryOut1 = 100011111100110001110000001111110001111000000110
carryOut0 = 000000010100111010010000000100011000011000000001000
carryOut1 = 100010100001000100001101100001000111001000000100
carryOut0 = 000000011110111011010000000110011100011000000001000
carryOut1 = 110011100001000100001111110001000111001100000110
carryOut0 = 000000011110111011110000000111011100011000000001000
carryOut1 = 111011100001000100001111110001000111001110000111
carryOut0 = 000000011110111011110000000111011100011000000001000
carryOut1 = 111111100001000100001111110001000111001111000111
carryOut0 = 000000011110111011110000000111011100011000000001000
carryOut1 = 111111100001000100001111110001000111001111100111
carryOut0 = 000000011110111011110000000111011100011000000001000
carryOut1 = 111111100001000100001111110001000111001111110111
carryOut0 = 101000100000011000000000101101000100000011000100
carryOut1 = 000001000101000000001000001001001000110110000001000
carryOut0 = 111100110000011100000000010110110011000000111000110
carryOut1 = 000001000111100000001100001001001100111110000001000
carryOut0 = 111110111000011110000000010110110011000000111101111
carryOut1 = 000001000111100000001110001001001100111110000001000
carryOut0 = 111110111000011110000000010110110011000000111101111
carryOut1 = 000001000111100000001111001001001100111110000001000
carryOut0 = 111110111000011111000000101101100110000001111011 1
carryOut1 = 000001000111100000001111101001001100111110000001000
carryOut0 = 100100000010000001011000000000000000100001000001000
carryOut1 = 001000000101111100000001001000100000010000001110000
carryOut0 = 110110000100000011110000000000000000100001100001100
carryOut1 = 001000000101111100000011011001100001100000011100000
carryOut0 = 110111000100000011111000000000000000100001110001110
carryOut1 = 001000000101111100000011111101110001110000011100000
carryOut0 = 110111100100000011111000000000000000100001110001111
carryOut1 = 001000000101111100000011111111111001111000111000 0
carryOut0 = 110111110100000011111000000000000000100001110001111
carryOut1 = 001000000101111100000011111111111101111000111000 0
The operand size = 48
The average delay =13.572
```

Fig 3. Carryin and Carryout in each cycle

5. Code for print out this part is:

```
cout<<"carryOut0 = ";
```

```
for(j=0;j<operandSize;j++) cout<<carryOut0[j];
cout<<"\n";
cout<<"carryOut1 = ";
for(j=0;j<operandSize;j++) cout<<carryOut1[j];
cout<<"\n";
```

6. Conclusion:

From the results and figures above, we can find out that when the operand size is greater, the average gate delay and the operand size may has linear relationship. However, we had not taken enough test sets to test it, the more tests we take, the more accurate the result will be.