Project 2

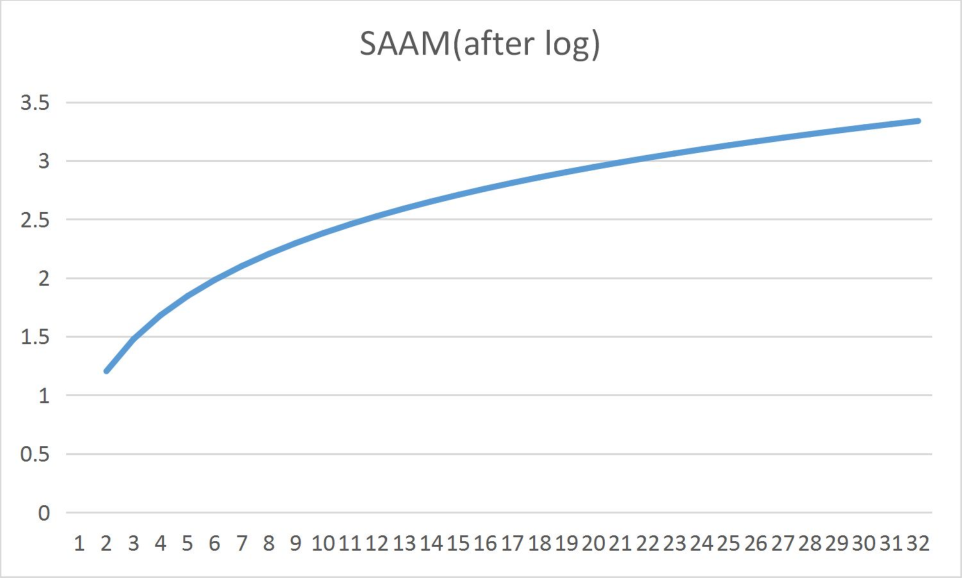
Name: Zhexin Zhang
Student ID:ipk693

1. **programs for part 1:**

```
#include <cstdlib>
#include <iostream>
#include <time.h>
#define TESTSET 10000
#define MAXBIT 32
using namespace std;
int main()
{
    cout<<"For shift and add multiplier,worst case delay=n*(2+2*n+2)d,n is the number of bit,both 1 bit shift and mux need 2d."<<endl;
    int n=2,t=0,delay[32],a[32],b[32],zero[32],carry[32],partial[32],c[64];
    srand((unsigned)time(NULL));
    for(n=2;n<=MAXBIT;n++)//different bit multiplication
    {
        for(t=0;t<TESTSET;t++)//for each bit status,we will have 10000 random testset
        {

            for(int i=0;i<n;i++)//generate random numbers a and b for multiplication
            {
                a[i]=rand()%2;
                b[i]=rand()%2;
                partial[i]=0;//initialize partial result to 0
                zero[i]=0;
            }
            for(int i=2;i<=32;i++)delay[i]=0;
            /*output a and b*/
            //cout<<"a=";
            //for(int j=n-1;0<=j;j--) cout<<a[j];
            //cout<<"\n"<<"b=";
            //for(int j=n-1;0<=j;j--) cout<<b[j];
            //cout<<"\n"<<"partial=";
            //for(int j=n-1;0<=j;j--) cout<<partial[j];
            //cout<<"\n";
            /*begin the calculation*/
            carry[0]=0;//initialize the first carryin to 0, every number such as a,b,partial,carry and c
here need to reverse output
            for(int i=0;i<2*n;i++) c[i]=0;//initialize final result to 0
            for(int count=0;count<n;count++)//first n-1 times shift,pick up b[0] to determine what
should we add
            {

                    //cout<<"Mux signal="<<b[0]<<endl;//last bit of b which we will use for
deciding to add A or 0
                    delay[n]=delay[n]+2;//2d for MUX
                    if(b[0]==0) for(int p=0;p<n;p++)
                        {
                            if(p==0)
                             {
                                carry[p]=0;
                                partial[p]=(zero[p]^partial[p]^0);

                            }
                            else
                            {

    carry[p]=(zero[p]*partial[p])||(zero[p]*carry[p-1])||(partial[p]*carry[p-1]);
                                partial[p]=(zero[p]^partial[p]^carry[p-1]);

                            }
                            delay[n]=delay[n]+2;
                        }

                    else if(b[0]==1) for(int p=0;p<n;p++)
                        {
                            if(p==0)
                             {
                                carry[p]=0;
                                partial[p]=(a[p]^partial[p]^0);
                            }
                            else
                            {

    carry[p]=(a[p]*partial[p])||(a[p]*carry[p-1])||(partial[p]*carry[p-1]);
                                partial[p]=(a[p]^partial[p]^carry[p-1]);
```

```
                                        }
                                        delay[n]=delay[n]+2;
                                }
                                c[count]=partial[0];//finalize 1 bit each cycle

                                /*shift 1 bit in register B and register partial result*/
                                for(int j=0;j<n-1;j++) b[j]=b[j+1];//register b shift 1 bit to generate the mux signal
                                b[n-1]=partial[0];//after shift,partial[0] will become the highest bit of b
                                for(int j=0;j<n-1;j++) partial[j]=partial[j+1];
                                partial[n-1]=carry[n-1];
                                delay[n]=delay[n]+2;//2d for shift



                        }
                        for(int z=0;z<n;z++) c[n+z]=partial[z];
                        /*print out the result*/
                        //cout<<"c=";
                        //for(int k=2*n-1;0<=k;k--) cout<<c[k];
                        //      cout<<"\n";
                }

                cout<<"n="<<n<<" delay="<<delay[n]<<"d."<<endl;
        }
}
```

2. **The result of program 1:**



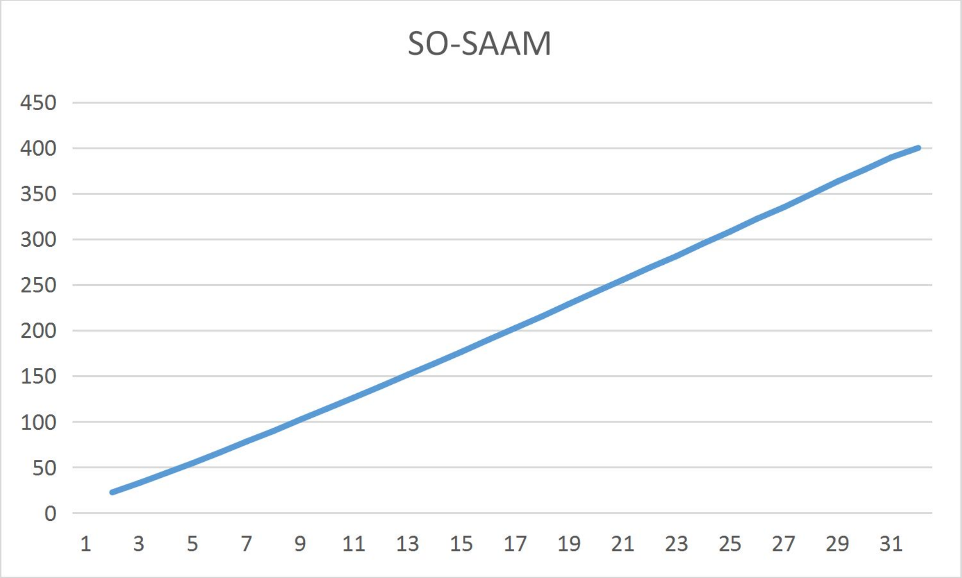
3. **Program for technique 2:**

```
#include <cstdlib>
#include <iostream>
#include <time.h>
#define TESTSET 10000
#define MAXBIT 32

using namespace std;

/*CCA 代码此处作为子程序以备调用*/
int CCA(int operandSize,int numA[33],int numB[33])
{
    int realoperandSize=operandSize+1;
    int carryIn1[realoperandSize], carryIn0[realoperandSize], carryOut0[realoperandSize],
carryOut1[realoperandSize],sum[33];
    int i, j, cycle=0,temp1,delayCCA;
    bool done[realoperandSize], CC=false;
    delayCCA=0;
    for (j = 0; j < realoperandSize; j++)/*generate the two operands and initialize other outputs to
zero*/
    {
        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<realoperandSize))/*every 2d delay,check if the carries has
been completed*/
            {
                for (j = 0; j<realoperandSize; j++)//set all the done for each bits in this cycle
                {
                    done[j] = ((carryOut0[j] || carryOut1[j]) ? true : false);
                }
                for (j = 0; j<realoperandSize; 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<realoperandSize; 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<realoperandSize; j++)
        {
            carryIn0[j] = carryOut0[j - 1];
            carryIn1[j] = carryOut1[j - 1];
        }
        cycle++;//need 1 more cycle
    }
    delayCCA=cycle * 2 + 2;
    //cout<<"delayCCA ="<<delayCCA<<endl;
    //cout<<"sum=";
    for(int k=realoperandSize-1;0<=k;k--)
    {
        numB[k]=sum[k];
        //cout<<numB[k];
    }
    return delayCCA;

}
int twosComp(int bits,int a[33],int A2[33])
{
    int one[33]={0},carrya[33]={0},A[33]={0};
    one[0]=1;
    int newBits=bits+1;
    for(int k=0;k<newBits;k++) A[k]=!a[k];
    //cout<<"1's compliment a=";
    //for (int k=bits;0<=k;k--) cout<<A[k];
    for(int k=0;k<newBits;k++)
        {
            if(k!=0)
            {
                carrya[k]=(A[k]*one[k])||(A[k]*carrya[k-1])||(carrya[k-1]*one[k]);
                A2[k]=A[k]^one[k]^carrya[k-1];
            }
            else if (k==0)
            {
                carrya[k]=A[k]*one[k];
                A2[k]=(A[k]^one[k]^0);
            }

        }
    //cout<<"\n2's compliment a=";
    //for (int k=bits;0<=k;k--) cout<<A2[k];
    //cout<<"\n";
}
int main()
{
    cout<<"For shift and add multiplier with shift over 0s and 1s,"<<endl;
    int
n=2,t=0,a[33]={0},b[33]={0},B,A[33]={0},partial[33]={0},result[66]={0},delayCCA,delay=0,lastbit=0;
    float Delay[32]={0};//to store the total delay for each bit lenth
    bool addSubtract;//signal to select add a or subtract a
    srand((unsigned)time(NULL));
    for(int n=2;n<=MAXBIT;n++)
    {
        for(int set=0;set<TESTSET;set++)
        {
```

```
for(int i=0;i<n+1;i++)//generate random numbers a and b for multiplication
    {
        a[i]=rand()%2;
        b[i]=rand()%2;
        partial[i]=0;//every testset need to reset them to 0
        result[i]=0;
    }
    a[n]=0;//pad a 0 to the highest bit of a
    b[n]=0;//pad a 0 to the highest bit of b
    delay=0;
    //cout<<"a=";
    //for(int p=n;0<=p;p--) cout<<a[p];
    //cout<<"\nb=";
    //for(int p=n;0<=p;p--) cout<<b[p];
    //cout<<"\n";
    twosComp(n,a,A);//generate 2's compliment of a,prepare for subtract a
    //cout<<"A=";
    //for(int k=n;0<=k;k--) cout<<A[k];
    //cout<<"\n";
    /*Every bit in b,it will go through MUX, add/subtract, and shift*/
    for(int count=0;count<n+1;count++)//for every bit of b,do the add and shift process
            {
        //cout<<"count="<<count<<endl;
                /*MUX*/
                {
                    if (count==0) B=b[0];
                    else B=(b[0]^lastbit);//set the select signal for mux
                    //cout<<"B="<<B<<endl;
                    delay=delay+2;//2d for MUX
                }
        /*add and shift over process*/
                {
                    if(B==0)//directly shift over
                    {

                        //cout<<"Shift over 1 bit"<<endl;
                        delayCCA=0;
                    }
                    else if (B==1)//add or subtract A
                    {
                        if(b[0]==0)//add A(that is for shifting 0s or 1s)
                        {
                            addSubtract=1;
                            delayCCA=CCA(n,a,partial);
                        //cout<<"after add a,partial=";
                        //for(int p=n;0<=p;p--) cout<<partial[p];
                        }
                        else if (b[0]==1)//subtract A
                        {
                            addSubtract=0;
                            delayCCA=CCA(n,A,partial);
                            //cout<<"after subtract a,partial=";
                            //for(int p=n;0<=p;p--) cout<<partial[p];
                        }
                    }
                        /*shift 1 bit*/
                        result[count]=partial[0];//finalize 1 bit each cycle
                        lastbit=b[0];
                        //cout<<"lastbit="<<lastbit<<endl;
                        /*shift 1 bit in register B and register partial result*/
                        for(int j=0;j<n;j++) b[j]=b[j+1];//register b shift 1 bit to generate the mux
signal
                        b[n]=partial[0];//after shift,partial[0] will become the highest bit of b

                    if(count!=n)//last cycle,partial don't shift any more,it's the higer part of the
final result.
                    {
                        for(int j=0;j<n;j++) partial[j]=partial[j+1];
                        partial[n]=partial[n-1];
                        /*cout<<"after shift,partial=";
                        for(int p=n;0<=p;p--) cout<<partial[p];*/
                        delay=delay+2+delayCCA;//2d for shift,and delay of CCA
                    }

                }
                    delay=delay+2;//another 2d for trigger next cycle;
            }
    for(int z=0;z<n+1;z++) result[n+z]=partial[z];
    /*cout<<"result=";
    for(int j=2*n-1;0<=j;j--) cout<<result[j];
            cout<<"\n";
```

```
        */Delay[n]=delay+Delay[n];
        }
                cout<<n<<" bit multiplier has an average Delay="<<Delay[n]/TESTSET<<endl;

        }
}
```

4. **Result of technique 2:**



5. **Program for testing x4b7 and x6c9:**

```
#include <cstdlib>
#include <iostream>
using namespace std;
#include <cstdlib>
#include <iostream>
#include <time.h>
#define TESTSET 10000
#define MAXBIT 32

using namespace std;

/*CCA 代码此处作为子程序以备调用*/
int CCA(int operandSize,int numA[33],int numB[33])
{
        int realoperandSize=operandSize+1;
        int carryIn1[realoperandSize], carryIn0[realoperandSize], carryOut0[realoperandSize], carryOut1[realoperandSize],sum[33];
        int i, j, cycle=0,temp1,delayCCA;
        bool done[realoperandSize], CC=false;
```

```
    delayCCA=0;
    for (j = 0; j < realoperandSize; j++)/*generate the two operands and initialize other outputs to zero*/
    {
        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<realoperandSize))/*every 2d delay,check if the carries has been completed*/
            {
                for (j = 0; j<realoperandSize; j++)//set all the done for each bits in this cycle
                {
                    done[j] = ((carryOut0[j] || carryOut1[j]) ? true : false);
                }
                for (j = 0; j<realoperandSize; 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<realoperandSize; 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<realoperandSize; j++)
        {
            carryIn0[j] = carryOut0[j - 1];
            carryIn1[j] = carryOut1[j - 1];
        }
        cycle++;//need 1 more cycle
    }
    delayCCA=cycle * 2 + 2;
    //cout<<"delayCCA ="<<delayCCA<<endl;
    //cout<<"sum=";
    for(int k=realoperandSize-1;0<=k;k--)
    {
        numB[k]=sum[k];
        //cout<<numB[k];
    }
    return delayCCA;

}

int twosComp(int bits,int a[33],int A2[33])
{
    int one[33]={0},carrya[33]={0},A[33]={0};
    one[0]=1;
    int newBits=bits+1;
    for(int k=0;k<newBits;k++) A[k]=!a[k];
    //cout<<"1's compliment a=";
    //for (int k=bits;0<=k;k--) cout<<A[k];
    for(int k=0;k<newBits;k++)
        {
            if(k!=0)
            {
                carrya[k]=(A[k]*one[k])||(A[k]*carrya[k-1])||(carrya[k-1]*one[k]);
                A2[k]=A[k]^one[k]^carrya[k-1];
            }
            else if (k==0)
            {
                carrya[k]=A[k]*one[k];
                A2[k]=(A[k]^one[k]^0);
            }

        }
    //cout<<"\n2's compliment a=";
    //for (int k=bits;0<=k;k--) cout<<A2[k];
    //cout<<"\n";
}

int main()
{
    int
n=12,t=0,a[33]={0},b[33]={0},B,A[33]={0},partial[33]={0},result[66]={0},delayCCA,delay=0,lastbit=0;
    float Delay[32]={0};//to store the total delay for each bit lenth
    bool addSubtract;//signal to select add a or subtract a
    cout<<"enter a:";
    for(int k=n-1;0<=k;k--) cin>>a[k];
    cout<<"\nenter b:";
    for(int k=n-1;0<=k;k--) cin>>b[k];
    a[n]=0;//pad a 0 to the highest bit of a
    b[n]=0;//pad a 0 to the highest bit of b
    delay=0;
    cout<<"a=";
    for(int p=n;0<=p;p--) cout<<a[p];
    cout<<"\nb=";
    for(int p=n;0<=p;p--) cout<<b[p];
    cout<<"\n";
    twosComp(n,a,A);//generate 2's compliment of a,prepare for subtract a
    cout<<"A=";
    //for(int k=n;0<=k;k--) cout<<A[k];
    cout<<"\n";
    /*Every bit in b,it will go through MUX, add/subtract, and shift*/
    for(int count=0;count<n+1;count++)//for every bit of b,do the add and shift process
            {
        cout<<"count="<<count<<endl;
```

```
                /*MUX*/
                {
                    if (count==0) B=b[0];
                    else B=(b[0]^lastbit);//set the select signal for mux
                    cout<<"B="<<B<<endl;
                    delay=delay+2;//2d for MUX
                }
        /*add and shift over process*/
                {
                    if(B==0)//directly shift over
                    {

                        cout<<"Shift over 1 bit"<<endl;
                        delayCCA=0;
                    }
                    else if (B==1)//add or subtract A
                    {
                        if(b[0]==0)//add A(that is for shifting 0s or 1s)
                        {
                            addSubtract=1;
                            delayCCA=CCA(n,a,partial);
                        cout<<"after add a,partial=";
                        for(int p=n;0<=p;p--) cout<<partial[p];
                        }
                        else if (b[0]==1)//subtract A
                        {
                            addSubtract=0;
                            delayCCA=CCA(n,A,partial);
                            cout<<"after subtract a,partial=";
                            for(int p=n;0<=p;p--) cout<<partial[p];
                        }
                    }
                        /*shift 1 bit*/
                        result[count]=partial[0];//finalize 1 bit each cycle
                        lastbit=b[0];
                        cout<<"lastbit="<<lastbit<<endl;
                        /*shift 1 bit in register B and register partial result*/
                        for(int j=0;j<n;j++) b[j]=b[j+1];//register b shift 1 bit to generate the mux signal

                        b[n]=partial[0];//after shift,partial[0] will become the highest bit of b
                    if(count!=n)//last cycle,partial don't shift any more,it's the higer part of the final result.
                    {
                        for(int j=0;j<n;j++) partial[j]=partial[j+1];
                        partial[n]=partial[n-1];
                        cout<<"after shift,partial=";
                        for(int p=n;0<=p;p--) cout<<partial[p];
                        delay=delay+2+delayCCA;//2d for shift,and delay of CCA
                    }

                }
                    delay=delay+2;//another 2d for trigger next cycle;
            }
    for(int z=0;z<n+1;z++) result[n+z]=partial[z];
    cout<<"result=";
    for(int j=2*n-1;0<=j;j--) cout<<result[j];
            cout<<"\n";
}
```

6. Result of test program:

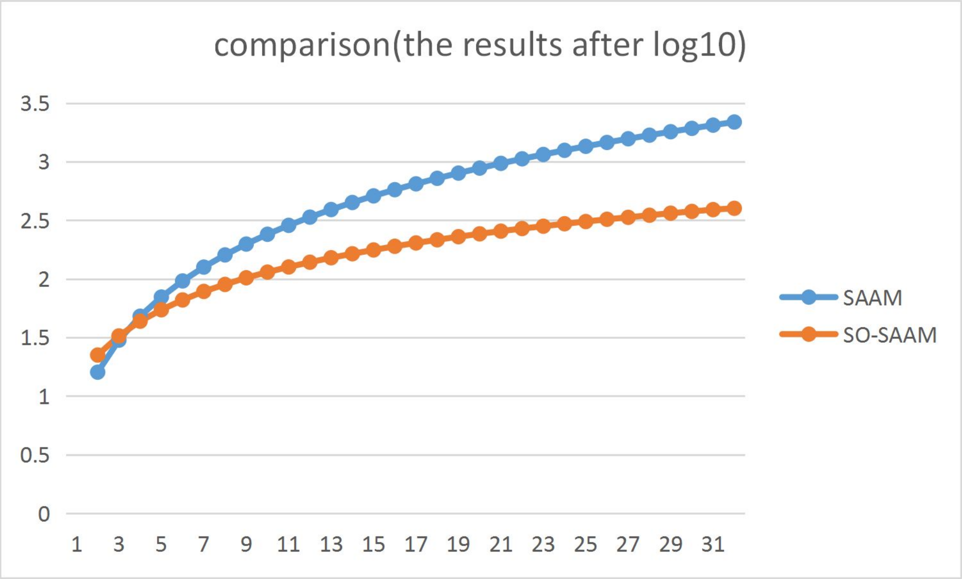
```
enter a:0 1 0 0 1 0 1 1 0 1 1 1
enter b:0 1 1 0 1 1 0 0 1 0 0 1
a=0010010110111
b=0011011001001
count=0
B=1
after subtract a,partial=1101101001001lastbit=1
after shift,partial=1110110100100count=1
B=1
after add a,partial=0001001011011lastbit=0
after shift,partial=0000100101101count=2
B=0
Shift over 1 bit
lastbit=0
after shift,partial=0000010010110count=3
B=1
after subtract a,partial=1101111011111lastbit=1
after shift,partial=1110111101111count=4
B=1
after add a,partial=0001010000110lastbit=0
after shift,partial=0000101000011count=5
B=0
```

```
Shift over 1 bit
lastbit=0
after shift,partial=0000010100001count=6
B=1
after subtract a,partial=1101111101010lastbit=1
after shift,partial=1110111110101count=7
B=0
Shift over 1 bit
lastbit=1
after shift,partial=1111011111010count=8
B=1
after add a,partial=0001110100001lastbit=0
after shift,partial=0000111010000count=9
B=1
after subtract a,partial=1110100011001lastbit=1
after shift,partial=1111010001100count=10
B=0
Shift over 1 bit
lastbit=1
after shift,partial=1111101000110count=11
B=1
after add a,partial=0001111111101lastbit=0
after shift,partial=0000111111110count=12
B=0
Shift over 1 bit
lastbit=0
result=000111111110101110101111
```

I am confused about this test actually, because every step the manipulation is correct, however, the result is not correct, I use my program test other numbers, it works perfect. I will figure it out later.

7. **Conclusion:**

   As we suspect before, shift over technique can lead to a faster multiplier than without shift over. And Skip Over one almost have a liner delay-bit relationship.


comparison(the results after log10)
3.5
3
2.5
2
1.5
1
0.5
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
SAAM
SO-SAAM