Computer Arithmetic Project 3

Vinay Chittam

Student Id: fet110

Aim: To find the efficiency of determining quotient bits by using less adders/subtractors for shift over 0’s and 1’s technique using comapritors [1/2,-1/2] and subtractors [d,-d]

Software used: C

Algorithm:

If we see 00, we shift and do not perform –D. And quotient becomes 0. If we see 11, we can shift over 1 and do not perform +D by keeping quotient as 1. This is the algorithm followed to program.

Code:

#include<stdio.h>

#include<conio.h>

#include<alloc.h>

#include<stdlib.h>

int \*sum,\*a,\*na,\*b,\*db,\*c0,\*c1,\*t0,\*t1,\*q,\*r,flag,n,afirst,alast,addercount;

float efficiency;

int\* comp2(int \*p,int last);

int\* ltshift(int \*p,int max);

void add();

void setB();

void setr();

void shiftra();

void adder(){

int i;

for(i=n-1;i>=0;i--){

sum[i]=a[i]^b[i]^c1[i+1];

//printf("%d ",sum[i]);

}

addercount++;

printf("\nadd counter %d",addercount);

getch();

}

void set(){ //setting t0 t1 from c0 c1

int j;

for(j=0;flag==1 && j<=n;j++){

t1[j]=c1[j];

t0[j]=c0[j];

}

}

void bits(int \*a,int \*b,int \*c0,int \*c1,int \*t0, int \*t1){ //co c1 t0 t1

int i,j,s,m;

for(i=n-1;i>=0;i--){

s=a[i]+b[i];

if(s==0){

c0[i]=1;

c1[i]=0;

}

else if(s>1){

c0[i]=0;

c1[i]=1;

}

else if(s==1){

if(t0[i+1]==t1[i+1]){

c0[i]=0;

c1[i]=0;

}

else{

m=s+t1[i+1];

if(m>1){

c0[i]=0;

c1[i]=1;

}

else if(m<=1){

c0[i]=1;

c1[i]=0;

}

}

}

}

}

void setc0c1(){

int i;

for(i=0;i<n;i++){

t1[i]=0;

t0[i]=0;

c1[i]=0;

c0[i]=0;

}

t1[n]=0;

t0[n]=1;

c1[n]=0;

c0[n]=1;

}

void seta(int f,int l){ //f inclusive and l exclusive

int i,j;

for(i=f,j=0;i<l;i++,j++){

a[j]=na[j];

}

}

void division(){

int k,i=0;afirst=i,alast=n;

seta(afirst,alast);

for(k=0;k<n;k++){

q[k]=9;

}

q[k]=9;

for(i=0;alast<n+n;i++){

if(a[0]==0 && a[1]==0){

int k;

q[i]=0;

a=ltshift(a,n);

afirst++;

alast++;

printf("\nshift over 0");

printf("\na :");

for(k=0;k<n;k++){

printf(" %d",a[k]);

}

printf("\n");

}

else if(a[0]==1 && a[1]==1){

int k;

q[i]=1;

a=ltshift(a,n);

afirst++;

alast++;

printf("\nshift over 1");

printf("\na :");

for(k=0;k<n;k++){

printf(" %d",a[k]);

}

printf("\n");

}

else if(a[0]==1 && a[1]==0){

int k;

setB();

printf("\na :");

for(k=0;k<n;k++){

printf(" %d",a[k]);

}

printf("\nd+:");

for(k=0;k<n;k++){

printf(" %d",b[k]);

}

printf("\n");

add();

setr();

shiftra();

if(r[0]==1)

q[i]=0;

else if(r[0]==0)

q[i]=1;

a=ltshift(a,n);

afirst++;

alast++;

}

else if(a[0]==0 && a[1]==1){

int k;

b=comp2(db,n);

printf("\na :");

for(k=0;k<n;k++){

printf(" %d",a[k]);

}

printf("\nd-:");

for(k=0;k<n;k++){

printf(" %d",b[k]);

}

printf("\n");

add();

setr();

shiftra();

if(a[0]==1)

q[i]=0;

else if(a[0]==0)

q[i]=1;

a=ltshift(a,n);

afirst++;

alast++;

}

}

}

void afterdiv(int \*a){

if(a[0]==1){

int k;

setB();

printf("\na :");

for(k=0;k<n;k++){

printf(" %d",a[k]);

}

printf("\nd+:");

for(k=0;k<n;k++){

printf(" %d",b[k]);

}

printf("\n");

add();

setr();

shiftra();

if(r[0]==1){

q[n]=0;

}

else if(r[0]==0){

q[n]=1;

}

}

else if(a[0]==0){

int k;

b=comp2(db,n);

printf("\na :");

for(k=0;k<n;k++){

printf(" %d",a[k]);

}

printf("\nd-:");

for(k=0;k<n;k++){

printf(" %d",b[k]);

}

printf("\n");

add();

setr();

shiftra();

if(a[0]==1)

q[n]=0;

else if(a[0]==0)

q[n]=1;

}

if(a[0]==1){

int k;

setB();

printf("\na :");

for(k=0;k<n;k++){

printf(" %d",a[k]);

}

printf("\nd+:");

for(k=0;k<n;k++){

printf(" %d",b[k]);

}

printf("\n");

add();

setr();

shiftra();

}

}

void main(){

int k,i,\*bc;

clrscr();

n=16,alast=n,afirst=0;

//

a=(int \*)malloc(n\*sizeof(int));

na=(int \*)malloc((n+n)\*sizeof(int));

b=(int \*)malloc(n\*sizeof(int));

db=(int \*)malloc(n\*sizeof(int));

//bc=(int \*)malloc(n\*sizeof(int));

c1=(int \*)malloc((n+1)\*sizeof(int));

c0=(int \*)malloc((n+1)\*sizeof(int));

t1=(int \*)malloc((n+1)\*sizeof(int));

t0=(int \*)malloc((n+1)\*sizeof(int));

sum=(int \*)malloc((n+1)\*sizeof(int));

q=(int \*)malloc((n+1)\*sizeof(int));

r=(int \*)malloc((n)\*sizeof(int));

//na[0]=0;na[1]=0;na[2]=0;na[3]=1;na[4]=1;na[5]=1;na[6]=1;na[7]=1;na[8]=0;na[9]=1;

//db[0]=0;db[1]=1;db[2]=0;db[3]=1;db[4]=1;

na[0]=0,na[1]=0,na[2]=0,na[3]=0,na[4]=1,na[5]=0,na[6]=0,na[7]=1,na[8]=1,na[9]=0,na[10]=1,na[11]=0,na[12]=0,na[13]=0,na[14]=0,na[15]=0,na[16]=1,na[17]=0,na[18]=1,na[19]=0,na[20]=1,na[21]=0,na[22]=1,na[23]=0,na[24]=1,na[25]=0,na[26]=1,na[27]=0,na[28]=1,na[29]=0,na[30]=1,na[31]=0;

db[0]=0,db[1]=1,db[2]=1,db[3]=1,db[4]=1,db[5]=0,db[6]=1,db[7]=1,db[8]=0,db[9]=0,db[10]=1,db[11]=0,db[12]=1,db[13]=0,db[14]=1,db[15]=0;

printf("\nnumerator :");

for(k=0;k<n+n;k++){

printf("%d ",na[k]);

} /\*

printf("\na :");

for(k=0;k<n;k++){

a[k]=na[k];

printf("%d ",a[k]);

}

printf("\ndenominator:");

for(k=0;k<n+n;k++){

printf("%d ",na[k]);

} \*/

printf("\ndenminator :");

for(k=0;k<n;k++){

printf("%d ",db[k]);

}

getch();

division();

afterdiv(a);

printf("\nna value :");

for(k=0;k<n+n;k++){

printf("%d ",na[k]);

}

printf("\nreminder is :");

for(k=0;k<n;k++){

printf("%d ",a[k]);

}

printf("\nquotient is :");

for(k=0;k<=n;k++){

printf("%d ",q[k]);

}

efficiency=(float)(n+1)/addercount;

printf("\nadder count is :%d",addercount);

printf("\nefficiency :%f",efficiency);

//

getch();

}

int\* comp2(int \*p,int last){

int i,carry=0,n=last;

int \*comp;

comp=(int \*)calloc(n,sizeof(int));

for(i=last-1;i>=0;i--){

if(p[i]==0){

comp[i]=1;

}

else if(p[i]==1){

comp[i]=0;

}

}

for(i=last-1;i>=0;i--){

if(i==last-1){

if(comp[i]==0){

comp[i]=1;

}

else if(c0[i]==1){

comp[i]=0;

carry=1;

}

}

else{

if(comp[i]==0 && carry==1){

comp[i]=1;

carry=0;

}

else if(c0[i]==1 && carry==1){

comp[i]=0;

carry=1;

}

}

}

return comp;

}

int\* ltshift(int \*p,int max){

int i;

for(i=0;i<max-1;i++){

p[i]=p[i+1];

}

if(alast<n+n){

p[i]=na[alast];

}

return p;

}

void setB(){

int k;

for(k=0;k<n;k++)

b[k]=db[k];

}

void setr(){

int k;

for(k=0;k<n;k++)

r[k]=sum[k];

}

void shiftra(){

int k;

for(k=0;k<n;k++)

a[k]=r[k];

}

void add(){

setc0c1();

do{ //addition process begins

int k;

bits(a,b,c0,c1,t0,t1);

flag=0;

for(k=0;k<=n;k++){

if(c0[k]==c1[k]){

flag=1;

set();

}

}

if(flag==0){

break;

}

else{

}

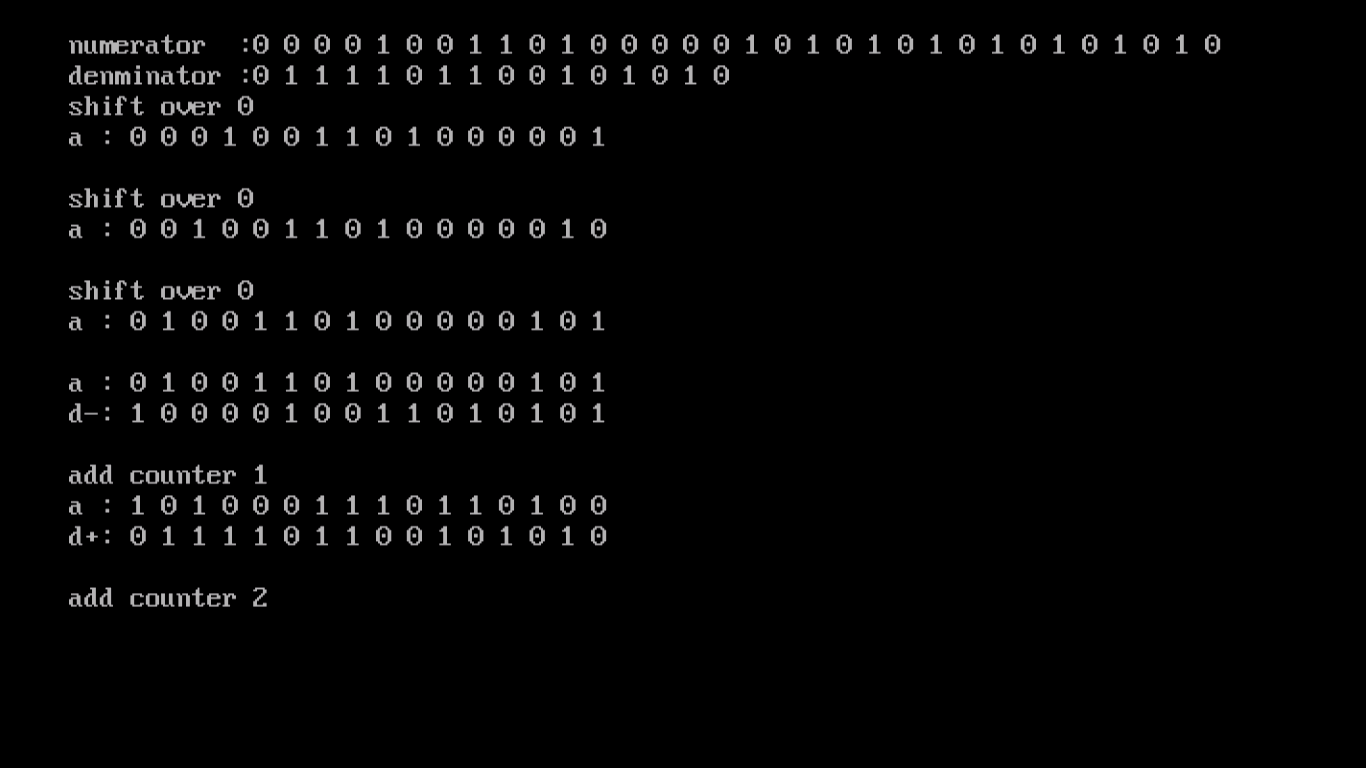
}

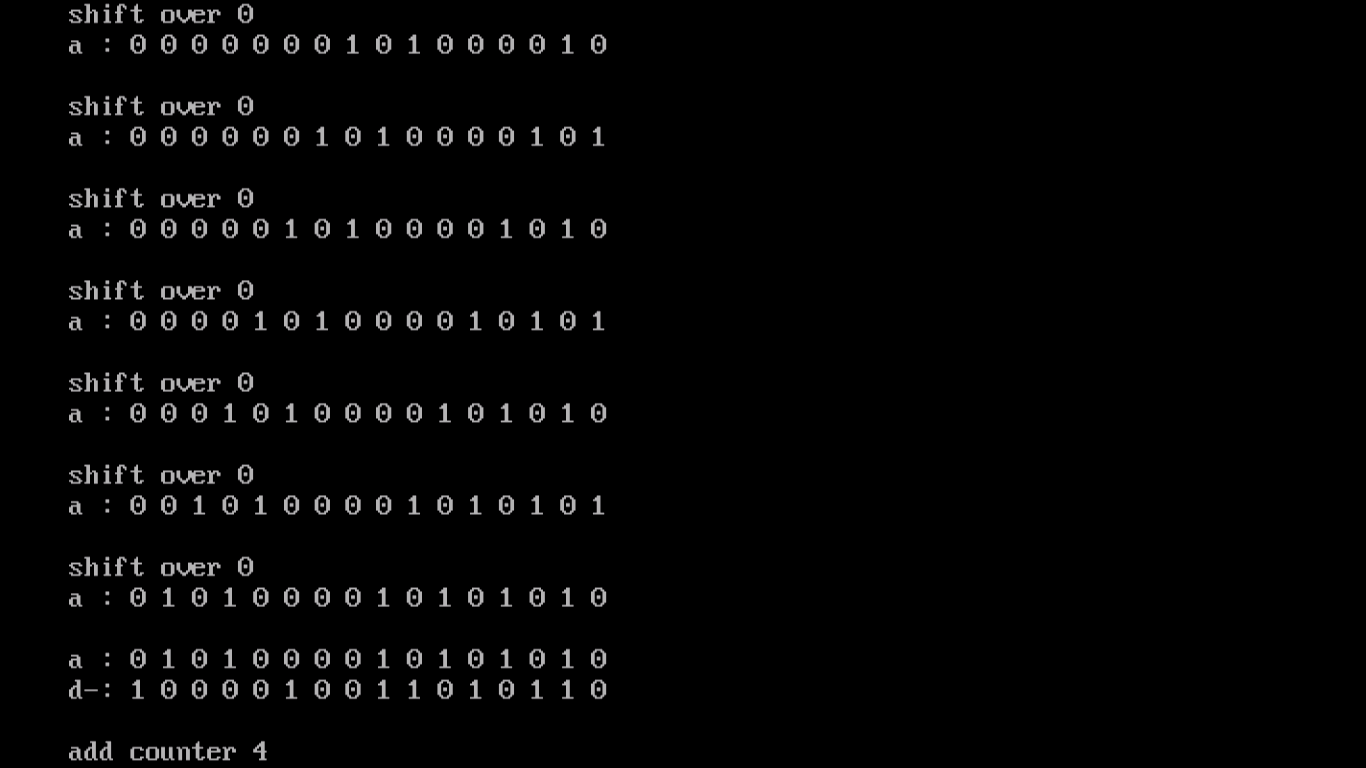
while(flag==1); //addition process ends

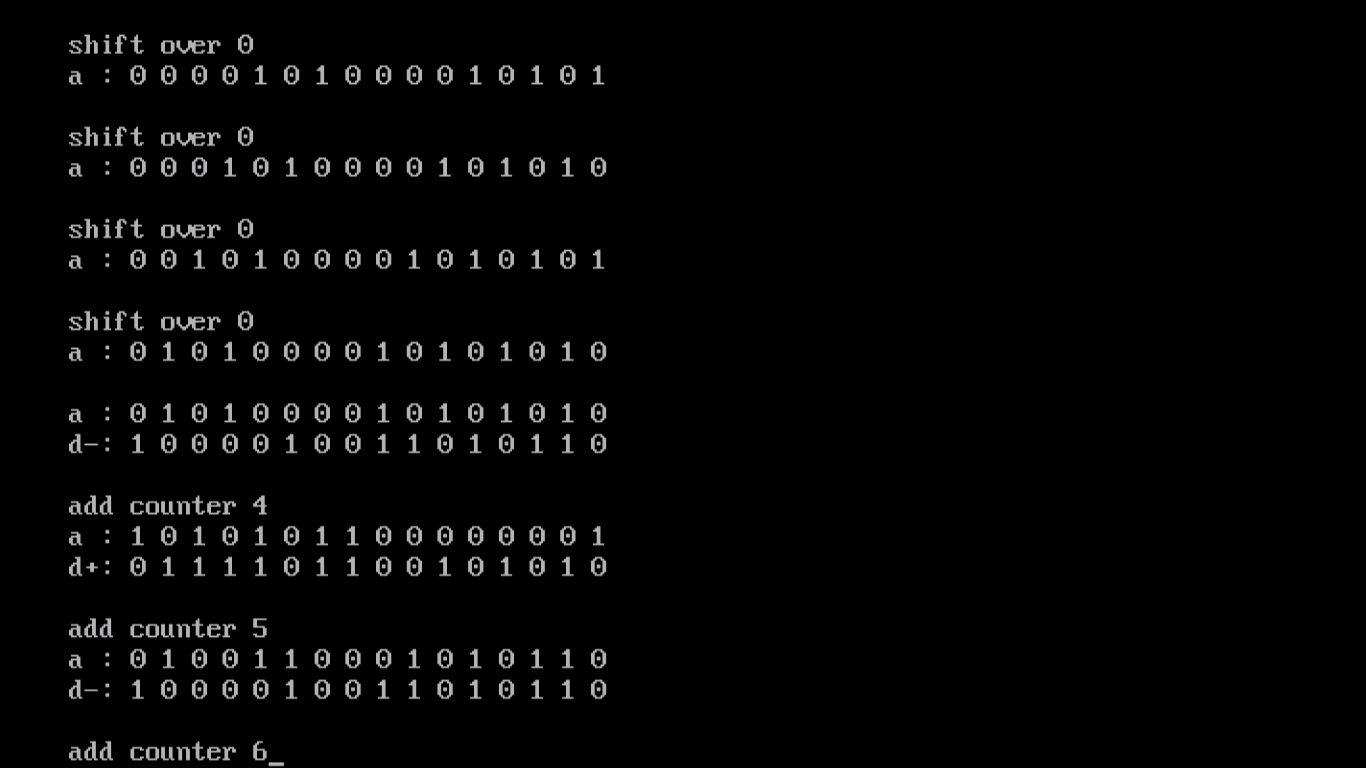
adder();

}

Simulation result:









Observation: For the given 16 bit divisor given in the question, I used the algorithm mentioned above and got the remainder as “0100110001010110” and quotient as “0001010000000010”.

And to obtain this 17 bit quotient I used 7 adders/ subtractors only. There the efficiency is 17/7=2.42857.

2. Doing the same process for 10,000 times for random bits when n is varying from 16,18,20….32 and finding the average efficiency.

I used the same algorithm and obtained the following result:

Code:

#include<stdio.h>

#include<conio.h>

#include<alloc.h>

#include<stdlib.h>

int \*sum,\*a,\*na,\*b,\*db,\*c0,\*c1,\*t0,\*t1,\*q,\*r,flag,n,afirst,alast,addercount;

float efficiency;

int\* comp2(int \*p,int last);

int\* ltshift(int \*p,int max);

void add();

void setB();

void setr();

void shiftra();

void adder(){

int i;

for(i=n-1;i>=0;i--){

sum[i]=a[i]^b[i]^c1[i+1];

//printf("%d ",sum[i]);

}

addercount++;

}

void set(){ //setting t0 t1 from c0 c1

int j;

for(j=0;flag==1 && j<=n;j++){

t1[j]=c1[j];

t0[j]=c0[j];

}

}

void bits(int \*a,int \*b,int \*c0,int \*c1,int \*t0, int \*t1){ //co c1 t0 t1

int i,j,s,m;

for(i=n-1;i>=0;i--){

s=a[i]+b[i];

if(s==0){

c0[i]=1;

c1[i]=0;

}

else if(s>1){

c0[i]=0;

c1[i]=1;

}

else if(s==1){

if(t0[i+1]==t1[i+1]){

c0[i]=0;

c1[i]=0;

}

else{

m=s+t1[i+1];

if(m>1){

c0[i]=0;

c1[i]=1;

}

else if(m<=1){

c0[i]=1;

c1[i]=0;

}

}

}

}

}

void setc0c1(){

int i;

for(i=0;i<n;i++){

t1[i]=0;

t0[i]=0;

c1[i]=0;

c0[i]=0;

}

t1[n]=0;

t0[n]=1;

c1[n]=0;

c0[n]=1;

}

void seta(int f,int l){ //f inclusive and l exclusive

int i,j;

for(i=f,j=0;i<l;i++,j++){

a[j]=na[j];

}

}

void division(){

int k,i=0;afirst=i,alast=n;

seta(afirst,alast);

for(k=0;k<n;k++){

q[k]=9;

}

q[k]=9;

for(i=0;alast<n+n;i++){

if(a[0]==0 && a[1]==0){

int k;

q[i]=0;

a=ltshift(a,n);

afirst++;

alast++;

}

else if(a[0]==1 && a[1]==1){

int k;

q[i]=1;

a=ltshift(a,n);

afirst++;

alast++;

}

else if(a[0]==1 && a[1]==0){

int k;

setB();

add();

setr();

shiftra();

if(r[0]==1)

q[i]=0;

else if(r[0]==0)

q[i]=1;

a=ltshift(a,n);

afirst++;

alast++;

}

else if(a[0]==0 && a[1]==1){

int k;

b=comp2(db,n);

add();

setr();

shiftra();

if(a[0]==1)

q[i]=0;

else if(a[0]==0)

q[i]=1;

a=ltshift(a,n);

afirst++;

alast++;

}

}

}

void afterdiv(int \*a){

if(a[0]==1){

int k;

setB();

add();

setr();

shiftra();

if(r[0]==1)

q[n]=0;

else if(r[0]==0)

q[n]=1;

}

else if(a[0]==0){

int k;

b=comp2(db,n);

add();

setr();

shiftra();

if(a[0]==1)

q[n]=0;

else if(a[0]==0)

q[n]=1;

}

if(a[0]==1){

setB();

add();

setr();

shiftra();

}

}

void main(){

int k,i,\*bc,j,l,loop=9;

long totaladcount=0;

float totalefficiency=0.0,averagecount=0.0,averageefficiency=0.0;

clrscr();

n=16,alast=n,afirst=0;

//

for(n=16;n<=32;n++){ // loop for 16 to 32

a=(int \*)malloc(n\*sizeof(int));

na=(int \*)malloc((n+n)\*sizeof(int));

b=(int \*)malloc(n\*sizeof(int));

db=(int \*)malloc(n\*sizeof(int));

//bc=(int \*)malloc(n\*sizeof(int));

c1=(int \*)malloc((n+1)\*sizeof(int));

c0=(int \*)malloc((n+1)\*sizeof(int));

t1=(int \*)malloc((n+1)\*sizeof(int));

t0=(int \*)malloc((n+1)\*sizeof(int));

sum=(int \*)malloc((n+1)\*sizeof(int));

q=(int \*)malloc((n+1)\*sizeof(int));

r=(int \*)malloc((n)\*sizeof(int));

//na[0]=0;na[1]=0;na[2]=0;na[3]=1;na[4]=1;na[5]=1;na[6]=1;na[7]=1;na[8]=0;na[9]=1;

//db[0]=0;db[1]=1;db[2]=0;db[3]=1;db[4]=1;

//na[0]=0,na[1]=0,na[2]=0,na[3]=0,na[4]=1,na[5]=0,na[6]=0,na[7]=1,na[8]=1,na[9]=0,na[10]=1,na[11]=0,na[12]=0,na[13]=0,na[14]=0,na[15]=0,na[16]=1,na[17]=0,na[18]=1,na[19]=0,na[20]=1,na[21]=0,na[22]=1,na[23]=0,na[24]=1,na[25]=0,na[26]=1,na[27]=0,na[28]=1,na[29]=0,na[30]=1,na[31]=0;

//db[0]=0,db[1]=1,db[2]=1,db[3]=1,db[4]=1,db[5]=0,db[6]=1,db[7]=1,db[8]=0,db[9]=0,db[10]=1,db[11]=0,db[12]=1,db[13]=0,db[14]=1,db[15]=0;

for(l=0;l<loop;l++){ //looping

addercount=0;

for(i=0;i<n+n;i++){

na[i]=((rand())%2);

//b[i]=((rand())%2);

}

for(i=0;i<n;i++){

db[i]=((rand())%2);

//b[i]=((rand())%2);

}

na[0]=0;

db[0]=0;

db[1]=1;

/\*

printf("\n%dth loop a value : ",n);

for(k=0;k<n;k++){

a[k]=na[k];

printf("%d ",a[k]);

}

printf("\n%dth loop na value :",n);

for(k=0;k<n+n;k++){

printf("%d ",na[k]);

}

printf("\n%dth loop db value :",n);

for(k=0;k<n;k++){

printf("%d ",db[k]);

} \*/

//getch();

division();

afterdiv(a);

/\*

printf("\nna value :");

for(k=0;k<n+n;k++){

printf("%d ",na[k]);

}

printf("\nreminder is :");

for(k=0;k<n;k++){

printf("%d ",a[k]);

}

printf("\nquotient is :");

for(k=0;k<=n;k++){

printf("%d ",q[k]);

} \*/

//printf("looping n value is :%d",n);

efficiency=(float)(n+1)/addercount;

totalefficiency+=efficiency;

totaladcount+=addercount;

//printf("\nadder count is :%d",addercount); long totaladcount=0;

averagecount=(float)totaladcount/loop;

averageefficiency=(float)totalefficiency/loop;

// float averagecount=0.0;

//printf("\nefficiency :%f",efficiency);

}// looping

//averagecount=(float)totaladcount/loop;

//averageefficiency=(float)totalefficiency/loop;

/\*

printf("\ntotal adder count for %d bit is:%ld ",n,totaladcount);

printf("average counter :%f",averagecount);

printf("\ntotal efficiency for %d bit is:%f ",n,totalefficiency);

printf("average efficiency is:%f\n",averageefficiency);

\*/

if(n%2==0)

printf("\naverage efficiency for %d bit is %f",n,efficiency);

getch();

} //loop for 16 to 32

getch();

}

int\* comp2(int \*p,int last){

int i,carry=0,n=last;

int \*comp;

comp=(int \*)calloc(n,sizeof(int));

for(i=last-1;i>=0;i--){

if(p[i]==0){

comp[i]=1;

}

else if(p[i]==1){

comp[i]=0;

}

}

for(i=last-1;i>=0;i--){

if(i==last-1){

if(comp[i]==0){

comp[i]=1;

}

else if(c0[i]==1){

comp[i]=0;

carry=1;

}

}

else{

if(comp[i]==0 && carry==1){

comp[i]=1;

carry=0;

}

else if(c0[i]==1 && carry==1){

comp[i]=0;

carry=1;

}

}

}

return comp;

}

int\* ltshift(int \*p,int max){

int i;

for(i=0;i<max-1;i++){

p[i]=p[i+1];

}

if(alast<n+n){

p[i]=na[alast];

}

return p;

}

void setB(){

int k;

for(k=0;k<n;k++)

b[k]=db[k];

}

void setr(){

int k;

for(k=0;k<n;k++)

r[k]=sum[k];

}

void shiftra(){

int k;

for(k=0;k<n;k++)

a[k]=r[k];

}

void add(){

setc0c1();

do{ //addition process begins

int k;

bits(a,b,c0,c1,t0,t1);

flag=0;

for(k=0;k<=n;k++){

if(c0[k]==c1[k]){

flag=1;

set();

}

}

if(flag==0){

break;

}

else{

}

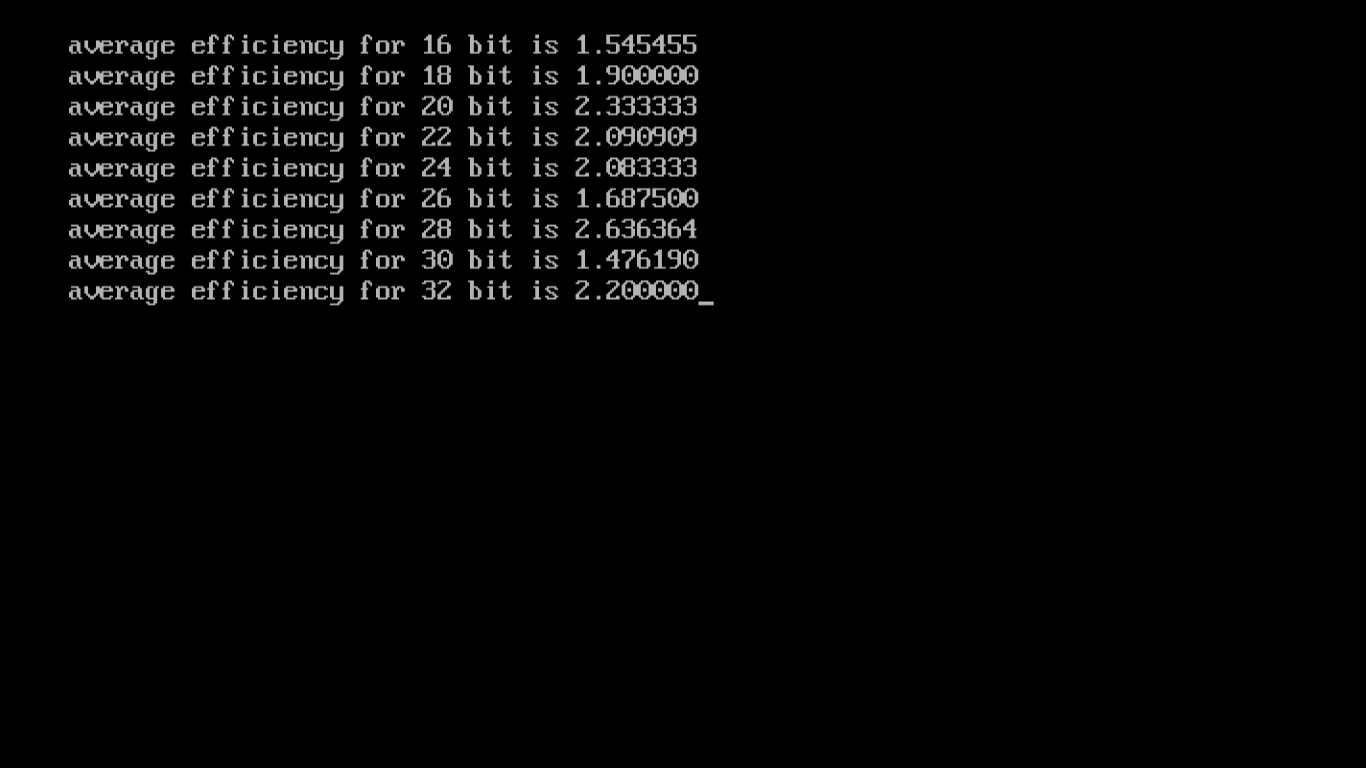
}

while(flag==1); //addition process ends

adder();

}

Simulation results:



Plot of efficiency and number of bits: