//program for synchronous multiplier

#include<stdio.h>

#include<stdlib.h>

int main()

{

int n\_bit; // number of bits.

int delay;

int total\_delay;

for(n\_bit=2;n\_bit<=32;n\_bit++)

{

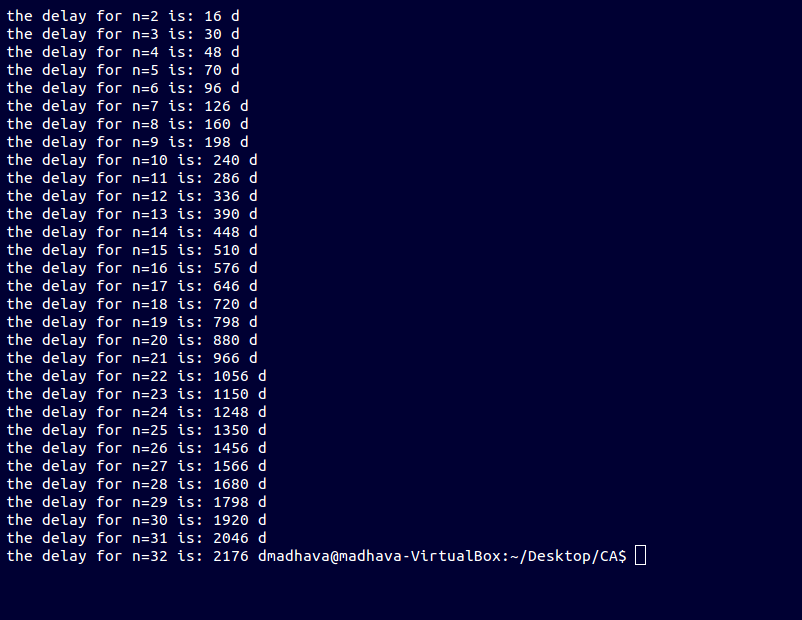
delay = 2+(2\*n\_bit)+2;

total\_delay = n\_bit\*delay;

printf("\n the delay for n=%d is: %d d",n\_bit,total\_delay,"\n");

}

}



//program for shift over 0 and 1 asynchronous multiplier

#include<stdio.h>

#include<stdlib.h>

int main()

{

int a[100],a\_prime[100],a\_neg[100],b[100],partial\_product[100],partial\_product\_inv[100],carry[100],c1[100],c0[100],c1\_next[100],c0\_next[100];

int n,test,delay,p,q,r,s,t,x,y,z,i,j,l;

int sum,average;

b[100]=0;

for(n=2;n<=32;n++)

{

sum=0;

for(test=1;test<=10000;test++)

{

for( p=0;p<=32;p++)

{

a[p]=0;a\_neg[p]=0;b[p]=0;partial\_product[p]=0;partial\_product\_inv[p]=0;c1[p]=0;c0[p]=0;c1\_next[p]=0;c0\_next[p]=0;

}

for(q=32;q>(32-n);q--)

{

a[q] = rand()%2;

b[q] = rand()%2;

a\_prime[q] = 1-a[q];

partial\_product[q]=0;

}

delay=0;

for(r=32;r>(32-n);r--)

{

delay=delay+2;

if((b[r]==0 && b[r+1]==0)||(b[r]==1 && b[r+1]==1))

{

for(s=0;s<=32;s++)

{

partial\_product[s]=partial\_product[s+1];

}

delay=delay+2;

delay=delay+2;

}

else if(b[r]==1 && b[r+1]==0)

{

for(t=1;t<=1;t++)

{

delay=delay+2;

c1[32]=0;

c0[32]=1;

for(x=31;x>=0;x--)

{

c0[x]=0;

c1[x]=0;

}

for(x=32;x>=0;x--)

{

partial\_product\_inv[x]=1-partial\_product[x];

a\_neg[x]=1-a\_prime[x];

}

repeat:

for(y=31;y>=0;y--)

{

c0\_next[y]=(a\_neg[y]&partial\_product\_inv[y])|(c0[y+1]&(a\_prime[y]^partial\_product[y]));

c1\_next[y]=(a\_prime[y]&partial\_product[y])|(c1[y+1]&(a\_prime[y]^partial\_product[y]));

}

for(z=31;z>=0;z--)

{

c0[z]=c0\_next[z];

c1[z]=c1\_next[z];

}

for(i=32;i>=0;i--)

{

if(c0[i]==0 & c1[i]==0)

{

delay=delay+2;

goto repeat;

}

}

carry[33]=0;

for(j=32;j>=0;j--)

{

carry[j]=(a\_prime[j]&partial\_product[j])|(a\_prime[j]&carry[j+1])|(b[j]&carry[j+1]);

partial\_product[j]=a\_prime[j]^partial\_product[j]^carry[j+1];

}

}

for(l=0;l<=32;l++)

{

partial\_product[l]=partial\_product[l+1];

}

delay=delay+2;

delay=delay+2;

}

else if(b[r]==0 && b[r+1]==1)

{

for(t=1;t<=1;t++)

{

delay=delay+2;

c1[32]=0;

c0[32]=1;

for(x=31;x>=0;x--)

{

c0[x]=0;

c1[x]=0;

}

for(x=32;x>=0;x--)

{

partial\_product\_inv[x]=1-partial\_product[x];

a\_neg[x]=1-a[x];

}

loop:

for(y=31;y>=0;y--)

{

c0\_next[y]=(a\_neg[y]&partial\_product\_inv[y])|(c0[y+1]&(a[y]^partial\_product[y]));

c1\_next[y]=(a[y]&partial\_product[y])|(c1[y+1]&(a[y]^partial\_product[y]));

}

for(z=31;z>=0;z--)

{

c0[z]=c0\_next[z];

c1[z]=c1\_next[z];

}

for(i=32;i>=0;i--)

{

if(c0[i]==0 & c1[i]==0)

{

delay=delay+2;

goto loop;

}

}

carry[33]=0;

for(j=32;j>=0;j--)

{

carry[j]=(a[j]&partial\_product[j])|(a[j]&carry[j+1])|(b[j]&carry[j+1]);

partial\_product[j]=a[j]^partial\_product[j]^carry[j+1];

}

}

for(l=0;l<=32;l++)

{

partial\_product[l]=partial\_product[l+1];

}

delay=delay+2;

delay=delay+2;

delay=delay+2;

}

}

sum=sum+delay;

}

//printf("\n the final sum delay of the multiplier for n= %d is %d d",n,sum,"\n");

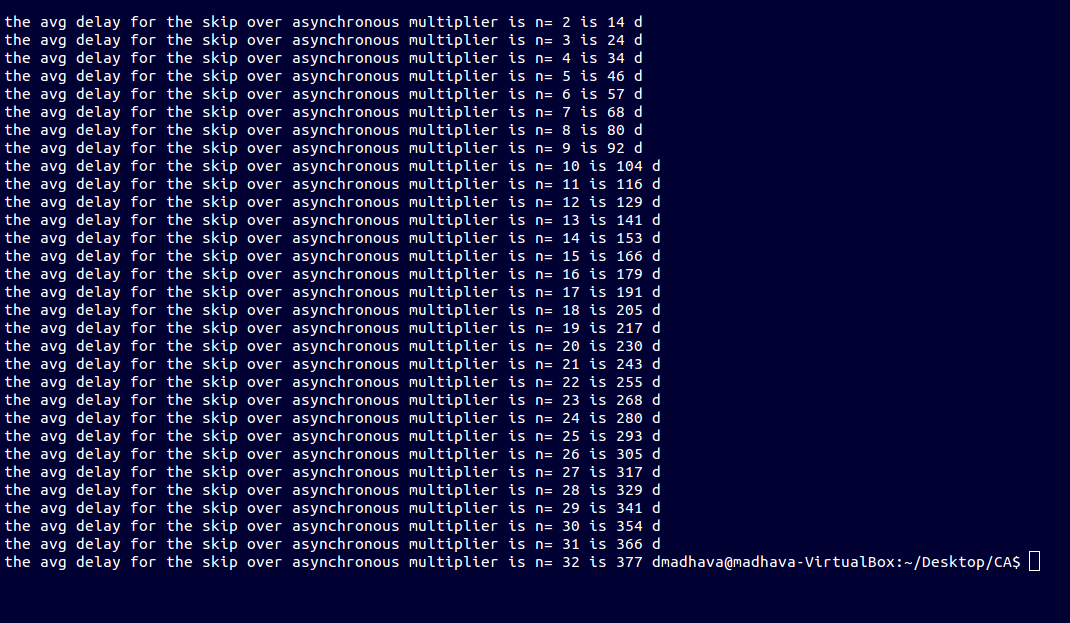
average=(sum/10000);

printf("\n the avg delay for the skip over asynchronous multiplier is n= %d is %d d",n,average,"\n");

}

return 0;

}



//program to multiply X4b7 and X6c9

#include<stdio.h>

#include<stdlib.h>

int main()

{

int a[33];int a\_prime[33];int a\_neg[33];int b[33];int partial\_product[33];int partial\_product\_inv[33];int carry[33];b[33]=0;int c0[33];int c1[33];int c0\_next[33];int c1\_next[33];

a[32]=1;a[31]=1;a[30]=1;a[29]=0;a[28]=1;a[27]=1;a[26]=0;a[25]=1;a[24]=0;a[23]=0;a[22]=1;a[21]=0;

b[32]=1;b[31]=0;b[30]=0;b[29]=1;b[28]=0;b[27]=0;b[26]=1;b[25]=1;b[24]=0;b[23]=1;b[22]=1;a[21]=0;

a\_prime[32]=1;a\_prime[31]=0;a\_prime[30]=0;a\_prime[29]=1;a\_prime[28]=0;a\_prime[27]=0;a\_prime[26]=1;a\_prime[25]=0;a\_prime[24]=1;a\_prime[23]=1;a\_prime[22]=0;a\_prime[21]=1;

for(int n=12;n<=12;n++)

{

for(int m=32;m>(32-n);m--)

{

partial\_product[m]=0;

}

int delay=0;

for(int i=32;i>(32-n);i--)

{

printf("\n\n\ni=%d",i);

printf("\nb[i]=%d,b[i+1]=%d",b[i],b[i+1]);

delay=delay+2;

printf("\n 2d for Multiplexer");

if((b[i]==0 && b[i+1]==0)||(b[i]==1 && b[i+1]==1))

{

for(int y=32;y<=21;y--)

{

partial\_product[y]=partial\_product[y+1];

}

delay=delay+2;

delay=delay+2;

}

else if((b[i]==1) & (b[i+1]==0))

{

for(int j=1;j<=1;j++)

{

delay=delay+2;

printf("\n 2d minimum delay for CCA");

c1[32]=0;

c0[32]=1;

for(int k=31;k>=0;k--)

{

c0[k]=0;

c1[k]=0;

}

for(int s=32;s>=0;s--)

{

partial\_product\_inv[s]=1-partial\_product[s];

a\_neg[s]=1-a\_prime[s];

}

loop:

for(int l=31;l>=0;l--)

{

c0\_next[l]=(a\_neg[l]&partial\_product\_inv[l])|(c0[l+1]&(a\_prime[l]^partial\_product[l]));

c1\_next[l]=(a\_prime[l]&partial\_product[l])|(c1[l+1]&(a\_prime[l]^partial\_product[l]));

}

for(int m=31;m>=0;m--)

{

c0[m]=c0\_next[m];

c1[m]=c1\_next[m];

}

for(int n=32;n>=0;n--)

{

if(c0[n]==0&c1[n]==0)

{

delay=delay+2;

printf("\n 2d delay for CCA(+A)");

goto loop;

}

}

carry[33]=0;

for(int m=32;m>=0;m--)

{

carry[m]=(a\_prime[m]&partial\_product[m])|(a\_prime[m]&carry[m+1])|(b[m]&carry[m+1]);

partial\_product[m]=a\_prime[m]^partial\_product[m]^carry[m+1];

}

}

for(int p=0;p<=32;p++)

{

partial\_product[p]=partial\_product[p+1];

}

delay+=2;

printf("\n 2d delay for shift");

delay+=2;

printf("\n 2d delay for triggering");

}

else if(b[i]==0 && b[i+1]==1)

{

{

for(int j=1;j<=1;j++)

{

delay=delay+2;

printf("\n 2d for CCA(-A)");

c1[32]=0;

c0[32]=1;

for(int k=31;k>=0;k--)

{

c0[k]=0;

c1[k]=0;

}

for(int k=32;k>=0;k--)

{

partial\_product\_inv[k]=1-partial\_product[k];

a\_neg[k]=1-a[k];

}

loop2:

for(int l=31;l>=0;l--)

{

c0\_next[l]=(a\_neg[l]&partial\_product\_inv[l])|(c0[l+1]&(a[l]^partial\_product[l]));

c1\_next[l]=(a[l]&partial\_product[l])|(c1[l+1]&(a[l]^partial\_product[l]));

}

for(int m=31;m>=0;m--)

{

c0[m]=c0\_next[m];

c1[m]=c1\_next[m];

}

for(int n=32;n>=0;n--)

{

if(c0[n]==0&c1[n]==0)

{

delay=delay+2;

printf("\n 2d delay for CCA carry not completed");

goto loop2;

}

}

carry[33]=0;

for(int m;m>=0;m--)

{

carry[m]=(a[m]&partial\_product[m])|(a[m]&carry[m+1])|(b[m]&carry[m+1]);

partial\_product[m]=a[m]^partial\_product[m]^carry[m+1];

}

}

for(int p=0;p<=32;p++)

{

partial\_product[p]=partial\_product[p+1];

}

delay+=2;

printf("\n 2d delay for shift");

delay+=2;

printf("\n 2d delay for triggering");

}

}

printf("\n the delay for skip over multiplier for n= %d is %d d",n,delay,"\n");

}

}

}

