**Computer Arthematic**

**Spring2015**

**Name:Vinay Chittam**

**Student ID:FET110**

**Project#1: CCA**

**Aim:** For an n-bit Carry Completion Adder, write a C program to determine

the relationship between the average delay and operand size (n).

**Procedure:** Consider the addition of two 4 bit numbers. A= 0 1 1 0 and B= 1 1 0 0. When the completion adder is performed, the following steps take place :

1. Like any other normal addition, the addition takes place from the right most bit. And every addition requires an incoming carry. The right most bits of A and B are 0,0. The incoming carry to these bits is 0. The incoming carry is always 0 for the addition of extreme right most bits. Since the incoming carry is 0, set c0 =1 and c1 =0( acts like a switch). Set all the other incoming carries to other bits as c1=0 and c0=0(switch is neither on nor off i.e not finalized). This entire cycle is “T0” which doesn’t have any delay.
2. Now the previous incoming carry for extreme right most bit which set in the last cycle is used by the extreme right most bit to generate a carry out. i.e 0+0+0 which again gives a c0=1 and c1=0(i.e there is 0 carry). The last but one bit i.e the 3rd bit has an incoming carry that is not finalized , therefore the carryout is not finalized as well. For the second bit i.e A=1 and B=1, the incoming carry is not finalzied, yet the carryout is c1=1 (because when the two bits are 1 and 1, no matter what the incoming carry is, the carryout will always be 1). The first bit has an incoming carry which is not finalized, and since we have A=0 and B=1, the carryout is not finalzed. This entire process is cycle “T1”
3. All the carryouts generated in cycle “T1” goes as carry in to the bits of A and B in cycle “T2”. Here we try to finalze the bits, which were not finalized in the cycle “T1”.
4. This process is continued till all the bits are finalized.
5. This process of addition can be simulated in C language. The code is given below.

**Softwares used for simmulation:** C and microsoft excell.

**C Program:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<alloc.h>

int k,\*a,\*b,\*c0,\*c1,\*t0,\*t1,i,j,n,s,m,l,td,flag;

long ttd; // variable to store delay

float avgd; // variable to store average delay

void set(){

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

t1[j]=c1[j];

t0[j]=c0[j];

} // Set function is used to store all the carryout bits in a temporary array, which acts as carry-in bits

}

void bits(int \*c0,int \*c1,int \*t0,int \*t1){

//// bit function checks for all conditions possible and generates carry-out bits

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 main(){

clrscr();

for(n=2;n<=48;n++){

ttd=0;

a=(int \*)malloc(n\*sizeof(int)); // creates memory for the array to store the variable no of bits

b=(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));

for(l=0;l<1000;l++){ //for looping 1000

td=2; //2 because we need to add 2d delay at the end when all the carries get finalized.

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

a[i]=((rand())%2); // random bits will be stored in a[i]

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

}

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

t1[i]=0; // initializing all the bits to 0

t0[i]=0; // initializing all the bits to 0

c1[i]=0;

c0[i]=0;

}

t1[n]=0; // finalizing the carry –in bit, because the incoming carry will always be 0 so t1[n]=0

t0[n]=1;

c1[n]=0;

c0[n]=1;

do{

bits(c0,c1,t0,t1); // control goes to the function “bits()” and executes the logic there and returns

flag=0; // flag=0 means the carry outs are finalized

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

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

flag=1; // carry outs are not finalized

set(); // calls the function “set()”

}

}

if(flag==0){

td+=2; // calculation of delay for each cycle when bits are finalzied

ttd+=td; // calculation of total delay

break;

}

else{

td+=2; //calculation of delay when bits are not finalized

}

}

while(flag==1);

}

avgd=(float)ttd/1000; // calculation of average delay

printf("\ntotal delay for %dth bit number is %ld and average delay is %f",n,ttd,avgd);

}

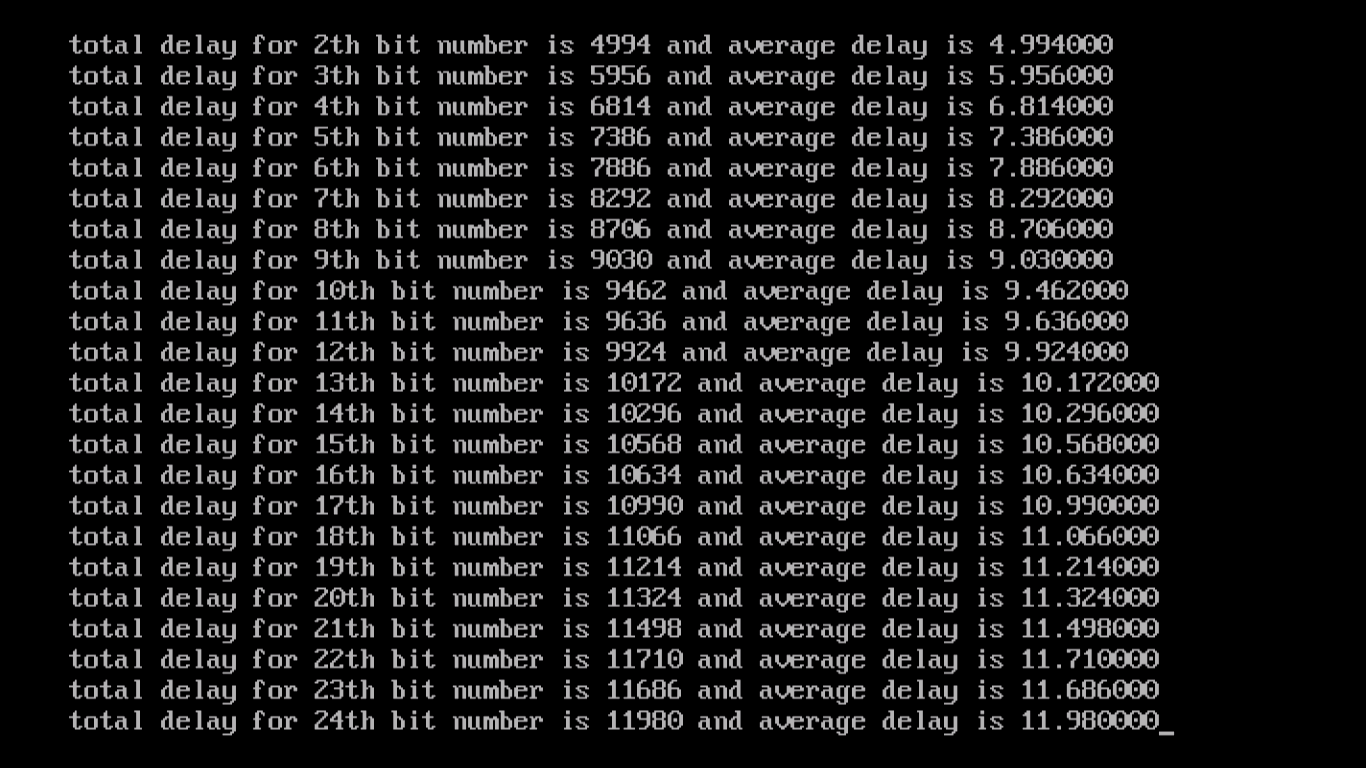
getch();

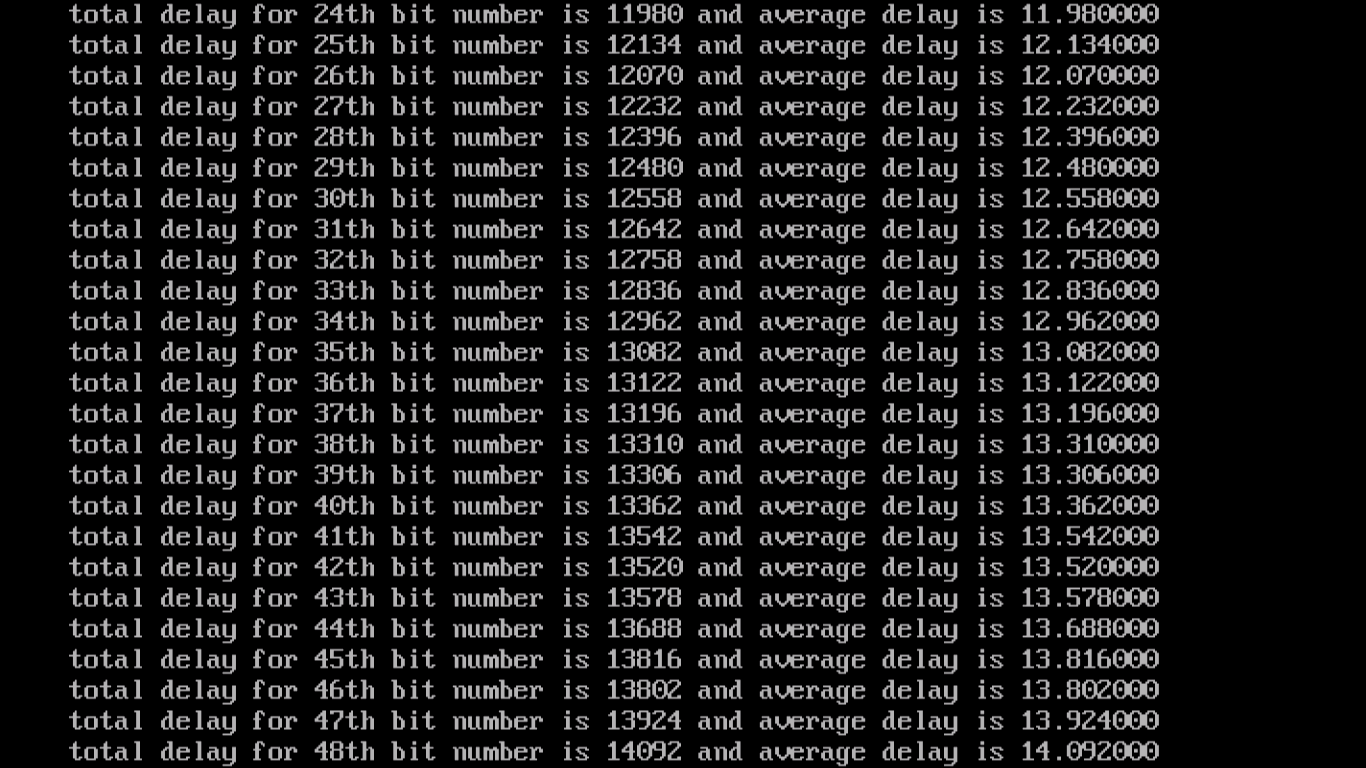
}

**Code Explanation:**

1. Since we are asked to add two numbers of variable bit size from 2 to 48, I created for loop from n=2 to n<=48.
2. Then I created 6 pointer variabes, a , b , c0, c1,T1,T0. (a and b hold the address of a memory location used to hold n\*size of int, which means if the number of bits 2, 4 bytes are allocated. Thus the size allocated depends on the value “n” ). Similarly c0 ,c1,T0,T1 hold (n+1 )\*size of int, which means c0 ,T1,T0, c1 hold the address of memory location that can store (n+1) \* size of int. For example if n=2, then c0, c1,T0,T1 will have a memory of 3\*2=6 bytes of memory.
3. We are required to take 1000 sets of random inputs to a and b. Therefore I created a for loop for(l=0;l<1000;l++) which iterates 1000 times and in each iteration a[i] and b[i] takes random input which are 1’s and 0’s. For this to happen, I made another for loop inside, which assigns random inputs to a[i] and b[i] using the random function “rand()%2”. This forloop ends here, while the forloop for 1000 iteration still runs.
4. Inside the 1000 iteration for loop, I again created a forloop to initialize all the c0[i], c1[i], T0[i] and T1[i] to 0. This for loop has a condition that i=0;i<n. So all the values less than n will be 0. Outside the forloop c0[n]=1,c1[n]=0,T0[n]=1 and T1[n]=0 will be initilized (since the carry in for last bit is 0).
5. Next I created a do while loop. The body of do{} contains a function called bits(). In this function the main logic lies. For any addition, the operation takes place from the right most bit, so I created a for loop “for(i=n-1;i>=0;i--)”, which starts from the last bit. The two last bits are added and stored in an array s[i](note that the addition is not xor, but it’s a normal addition), and in this for loop I gave an if condition stating that if s[i]=0, the c0[i]=1 and c1[i]=0, and if s[i]>1, c0[i]=0 and c1[i]=1. If s[i]=1, and T0[i+1] =T1[i+1], then c0[i]=c1[i]=0 i.e carry out is not finalized. (T0 and T1 are temporary bits which hold the carry in bits and c0 and c1 are the carry out bits). If s[i]+ T1[i+1]>1 then c0[i]=0 and c1[i]=1 and if s[i]+T1[i+1]<1 then c0[i]=1 and c1[i]=0. Thus all the posible conditions are chechked in this do loop.
6. To know if the carry out bits are finalized or not, I created a variable flag=0, which means says that the bits are finalized and if flag =1 then they are not finalized.The flag should become 1 when the two bits c1[k] and c0[k] are equal to 0. So I gave that condition in a for loop and if its met flag becomes 1. The do loop executes while flag=1 and doesn’t execute when flag=0.
7. Then I call a function calle set(). In set function I store the carry out bits in a temporary array, which act as carry -in bits for the bits in next cycle.
8. For one iteration i.e the 1000 iteration loop the delay for each cycle is calculated as 2d and added for all 1000 iterations. At the end of each iteration the calculated delay “td” is stored total time delay “ttd”. For example in the for loop i=0;i<1000; i++, for the first iteration i=0, the delay td is calculated. Say its 6d. Then for the second iteration i=1, the delay td is again calculated and added to the previous delay td and stored in ttd. and this continues for all the iterations. The final value is divided by 1000 to get the average delay for n bits. (Note that n is again varying, so this process repeats for each value of n). I have initialized td=2, because we need to add 2d after the last clock cycle to finalzie.

**Output of the code:**





**The output of program**

Total delay and average delay for bits varying from 2 to 48.

**Observation**: It can be observed that as the number of bits increases, the average delay increases. Operand size and average delay have a linear relationship. This can be shown clearly in the graph. I used the data and plotted the graph in Microsoft Excell.

**Figure showing the plot between Average Delay and Operand Size (n)**

**C program for n=24 bit pattern: A= 101001001100101101100101 and B=010101111000010010001011**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<alloc.h>

int \*sum,\*a,\*b,\*c0,\*c1,\*t0,\*t1,i,j,k,n,s,m,l,t,td,flag; // declaration of variables used in the program

long ttd; // variable to store total time delay

float avgd; // variable to store average delay

void adder(){

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

sum[i]=a[i]^b[i]^c1[i+1]; // XOR of a ,b and carry –in, to calculate the sum

}

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

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

}

void set(){

// Set function is used to store the carryout bits in one cycle in a temporary array which acts as a carry in for the bits in next clock cycle.

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

t1[j]=c1[j]; // As mentioned in the above comments carry out is being stored in temp array

t0[j]=c0[j];

}

}

void bits(){

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 main(){

clrscr();

n=24,t=0;

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

b=(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));

td=2; //Initializing to 2d, because at the last cycle, when all the carry bits get finalized we need to add 2d.

a[0]=1;a[1]=0;a[2]=1;a[3]=0;a[4]=0;a[5]=1;a[6]=0;a[7]=0;a[8]=1;a[9]=1;a[10]=0;a[11]=0;a[12]=1;a[13]=0;a[14]=1;a[15]=1;a[16]=0;a[17]=1;a[18]=1;a[19]=0;a[20]=0;a[21]=1;a[22]=0;a[23]=1;

b[0]=0;b[1]=1;b[2]=0;b[3]=1;b[4]=0;b[5]=1;b[6]=1;b[7]=1;b[8]=1;b[9]=0;b[10]=0;b[11]=0;b[12]=0;b[13]=1;b[14]=0;b[15]=0;b[16]=1;b[17]=0;b[18]=0;b[19]=0;b[20]=1;b[21]=0;b[22]=1;b[23]=1;

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;

printf("a :");

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

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

}

printf("\nb :");

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

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

}

printf("\n\n");

do{

printf("\nc1 values :");

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

printf("%d ",c1[k]);

}

printf("\nc0 values :");

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

printf("%d ",c0[k]);

}

printf("\nat cycle %d :",t);

printf("\n");

bits();

flag=0;

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

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

//printf("value at %d of c0 is %d and c1 is %d\n",j,c0[j],c1[j]);

flag=1;

set();

}

}

If(t=3){ //If this line and the next line “getch()” is commented, all the 8cycles will be printed, but the only the output that fits the screen will be displayed i.e first 3 cycles will not be displayed because it doesn’t fit on the screen, so in order to view even the first three cycles, the statements “if(t=3){getch()} should not be commented. If we wish to rest of the 8 cycles comment the statements “if(t=3){getch()}”

Getch();

}

if(flag==0){

t++;

td+=2;

ttd+=td;

break;

}

else{

t++;

td+=2;

}

}

while(flag==1);

avgd=(float)ttd/1000;

printf("\nfinal result c1: ");

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

printf("%d ",c1[i]);

}

printf("\nfinal result c0: ");

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

printf("%d ",c0[i]);

}

printf("\nat cycle %d ",t);

printf("final sum is :");

adder();

printf("\n");

printf("\ntotal delay for %dth bit number is %dd",n,td);

getch();

}

This program is same as the first program except that the loop n=2 to n=48 and the 1000 iteration loop have been removed and the input to A and B are not given randomly.

**Result:** The values of sum, Ci(0) and Ci(1) for each cycle are shown as follows.





**The output of c program for 24bit patter. It takes 18d to finalize all the carry out bits.**

**Conclusion:**

In each cycle some bits are finalized in parallel and some bits are not finalized. The bits that are finalized in a cycle are given as carry-in to A and B in the next cycle( This can be achieved by using a temporary array t0[i] and t1[i] ). This process continues till all the c0 and c1 bits are finalized. It takes 8 cycles to finalize all the bits in c0 and c1, and each cycle takes 2d delay. Therefore the total delay is 2d\*8= 16d and additional 2d, therefore 18d.

Although in each cycle some bits such as A=0 and B=0 or A=1 and B=1 parallelly generates the carry irrespective of the incoming carry it can be clearly noticed that when the number of bits “n” is more, it takes more cycles to finalize all the bits because when A=1 and B=0 or vice versa the incoming carry is to be finalized to generate a carryout.

Similarly when the number of bits is less, all the bits get finalized in very few cycles.