Name: SUYOG DHAKAL 075BCT092

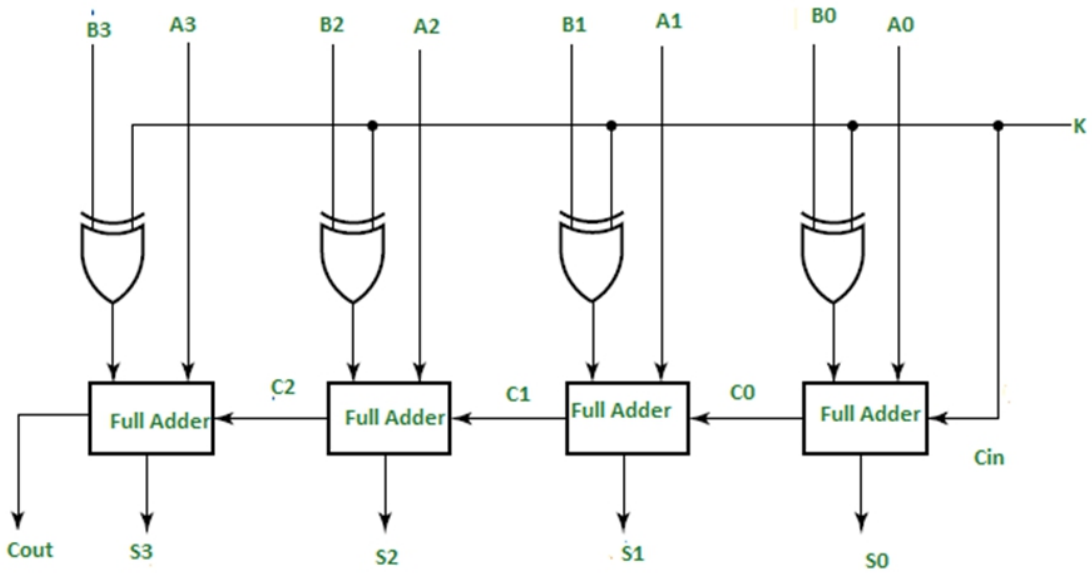
LAB2: SUBTRACTION OF TWO UNSIGNED INTEGER BINARY NUMBER

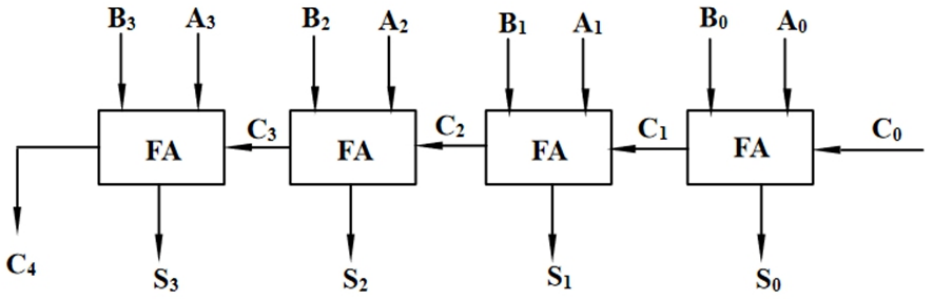
Objective:

To design n-bit(4bit) subtractor for unsigned integer binary numbers.

Theory:

Subtraction is the basic arithmetic operation. In digital computers complement is used for simplifying the subtraction. There are two types of compliments for base 2 system: 1's complement and 2's complement. The 1's complement of binary number is obtained by subtracting each digit from 1.However, the subtraction of a binary digit from 1 causes the bit to change from 0 to 1 or from 1to 0.Therefore, the 1's complement of a binary number is formed by changing 1's into 0's and 0's into 1's. For example 1's complement of 0111 is 1000. 2's complement can be formed by leaving all least significant 0's and first 1 unchanged, and then replacing 1's by 0's and 0's by 1's in all other higher significant bits.
This can be drawn as a circuit schematic as shown:





As shown in the figure, the first full adder has control line directly as its input(input carry Cin), The input A0 (The least significant bit of A) is directly input in the full adder. The third input is the xor of B0 and K. The two outputs produced are Difference and Carry.

Code:

AND.m

```
function var = AND(a,b)
  if(a==1 && b==1)
   var =1;
   else
    var=0;
   endif
endfunction
```

OR.m

```
function var = OR(a,b)
   if(a==0 && b==0)
    var = 0;
   else
    var = 1;
   endif
 endfunction
```

NOT.m

```
function var = NOT(a)
 if(a>=1)
   var=0;
 else
   var=1;
 endif
endfunction
```

XOR.m

```
function var = XOR(a,b)
 if(a!=b)
   var = 1;
 else
   var = 0;
  endif
endfunction
```

OnesComplement.m

```
function [ones_complement] = OneComplement(num)
 ones_complement = zeros(1,length(num));
 for i = length(num):-1:1
   ones_complement(i) = NOT(num(i));
 endfor
endfunction
```

TwosComplement.m

```
function [twos_complement] = TwosComplement(num)
 twos_complement = num;
 len = length(num);

 while(len>1 && num(len)==0)
   --len;
  endwhile

  if(len!=1)
   twos_complement(1:len-1) = OnesComplement(num(1:len-1));
  endif
endfunction
```

fulladder.m

```
function [sum,carry] = fulladder(a,b,c)
 sum = XOR(XOR(a,b),c);
 carry = OR(AND(a,b),AND(XOR(a,b),c));
endfunction
```

adder.m

```
function [sum,carry] = adder(num1,num2,sub=0)
 if(nargin<2)
  error("few arguments");
 endif
 if(!isvector(num1) && !isvector(num2))
  error("Requires vector arguments");
 endif
 i=length(num1);
 j=length(num2);
 if(i>j)
  gt=i;
 else
  gt=j;
 endif

 sum = zeros(1,gt);
 carry = sub;

 while(i>=1 && j>=1)
  [sum(gt), carry] = fulladder(num1(i--), XOR(num2(j--),sub),carry);
  --gt;
 endwhile

 while(i>=1)
  [sum(i),carry] = fulladder(num1(i--),XOR(0,sub),carry);
 endwhile

 while(j>=1)
  [sum(j), carry] = fulladder(0, XOR(num2(j--),sub),carry);
 endwhile

endfunction
```

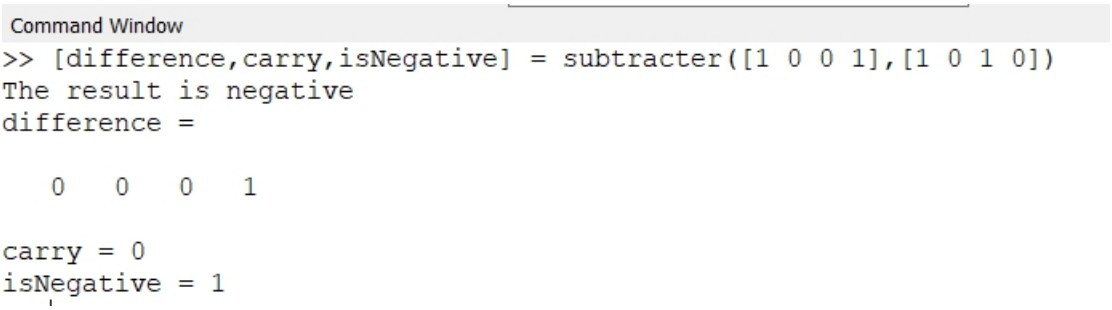
subtractor.m

```
function [difference,carry,isNegative] = subtracter(minuend, subtrahend)
  isNegative = 0;
  [difference,carry] = adder(minuend,subtrahend,1);

  #if there is no carry then the result is negative and the answer is
negative of 2's complement of the result

  if(!carry)
     isNegative = 1;
     difference = TwosComplement(difference);
     display("negative");
   endif
endfunction
```

OUTPUT:



Discussion and conclusion

Hence 4-bit binary subtractor can be designed using 4 full adders cascaded with one after another in series to perform 2’s compliment subtraction. The circuit implementation was visualized using functions and variables. At the fundamental level, the circuit consists of basic logic gates like AND, OR, XOR which were also implemented in code. It uses the full adder circuit for the addition. So the binary subtractor was built on top of the full adder circuit while the full adder circuit was built on top of the basic gates.