Digital Verilog HDL Project

Name: Nafe Abubaker

ID : 1200047

Section: 5

Description

Implementing a BCD Adder-Subtractor circuit :

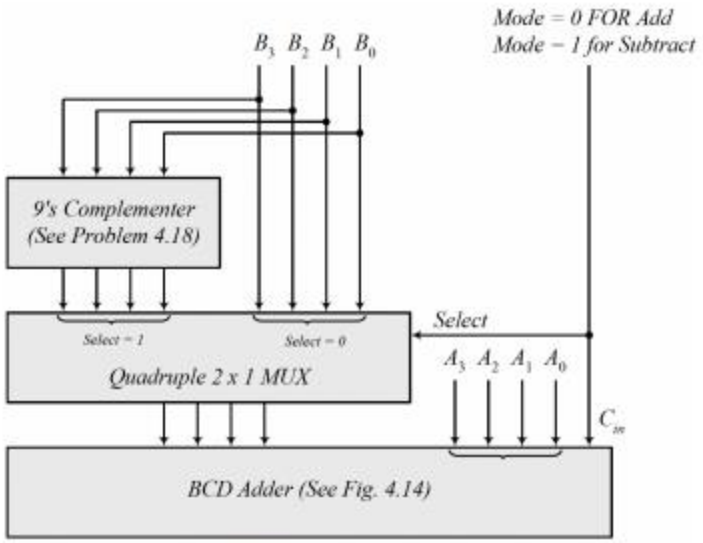


Figure 1 BCD adder-subtractor circuit

First we started with the 9's Complementer :

1) **Code :**

First we used the truth table (found at the end of the report), and after minimization we came up with several equatios.

1)A= w'x'y'

2)B=x'y + xy' (Which is XOR)

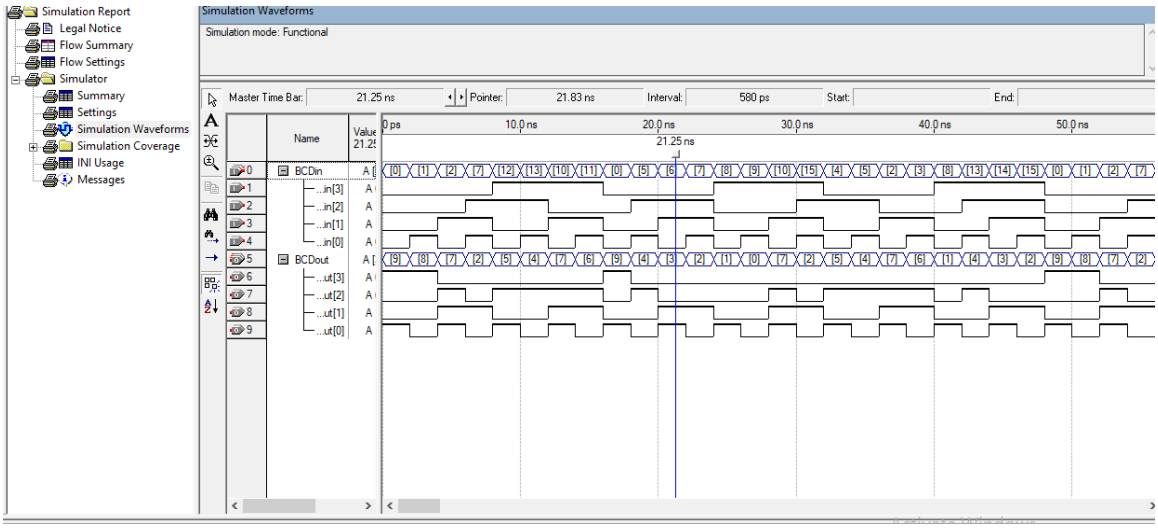
3)C= Y

4)D=z'

Then used dataflow to implement the found equations and came up with this code:

```
module Complement_9(BCDin,BCDout);


 input [3:0]BCDin;
 output [3:0]BCDout;

 //BCDin 3 = a
 //BCDin 2 = b
 //BCDin 1 = c
 //BCDin 0 = d
 //--------------
 //BCDout 3 = w
 //BCDout 2 = x
 //BCDout 1 = y
 //BCDout 0 = z

 assign BCDout[3] = ((~(BCDin[3])) & (~(BCDin[2])) & (~(BCDin[1])));
 assign BCDout[2] = BCDin[2] ^ BCDin[1];
 assign BCDout[1] = BCDin[1];
 assign BCDout[0] = ~BCDin[0];

 endmodule
```

2) Simulation:



Quad 2x1 Mux :

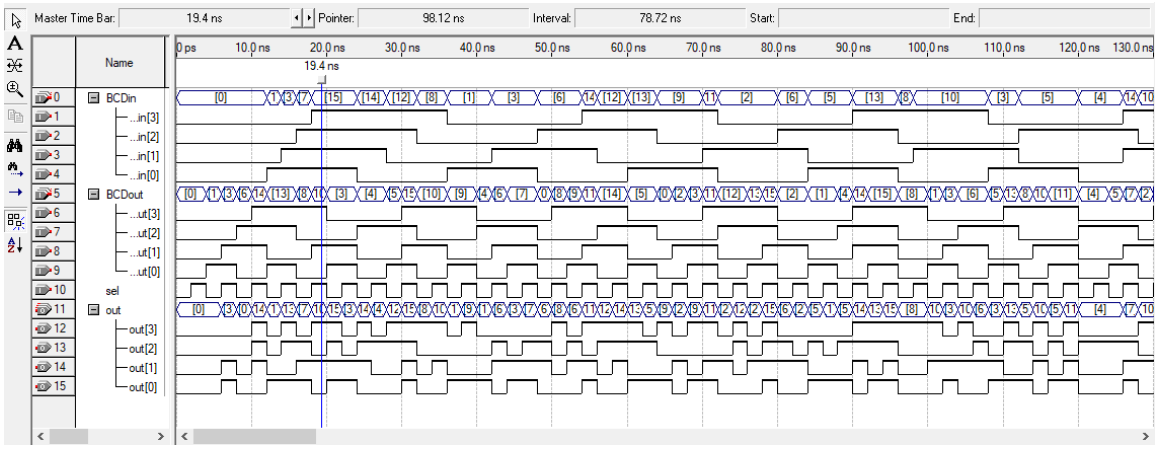
We used behavioral to implement 2x1 mux and came up with this code which will set the output to the first 4 bits if the selection equals zero, and will set the output to the

Next 4 bits if the selection equals one.

1) Code

```
1  module QuadMux(input [3:0]BCDout,BCDin,input sel,output reg [3:0]out);
2
3  always @(BCDout,BCDin,sel) begin
4   if(sel == 0) out = BCDin;
5
6   else out = BCDout;
7
8   end
9
10 endmodule
```

2) Simulation

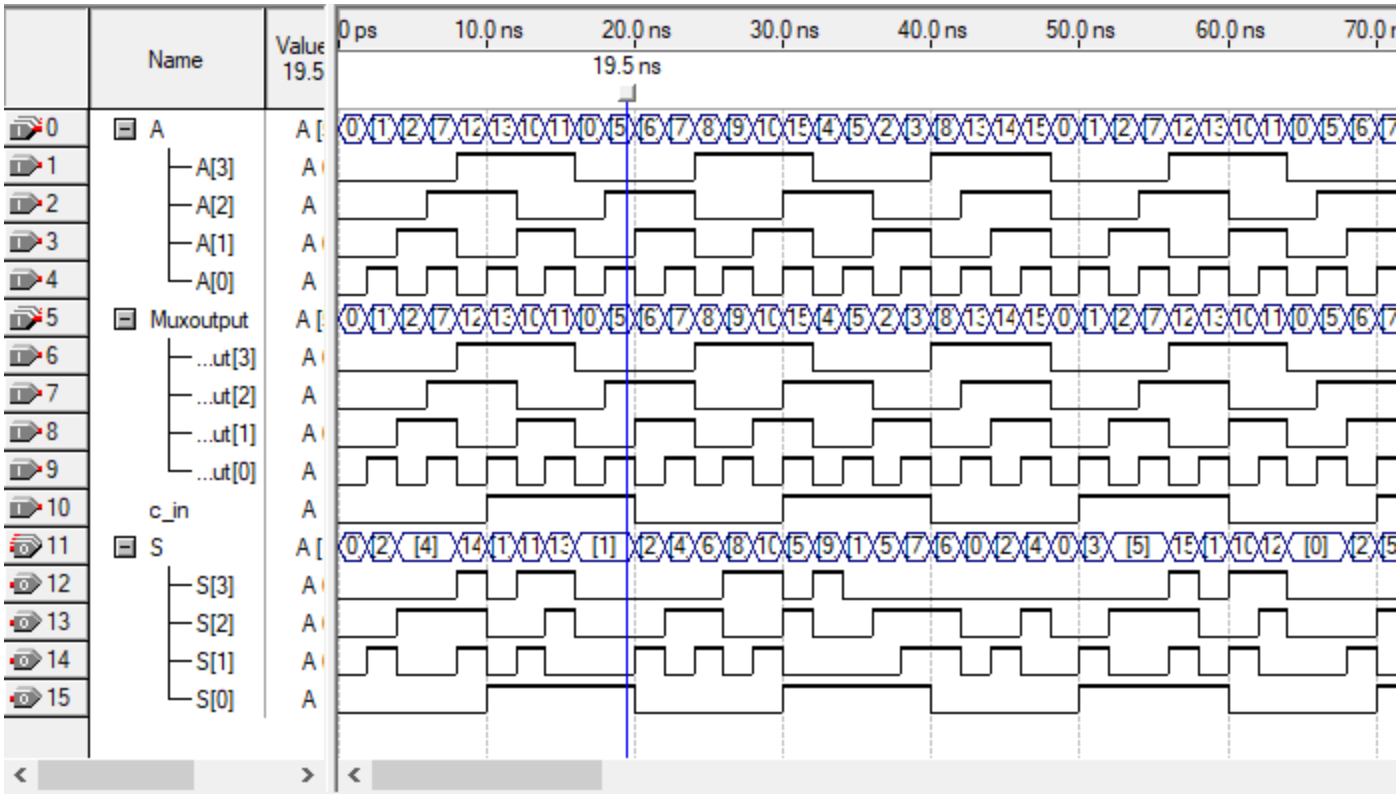


Last but not least : BCD Adder :

We used Full_Adder(Code is in the end of the report) function to implement the BCD Adder, First 4 bits will be added normally using the full adder, after the addition, We did another addition which will add 6 if the sum of the previous addition is more than 9. And came up with this code.(Using dataflow)

```
1  module BCD_Adder(input [3:0]Muxoutput,A,input c_in,output [3:0]S);
2
3  wire [3:0] Z,C_out,cout;
4  wire OutputCarry;
5
6
7  Full_Adder F1(Muxoutput[0],A[0],c_in,Z[0],C_out[0]);
8  Full_Adder F2(Muxoutput[1],A[1],C_out[0],Z[1],C_out[1]);
9  Full_Adder F3(Muxoutput[2],A[2],C_out[1],Z[2],C_out[2]);
10 Full_Adder F4(Muxoutput[3],A[3],C_out[2],Z[3],C_out[3]);
11
12 assign OutputCarry = ((C_out[3]) | (Z[3]&Z[2]) | (Z[3]&Z[1]));
13 //if OutputCarry =1 , then it will add 6(4b'0110) to the sum, if not
14 //, the sum will be added to 0(4b'0000) which will remain the same.
15 Full_Adder F5(Z[0],0,0,S[0],cout[0]);
16 Full_Adder F6(Z[1],OutputCarry,cout[0],S[1],cout[1]);
17 Full_Adder F7(Z[2],OutputCarry,cout[1],S[2],cout[2]);
18 Full_Adder F8(Z[3],0,cout[2],S[3],cout[3]);
19
20 endmodule
```

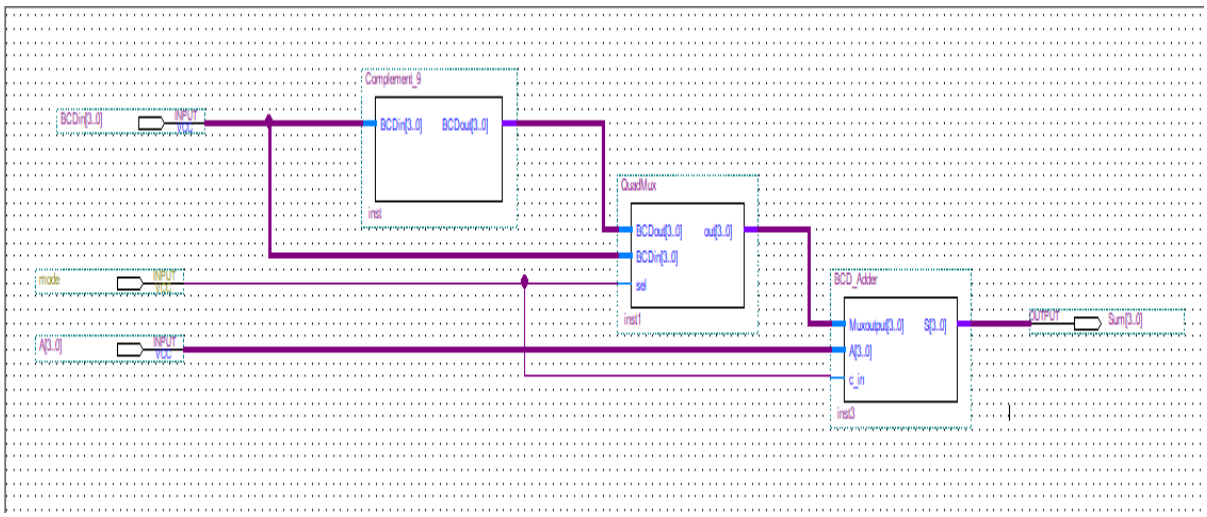
Simulation:



Finally :

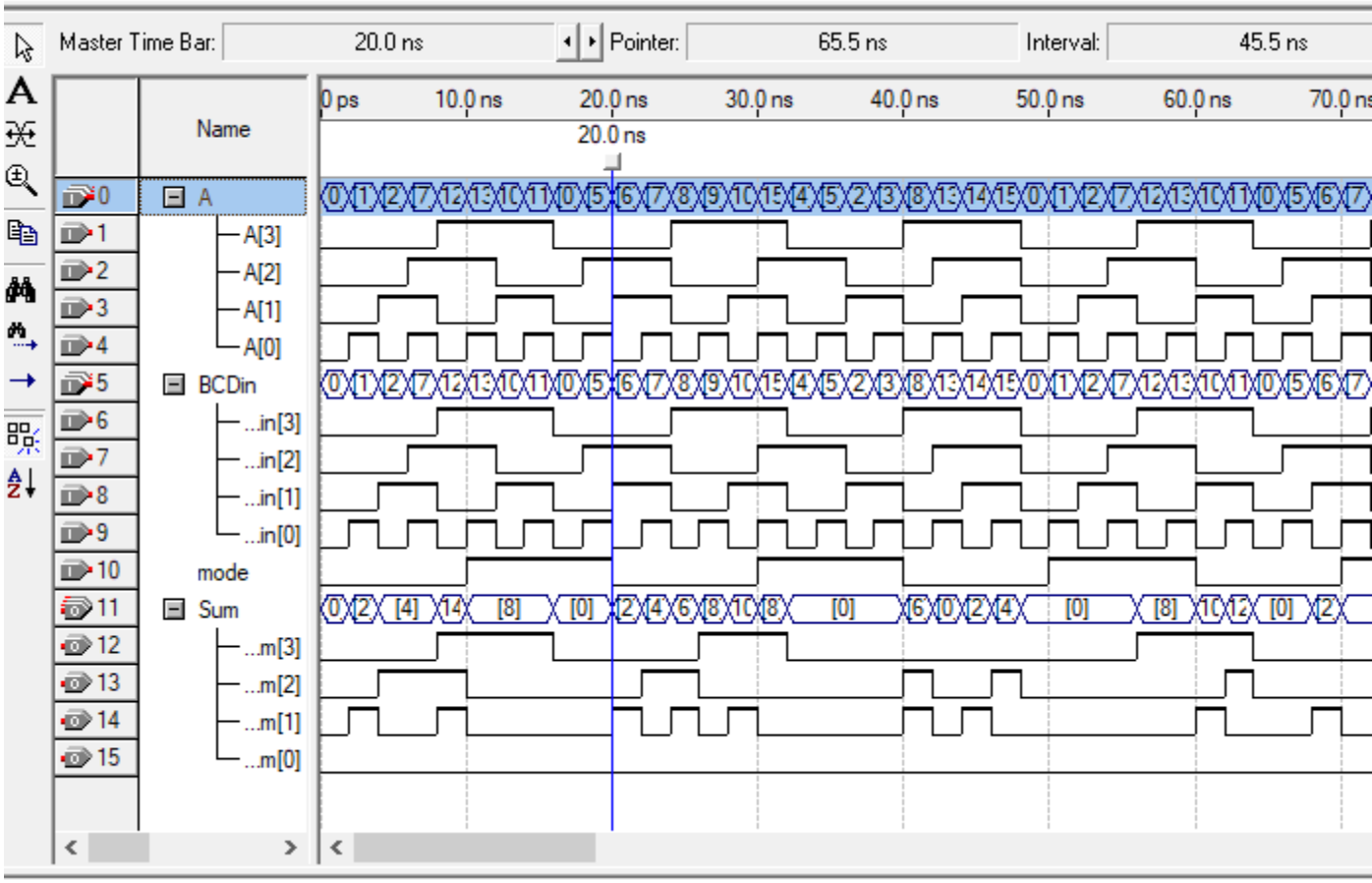
The block diagram :

Everything is connected successfully and the result is below.



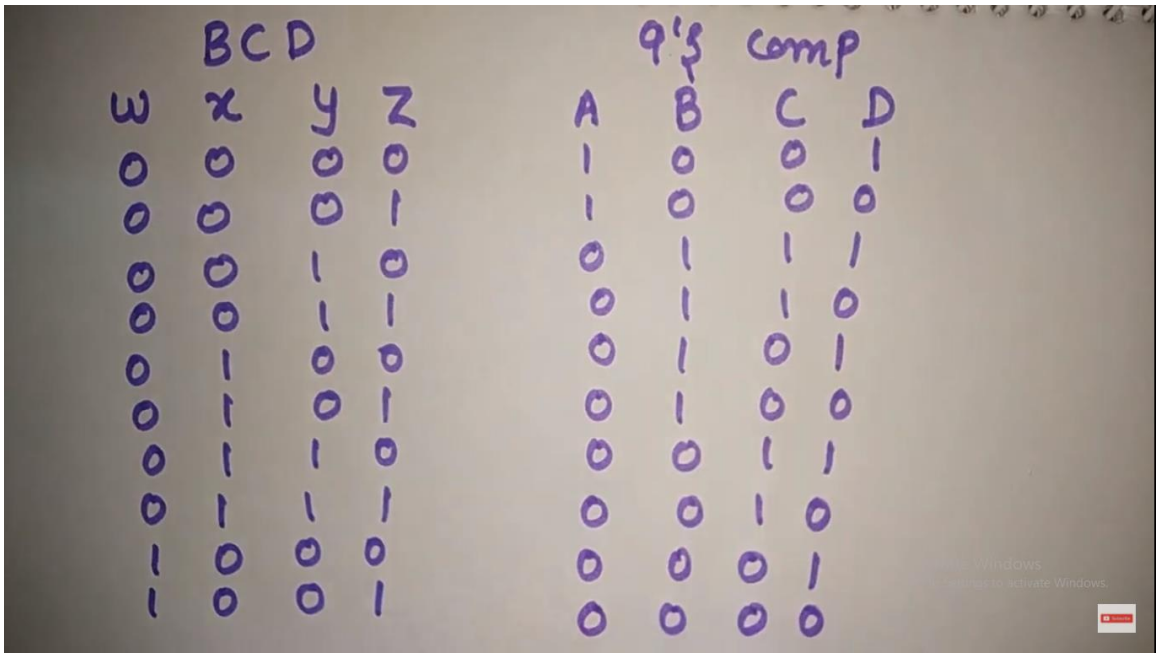

```
1   module final_code(input[3:0] BCD_in,A,input mode,output [3:0] S);
2
3   wire[3:0]BCD_out;
4   wire[3:0]out;
5
6   Complement_9 a1(BCD_in,BCD_out);
7   QuadMux a2(BCD_out,BCD_in,mode,out);
8   BCD_Adder a3(out,A,mode,S);
9
10  endmodule
```

Simulation


Master Time Bar:
20.0 ns
Pointer:
65.5 ns
Interval:
45.5 ns
Name
0 ps
10.0 ns
20.0 ns
30.0 ns
40.0 ns
50.0 ns
60.0 ns
70.0 ns
20.0 ns
0
A
1
A[3]
2
A[2]
3
A[1]
4
A[0]
5
BCDin
6
...in[3]
7
...in[2]
8
...in[1]
9
...in[0]
10
mode
11
Sum
12
...m[3]
13
...m[2]
14
...m[1]
15
...m[0]

Truth tables used:

1) **9's Complement**



BCD				9's comp			
W	X	Y	Z	A	B	C	D
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0

2) **QuadMux**

Select	Inputs		Output
0	0	0	0
0	0	1	1
1	1	0	1
1	1	1	1

3)BCD Adder

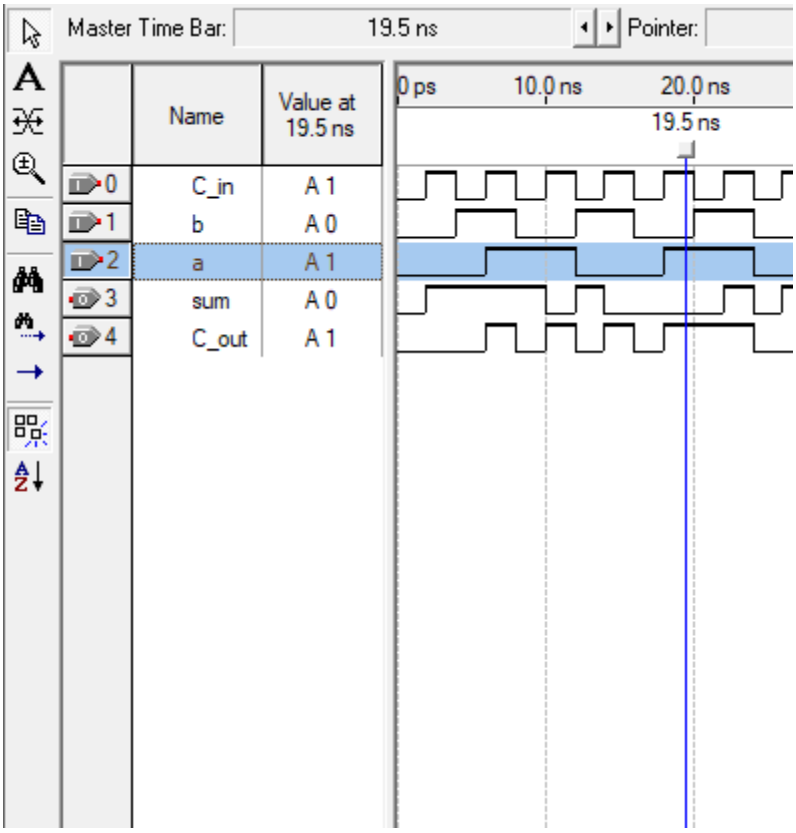
Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

Derivation of BCD adder

Full Adder code:

```
1  module Full_Adder(a,b,C_in,sum,C_out);
2
3  input a,b,C_in;
4  output sum,C_out;
5
6  assign C_out = (a&b) | (a&C_in) | (b&C_in);
7  assign sum = a^b^C_in;
8
9  endmodule
10
11
```

Full adder- simulation:



All Codes used(in writing) :

1)9's Complement:

module Complement_9(BCDin,BCDout);

```
input [3:0]BCDin;
output [3:0]BCDout;

//assign z = ~d;
//assign y = c;
//assign x = b^c;
//assign w = ~a & ~b & ~c;

//BCDin 3 = a
//BCDin 2 = b
//BCDin 1 = c
//BCDin 0 = d
//----------
//BCDout 3 = w
//BCDout 2 = x
//BCDout 1 = y
//BCDout 0 = z

assign BCDout[3] = ((~(BCDin[3])) & (~(BCDin[2])) & (~(BCDin[1])));
assign BCDout[2] = BCDin[2] ^ BCDin[1];
assign BCDout[1] = BCDin[1];
assign BCDout[0] = ~BCDin[0];

endmodule
```

2)MUX

```
module QuadMux(input [3:0]BCDout,BCDin,input sel,output reg [3:0]out);

always @(BCDout,BCDin,sel) begin
if(sel == 0) out = BCDin;

else out = BCDout;

end

endmodule
```

---

3)Full adder

```
module Full_Adder(a,b,C_in,sum,C_out);

input a,b,C_in;
output sum,C_out;

assign C_out = (a&b) | (a&C_in) | (b&C_in);
```

```
assign sum = a^b^C_in;

endmodule
```

---

4)BCD Adder

```
module BCD_Adder(input [3:0]Muxoutput,A,input c_in,output [3:0]S);

wire [3:0] Z,C_out,cout;
wire OutputCarry;

Full_Adder F1(Muxoutput[0],A[0],c_in,Z[0],C_out[0]);
Full_Adder F2(Muxoutput[1],A[1],C_out[0],Z[1],C_out[1]);
Full_Adder F3(Muxoutput[2],A[2],C_out[1],Z[2],C_out[2]);
Full_Adder F4(Muxoutput[3],A[3],C_out[2],Z[3],C_out[3]);

assign OutputCarry = ((C_out[3]) | (Z[3]&Z[2]) | (Z[3]&Z[1]));
//if OutputCarry =1 , then it will add 6(4b'0110) to the sum, if not, the
sum will be added to 0(4b'0000) which will remain the same.
Full_Adder F5(Z[0],0,0,S[0],cout[0]);
Full_Adder F6(Z[1],OutputCarry,cout[0],S[1],cout[1]);
Full_Adder F7(Z[2],OutputCarry,cout[1],S[2],cout[2]);
Full_Adder F8(Z[3],0,cout[2],S[3],cout[3]);
```

endmodule

---

5)Final code

```
module final_code(input[3:0] BCD_in,A,input mode,output [3:0] S);

wire[3:0]BCD_out;
wire[3:0]out;

Complement_9 a1(BCD_in,BCD_out);
QuadMux a2(BCD_out,BCD_in,mode,out);
BCD_Adder a3(out,A,mode,S);

endmodule
```

Done.