VLSI Lab Report
Assignment 4 Annexure1

BCSE 4th Year 2nd Semester

Name: ***Priti Shaw***
Roll No. : ***001710501076***
Batch: ***A3***

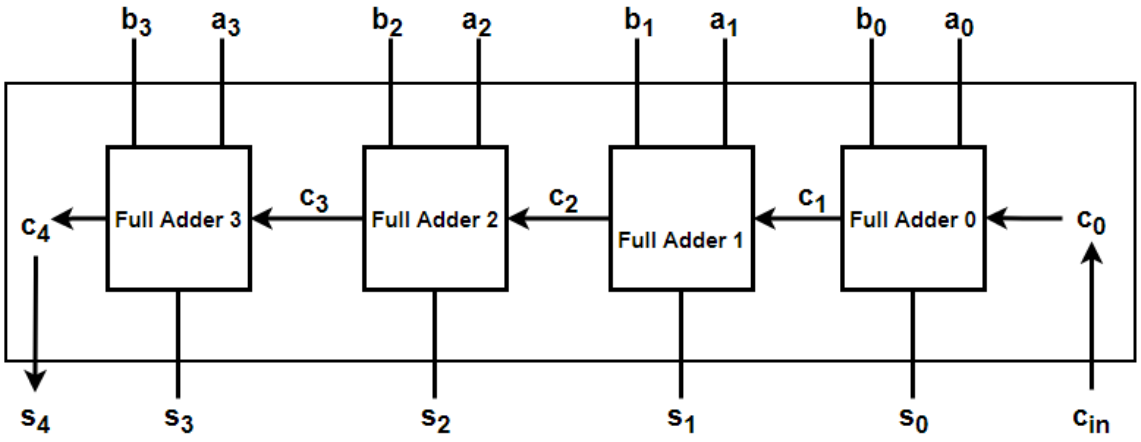
1. 4-Bit Ripple Carry Adder

Description

Implement a 4-bit ripple carry adder using the structural modelling. Also, design a testbench for 4-bit ripple carry adder.

Multiple full adder circuits can be cascaded in parallel to add an N-bit number. For an N-bit parallel adder, there must be N number of full adder circuits. A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry-in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage.In a 4-bit ripple carry adder, it takes two 4-bit binary numbers and one input carry bit as input and gives their sum(maximum five bits) as output. Each full adder gives one bit of final sum and the output carry bit of the last full adder is passed to MSB of output sum.

Block Diagram



Entity

```
entity fourBitRippleCarryUsingStructural is
    Port ( a_in : in  STD_LOGIC_VECTOR (3 downto 0);
           b_in : in  STD_LOGIC_VECTOR (3 downto 0);
           c_in : in  STD_LOGIC;
           s_out : out  STD_LOGIC_VECTOR (4 downto 0));
end fourBitRippleCarryUsingStructural;
```

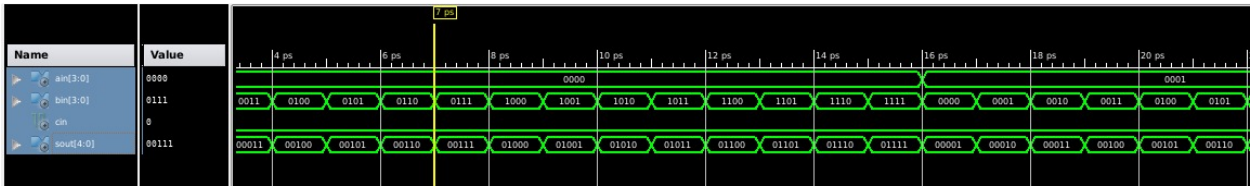
Architecture

```
architecture Behavioral of fourBitRippleCarryUsingStructural is

component fullAdder is
    Port ( ain : in  STD_LOGIC;
           bin : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           cout : out  STD_LOGIC;
           sout : out  STD_LOGIC);
end component;

signal c:std_logic_vector(4 downto 0);
begin
            c(0)<=c_in;
            gen: for k in 0 to 3 generate
                              prock:fullAdder port map
(a_in(k),b_in(k),c(k),c(k+1),s_out(k));
            end generate;
            s_out(4)<=c(4);

end Behavioral;
```

TestBench

```
- Stimulus process
   stim_proc: process
   variable j,k:integer;
      variable binA,binB:std_logic_vector(3 downto 0);
   begin
            for k in 0 to 15 loop
                  procA:decimalToBinaryProcedure(k,4,binA);
                  ain<=binA;
                  for j in 0 to 15 loop
                        procB:decimalToBinaryProcedure(j,4,binB);
                        bin<=binB;
                        cin<='0';
                        wait for 1 ps;
                  end loop;
            end loop;
   end process;
```

Timing Diagram



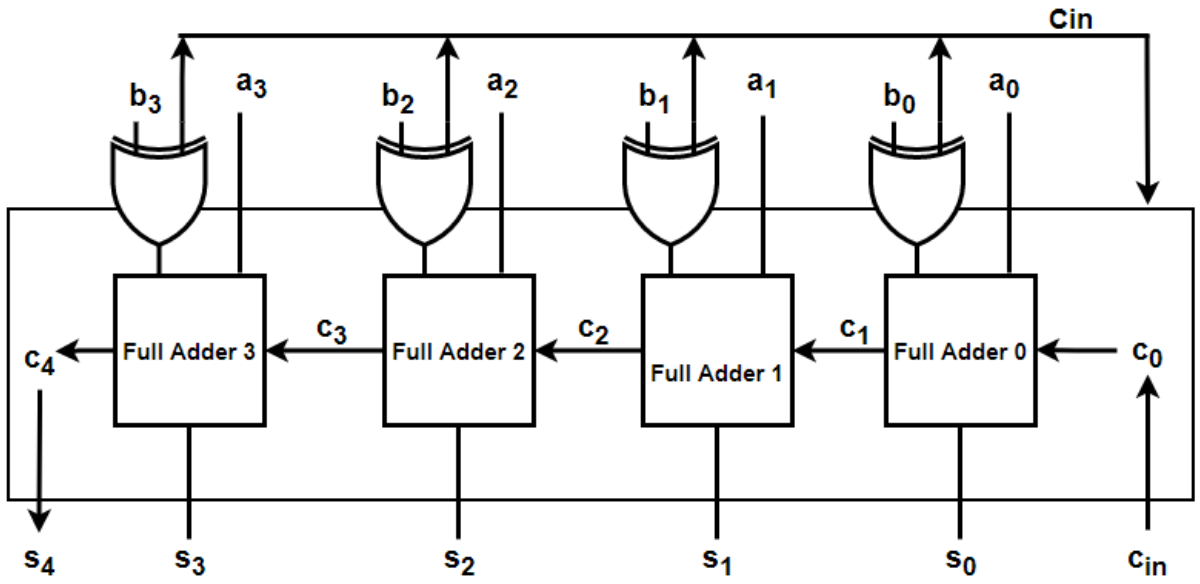
2. 4-bit Adder/Subtractor

Description

Implement a 4-bit Adder/Subtracter circuit using the structural modelling. Also, design a testbench for 4-bit adder/subtractor.

A 4 bit ripple carry adder is used to make a 4 bit Adder/Subtractor. It takes two input numbers (4 bit long) and one input carry bit and depending on the input carry bit value it gives addition or subtraction value as output(5 bit long). If input carry bit is 0 then it acts as adder and if input carry bit is 1 then it acts as a subtractor. For negative number output (MSB is 1), it gives the result in 2's complement format.

Block Diagram



Entity

```
entity fourBitAdderSubtractorusingStructural is
    Port ( ain : in  STD_LOGIC_VECTOR (3 downto 0);
           bin : in  STD_LOGIC_VECTOR (3 downto 0);
           cin : in  STD_LOGIC;
           s : out  STD_LOGIC_VECTOR (4 downto 0));
end fourBitAdderSubtractorusingStructural;
```

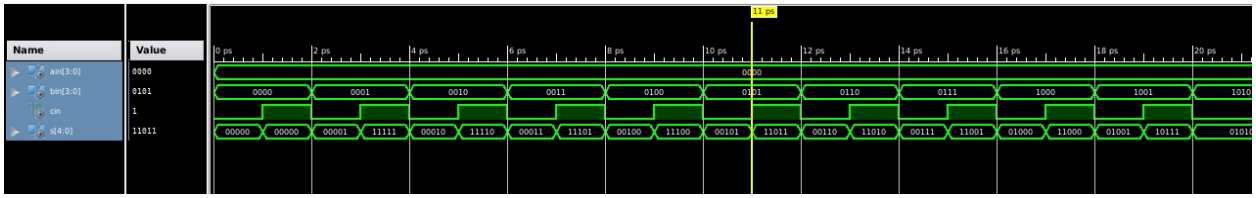
Architecture

```
architecture Behavioral of fourBitAdderSubtractorusingStructural is
component fourBitRippleCarryUsingStructural is
    Port ( a_in : in  STD_LOGIC_VECTOR (3 downto 0);
           b_in : in  STD_LOGIC_VECTOR (3 downto 0);
           c_in : in  STD_LOGIC;
           s_out : out  STD_LOGIC_VECTOR (4 downto 0));
end component;
      signal p:std_logic_vector(3 downto 0);
      signal ss:std_logic_vector(4 downto 0);

begin
            genk: for k in 0 to 3 generate
                  p(k) <= bin(k) xor cin;
            end generate;

            gen: fourBitRippleCarryUsingStructural port map (ain, p, cin,
ss);
            s(3 downto 0) <= ss(3 downto 0);
            s(4) <= ss(4) xor cin;
end Behavioral;
```

TestBench

```
stim_proc: process
   variable j,k:integer;
      variable binA,binB:std_logic_vector(3 downto 0);
   begin
            for k in 0 to 15 loop
                  procA:decimalToBinaryProcedure(k,4,binA);
                  ain<=binA;
                  for j in 0 to 15 loop
                        procB:decimalToBinaryProcedure(j,4,binB);
                        bin<=binB;
                        cin<='0';
                        wait for 1 ps;
                        cin<='1';
                        wait for 1 ps;
                  end loop;
            end loop;
   end process;
```

Timing Diagram



3. BCD Adder

Description

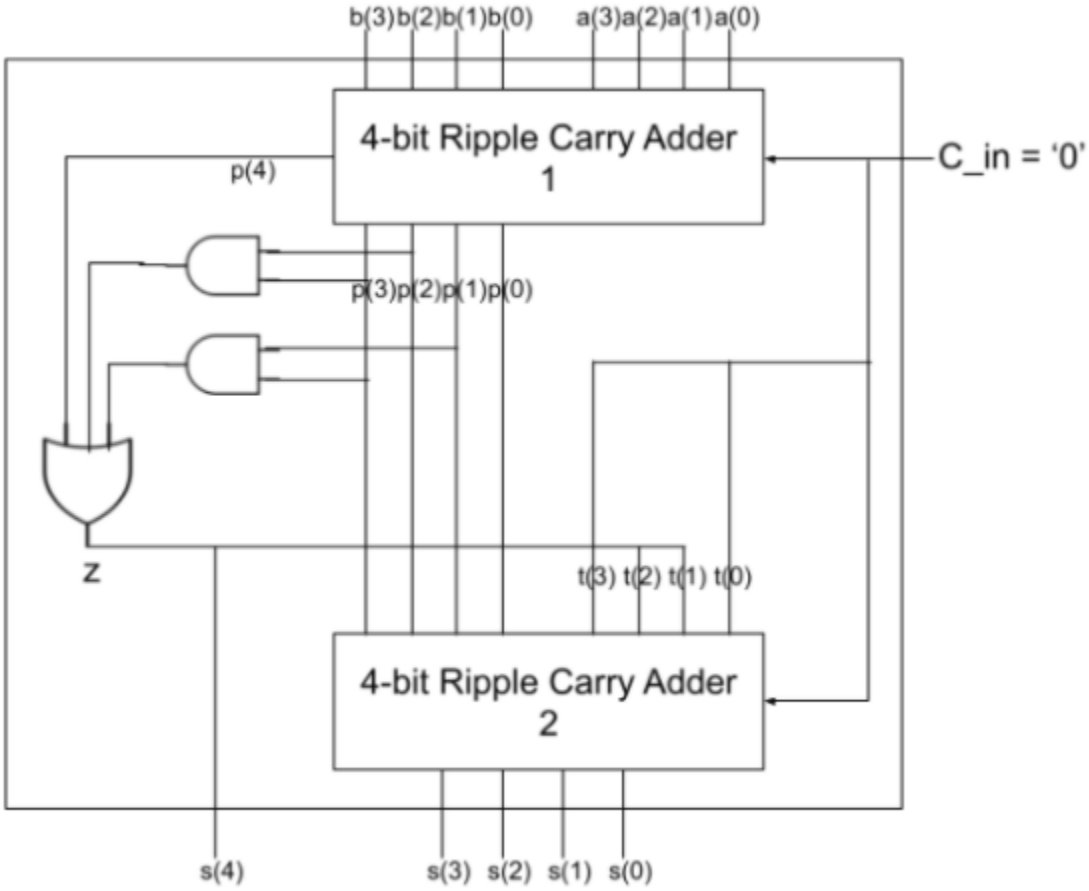
Implement a BCD adder using the structural modelling. Also, design a testbench for BCD adder.

In BCD or Binary Coded Decimal representation, every digit of a decimal number is converted to a four bit binary number. So, if the decimal number is 16 then the corresponding BCD number will be "00010110". Rest of the things are the same as a four bit ripple carry adder. It takes two four bit binary numbers (maximum 9 or 1001) as inputs and gives their sum as output (5 bit long) in BCD format.

Truth Table

Dec	**Binary Sum**					**BCD Sum**				
	p4	**p3**	**p2**	**p1**	**p0**	**q4(z)**	**q3**	**q2**	**q1**	**q0**
0-9	b4	b3	b2	b1	b0	b4	b3	b2	b1	b0
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
...	...	...	...	...	...	...	...	...	...	...
31	...	...	...	...	...	...	...	...	…	...

Block Diagram



Entity

```
entity BCDAdderUsingStructural is
    Port ( a : in  STD_LOGIC_VECTOR (3 downto 0);
           b : in  STD_LOGIC_VECTOR (3 downto 0);
           s : out  STD_LOGIC_VECTOR (4 downto 0));
end BCDAdderUsingStructural;
```

Architecture

```
architecture Behavioral of BCDAdderUsingStructural is

component fourBitRippleCarryUsingStructural is
    Port ( a_in : in  STD_LOGIC_VECTOR (3 downto 0);
           b_in : in  STD_LOGIC_VECTOR (3 downto 0);
```

```
           c_in : in  STD_LOGIC;
           s_out : out  STD_LOGIC_VECTOR (4 downto 0));
end component;

signal p,q:std_logic_vector(4 downto 0);
signal c:std_logic_vector(3 downto 0);
signal z:std_logic;

begin
                  c1: fourBitRippleCarryUsingStructural port map
(a,b,'0',p);
                  z <= p(4) or (p(3) and p(2)) or (p(3) and p(1));
                  c <= '0' & z & z & '0';
                  c2: fourBitRippleCarryUsingStructural port map(p(3 downto
0),c,'0',q);
                  s(3 downto 0) <= q(3 downto 0);
                  s(4) <= z;
end Behavioral;
```

TestBench

```
stim_proc: process
   variable j,k:integer;
      variable binA,binB:std_logic_vector(3 downto 0);
      begin
            for k in 0 to 9 loop
                  procA:decimalToBinaryProcedure(k,4,binA);
                  aa<=binA;
                  for j in 0 to 9 loop
                        procB:decimalToBinaryProcedure(j,4,binB);
                        bb<=binB;
                        wait for 1 ps;
                  end loop;
            end loop;
   end process;
```

Timing Diagram