VLSI
Assignment 4 Annexure

Md Sahil
BCSE IV
Roll-001710501029

1 Describtion

- Implement 4-bit ripple carry adder using structural modelling.
- Implement adder/subtractor using structural modelling.
- Implement BCD adder using structural modelling.

2 Block Diagram

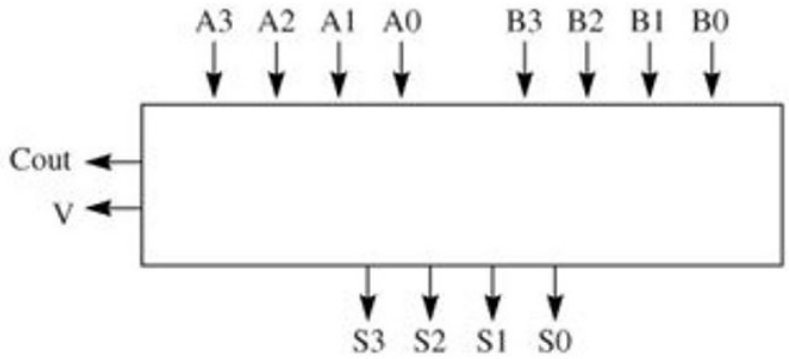


Figure 1: Ripple carry adder block diagram

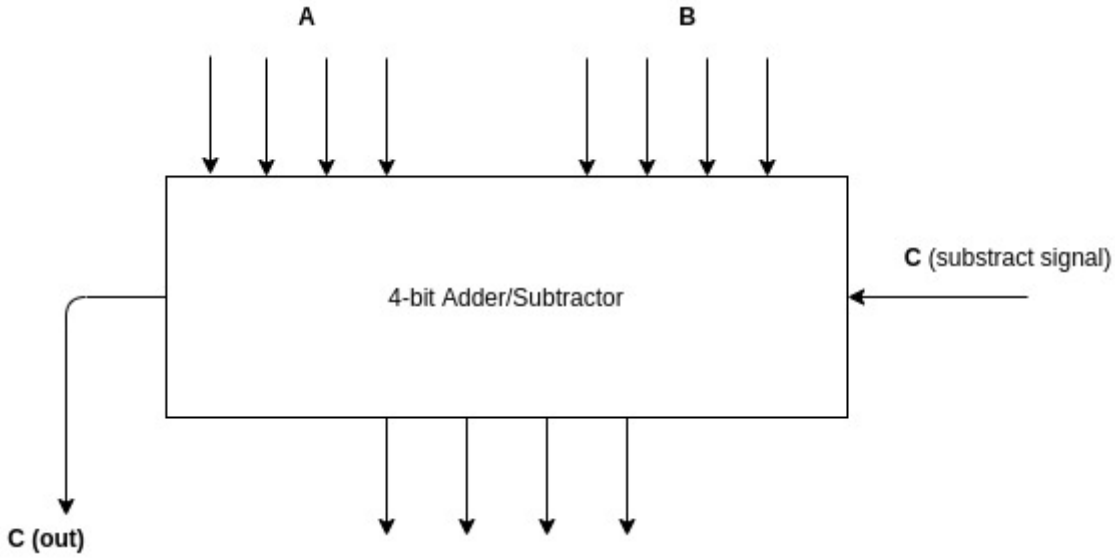


Figure 2: Adder/subtractor block diagram

2.1 BCD adder

3 Circuit Diagram

4 Package

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package adder_package is
```

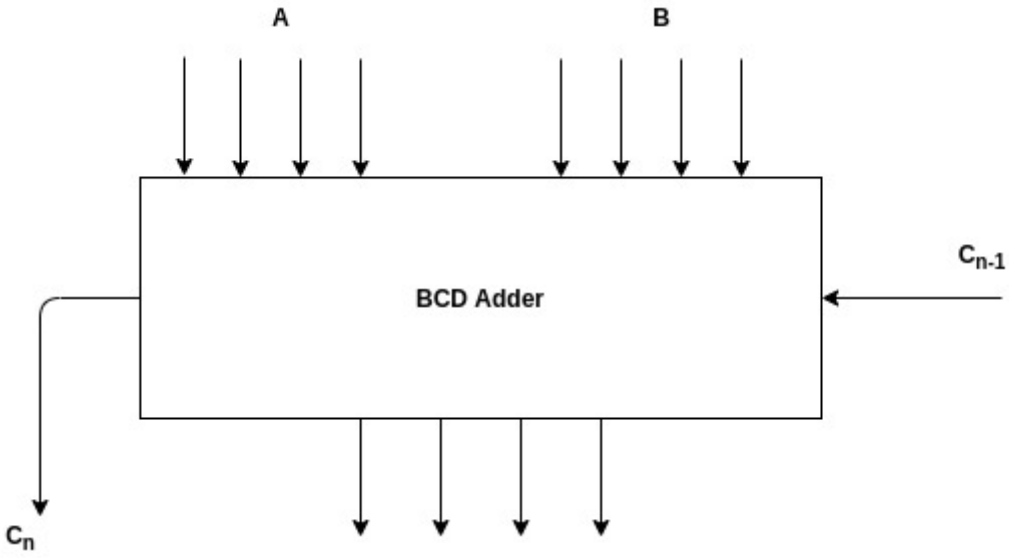


Figure 3: BCD adder block diagram

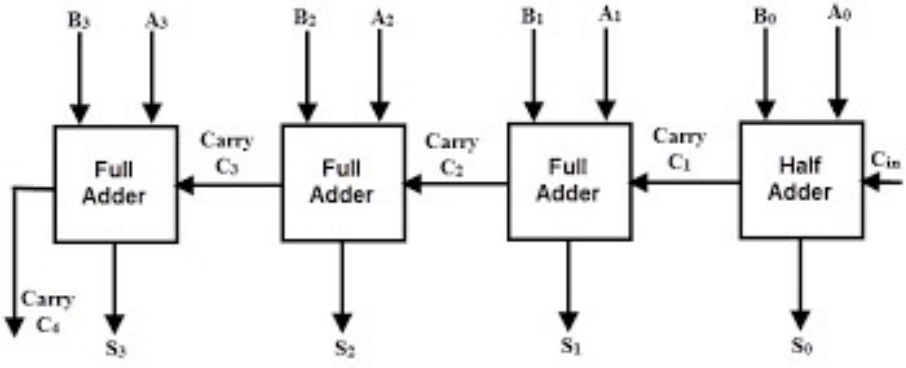


Figure 4: Ripple carry adder circuit diagram

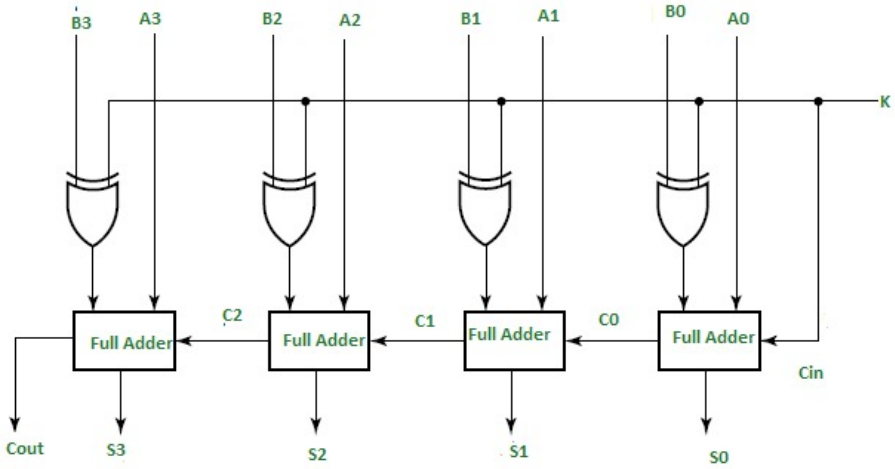


Figure 5: Adder subtractor circuit diagram

```
procedure half_adder (
A: in std_logic;
B: in std_logic;
S: out std_logic;
C: out std_logic);
```

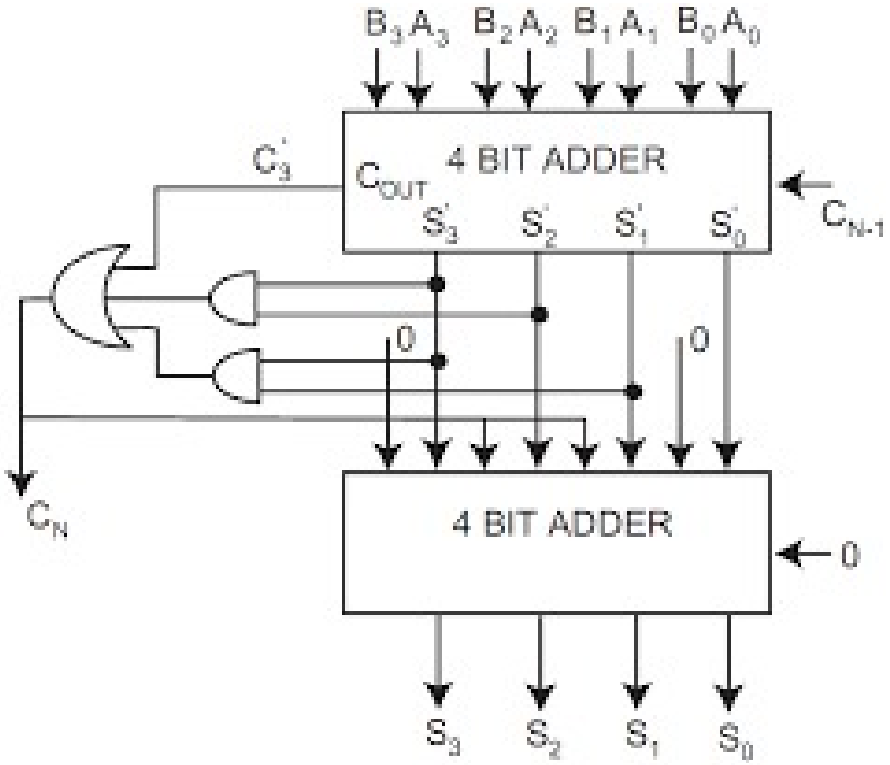


Figure 6: BCD adder circuit diagram

```
        procedure full_adder (
        A: in std_logic;
        B: in std_logic;
        Cin: in std_logic;
        S: out std_logic;
        C: out std_logic);
        procedure ripple_carry_adder_4bit (
        A: in std_logic_vector(3 downto 0);
        B: in std_logic_vector(3 downto 0);
        C: in std_logic;
        S: out std_logic_vector(4 downto 0));
        procedure adder_subtractor_4bit (
        A: in std_logic_vector(3 downto 0);
        B: in std_logic_vector(3 downto 0);
        C: in std_logic;
        S: out std_logic_vector(4 downto 0));
        procedure bcd_adder_4bit(
        A: in std_logic_vector(3 downto 0);
        B: in std_logic_vector(3 downto 0);
        s: out std_logic_vector(4 downto 0));
        procedure dec_to_bin_proc(
        decimal: in integer;
        num_of_bits: in integer;
        binary: out std_logic_vector);
end adder_package;

package body adder_package is

        procedure dec_to_bin_proc(
        decimal: in integer;
        num_of_bits: in integer;
        binary: out std_logic_vector) is
```

```
variable dec, bit_pos: integer;
begin
            dec := decimal;
            bit_pos := 0;
            while(bit_pos < num_of_bits) loop
                    if (dec rem 2) = 0 then
                            binary(bit_pos) := '0';
                    else
                            binary(bit_pos) := '1';
                    end if;
                    dec := dec/2;
                    bit_pos := bit_pos + 1;
            end loop;
end procedure;


    procedure half_adder (A: in std_logic;
                                                            B: in std_logic;
                                                            S: out std_logic;
                                                            C: out std_logic)  is
    begin
            S := A xor B;
            C := A and B;
    end half_adder;

    procedure full_adder (A: in std_logic;
                                                            B: in std_logic;
                                                            Cin: in std_logic;
                                                            S: out std_logic;
                                                            C: out std_logic)  is
            variable cc1, ss1, cc2, ss2 : std_logic;
    begin
            pc1: half_adder(A, B, ss1, cc1);
            pc2: half_adder(Cin, ss1, S, cc2);
            pc3: half_adder(cc1, cc2, C, ss2);
    end full_adder;


    procedure ripple_carry_adder_4bit (
            A: in std_logic_vector( 3 downto 0 );
            B: in std_logic_vector( 3 downto 0 );
            C: in std_logic;
            S: out std_logic_vector( 4 downto 0)
    )  is
            variable cc : std_logic_vector( 4 downto 0);
    begin
            cc(0):= C;
            for i in 0 to 3 loop
                    pc1: full_adder(A(i), B(i), cc(i), S(i), cc(i+1));
            end loop;
            S(4) := cc(4);
    end ripple_carry_adder_4bit;


    procedure adder_subtractor_4bit(
    A: in std_logic_vector(3 downto 0);
    B: in std_logic_vector(3 downto 0);
    C: in std_logic;
    S: out std_logic_vector(4 downto 0)) is
            variable p: std_logic_vector(3 downto 0);
```

```
                variable s1: std_logic_vector(4 downto 0);
        begin
                p(3 downto 0) := B(3 downto 0) xor (C & C & C & C);
                pc1: ripple_carry_adder_4bit(A,P,'0',s1);
                if c='1' then
                        s1(4) := not S1(4);
                end if;
                S := s1;
        end adder_subtractor_4bit;


        procedure bcd_adder_4bit(
        A: in std_logic_vector(3 downto 0);
        B: in std_logic_vector(3 downto 0);
        s: out std_logic_vector(4 downto 0)) is
                variable p: std_logic_vector(4 downto 0);
                variable z,c: std_logic;
                variable q: std_logic_vector(3 downto 0);
        begin
                c := '0';
                pc1: ripple_carry_adder_4bit(A,B,c,p);
                z:= p(4) or (p(3) and (p(2) or p(1)));
                q:= c & z & z & c;
                pc2: ripple_carry_adder_4bit(p(3 downto 0),q,c,s);
                s(4):=z;
        end bcd_adder_4bit;

end adder_package;
```

5 4 bit ripple carry adder

5.1 Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity a4_ax1 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 a4_ax1;

architecture Behavioral of a4_ax1 is

component a4_b is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           Cin : in  STD_LOGIC;
           S : out  STD_LOGIC;
           C : out  STD_LOGIC);
end component;

signal cc : std_logic_vector( 4 downto 0);

begin

                cc(0)<= '0';
                f1: for i in 0 to 3 generate
                        pc1: a4_b port map(A(i), B(i), cc(i), S(i), cc(i+1));
                end generate;
```

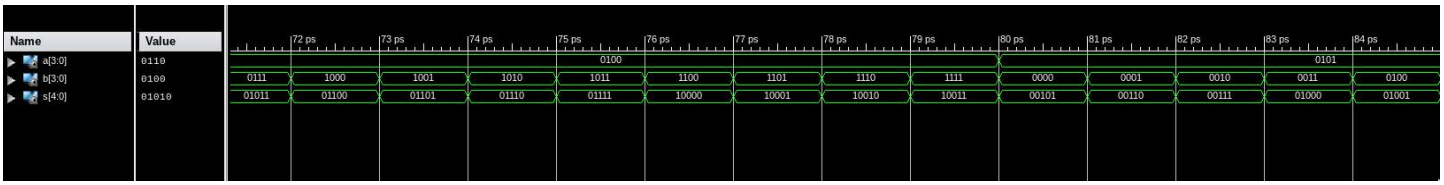
```
            S(4) <= cc(4);

end Behavioral;
```

5.2 Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use work.adder_package.ALL;

ENTITY tb_a4_ax1 IS
END tb_a4_ax1;

ARCHITECTURE behavior OF tb_a4_ax1 IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT a4_ax1
    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 COMPONENT;


   --Inputs
   signal A : std_logic_vector(3 downto 0) := (others => '0');
   signal B : std_logic_vector(3 downto 0) := (others => '0');

        --Outputs
   signal S : std_logic_vector(4 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name


BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: a4_ax1 PORT MAP (
          A => A,
          B => B,
          S => S
        );


   -- Stimulus process
   stim_proc: process
        variable aa,bb: std_logic_vector(3 downto 0);
   begin
                l1: for i in 0 to 15 loop
                        l2: for j in 0 to 15 loop
                                dec_to_bin_proc(i,4,aa);
                                A <= aa;
                                dec_to_bin_proc(j,4,bb);
                                B <= bb;
                                wait for 1 ps;
                        end loop;
                end loop;
   end process;
```

```
END;
```

5.3 Timing diagram



6 Adder/subtractor

6.1 Code

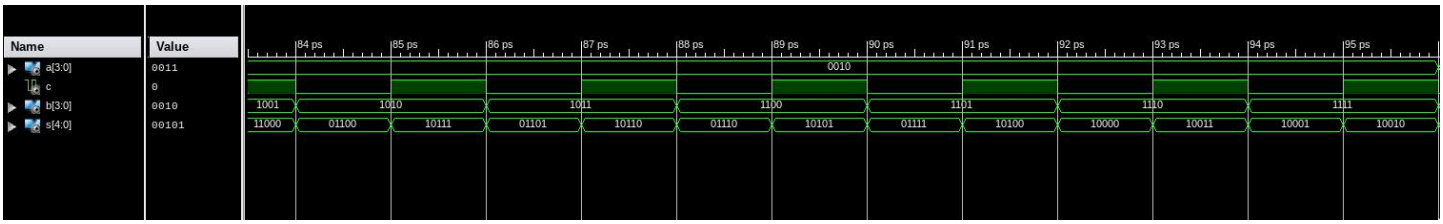
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity a4_ax2 is
    Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
           B : in  STD_LOGIC_VECTOR (3 downto 0);
                        C : in STD_LOGIC;
           S : out  STD_LOGIC_VECTOR (4 downto 0));
end a4_ax2;

architecture Behavioral of a4_ax2 is

    COMPONENT a4_ax1
    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 COMPONENT;

        signal p: std_logic_vector(3 downto 0);
        signal s1: std_logic_vector(4 downto 0);

        begin
                p(3 downto 0) <= B(3 downto 0) xor (C & C & C & C);
                pc1: a4_ax1 port map(A,p,s1);
                with C select S(4) <= not s1(4) when '1', s1(4) when others;
                S(3 downto 0) <= s1(3 downto 0);



end Behavioral;
```

6.2 Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use work.adder_package.ALL;

ENTITY tb_a4_ax2 IS
END tb_a4_ax2;

ARCHITECTURE behavior OF tb_a4_ax2 IS
```

```
    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT a4_ax2
    PORT(
         A : IN  std_logic_vector(3 downto 0);
         B : IN  std_logic_vector(3 downto 0);
                        C : IN std_logic;
         S : OUT  std_logic_vector(4 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal A : std_logic_vector(3 downto 0) := (others => '0');
        signal C : std_logic;
   signal B : std_logic_vector(3 downto 0) := (others => '0');

         --Outputs
   signal S : std_logic_vector(4 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name


BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: a4_ax2 PORT MAP (
          A => A,
          B => B,
                          C => C,
          S => S
        );


   -- Stimulus process
   stim_proc: process
        variable aa,bb: std_logic_vector(3 downto 0);
   begin
                l1: for i in 0 to 15 loop
                        l2: for j in 0 to 15 loop
                                dec_to_bin_proc(i,4,aa);
                                A <= aa;
                                dec_to_bin_proc(j,4,bb);
                                B <= bb;
                                C<='0';
                                wait for 1 ps;
                                dec_to_bin_proc(i,4,aa);
                                A <= aa;
                                dec_to_bin_proc(j,4,bb);
                                B <= bb;
                                C<='1';
                                wait for 1 ps;
                        end loop;
                end loop;
   end process;
END;
```

6.3 Timing diagram



7 BCD adder

7.1 Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity a4_ax3 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 a4_ax3;

architecture Behavioral of a4_ax3 is

component a4_d 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 component;

begin
        c1: a4_d port map(A,B,S);
end Behavioral;
```

7.2 Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use work.adder_package.ALL;

ENTITY tb_a4_ax3 IS
END tb_a4_ax3;

ARCHITECTURE behavior OF tb_a4_ax3 IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT a4_ax3
    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 COMPONENT;


   --Inputs
   signal A : std_logic_vector(3 downto 0) := (others => '0');
   signal B : std_logic_vector(3 downto 0) := (others => '0');
```

```
        --Outputs
    signal S : std_logic_vector(4 downto 0);
    -- No clocks detected in port list. Replace <clock> below with
    -- appropriate port name

BEGIN

        -- Instantiate the Unit Under Test (UUT)
    uut: a4_ax3 PORT MAP (
           A => A,
           B => B,
           S => S
         );


    -- Stimulus process
    stim_proc: process
         variable aa,bb: std_logic_vector(3 downto 0);
    begin
                 l1: for i in 0 to 15 loop
                        l2: for j in 0 to 15 loop
                                dec_to_bin_proc(i,4,aa);
                                A <= aa;
                                dec_to_bin_proc(j,4,bb);
                                B <= bb;
                                wait for 1 ps;
                        end loop;
                 end loop;
    end process;

END;
```

7.3 Timing diagram