



# Adders

VLSI Systems  
Assignment-4

**PREPARED BY**

Md Sahil

BCSE-IV

001710501029

Jadavpur University

## Description

Creating a package for different procedures of adders and using them to implement various adders.

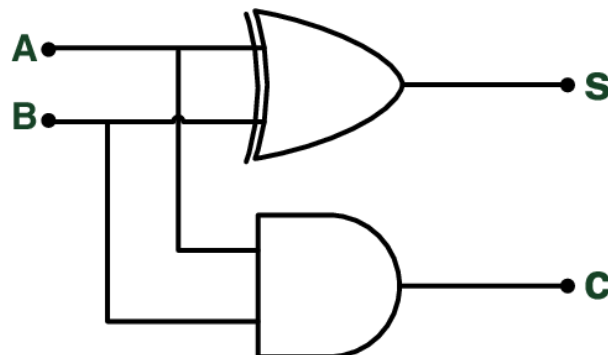
1. Half Adder
2. Full Adder
3. 4-bit ripple carry Adder
4. Adder/Subtractor
5. BCD Adder

## Half Adder

Block Diagram



Circuit Diagram



Truth Table

| Inputs |   | Outputs |     |
|--------|---|---------|-----|
| A      | B | Carry   | Sum |
| 0      | 0 | 0       | 0   |
| 0      | 1 | 0       | 1   |
| 1      | 0 | 0       | 1   |
| 1      | 1 | 1       | 0   |

## Code

### Package code

```
procedure half_adder_proc(a: in std_logic; b: in std_logic; c: out
std_logic; s: out std_logic) is
begin
    s:= a xor b;
    c:= a and b;
end procedure;
```

### Implementation

```
architecture Behavioral of half_adder is
begin
    p1: process(a, b)
        variable c,s: std_logic;
    begin
        proc: half_adder_proc(a, b, c, s);
        carry <= c;
        sum <= s;
    end process;
end Behavioral;
```

### Test Bench

```
ARCHITECTURE behavior OF half_adder_test_bench IS

-- Component Declaration
    COMPONENT half_adder
    PORT(
        a : in  STD_LOGIC;
        b : in  STD_LOGIC;
        sum : out STD_LOGIC;
        carry : out STD_LOGIC
    );
    END COMPONENT;

    SIGNAL a : std_logic;
    SIGNAL b : std_logic;
    SIGNAL sum: std_logic;
    SIGNAL carry: std_logic;

BEGIN

    uut: half_adder PORT MAP (
```

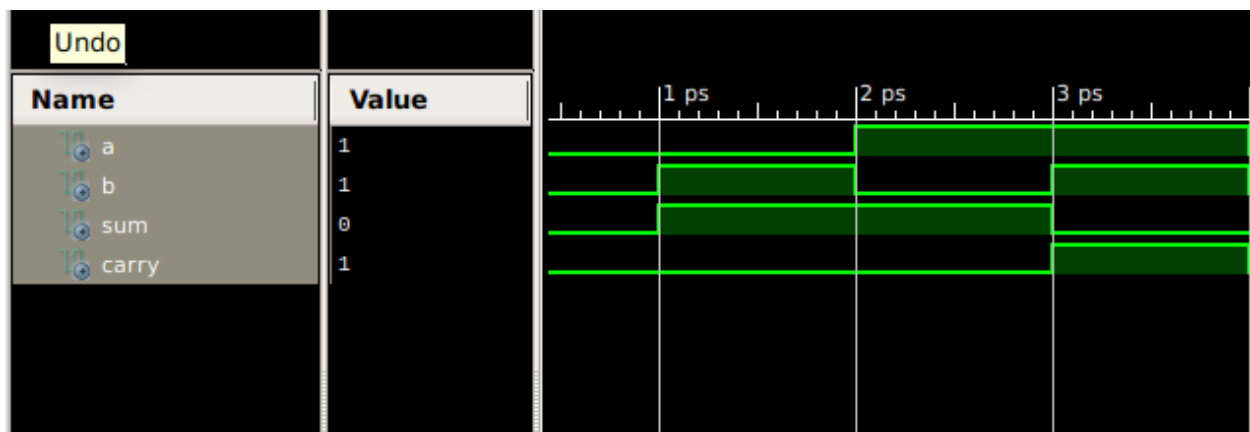
```

        a => a,
        b => b,
        sum => sum,
        carry => carry
    );

    tb : PROCESS
    BEGIN
        a <= '0';
        b <= '0';
        wait for 1ps;
        a <= '0';
        b <= '1';
        wait for 1ps;
        a <= '1';
        b <= '0';
        wait for 1ps;
        a <= '1';
        b <= '1';
        wait for 1ps;
    END PROCESS tb;

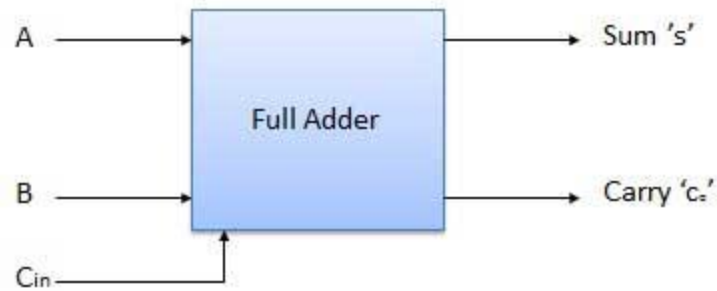
END;
```

Timing Diagram

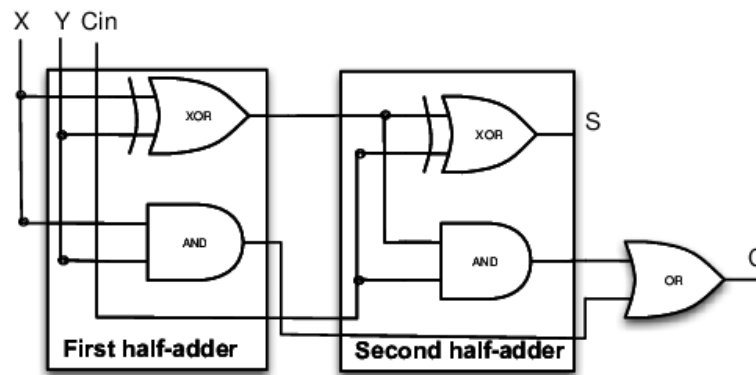


# Full Adder

## Block Diagram



## Circuit Diagram



## Truth Table

| Inputs |   |     | Outputs |     |
|--------|---|-----|---------|-----|
| A      | B | Cin | Carry   | Sum |
| 0      | 0 | 0   | 0       | 0   |
| 0      | 0 | 1   | 0       | 1   |
| 0      | 1 | 0   | 0       | 1   |
| 0      | 1 | 1   | 1       | 0   |
| 1      | 0 | 0   | 0       | 1   |
| 1      | 0 | 1   | 1       | 0   |
| 1      | 1 | 0   | 1       | 0   |
| 1      | 1 | 1   | 1       | 1   |

## Code

### Package code

```
procedure full_adder_proc(a: in std_logic; b: in std_logic; c: in
std_logic; carry: out std_logic; sum: out std_logic) is
    variable c1, s1, c2, s2: std_logic;
begin
    proc1: half_adder_proc(b, c, c1, s1);
    proc2: half_adder_proc(a, s1, c2, s2);
    carry := c1 or c2;
    sum := s2;
end procedure;
```

### Implementation

```
architecture Behavioral of full_adder is
begin
    p1: process(a, b, c)
        variable s, c_out: std_logic;
    begin
        proc: full_adder_proc(a, b, c, c_out, s);
        sum <= s;
        carry <= c_out;
    end process;
end Behavioral;
```

## Test Bench

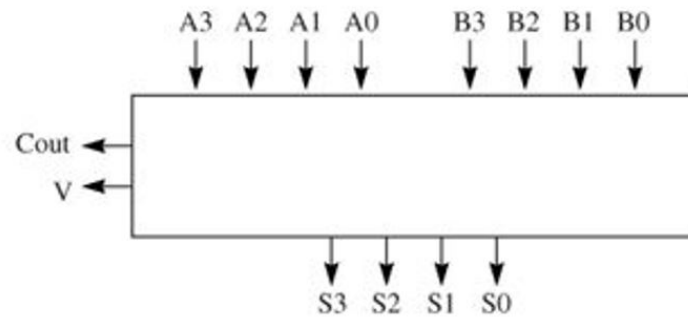
```
ARCHITECTURE behavior OF full_adder_test_bench IS
    COMPONENT full_adder
    PORT(
        a : IN  std_logic;
        b : IN  std_logic;
        c : IN  std_logic;
        sum : OUT std_logic;
        carry : OUT std_logic
    );
    END COMPONENT;

    --Inputs
    signal a : std_logic := '0';
    signal b : std_logic := '0';
    signal c : std_logic := '0';
```

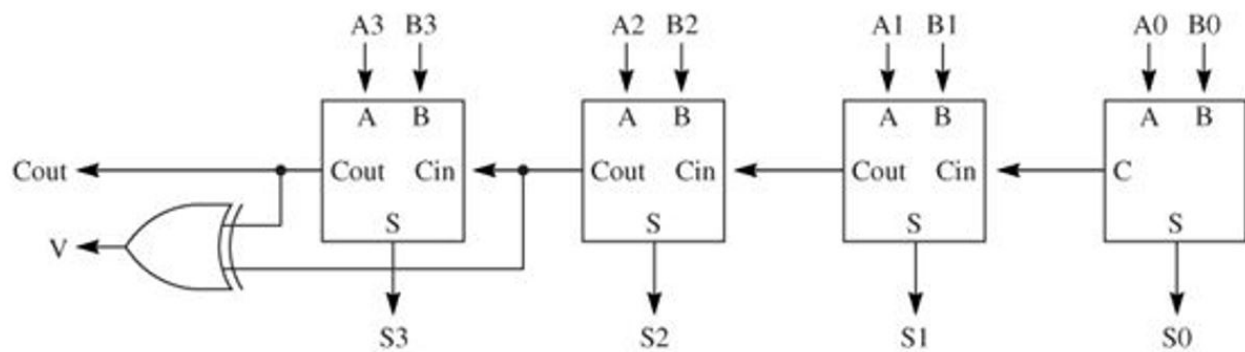


## 4-bit Ripple Carry Adder

### Block Diagram



### Circuit Diagram



### Truth Table

|     | A  |    |    |    | B  |    |    |    | Sum |    |    |    | Carry |
|-----|----|----|----|----|----|----|----|----|-----|----|----|----|-------|
| Cin | A3 | A2 | A1 | A0 | B3 | B2 | B1 | B0 | S3  | S2 | S1 | S0 | Cout  |
| 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0   | 0  | 0  | 0  | 0     |
| 0   | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0   | 0  | 1  | 0  | 0     |
| 0   | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0   | 1  | 0  | 0  | 0     |
| 0   | 0  | 0  | 1  | 1  | 0  | 0  | 1  | 1  | 0   | 1  | 1  | 0  | 0     |
| 0   | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 1   | 0  | 0  | 0  | 0     |
| 0   | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 1  | 1   | 0  | 1  | 0  | 0     |
| 0   | 0  | 1  | 1  | 0  | 0  | 1  | 1  | 0  | 1   | 1  | 0  | 0  | 0     |
| 0   | 0  | 1  | 1  | 1  | 0  | 1  | 1  | 1  | 1   | 1  | 1  | 0  | 0     |
| 0   | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0   | 0  | 0  | 0  | 1     |
| 0   | 1  | 0  | 0  | 1  | 1  | 0  | 0  | 1  | 0   | 0  | 1  | 0  | 1     |



|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## Code

### Package code

```

procedure ripple_carry_adder_4bit_proc(a: in std_logic_vector; b: in
std_logic_vector; c: in std_logic; s: out std_logic_vector) is
    variable c1, s1: std_logic_vector(4 downto 0);
    variable k: integer;
begin
    c1(0) := c;
    for k in 0 to 3 loop
        proc: full_adder_proc(a(k), b(k), c1(k), c1(k+1), s1(k));
    end loop;
    s1(4) := c1(4);
    s := s1;
end procedure;

```

### Implementation

```

entity ripple_carry_adder_4bit is
    Port ( a : in  STD_LOGIC_VECTOR (3 downto 0);
          b : in  STD_LOGIC_VECTOR (3 downto 0);
          c : in  STD_LOGIC;
          sum : out  STD_LOGIC_VECTOR (4 downto 0));
end ripple_carry_adder_4bit;

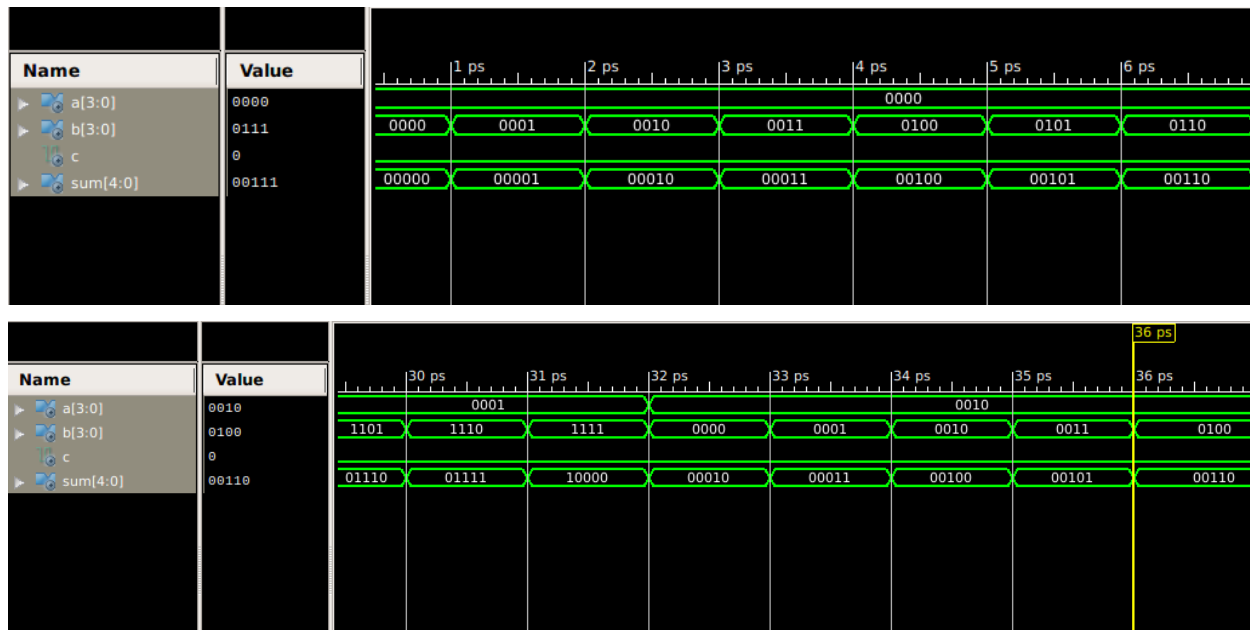
architecture Behavioral of ripple_carry_adder_4bit is
begin
    p1: process(a, b, c)
        variable s: std_logic_vector(4 downto 0);
    begin
        proc: ripple_carry_adder_4bit_proc(a, b, c, s);
        sum <= s;
    end process;
end Behavioral;

```

## Test Bench

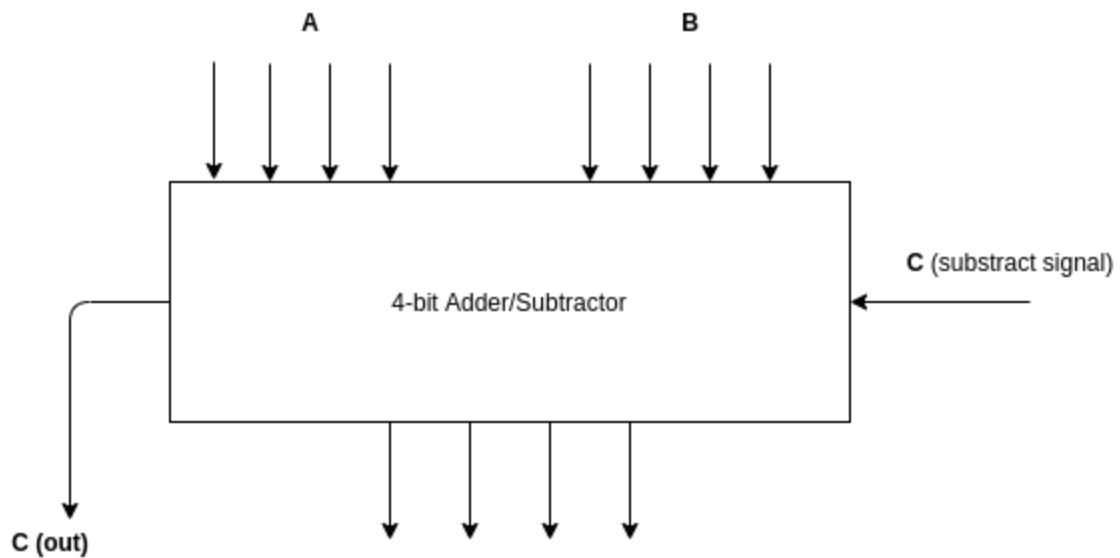
```
ARCHITECTURE behavior OF ripple_carry_adder_4bit_test_bench IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT ripple_carry_adder_4bit
    PORT(
        a : IN  std_logic_vector(3 downto 0);
        b : IN  std_logic_vector(3 downto 0);
        c : IN  std_logic;
        sum : 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');
    signal c : std_logic;
    --Outputs
    signal sum : std_logic_vector(4 downto 0);
BEGIN
    -- Instantiate the Unit Under Test (UUT)
    uut: ripple_carry_adder_4bit PORT MAP (
        a => a,
        b => b,
        c => c,
        sum => sum
    );
    -- Stimulus process
    stim_proc: process
        variable j,k: integer;
        variable bin_a, bin_b: std_logic_vector(3 downto 0);
    begin
        for j in 0 to 15 loop
            proc_a: dec_to_bin_proc(j, 4, bin_a);
            a <= bin_a;
            for k in 0 to 15 loop
                proc_b: dec_to_bin_proc(k, 4, bin_b);
                b <= bin_b;
                c <= '0';
                wait for 1ps;
            end loop;
        end loop;
    end process;
END;
```

## Timing Diagram

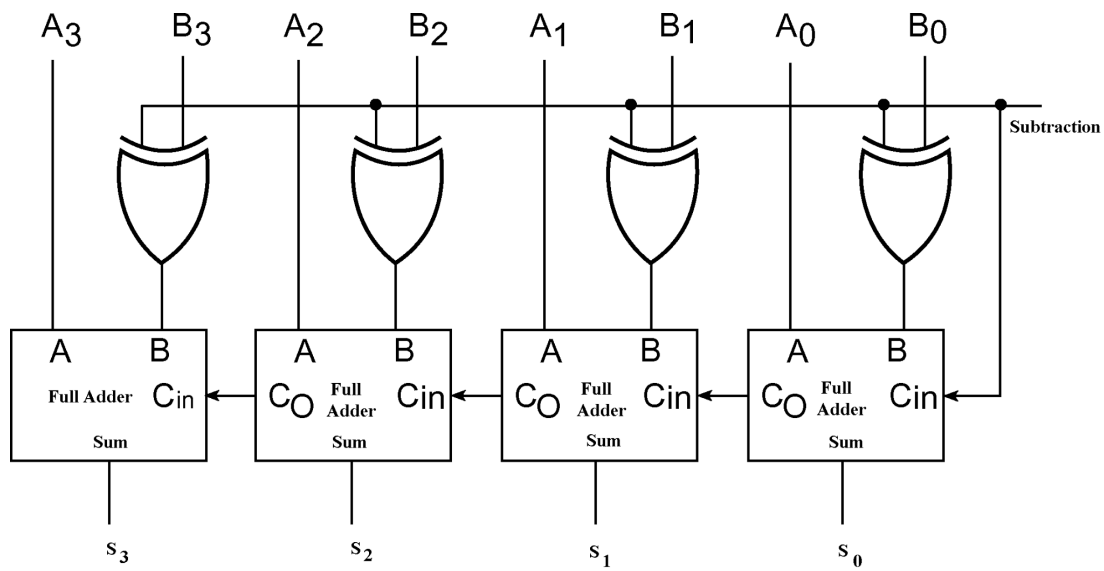


## Adder/Subtractor

## Block Diagram



## Circuit Diagram



## Code

## Package code

```

procedure adder_subtractor_proc(a: in std_logic_vector; b: in
std_logic_vector; c: in std_logic; s: out std_logic_vector) 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);
    proc: ripple_carry_adder_4bit_proc(a(3 downto 0), p(3 downto 0),
c, s1(4 downto 0));
    if c = '1' then
        s1(4) := not s1(4);
    end if;
    s := s1;
end procedure;

```

## Implementation

```

architecture Behavioral of adder_subtractor is
begin
    p1: process(a, b, c)
        variable s: std_logic_vector(4 downto 0);
    begin
        proc: adder_subtractor_proc(a, b, c, s);
        sum <= s;
    end process;
end Behavioral;

```

## Test Bench

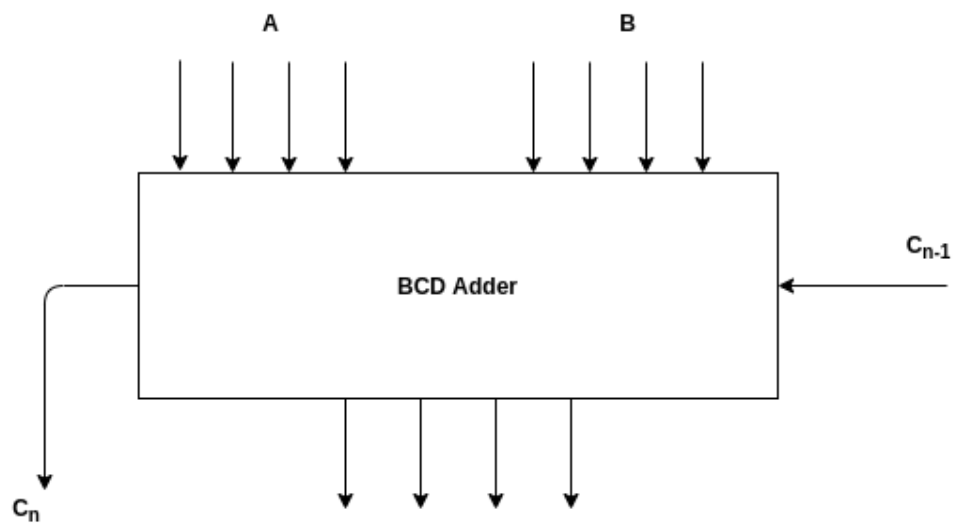
```
ARCHITECTURE behavior OF adder_subtractor_test_bench IS
    COMPONENT adder_subtractor
    PORT(
        a : IN  std_logic_vector(3 downto 0);
        b : IN  std_logic_vector(3 downto 0);
        c : IN  std_logic;
        sum : OUT std_logic_vector(4 downto 0)
    );
    END COMPONENT;
--Inputs
signal a : std_logic_vector(3 downto 0) := (others => '0');
signal b : std_logic_vector(3 downto 0) := (others => '0');
signal c : std_logic := '0';
--Outputs
signal sum : std_logic_vector(4 downto 0);
BEGIN
    -- Instantiate the Unit Under Test (UUT)
    uut: adder_subtractor PORT MAP (
        a => a,
        b => b,
        c => c,
        sum => sum
    );
    -- Stimulus process
    stim_proc: process
        variable bin_a, bin_b: std_logic_vector(3 downto 0);
    begin
        for i in 0 to 15 loop
            dec_to_bin_proc(i, 4, bin_a);
            a <= bin_a;
            for j in 0 to 15 loop
                dec_to_bin_proc(j, 4, bin_b);
                b <= bin_b;
                c <= '0';    -- for adding
                wait for 1ps;
                c <= '1';    -- for subtraction
                wait for 1ps;
            end loop;
        end loop;
    end process;
END;
```

## Timing Diagram

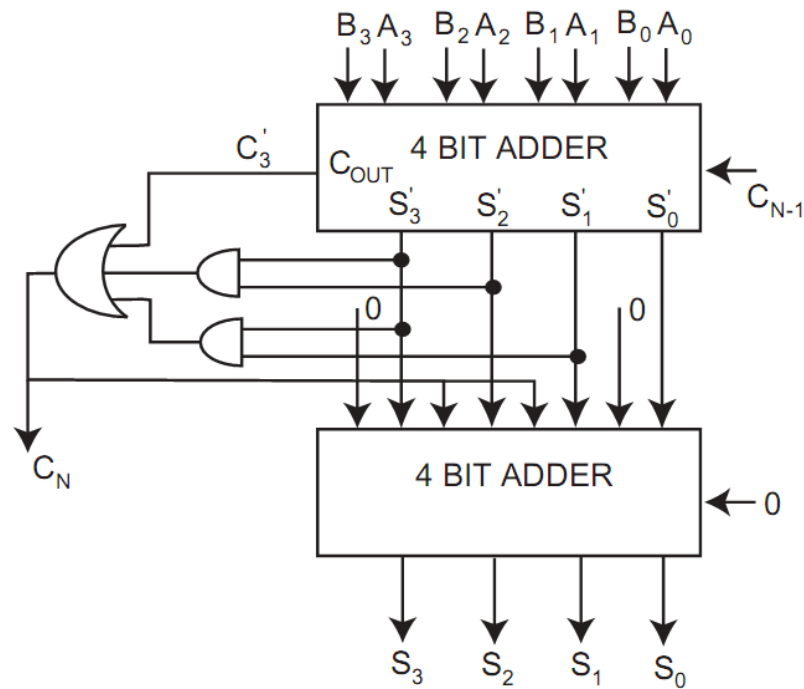


## BCD Adder

## Block Diagram



## Circuit Diagram



## Truth Table

| Binary Sum |    |    |    |    | BCD Sum |    |    |    |    | Decimal |
|------------|----|----|----|----|---------|----|----|----|----|---------|
| C          | B3 | B2 | B1 | B0 | C       | S3 | S2 | S1 | S0 |         |
| 0          | 0  | 0  | 0  | 0  | 0       | 0  | 0  | 0  | 0  | 0       |
| 0          | 0  | 0  | 0  | 1  | 0       | 0  | 0  | 0  | 1  | 1       |
| 0          | 0  | 0  | 1  | 0  | 0       | 0  | 0  | 1  | 0  | 2       |
| 0          | 0  | 0  | 1  | 1  | 0       | 0  | 0  | 1  | 1  | 3       |
| 0          | 0  | 1  | 0  | 0  | 0       | 0  | 1  | 0  | 0  | 4       |
| 0          | 0  | 1  | 0  | 1  | 0       | 0  | 1  | 0  | 1  | 5       |
| 0          | 0  | 1  | 1  | 0  | 0       | 0  | 1  | 1  | 0  | 6       |
| 0          | 0  | 1  | 1  | 1  | 0       | 0  | 1  | 1  | 1  | 7       |
| 0          | 1  | 0  | 0  | 0  | 0       | 1  | 0  | 0  | 0  | 8       |
| 0          | 1  | 0  | 0  | 1  | 0       | 1  | 0  | 0  | 1  | 9       |
| 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 |

## Code

### Package code

```

procedure bcd_adder_proc(a: in std_logic_vector; b: in std_logic_vector;
s: out std_logic_vector) 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';
    proc0: ripple_carry_adder_4bit_proc(a(3 downto 0), b(3 downto 0),
c, p(4 downto 0));
    z := p(4) or (p(3) and (p(2) or p(1)));
    q := c & z & z & c;
    proc1: ripple_carry_adder_4bit_proc(p(3 downto 0), q(3 downto 0),
c, s(4 downto 0));
    s(4) := z;
end procedure;

```

### Implementation

```

architecture Behavioral of bcd_adder is
begin

    p1: process(a, b)
        variable s: std_logic_vector(4 downto 0);
    begin
        proc: bcd_adder_proc(a, b, s);
        sum <= s;
    end process;

end Behavioral;

```



## Test Bench

```
ARCHITECTURE behavior OF bcd_adder_test_bench IS

    COMPONENT bcd_adder
    PORT (
        a : IN  std_logic_vector(3 downto 0);
        b : IN  std_logic_vector(3 downto 0);
        sum : 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 sum : std_logic_vector(4 downto 0);
BEGIN

    uut: bcd_adder PORT MAP (
        a => a,
        b => b,
        sum => sum
    );

    -- Stimulus process
    stim_proc: process
        variable bin_a, bin_b: std_logic_vector(3 downto 0);
    begin
        for i in 0 to 15 loop
            dec_to_bin_proc(i, 4, bin_a);
            a <= bin_a;
            for j in 0 to 15 loop
                dec_to_bin_proc(j, 4, bin_b);
                b <= bin_b;
                wait for 1ps;
            end loop;
        end loop;
    end process;
END;
```

## Timing Diagram

