ASSIGNMENT-07

VHDL

ECN-104

Digital Logic Design

Submitted By:

- Sarthak Shaurya
- 22116082
- O7
- ECE (B.Tech.)

QUESTION-1:

Develop a VHDL code for a 2 bit adder. Use any modeling style of your choice but mention which style you are using. Develop a testbench and show results.

SOLUTION:

A full adder has been designed using half adders. The full adder has then been used to design the two-bit binary adder.

The modelling style is **STRUCTURAL**.

CODE:

The code for designing half adder is as shown below:

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


entity ha is
Port (a,b : in STD_LOGIC;
sum, carry: out std_logic );
end ha;

architecture Behavioral of ha is

begin

process(a,b)begin
if(a='0' and b='0') then
sum<='0';
carry<='0';
elsif(a='1' and b='0') then
sum<='1';
carry<='0';
elsif(a='0' and b='1') then
sum<='1';
carry<='0';
elsif(a='1' and b='1') then
sum<='0';
```

```
carry<='1';

end if;

end process;
end Behavioral;
```

The code for designing full adder using half adder is as shown below:

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

entity fa is
  Port (a,b,cin: in std_logic;
    sum, carry: out std_logic );
end fa;


architecture Behavioral of fa is
    component ha is
    Port(a,b: in std_logic;
        sum,carry: out std_logic);
end component;
signal s1,s2,s3: std_logic;
begin

ha1: ha port map(a=>a, b=>b, sum=>s1, carry=>s2);
ha2: ha port map(a=>s1, b=>cin, sum=>sum, carry=>s3);

carry<=s3 or s2;

end Behavioral;
```

The code for designing the two-bit binary adder using full adder is as shown:

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

entity twobitadder is
  Port (A,B: in std_logic_vector(1 downto 0);
```

```
      Sum  : out std_logic_vector(1 downto 0);
      Carry: out std_logic);
end twobitadder;

architecture tba of twobitadder is

component fa is
   Port (a,b,cin: in std_logic;
      sum, carry: out std_logic );
end component;

signal c1: std_logic;
begin
fa1: fa port map(a=>A(0), b=>B(0), cin=>'0', sum=>Sum(0), carry=>c1);
fa2: fa port map(a=>A(1), b=>B(1), cin=>c1, sum=>Sum(1), carry=>Carry);


end tba;
```

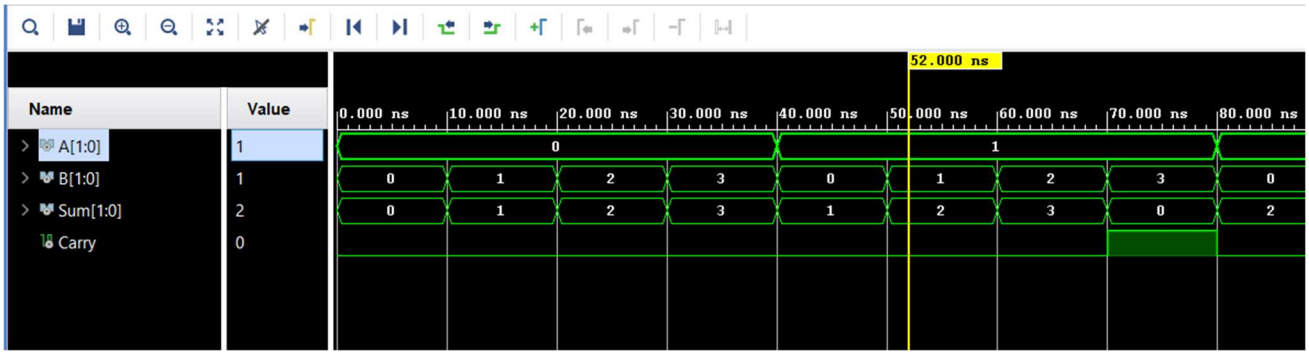
TESTBENCH:

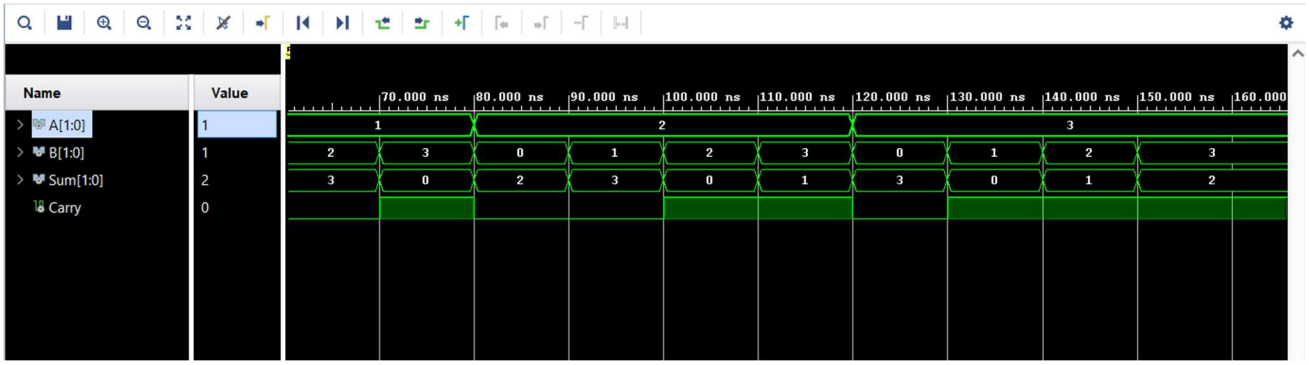
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity twobitadder_tb is
--  Port ( );
end twobitadder_tb;

architecture tba_tb of twobitadder_tb is
 component twobitadder is
  Port (A,B: in std_logic_vector(1 downto 0);
      Sum  : out std_logic_vector(1 downto 0);
      Carry: out std_logic);
 end component;

signal A,B: std_logic_vector(1 downto 0);
signal Sum: std_logic_vector(1 downto 0);
signal Carry: std_logic;

begin
```

```
uut: twobitadder port map ( A     => A,
                            B     => B,
                            Sum   => Sum,
                            Carry => Carry );

stimulus: process
begin
A<="00";
B<="00";
wait for 10ns;
A<="00";
B<="01";
wait for 10ns;
A<="00";
B<="10";
wait for 10ns;
A<="00";
B<="11";
wait for 10ns;
A<="01";
B<="00";
wait for 10ns;
A<="01";
B<="01";
wait for 10ns;
A<="01";
B<="10";
wait for 10ns;
A<="01";
B<="11";
wait for 10ns;
A<="10";
B<="00";
wait for 10ns;
A<="10";
B<="01";
wait for 10ns;
A<="10";
B<="10";
wait for 10ns;
A<="10";
B<="11";
wait for 10ns;
A<="11";
B<="00";
```

```
  wait for 10ns;
  A<="11";
  B<="01";
  wait for 10ns;
  A<="11";
  B<="10";
  wait for 10ns;
  A<="11";
  B<="11";
  wait for 10ns;
wait;
end process;
end;
```

SIMULATION RESULT:





QUESTION-2:

Develop a VHDL code for a J-K flip-flop with synchronous reset. Develop a testbench and show results.

SOLUTION:

The modelling style is **BEHAVIORAL**.

CODE:

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


entity jkff is
  Port (j,k,clk,reset: in std_logic;
        q: out std_logic);
end jkff;

architecture behavioral of jkff is



begin
process(clk, reset)
variable temp : std_logic;
begin

if(reset='1') then
temp:='0';
elsif(clk'event and clk='1') then
    temp:='0';
    if(j='1' and k='0') then
    temp:='1';
    elsif(j='0' and k='1') then
    temp:='0';
    elsif(j='1' and k='1') then
    temp:=not temp;
    elsif(j='0' and k='0') then
    temp:=temp;
    end if;
end if;
q<=temp;
end process;
```

```
end behavioral;
```

TESTBENCH:

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity jkff_tb is
-- Port();
end;

architecture bench of jkff_tb is

  component jkff
    Port (j,k,clk,reset: in std_logic;
          q: out std_logic);
  end component;

  signal j,k,clk,reset: std_logic;
  signal q: std_logic;

  constant clock_period: time := 10 ns;
  signal stop_the_clock: boolean;

begin

  uut: jkff port map ( j     => j,
                       k     => k,
                       clk   => clk,
                       reset => reset,
                       q     => q );

  stimulus: process
  begin
    j<='0';
    k<='0';
    reset<='0';
    wait for 10ns;
    j<='0';
    k<='1';
    reset<='0';
    wait for 10ns;
    j<='1';
```

```
    k<='0';
    reset<='0';
    wait for 10ns;
    j<='1';
    k<='1';
    reset<='0';
    wait for 10ns;
    j<='1';
    k<='1';
    reset<='1';
    wait for 10ns;
    j<='0';
    k<='0';
    reset<='0';
    wait for 10ns;
    j<='0';
    k<='1';
    reset<='0';
    wait for 10ns;
    j<='1';
    k<='0';
    reset<='0';
    wait for 10ns;
    j<='1';
    k<='1';
    reset<='0';
   stop_the_clock <= true;
   wait;
 end process;

 clocking: process
 begin
   while not stop_the_clock loop
     clk <= '0', '1' after clock_period / 2;
     wait for clock_period;
   end loop;
   wait;
 end process;
```

SIMULATION RESULT: