**DAY 1**

**OR Gate (Dataflow Model)**

**VHDL Code**

entity ORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end ORdf;

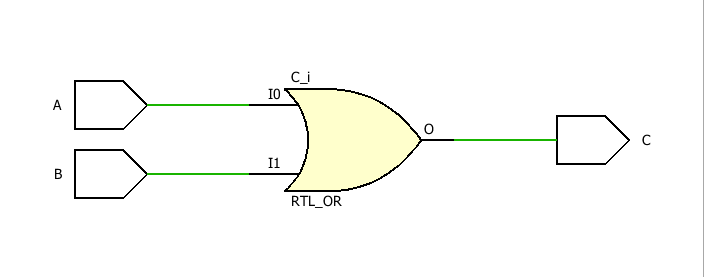
architecture Dataflow of ORdf is

begin

C <= A OR B;

end Dataflow;

**RTL Diagram**

****

**TBW Code**

entity ORdf\_tbw is

-- Port ( );

end ORdf\_tbw;

architecture Behavioral of ORdf\_tbw is

component ORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC:='0';

Signal B1: STD\_LOGIC:='0';

Signal C1: STD\_LOGIC;

begin

uut: ORdf port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

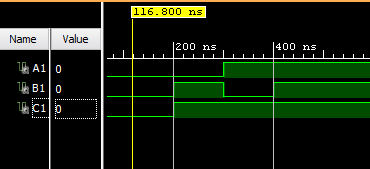
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**AND Gate (Dataflow Model)**

**VHDL Code**

entity ANDdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end ANDdf;

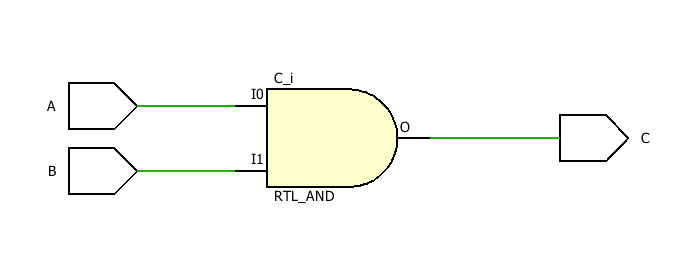
architecture Behavioral of ANDdf is

begin

C<=A AND B;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity ANDdf\_tbw is

-- Port ( );

end ANDdf\_tbw;

architecture Behavioral of ANDdf\_tbw is

component ANDdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC:='0';

Signal B1: STD\_LOGIC:='0';

Signal C1: STD\_LOGIC;

begin

uut: ANDdf port map(A=>A1,B=>B1,C=>C1);

stim\_proc: process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

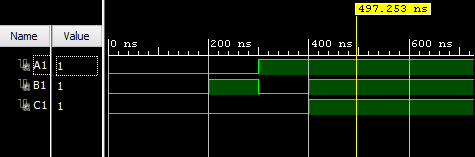
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**NOT Gate (Dataflow Model)**

**VHDL Code**

entity NOTdf is

Port ( A : in STD\_LOGIC;

B : out STD\_LOGIC);

end NOTdf;

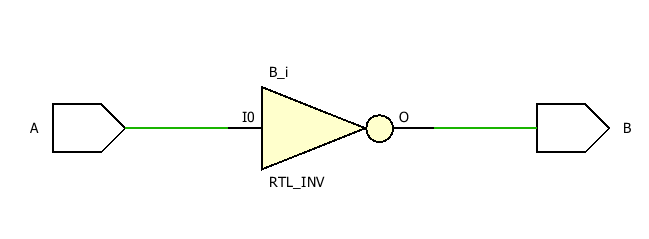
architecture Dataflow of NOTdf is

begin

B<=NOT A;

end Dataflow;

**RTL Diagram**

****

**TBW Code**

entity NOTdf\_tbw is

-- Port ( );

end NOTdf\_tbw;

architecture Behavioral of NOTdf\_tbw is

component NOTdf is

Port ( A : in STD\_LOGIC;

B : out STD\_LOGIC);

end component;

Signal A1:STD\_LOGIC:='0';

Signal B1:STD\_LOGIC;

begin

uut: NOTdf port map(A=>A1,B=>B1);

stim\_proc: process

begin

wait for 100 ns;

A1<='0';

wait for 100 ns;

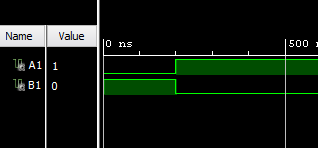
A1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**NAND Gate (Dataflow Model)**

**VHDL Code**

entity NANDdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end NANDdf;

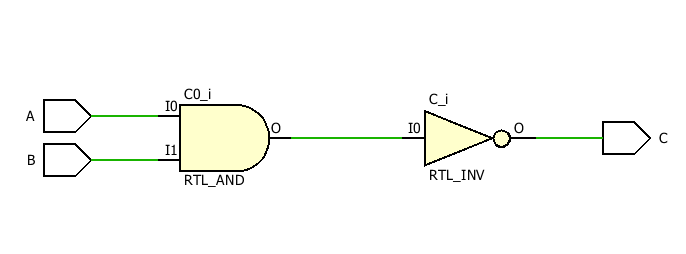
architecture Behavioral of NANDdf is

begin

C<=A NAND B;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity NANDdf\_tbw is

-- Port ( );

end NANDdf\_tbw;

architecture Behavioral of NANDdf\_tbw is

component NANDdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC:='0';

Signal B1: STD\_LOGIC:='0';

Signal C1: STD\_LOGIC;

begin

uut: NANDdf port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

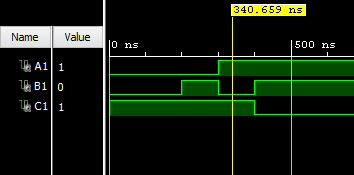
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**NOR Gate (Dataflow Model)**

**VHDL Code**

entity NORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end NORdf;

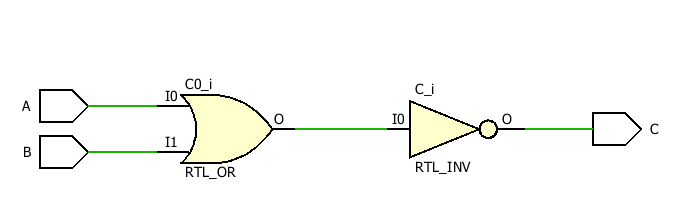
architecture Behavioral of NORdf is

begin

C<=A NOR B;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity NORdf\_tbw is

-- Port ( );

end NORdf\_tbw;

architecture Behavioral of NORdf\_tbw is

component NORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1:STD\_LOGIC:='0';

Signal B1:STD\_LOGIC:='0';

Signal C1:STD\_LOGIC;

begin

uut: NORdf port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

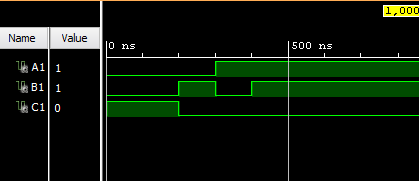
A1<='1';

B1<='1';

wait;

end process;

**Waveform**



**XOR Gate (Dataflow Model)**

**VHDL Code**

entity XORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XORdf;

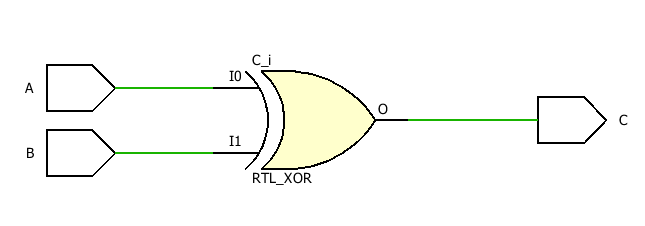
architecture Dataflow of XORdf is

begin

C<=A XOR B;

end Dataflow;

**RTL Diagram**

****

**TBW Code**

entity XORdf\_tbw is

-- Port ( );

end XORdf\_tbw;

architecture Behavioral of XORdf\_tbw is

component XORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1:STD\_LOGIC:='0';

Signal B1:STD\_LOGIC:='0';

Signal C1:STD\_LOGIC;

begin

uut: XORdf port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

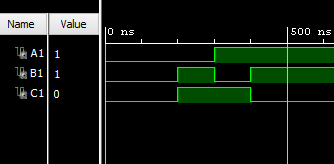
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**XNOR Gate (Dataflow Model)**

**VHDL Code**

entity XNORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XNORdf;

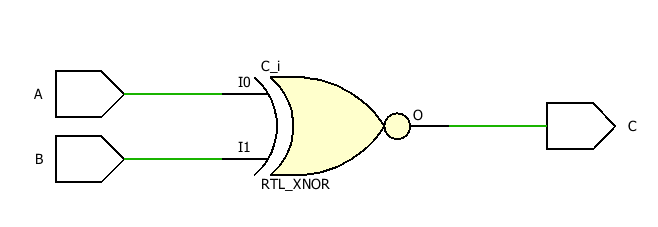
architecture Behavioral of XNORdf is

begin

C<=A XNOR B;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity XNORdf\_tbw is

-- Port ( );

end XNORdf\_tbw;

architecture Behavioral of XNORdf\_tbw is

component XNORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1:STD\_LOGIC:='0';

Signal B1:STD\_LOGIC:='0';

Signal C1:STD\_LOGIC;

begin

uut: XNORdf port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

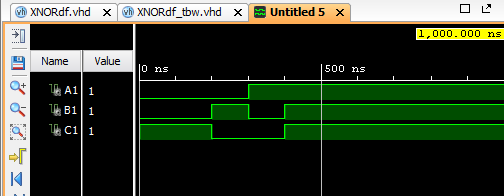
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**DAY 2**

**AND Gate using NAND (Dataflow Model)**

**VHDL Code**

entity NAND\_And is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end NAND\_And;

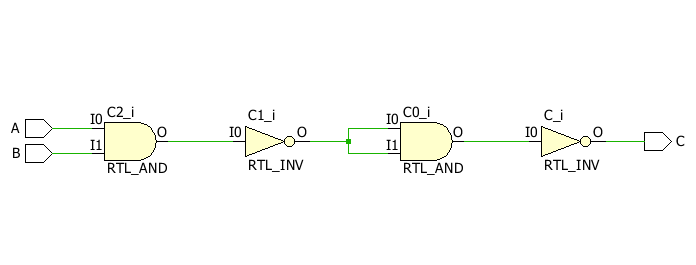
architecture Dataflow of NAND\_And is

begin

C<=(A NAND B) NAND (A NAND B);

end Dataflow;

**RTL Diagram**



**TBW Code**

entity NAND\_And\_tbw is

-- Port ( );

end NAND\_And\_tbw;

architecture Behavioral of NAND\_And\_tbw is

component NAND\_And is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: NAND\_And port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

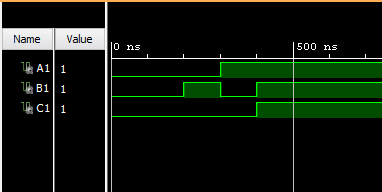
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**OR Gate using NAND (Dataflow Model)**

**VHDL Code**

entity NAND\_Or is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end NAND\_Or;

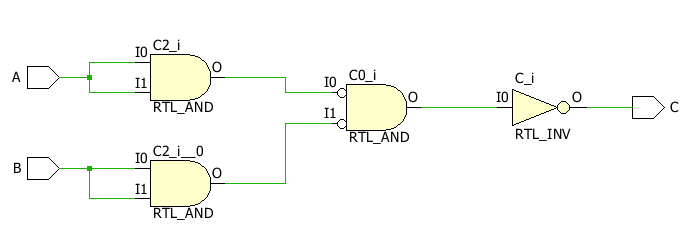
architecture Dataflow of NAND\_Or is

begin

C<=(A NAND A) NAND (B NAND B);

end Dataflow;

**RTL Diagram**



**TBW Code**

entity NAND\_Ortbw is

-- Port ( );

end NAND\_Ortbw;

architecture Behavioral of NAND\_Ortbw is

component NAND\_Or is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: NAND\_Or port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

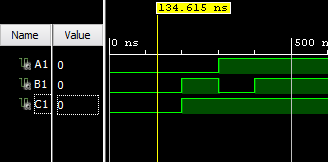
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**NOT Gate using NAND (Dataflow Model)**

**VHDL Code**

entity NAND\_Not is

Port ( A : in STD\_LOGIC;

C : out STD\_LOGIC);

end NAND\_Not;

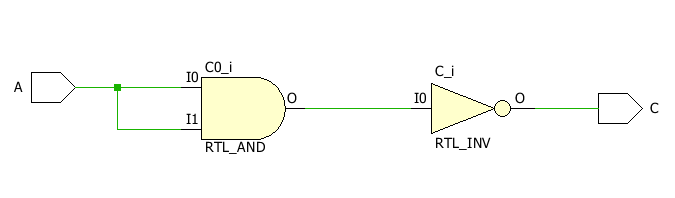
architecture Dataflow of NAND\_Not is

begin

C<=A NAND A;

end Dataflow;

**RTL Diagram**



**TBW Code**

entity NAND\_Nottbw is

-- Port ( );

end NAND\_Nottbw;

architecture Behavioral of NAND\_Nottbw is

component NAND\_Not is

Port ( A : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: NAND\_Not port map(A=>A1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

wait for 100 ns;

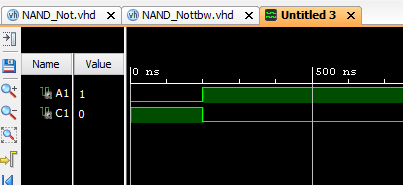
A1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**AND Gate using NOR (Dataflow Model)**

**VHDL Code**

entity NOR\_And is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end NOR\_And;

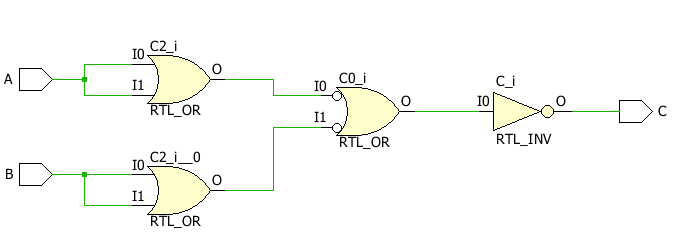
architecture Dataflow of NOR\_And is

begin

C<=(A NOR A)NOR(B NOR B);

end Dataflow;

**RTL Diagram**



**TBW Code**

entity NOR\_Andtbw is

-- Port ( );

end NOR\_Andtbw;

architecture Behavioral of NOR\_Andtbw is

component NOR\_And is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: NOR\_And port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

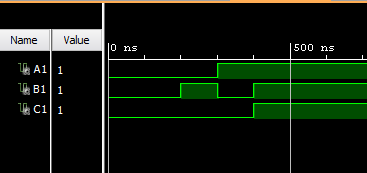
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**OR Gate using NOR (Dataflow Model)**

**VHDL Code**

entity NOR\_Or is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end NOR\_Or;

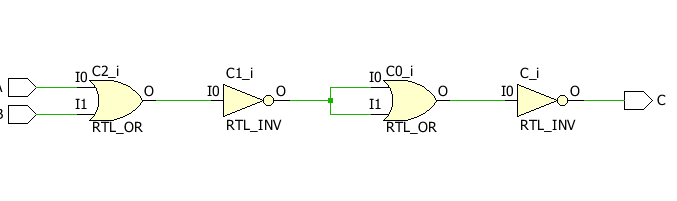
architecture Dataflow of NOR\_Or is

begin

C<=(A NOR B) NOR (A NOR B);

end Dataflow;

**RTL Diagram**



**TBW Code**

entity NOR\_Ortbw is

-- Port ( );

end NOR\_Ortbw;

architecture Behavioral of NOR\_Ortbw is

component NOR\_Or is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: NOR\_Or port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

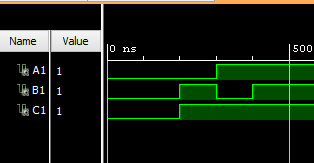
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**NOT Gate using NOR (Dataflow Model)**

**VHDL Code**

entity NOR\_Not is

Port ( A : in STD\_LOGIC;

C : out STD\_LOGIC);

end NOR\_Not;

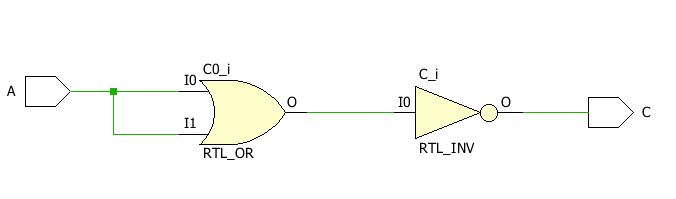
architecture Dataflow of NOR\_Not is

begin

C<=(A NOR A);

end Dataflow;

**RTL Diagram**



**TBW Code**

entity NOR\_Nottbw is

-- Port ( );

end NOR\_Nottbw;

architecture Behavioral of NOR\_Nottbw is

component NOR\_Not is

Port ( A : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: NOR\_Not port map(A=>A1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

wait for 100 ns;

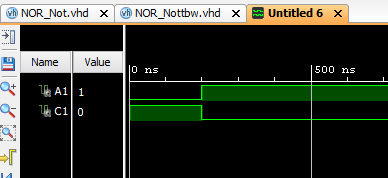
A1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**DAY 3**

**AND Gate (Behavioral Model)**

**VHDL Code**

entity AndBh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end AndBh;

architecture Behavioral of AndBh is

begin

process(A,B)

begin

if (A = '1' AND B = '1') then

C<='1';

else

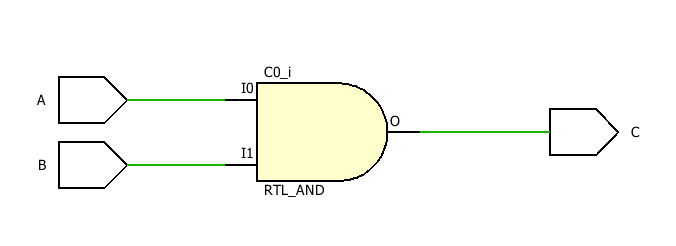
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

entity AndBh\_tbw is

-- Port ( );

end AndBh\_tbw;

architecture Behavioral of AndBh\_tbw is

component AndBh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: AndBh port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

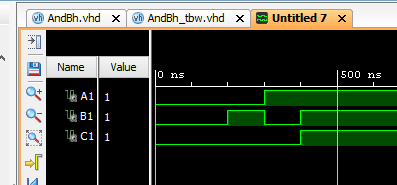
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**OR Gate (Behavioral Model)**

**VHDL Code**

entity ORbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end ORbh;

architecture Behavioral of ORbh is

begin

process(A,B)

begin

if(A = '1' OR B = '1') then

C<='1';

else

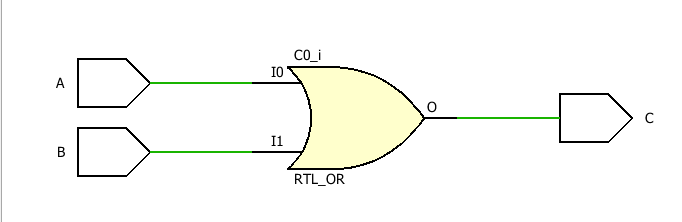
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: OrBh port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

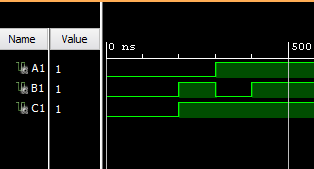
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**NOT Gate (Behavioral Model)**

**VHDL Code**

entity Notbh is

Port ( A : in STD\_LOGIC;

C : out STD\_LOGIC);

end Notbh;

architecture Behavioral of Notbh is

begin

process(A)

if (A = '1') then

B<='0';

elsif (A = '0') then

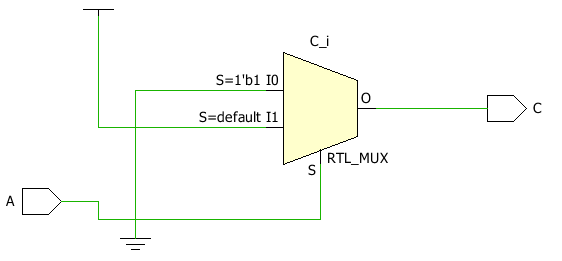
B<='1';

end elif;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

entity Notbh\_tbw is

-- Port ( );

end Notbh\_tbw;

architecture Behavioral of Notbh\_tbw is

component Notbh is

Port ( A : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: Notbf port map(A=>A1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

wait for 100 ns;

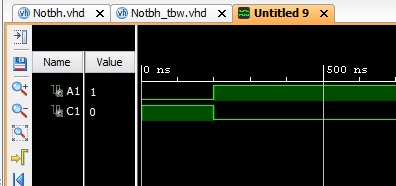
A1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**NAND Gate (Behavioral Model)**

**VHDL Code**

entity Nandbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end Nandbh;

architecture Behavioral of Nandbh is

begin

process(A,B)

begin

if(A = '0' OR B = '0') then

C<='1';

else

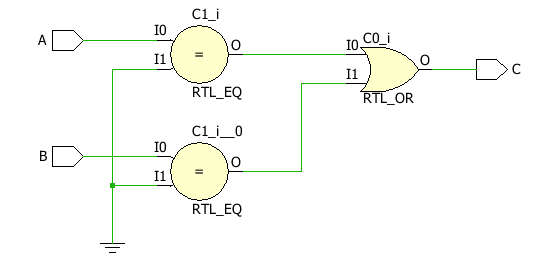
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

entity Nandbf\_tbw is

-- Port ( );

end Nandbf\_tbw;

architecture Behavioral of Nandbf\_tbw is

component Nandbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: Nandbh port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

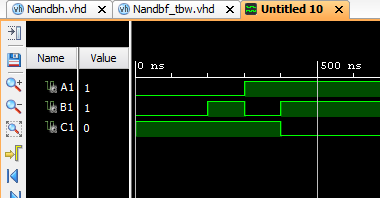
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**NOR Gate (Behavioral Model)**

**VHDL Code**

entity Norbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end Norbh;

architecture Behavioral of Norbh is

begin

process(A,B)

begin

if ( A = '0' AND B = '0') then

C<='1';

else

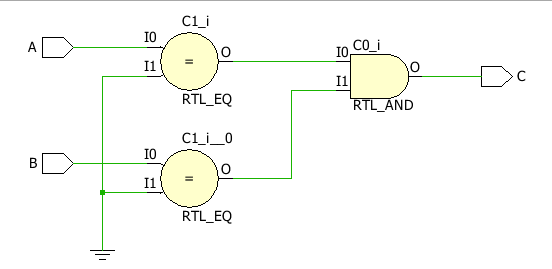
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

entity Norbh\_tbw is

-- Port ( );

end Norbh\_tbw;

architecture Behavioral of Norbh\_tbw is

component Norbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: Norbh port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

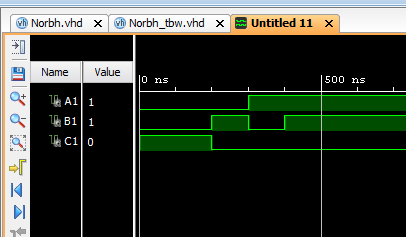
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**XOR Gate (Behavioral Model)**

**VHDL Code**

entity Xorbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end Xorbh;

architecture Behavioral of Xorbh is

begin

process(A,B)

begin

if(A = B) then

C<='0';

else

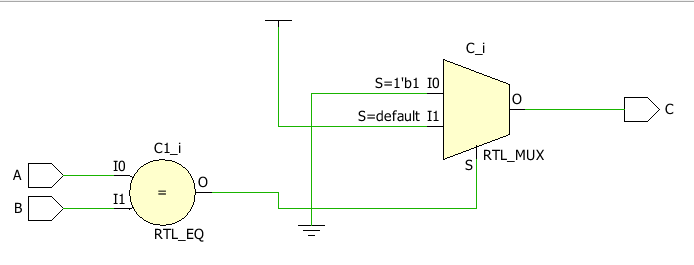
C<='1';

end if;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

entity Xorbh\_tbw is

-- Port ( );

end Xorbh\_tbw;

architecture Behavioral of Xorbh\_tbw is

component Xorbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: Xorbh port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

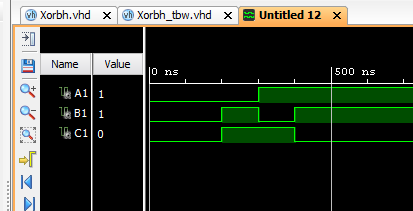
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**XNOR Gate (Behavioral Model)**

**VHDL Code**

entity Xnorbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end Xnorbh;

architecture Behavioral of Xnorbh is

begin

process(A,B)

begin

if(A = B) then

C<='1';

else

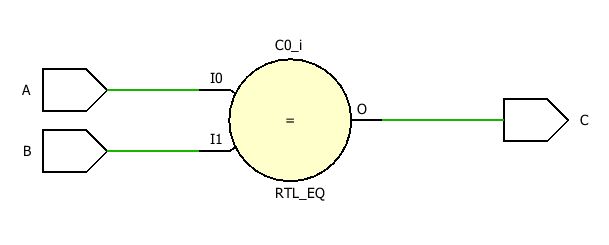
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

entity Xnorbf\_tbw is

-- Port ( );

end Xnorbf\_tbw;

architecture Behavioral of Xnorbf\_tbw is

component Xnorbh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

begin

uut: Xnorbh port map(A=>A1,B=>B1,C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

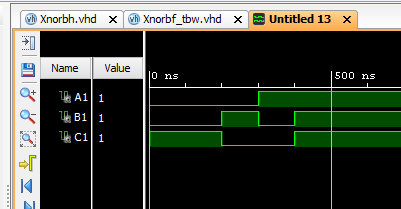
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**DAY 4**

**Half Adder (Dataflow)**

**VHDL Code**

entity HalfAdder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end HalfAdder;

architecture Dataflow of HalfAdder is

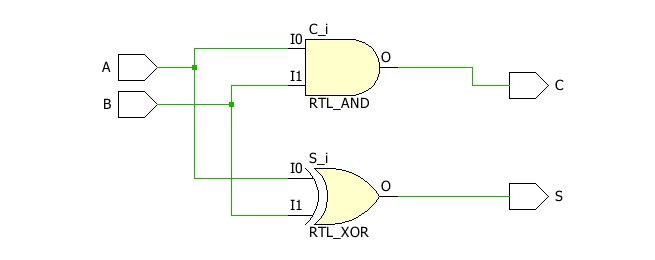
begin

S<=A XOR B;

C<=A AND B;

end Dataflow;

**RTL Diagram**

****

**TBW Code**

entity HalfAdder\_tbw is

-- Port ( );

end HalfAdder\_tbw;

architecture Behavioral of HalfAdder\_tbw is

component HalfAdder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC;

Signal S1: STD\_LOGIC;

begin

uut: HalfAdder port map(A=>A1,B=>B1,C=>C1,S=>S1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

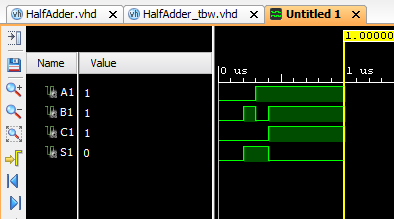
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**Full Adder (Dataflow)**

**VHDL Code**

entity FullAdder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

S : out STD\_LOGIC;

Y : out STD\_LOGIC);

end FullAdder;

architecture Dataflow of FullAdder is

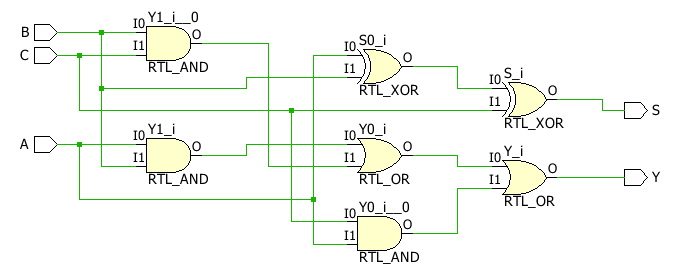
begin

S<=(A XOR B) XOR C;

Y<=(A AND B) OR (B AND C) OR (C AND A);

end Dataflow;

**RTL Diagram**

****

**TBW Code**

entity FullAdder\_tbw is

-- Port ( );

end FullAdder\_tbw;

architecture Behavioral of FullAdder\_tbw is

component FullAdder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

S : out STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC := '0';

Signal S1: STD\_LOGIC;

Signal Y1: STD\_LOGIC;

begin

uut: FullAdder port map(A=>A1,B=>B1,C=>C1,S=>S1,Y=>Y1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

C1<='0';

wait for 100 ns;

A1<='0';

B1<='0';

C1<='1';

wait for 100 ns;

A1<='0';

B1<='1';

C1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

C1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

C1<='0';

wait for 100 ns;

A1<='1';

B1<='0';

C1<='1';

wait for 100 ns;

A1<='1';

B1<='1';

C1<='0';

wait for 100 ns;

A1<='1';

B1<='1';

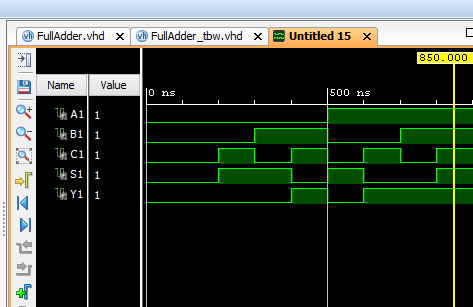
C1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**Half Adder (Behavioural)**

**VHDL Code**

entity HalfAdderBh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end HalfAdderBh;

architecture Behavioral of HalfAdderBh is

begin

process(A,B)

begin

if(A=B) then

S<='0';

else

S<='1';

end if;

if(A='1' AND B='1') then

C<='1';

else

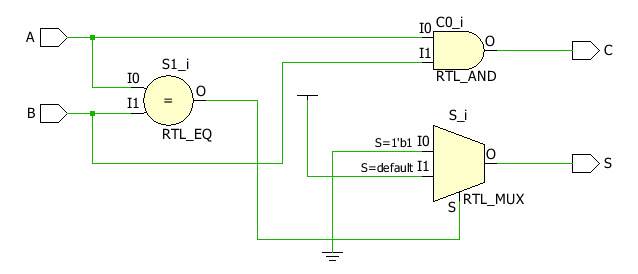
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity HalfAdderBh\_tbw is

-- Port ( );

end HalfAdderBh\_tbw;

architecture Behavioral of HalfAdderBh\_tbw is

component HalfAdderBh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '1';

Signal B1: STD\_LOGIC := '0';

Signal S1: STD\_LOGIC;

Signal C1: STD\_LOGIC;

begin

uut: HalfAdderBh port map(A=>A1, B=>B1, S=>S1, C=>C1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

wait for 100 ns;

A1<='1';

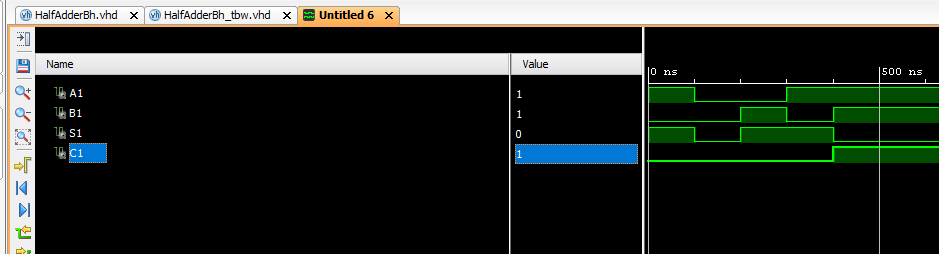
B1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**Full Adder (Behavioural)**

**VHDL Code**

entity FullAdderBh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

S : out STD\_LOGIC;

Y : out STD\_LOGIC);

end FullAdderBh;

architecture Behavioral of FullAdderBh is

begin

process(A,B,C)

begin

if(A='0') then

if(B=C) then

S<='0';

else

S<='1';

end if;

if(B='1' AND C='1') then

Y<='1';

else

Y<='0';

end if;

else

if(B=C) then

S<='1';

else

S<='0';

end if;

if(B='1' OR C='1') then

Y<='1';

else

Y<='0';

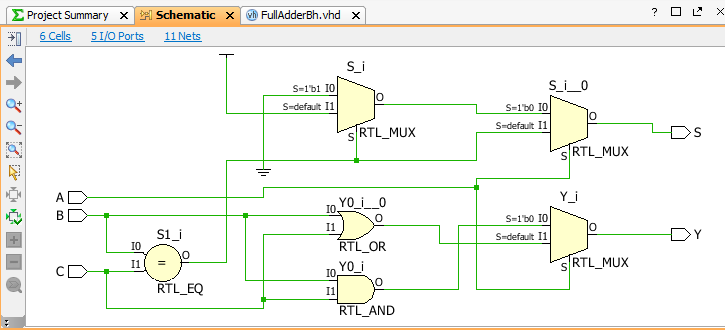
end if;

end if;

end process;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity FullAdderBh\_tbw is

-- Port ( );

end FullAdderBh\_tbw;

architecture Behavioral of FullAdderBh\_tbw is

component FullAdderBh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

S : out STD\_LOGIC;

Y : out STD\_LOGIC);

end Component;

Signal A1: STD\_LOGIC := '0';

Signal B1: STD\_LOGIC := '0';

Signal C1: STD\_LOGIC := '0';

Signal S1: STD\_LOGIC;

Signal Y1: STD\_LOGIC;

begin

uut: FullAdderBh port map(A=>A1,B=>B1,C=>C1,S=>S1,Y=>Y1);

stim\_proc:process

begin

wait for 100 ns;

A1<='0';

B1<='0';

C1<='0';

wait for 100 ns;

A1<='0';

B1<='0';

C1<='1';

wait for 100 ns;

A1<='0';

B1<='1';

C1<='0';

wait for 100 ns;

A1<='0';

B1<='1';

C1<='1';

wait for 100 ns;

A1<='1';

B1<='0';

C1<='0';

wait for 100 ns;

A1<='1';

B1<='0';

C1<='1';

wait for 100 ns;

A1<='1';

B1<='1';

C1<='0';

wait for 100 ns;

A1<='1';

B1<='1';

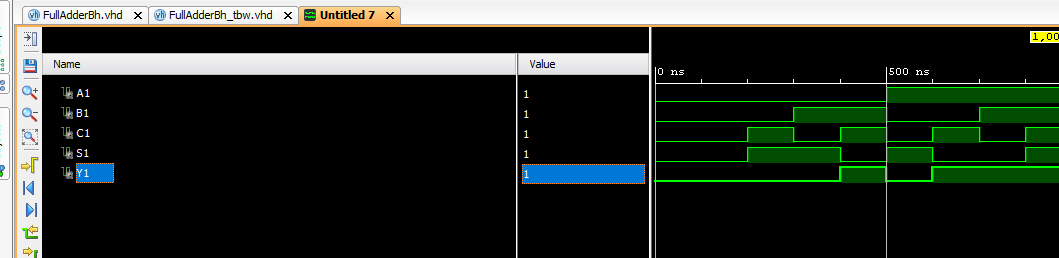
C1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**2:1 MUX (Dataflow)**

**VHDL Code**

entity MUX21df is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end MUX21df;

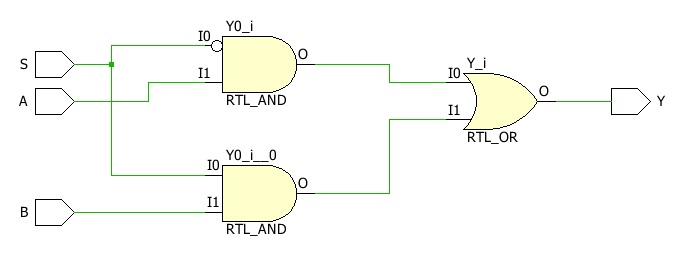
architecture Dataflow of MUX21df is

begin

Y<=((NOT S) AND A) OR (S AND B);

end Dataflow;

**RTL Diagram**

****

**TBW Code**

entity MUX21df\_tbw is

-- Port ( );

end MUX21df\_tbw;

architecture Behavioral of MUX21df\_tbw is

component MUX21df is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '1';

Signal B1: STD\_LOGIC := '0';

Signal S1: STD\_LOGIC := '0';

Signal Y1: STD\_LOGIC;

begin

uut: MUX21df port map(A=>A1, B=>B1, S=>S1, Y=>Y1);

stim\_proc:process

begin

wait for 100 ns;

S1<='0';

wait for 100 ns;

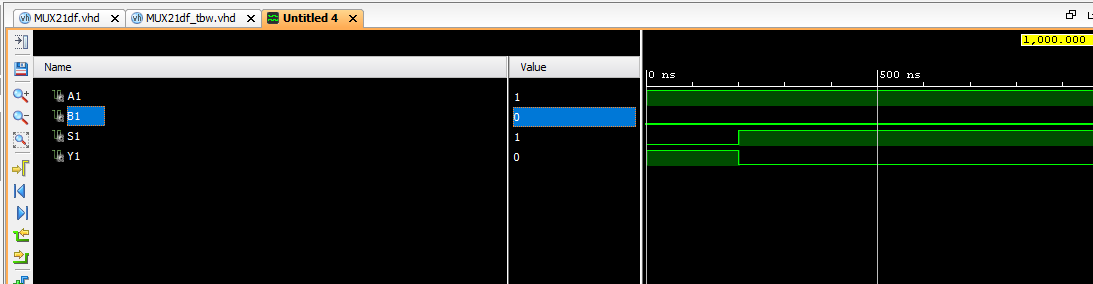
S1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**2:1 MUX (Behavioural)**

**VHDL Code**

entity MUX21Bh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end MUX21Bh;

architecture Behavioral of MUX21Bh is

begin

process(A,B)

begin

if(S = '1') then

Y<=B;

else

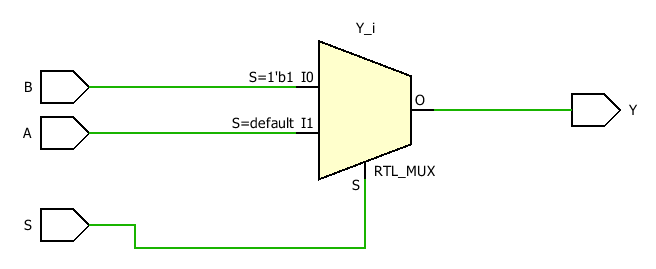
Y<=A;

end if;

end process;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity MUX21Bh\_tbw is

-- Port ( );

end MUX21Bh\_tbw;

architecture Behavioral of MUX21Bh\_tbw is

component MUX21Bh is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC := '1';

Signal B1: STD\_LOGIC := '0';

Signal S1: STD\_LOGIC := '0';

Signal Y1: STD\_LOGIC;

begin

uut: MUX21Bh port map(A=>A1, B=>B1, S=>S1, Y=>Y1);

stim\_proc:process

begin

wait for 100 ns;

S1<='0';

wait for 100 ns;

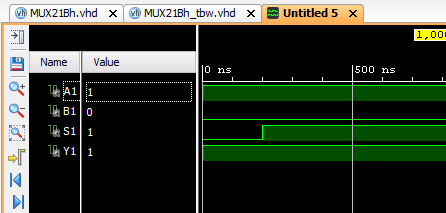
S1<='1';

wait;

end process;

end Behavioral;

**Waveform**



**DAY 5**

**4:1 MUX (Dataflow)**

**VHDL Code**

entity MUX41 is

Port ( ip : in STD\_LOGIC\_VECTOR (3 downto 0);

sel : in STD\_LOGIC\_VECTOR (1 downto 0);

Y : out STD\_LOGIC);

end MUX41;

architecture dataflow of MUX41 is

begin

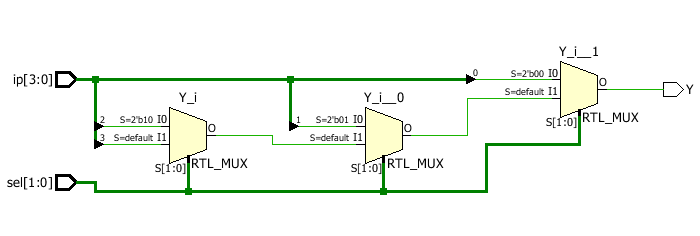
Y <= ip(0) when sel="00" else

ip(1) when sel="01" else

ip(2) when sel="10" else ip(3);

end dataflow;

**RTL Diagram**

****

**TBW Code**

entity MUX41\_tbw is

-- Port ( );

end MUX41\_tbw;

architecture Behavioral of MUX41\_tbw is

component MUX41 is

Port ( ip : in STD\_LOGIC\_VECTOR (3 downto 0);

sel : in STD\_LOGIC\_VECTOR (1 downto 0);

Y : out STD\_LOGIC);

end component;

Signal ip1: STD\_LOGIC\_VECTOR (3 downto 0) := "1010";

Signal sel1: STD\_LOGIC\_VECTOR (1 downto 0):= "00";

Signal y1: STD\_LOGIC;

begin

uut: MUX41 port map(ip=>ip1,sel=>sel1,Y=>y1);

stim\_proc:process

begin

wait for 100 ns;

sel1<="00";

wait for 100 ns;

sel1<="01";

wait for 100 ns;

sel1<="10";

wait for 100 ns;

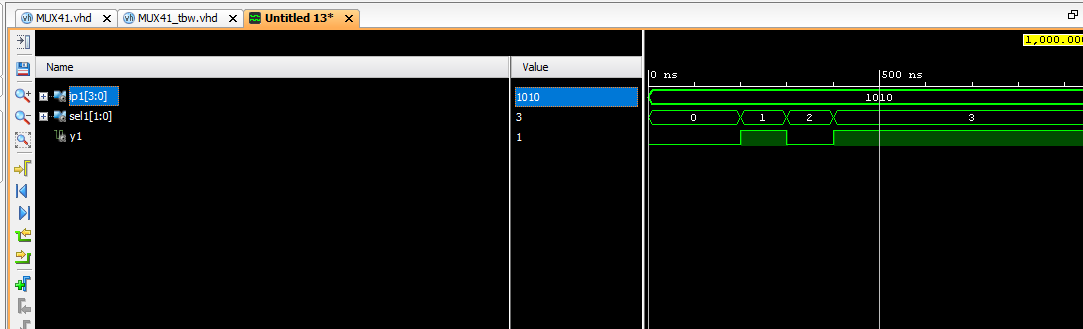
sel1<="11";

wait;

end process;

end Behavioral;

**Waveform**

****

**4:1 MUX (Behavioural)**

**VHDL Code**

entity mux41beh is

Port ( ip : in STD\_LOGIC\_VECTOR (3 downto 0);

sel : in STD\_LOGIC\_VECTOR (1 downto 0);

y : out STD\_LOGIC);

end mux41beh;

architecture Behavioral of mux41beh is

begin

process(ip, sel)

begin

case sel is

when "00" => y<=ip(0);

when "01" => y<=ip(1);

when "10" => y<=ip(2);

when "11" => y<=ip(3);

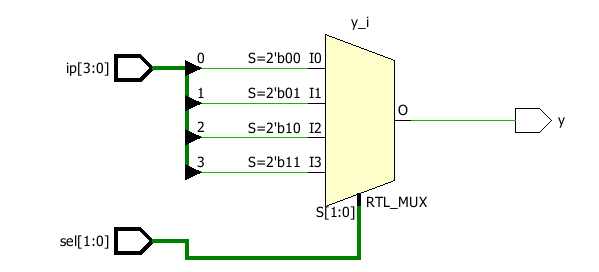
when others => NULL;

end case;

end process;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity mux41beh\_tbw is

-- Port ( );

end mux41beh\_tbw;

architecture Behavioral of mux41beh\_tbw is

component mux41beh is

Port ( ip : in STD\_LOGIC\_VECTOR (3 downto 0);

sel : in STD\_LOGIC\_VECTOR (1 downto 0);

y : out STD\_LOGIC);

end component;

Signal ip1: STD\_LOGIC\_VECTOR (3 downto 0) := "1010";

Signal sel1: STD\_LOGIC\_VECTOR (1 downto 0):= "00";

Signal y1: STD\_LOGIC;

begin

uut: mux41beh port map(ip=>ip1,sel=>sel1,y=>y1);

stim\_proc:process

begin

wait for 100 ns;

sel1<="00";

wait for 100 ns;

sel1<="01";

wait for 100 ns;

sel1<="10";

wait for 100 ns;

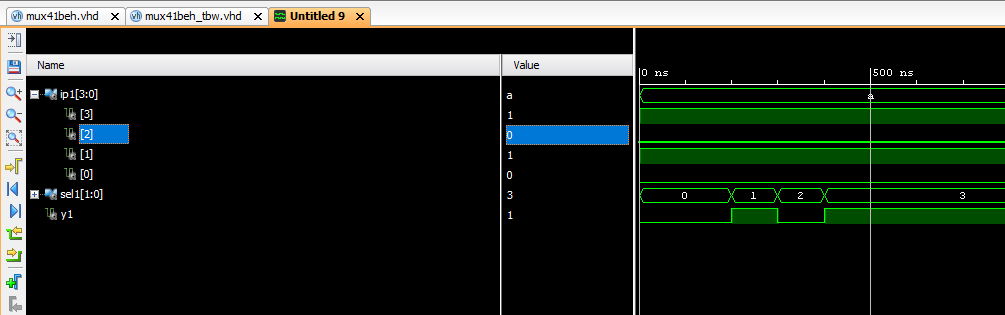
sel1<="11";

wait;

end process;

end Behavioral;

**Waveform**



**3:8 Decoder (Dataflow)**

**VHDL Code**

entity Decoder38 is

Port ( ip : in STD\_LOGIC\_VECTOR (2 downto 0);

op : out STD\_LOGIC\_VECTOR (7 downto 0));

end Decoder38;

architecture Dataflow of Decoder38 is

begin

op<=”00000000”;

op(0)<='1' when ip="000";

op(1)<='1' when ip="001";

op(2)<='1' when ip="010";

op(3)<='1' when ip="011";

op(4)<='1' when ip="100";

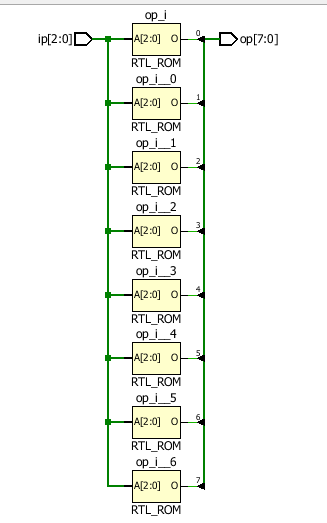
op(5)<='1' when ip="101";

op(6)<='1' when ip="110";

op(7)<='1' when ip="111";

end Dataflow;

**RTL Diagram**

****

**TBW Code**

architecture Behavioral of Decoder38\_tbw is

component Decoder38 is

Port ( ip : in STD\_LOGIC\_VECTOR (2 downto 0);

op : out STD\_LOGIC\_VECTOR (7 downto 0));

end component;

Signal ip1:STD\_LOGIC\_VECTOR (2 downto 0) := "000";

Signal op1:STD\_LOGIC\_VECTOR (7 downto 0);

begin

uut: Decoder38 port map(ip=>ip1,op=>op1);

stim\_proc:process

begin

wait for 20ns;

ip1<="001";

wait for 20ns;

ip1<="010";

wait for 20ns;

ip1<="011";

wait for 20ns;

ip1<="100";

wait for 20ns;

ip1<="101";

wait for 20ns;

ip1<="110";

wait for 20ns;

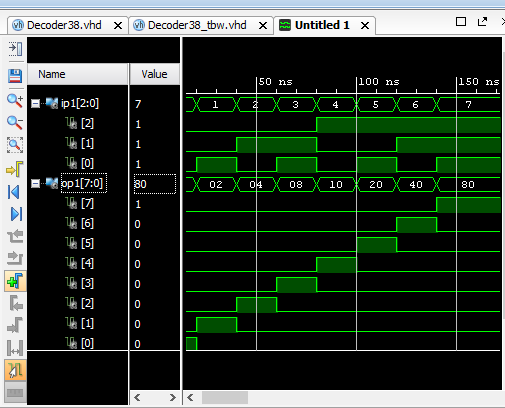
ip1<="111";

wait;

end process;

end Behavioral;

**Waveform**

****

**3:8 Decoder (Behavioural)**

**VHDL Code**

entity Decoder38Beh is

Port ( ip : in STD\_LOGIC\_VECTOR (2 downto 0);

op : out STD\_LOGIC\_VECTOR (7 downto 0));

end Decoder38Beh;

architecture Behavioral of Decoder38Beh is

begin

process(ip)

begin

op<="00000000";

case ip is

when "000" => op(0) <= '1' ;

when "001" => op(1) <= '1' ;

when "010" => op(2) <= '1' ;

when "011" => op(3) <= '1' ;

when "100" => op(4) <= '1' ;

when "101" => op(5) <= '1' ;

when "110" => op(6) <= '1' ;

when "111" => op(7) <= '1' ;

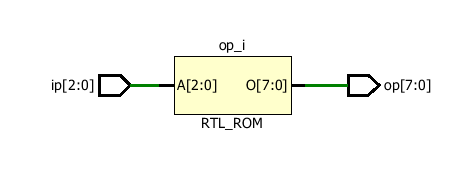
when others => NULL;

end case;

end process;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity Decoder38Beh\_tbw is

-- Port ( );

end Decoder38Beh\_tbw;

architecture Behavioral of Decoder38Beh\_tbw is

component Decoder38Beh is

Port ( ip : in STD\_LOGIC\_VECTOR (2 downto 0);

op : out STD\_LOGIC\_VECTOR (7 downto 0));

end component;

Signal ip1:STD\_LOGIC\_VECTOR (2 downto 0) := "000";

Signal op1:STD\_LOGIC\_VECTOR (7 downto 0);

begin

uut: Decoder38Beh port map(ip=>ip1,op=>op1);

stim\_proc:process

begin

wait for 20ns;

ip1<="001";

wait for 20ns;

ip1<="010";

wait for 20ns;

ip1<="011";

wait for 20ns;

ip1<="100";

wait for 20ns;

ip1<="101";

wait for 20ns;

ip1<="110";

wait for 20ns;

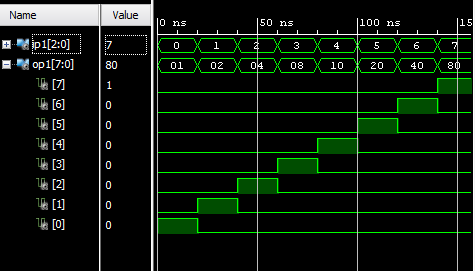
ip1<="111";

wait;

end process;

end Behavioral;

**Waveform**

****

**4 Bit Comparator (Dataflow)**

**VHDL Code**

entity Comp4 is

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0);

eq : out STD\_LOGIC;

gt : out STD\_LOGIC;

lt : out STD\_LOGIC);

end Comp4;

architecture Dataflow of Comp4 is

begin

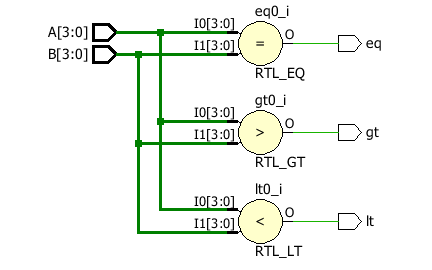
eq<='1' when A=B else '0';

gt<='1' when A>B else '0';

lt<='1' when A<B else '0';

end Dataflow;

**RTL Diagram**

****

**TBW Code**

entity Comp4\_tbw is

-- Port ( );

end Comp4\_tbw;

architecture Behavioral of Comp4\_tbw is

component Comp4 is

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0);

eq : out STD\_LOGIC;

gt : out STD\_LOGIC;

lt : out STD\_LOGIC);

end component;

Signal A1:STD\_LOGIC\_VECTOR (3 downto 0):="0000";

Signal B1:STD\_LOGIC\_VECTOR (3 downto 0):="0000";

Signal eq1:STD\_LOGIC;

Signal gt1:STD\_LOGIC;

Signal lt1:STD\_LOGIC;

begin

uut:Comp4 port map(A=>A1,B=>B,eq=>eq1,gt=>gt1,lt=>lt1);

stim\_proc:process

begin

wait for 50 ns;

A1<="0001";

B1<="1000";

wait for 50 ns;

A1<="1100";

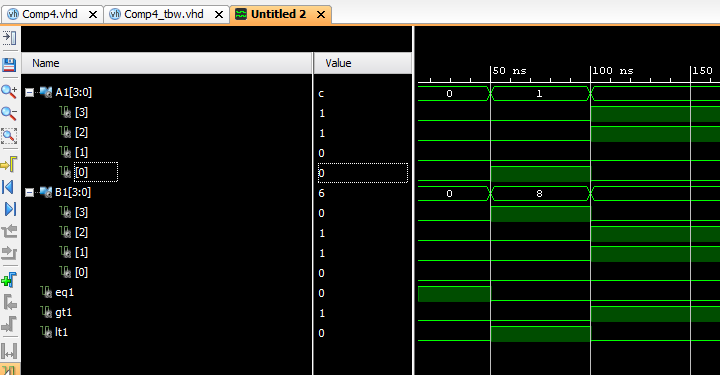
B1<="0110";

wait for 50 ns;

end process;

end Behavioral;

**Waveform**



**DAY 6**

**ALU (Behavioural)**

**VHDL Code**

use IEEE.NUMERIC\_STD.ALL;

entity AluBehav is

Port ( A : in unsigned (3 downto 0);

B : in unsigned (3 downto 0);

C : in STD\_LOGIC\_VECTOR (2 downto 0);

res : out unsigned (3 downto 0));

end AluBehav;

architecture Behavioral of AluBehav is

begin

process(A,B,C)

begin

case C is

when "000" => res <= A+B;

when "001" => res <= A-B;

when "010" => res <= A+ 1;

when "011" => res <= A-1;

when "100" => res <= A AND B;

when "101" => res <= A OR B;

when "110" => res <= NOT A;

when "111" => res <= A NAND B;

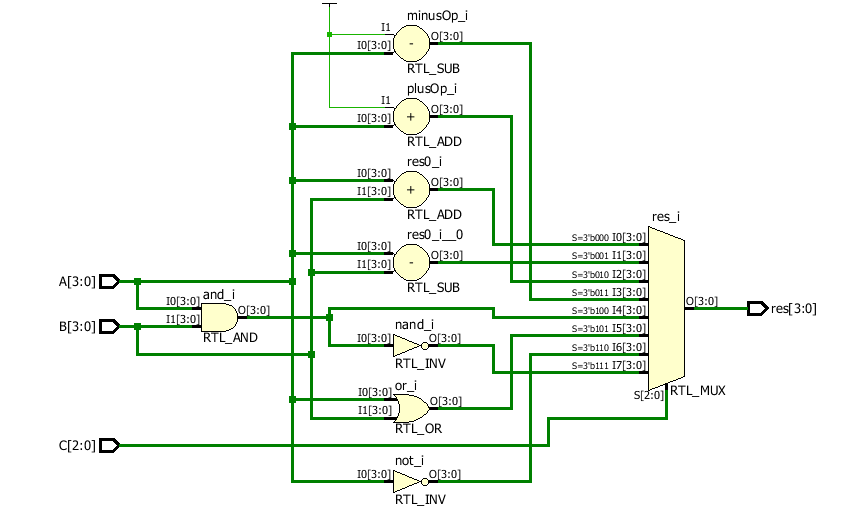
when others => NULL;

end case;

end process;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

use IEEE.NUMERIC\_STD.ALL;

entity AluBehav\_tbw is

-- Port ( );

end AluBehav\_tbw;

architecture Behavioral of AluBehav\_tbw is

component AluBehav is

Port ( A : in unsigned (3 downto 0);

B : in unsigned (3 downto 0);

C : in STD\_LOGIC\_VECTOR (2 downto 0);

res : out unsigned (3 downto 0));

end component;

Signal A1:unsigned (3 downto 0) := "1011";

Signal B1:unsigned (3 downto 0) := "0010";

Signal C1:STD\_LOGIC\_VECTOR (2 downto 0) := "000";

Signal res1:unsigned (3 downto 0);

begin

uut: AluBehav port map(A=>A1,B=>B1,C=>C1,res=>res1);

stim\_proc:process

begin

wait for 100 ns;

C1<="001";

wait for 100 ns;

C1<="010";

wait for 100 ns;

C1<="011";

wait for 100 ns;

C1<="100";

wait for 100 ns;

C1<="101";

wait for 100 ns;

C1<="110";

wait for 100 ns;

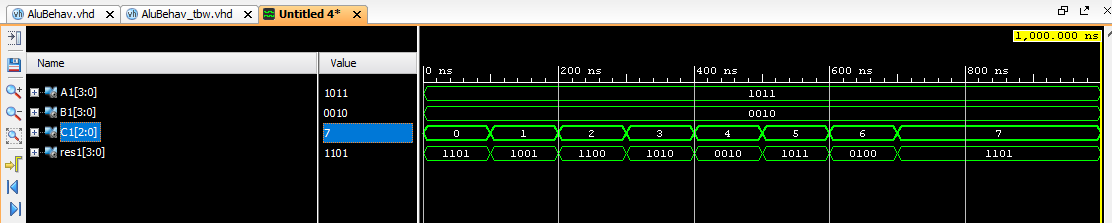
C1<="111";

wait;

end process;

end Behavioral;­­­­

**Waveform**



**Parity Checker Even and Odd (Behavioral)**

**VHDL Code­**

entity ParityGen is

Port ( ip : in STD\_LOGIC\_VECTOR (7 downto 0);

Po : out STD\_LOGIC;

Pe : out STD\_LOGIC);

end ParityGen;

architecture Behavioral of ParityGen is

begin

process(ip)

variable temp:STD\_LOGIC;

begin

temp := ip(0) XOR ip(1);

for i in 2 to 7 loop

temp := ip(i) XOR temp;

end loop;

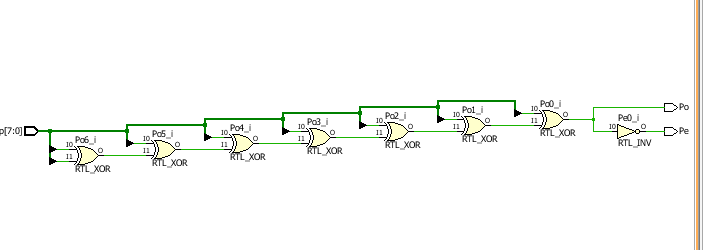
Po<=temp;

Pe<=not temp;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

entity ParityGen\_tbw is

-- Port ( );

end ParityGen\_tbw;

architecture Behavioral of ParityGen\_tbw is

component ParityGen is

Port ( ip : in STD\_LOGIC\_VECTOR (7 downto 0);

Po : out STD\_LOGIC;

Pe : out STD\_LOGIC);

end component;

Signal ip1 : STD\_LOGIC\_VECTOR (7 downto 0) := "00000000";

Signal Po1 : STD\_LOGIC;

Signal Pe1 : STD\_LOGIC;

begin

uut: ParityGen port map(ip=>ip1,Po=>Po1,Pe=>Pe1);

stim\_proc:process

begin

wait for 20ns;

ip1<="01101110";

wait for 20ns;

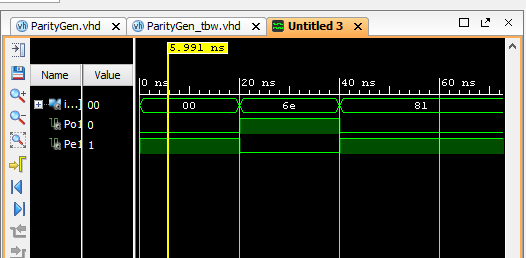
ip1<="10000001";

wait;

end process;

end Behavioral;

**Waveform**



**2’s Complement (Behavioral)**

**VHDL Code­**

entity twosComplement is

Port ( A : in unsigned (3 downto 0);

B : out unsigned (3 downto 0));

end twosComplement;

architecture Behavioral of twosComplement is

begin

process(A)

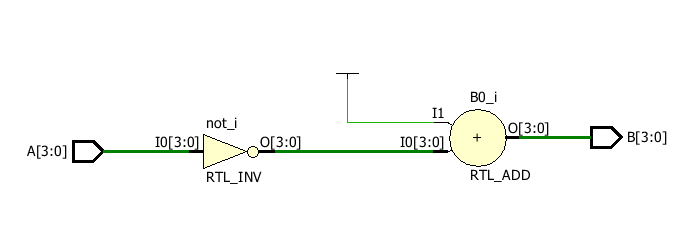
begin

B<= NOT A + "0001";

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

use IEEE.NUMERIC\_STD.ALL;

entity twosComplement\_tbw is

-- Port ( );

end twosComplement\_tbw;

architecture Behavioral of twosComplement\_tbw is

component twosComplement is

Port ( A : in unsigned (3 downto 0);

B : out unsigned (3 downto 0));

end component;

Signal A1: unsigned (3 downto 0):= "0000";

Signal B1: unsigned (3 downto 0);

begin

uut: twosComplement port map(A=>A1,B=>B1);

stim\_proc:process

begin

A1<="0101";

wait for 50 ns;

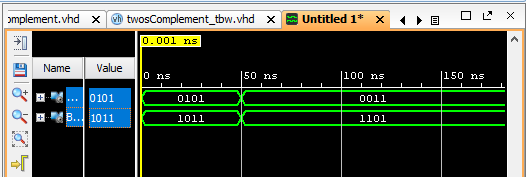
A1<="0011";

wait;

end process;

end Behavioral;

**Waveform**

****

**DAY 7**

**Ripple Carry Adder (Structural)**

**VHDL Code­**

entity RippleCarryAdder is

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0);

Cin : in STD\_LOGIC;

Cout : out STD\_LOGIC;

Sum : out STD\_LOGIC\_VECTOR (3 downto 0));

end RippleCarryAdder;

architecture Behavioral of RippleCarryAdder is

component FullAdder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

S : out STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

signal c1,c2,c3:STD\_LOGIC;

begin

L1:FullAdder port map(A(0),B(0),Cin,Sum(0),c1);

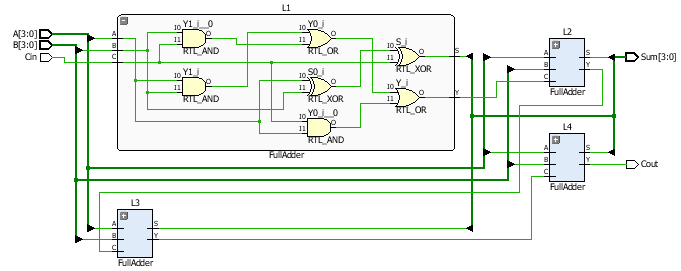
L2:FullAdder port map(A(1),B(1),c1,Sum(1),c2);

L3:FullAdder port map(A(2),B(2),c2,Sum(2),c3);

L4:FullAdder port map(A(3),B(3),c3,Sum(3),Cout);

end Behavioral;

**RTL Diagram**



**TBW Code**

entity RippleCarryAdder\_tbw is

-- Port ( );

end RippleCarryAdder\_tbw;

architecture Behavioral of RippleCarryAdder\_tbw is

component RippleCarryAdder is

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0);

Cin : in STD\_LOGIC;

Cout : out STD\_LOGIC;

Sum : out STD\_LOGIC\_VECTOR (3 downto 0));

end component;

Signal A1: STD\_LOGIC\_VECTOR (3 downto 0) := "1100";

Signal B1: STD\_LOGIC\_VECTOR (3 downto 0) := "1001";

Signal Cin1: STD\_LOGIC:= '0';

Signal Cout1: STD\_LOGIC;

Signal Sum1: STD\_LOGIC\_VECTOR (3 downto 0);

begin

uut: RippleCarryAdder port map(A=>A1,B=>B1,Cin=>Cin1,Cout=>Cout1,Sum=>Sum1);

stim\_proc:process

begin

wait for 50 ns;

A1<="1000";

B1<="0110";

wait for 50 ns;

A1<="1001";

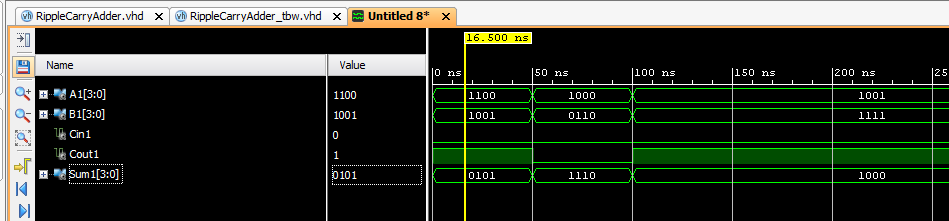
B1<="1111";

wait;

end process;

end Behavioral;

**Waveform**



**Full Adder using Half Adder (Structural)**

**VHDL Code­**

entity fulladderstr is

Port ( X : in STD\_LOGIC;

Y : in STD\_LOGIC;

Z : in STD\_LOGIC;

Sum : out STD\_LOGIC;

Carry : out STD\_LOGIC);

end fulladderstr;

architecture Structural of fulladderstr is

component HalfAdder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end component;

component ORdf is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

signal s1, c1, c2: std\_logic;

begin

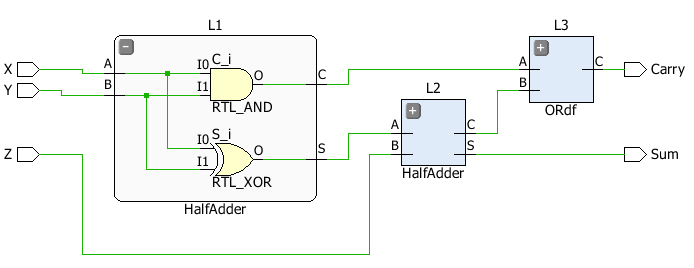
L1: HalfAdder port map(X, Y, s1, c1);

L2: HalfAdder port map(s1, Z, Sum, c2);

L3: ORdf port map(c1, c2, Carry);

end Structural;

**RTL Diagram**

****

**TBW Code**

entity fulladderstr\_tbw is

-- Port ( );

end fulladderstr\_tbw;

architecture Behavioral of fulladderstr\_tbw is

component fulladderstr is

Port ( X : in STD\_LOGIC;

Y : in STD\_LOGIC;

Z : in STD\_LOGIC;

Sum : out STD\_LOGIC;

Carry : out STD\_LOGIC);

end component;

signal X1: STD\_LOGIC:= '0';

signal Y1: STD\_LOGIC:= '0';

signal Z1: STD\_LOGIC:= '0';

signal Sum1: STD\_LOGIC;

signal Carry1: STD\_LOGIC;

begin

uut: fulladderstr port map(X=>X1,Y=>Y1,Z=>Z1,Sum=>Sum1,Carry=>Carry1);

stim\_proc:process

begin

wait for 50 ns;

X1<='0';

Y1<='0';

Z1<='1';

wait for 50 ns;

X1<='0';

Y1<='1';

Z1<='0';

wait for 50 ns;

X1<='0';

Y1<='1';

Z1<='1';

wait for 50 ns;

X1<='1';

Y1<='0';

Z1<='0';

wait for 50 ns;

X1<='1';

Y1<='0';

Z1<='1';

wait for 50 ns;

X1<='1';

Y1<='1';

Z1<='0';

wait for 50 ns;

X1<='1';

Y1<='1';

Z1<='1';

wait;

end process;

end Behavioral;

**Waveform**

**Adder Subtract Composite Unit (Behavioral)**

**VHDL Code­**

entity AddSubComp is

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0);

Sum : out STD\_LOGIC\_VECTOR (3 downto 0);

Ch : in STD\_LOGIC;

Cin : in STD\_LOGIC;

Cout : out STD\_LOGIC);

end AddSubComp;

architecture Structural of AddSubComp is

component FullAdder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

S : out STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

signal c1,c2,c3:STD\_LOGIC;

Signal temp:STD\_LOGIC\_VECTOR (3 downto 0);

begin

gk: for i in 0 to 3 generate

temp(i) <= B(i) XOR Ch;

end generate;

L1:FullAdder port map(A(0),temp(0),Cin,Sum(0),c1);

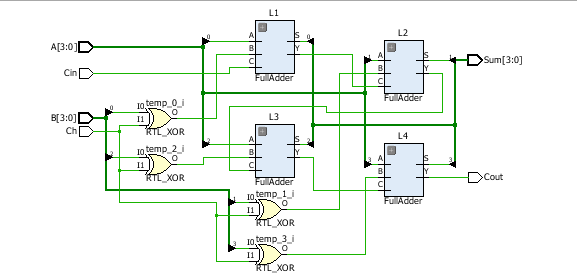
L2:FullAdder port map(A(1),temp(1),c1,Sum(1),c2);

L3:FullAdder port map(A(2),temp(2),c2,Sum(2),c3);

L4:FullAdder port map(A(3),temp(3),c3,Sum(3),Cout);

end Structural;

**RTL Diagram**



**TBW Code**

entity AddSubComp\_tbw is

-- Port ( );

end AddSubComp\_tbw;

architecture Behavioral of AddSubComp\_tbw is

component AddSubComp is

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (3 downto 0);

Sum : out STD\_LOGIC\_VECTOR (3 downto 0);

Cin : in STD\_LOGIC;

Cout : out STD\_LOGIC);

end component;

Signal A1: STD\_LOGIC\_VECTOR (3 downto 0) := "1010";

Signal B1: STD\_LOGIC\_VECTOR (3 downto 0) := "1101";

Signal Cin1: STD\_LOGIC:= '1';

Signal Cout1: STD\_LOGIC;

Signal Sum1: STD\_LOGIC\_VECTOR (3 downto 0);

begin

uut: AddSubComp port map(A=>A1,B=>B1,Cin=>Cin1,Cout=>Cout1,Sum=>Sum1);

stim\_proc:process

begin

wait for 50 ns;

A1<="1000";

B1<="0110";

Cin1<='1';

wait for 50 ns;

A1<="1001";

B1<="1111";

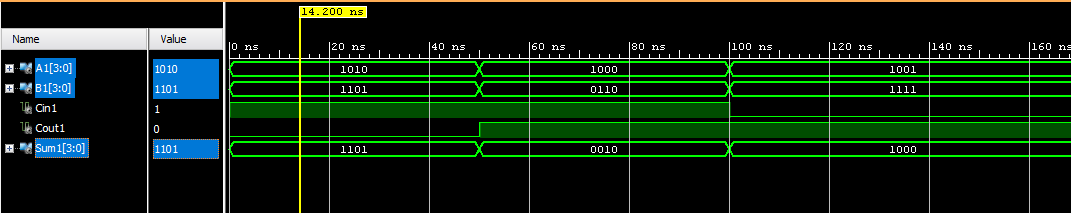
Cin1<='0';

wait;

end process;

end Behavioral;

**Waveform**

****

**8:1 MUX using 4:1 and 2:1 (Structural)**

**VHDL Code­**

entity MUX81 is

Port ( ip : in STD\_LOGIC\_VECTOR (7 downto 0);

sel : in STD\_LOGIC\_VECTOR (2 downto 0);

op : out STD\_LOGIC);

end MUX81;

architecture Behavioral of MUX81 is

component MUX41 is

Port ( ip : in STD\_LOGIC\_VECTOR (3 downto 0);

sel : in STD\_LOGIC\_VECTOR (1 downto 0);

Y : out STD\_LOGIC);

end component;

component MUX21df is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

Signal Y1,Y2:STD\_LOGIC;

begin

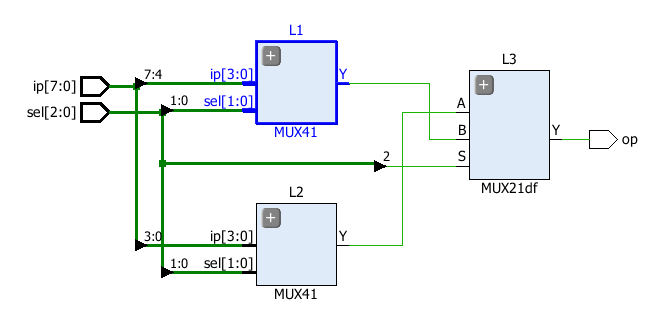
L1:MUX41 port map(ip(7 downto 4),sel(1 downto 0),Y1);

L2:MUX41 port map(ip(3 downto 0),sel(1 downto 0),Y2);

L3:MUX21df port map(Y2,Y1,sel(2),op);

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity MUX81\_tbw is

-- Port ( );

end MUX81\_tbw;

architecture Behavioral of MUX81\_tbw is

component MUX81 is

Port ( ip : in STD\_LOGIC\_VECTOR (7 downto 0);

sel : in STD\_LOGIC\_VECTOR (2 downto 0);

op : out STD\_LOGIC);

end component;

Signal ip1:STD\_LOGIC\_VECTOR (7 downto 0):="11101110";

Signal sel1:STD\_LOGIC\_VECTOR (2 downto 0):="000";

Signal op1:STD\_LOGIC;

begin

uut:MUX81 port map(ip=>ip1,sel=>sel1,op=>op1);

stim\_proc:process

begin

wait for 50ns;

sel1<="001";

wait for 50ns;

sel1<="010";

wait for 50ns;

sel1<="011";

wait for 50ns;

sel1<="100";

wait for 50ns;

sel1<="101";

wait for 50ns;

sel1<="110";

wait for 50ns;

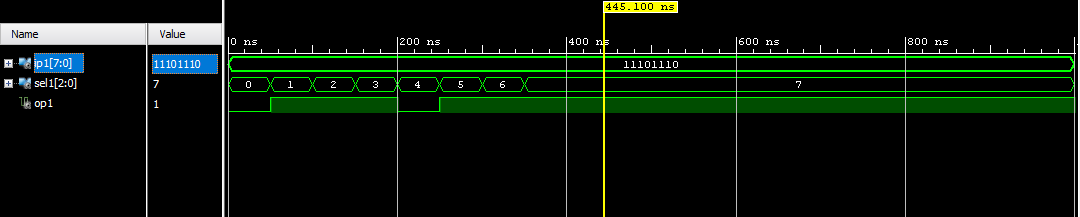
sel1<="111";

wait;

end process;

end Behavioral;

**Waveform**

****

**DAY 8**

**JK Flip-Flop (Behavioral)**

**VHDL Code­**

entity jkff is

Port ( J : in STD\_LOGIC;

K : in STD\_LOGIC;

clk : in STD\_LOGIC;

Q : inout STD\_LOGIC;

Qn : inout STD\_LOGIC);

end jkff;

architecture Behavioral of jkff is

begin

process(J,K,clk)

begin

if (clk'event and clk='1') then

if(J='0' and K='0') then

Q<=Q;

elsif (J='0' and K='1') then

Q<='0';

elsif (J='1' and K='1') then

Q<='1';

elsif (J='0' and K='1') then

Q<=not Q;

end if;

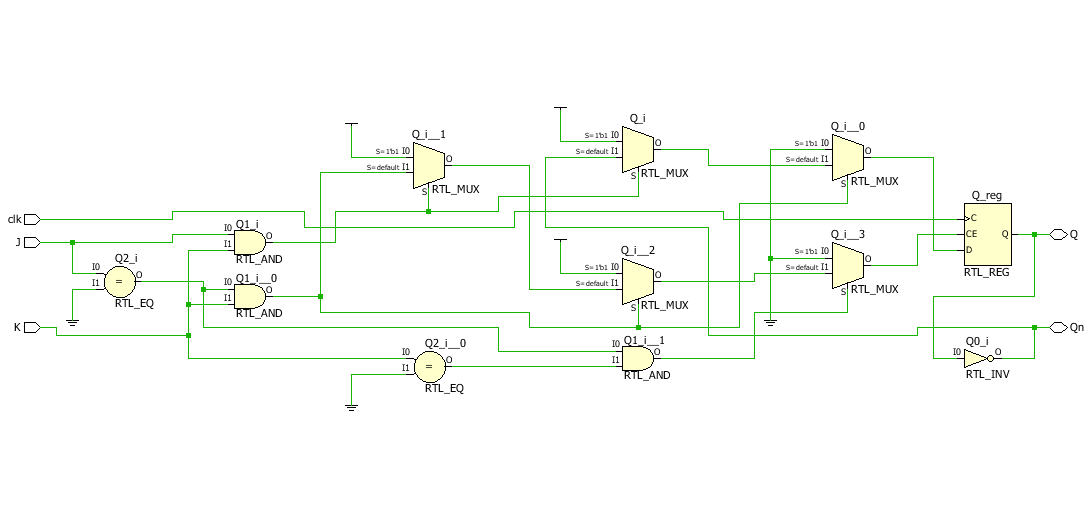
end if;

end process;

Qn<= not Q;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity jkff\_tbw is

-- Port ( );

end jkff\_tbw;

architecture Behavioral of jkff\_tbw is

component jkff is

Port ( J : in STD\_LOGIC;

K : in STD\_LOGIC;

clk : in STD\_LOGIC;

Q : inout STD\_LOGIC;

Qn : inout STD\_LOGIC);

end component;

constant clock\_period:time:=60ns;

Signal J1,K1,clk1:STD\_LOGIC:='0';

Signal Q1,Qn1:STD\_LOGIC;

begin

uut: jkff port map(J=>J1,K=>K1,clk=>clk1,Q=>Q1,Qn=>Qn1);

clk1<=not clk1 after clock\_period/2;

stim\_proc:process

begin

J1<='0';

K1<='1';

wait for 100ns;

J1<='0';

K1<='0';

wait for 100ns;

J1<='1';

K1<='0';

wait for 100ns;

J1<='1';

K1<='1';

wait for 100ns;

J1<='0';

K1<='0';

wait for 100ns;

J1<='1';

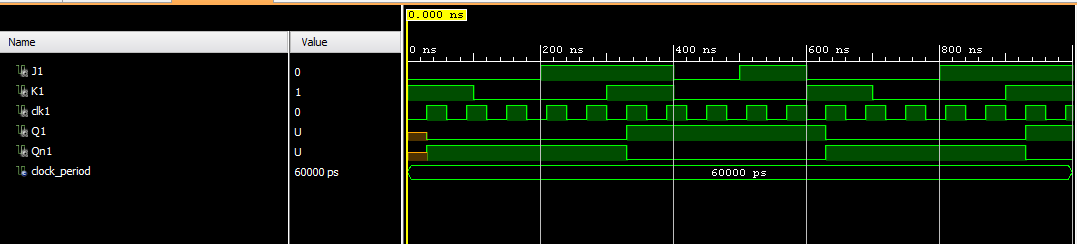
K1<='0';

wait for 100ns;

end process;

end Behavioral;

**Waveform**



**SR Flip-Flop (Behavioral)**

**VHDL Code­**

entity srff is

Port ( S : in STD\_LOGIC;

R : in STD\_LOGIC;

clk : in STD\_LOGIC;

Q : out STD\_LOGIC;

Qn : out STD\_LOGIC);

end srff;

architecture Behavioral of srff is

begin

process(S,R,clk)

begin

if(clk'event and clk='1') then

if(S='0' and R='1') then

Q<='0';

Qn<='1';

elsif(S='1' and R='0') then

Q<='1';

Qn<='0';

elsif(S='1' and R='1') then

Q<='X';

Qn<='X';

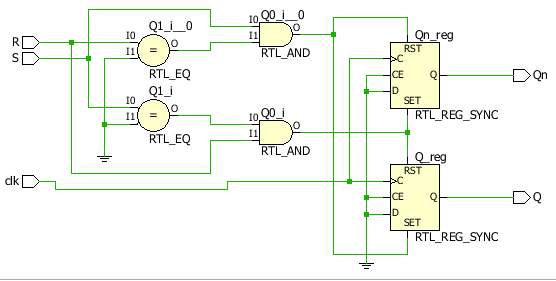
end if;

end if;

end process;

end Behavioral;

**RTL Diagram**



**TBW Code**

entity srff\_tbw is

-- Port ( );

end srff\_tbw;

architecture Behavioral of srff\_tbw is

component srff is

Port ( S : in STD\_LOGIC;

R : in STD\_LOGIC;

clk : in STD\_LOGIC;

Q : out STD\_LOGIC;

Qn : out STD\_LOGIC);

end component;

constant clock\_period:time:=60ns;

Signal S1,R1,clk1:STD\_LOGIC:='0';

Signal Q1,Qn1:STD\_LOGIC;

begin

uut:srff port map(S=>S1,R=>R1,clk=>clk1,Q=>Q1,Qn=>Qn1);

clk1<=not clk1 after clock\_period/2;

stim\_proc:process

begin

wait for 100ns;

S1<='0';

R1<='1';

wait for 100ns;

S1<='0';

R1<='0';

wait for 100ns;

S1<='1';

R1<='0';

wait for 100ns;

S1<='1';

R1<='1';

wait for 100ns;

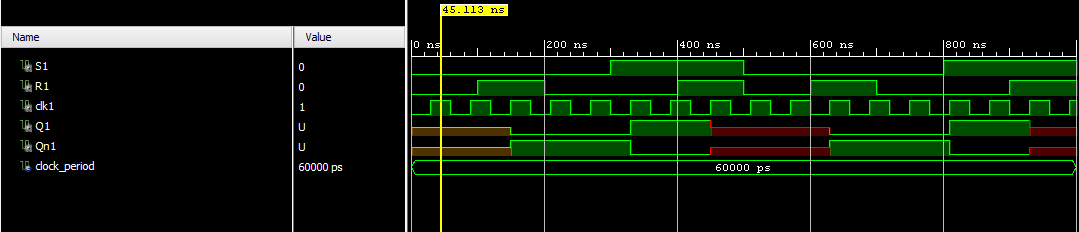
S1<='0';

R1<='0';

end process;

end Behavioral;

**Waveform**



**D Flip-Flop (Behaviour)**

**VHDL Code­**

**RTL Diagram**

**TBW Code**

**Waveform**

**T Flip-Flop (Behaviour)**

**VHDL Code­**

entity tff is

Port ( T : in STD\_LOGIC;

clk : in STD\_LOGIC;

Q : out STD\_LOGIC);

end tff;

architecture Behavioral of tff is

Signal temp:STD\_LOGIC:='0';

begin

process(T,clk)

begin

if(clk'event and clk='1') then

if(T='1') then

temp<=not temp;

else

temp<=temp;

end if;

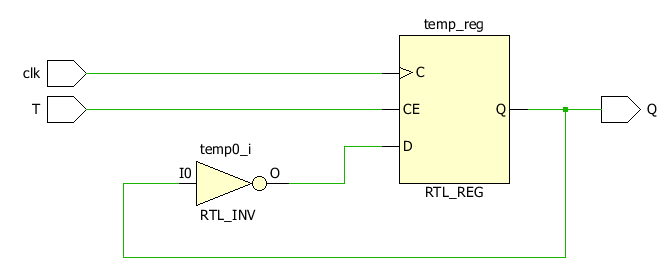
end if;

Q<=temp;

end process;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

entity tff\_tbw is

-- Port ( );

end tff\_tbw;

architecture Behavioral of tff\_tbw is

component tff is

Port ( T : in STD\_LOGIC;

clk : in STD\_LOGIC;

Q : out STD\_LOGIC);

end component;

Signal T1,clk1:STD\_LOGIC:='0';

Signal Q1:STD\_LOGIC;

constant clock\_period:time:=60ns;

begin

uut:tff port map(T=>T1,clk=>clk1,Q=>Q1);

clk1 <= not clk1 after clock\_period/2;

stim\_proc:process

begin

wait for 50ns;

T1<='0';

wait for 50ns;

wait for 50ns;

T1<='1';

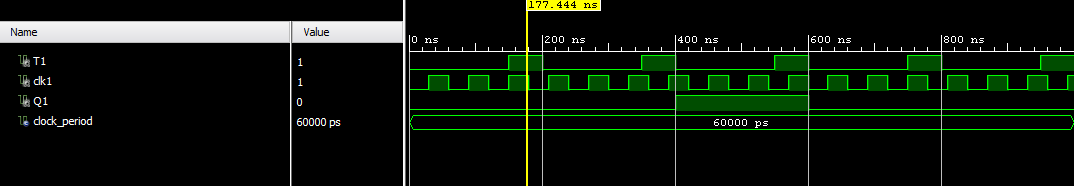
wait for 50ns;

T1<='0';

end process;

end Behavioral;

**Waveform**



**4 Bit Shift Register (Structural)**

**VHDL Code­**

entity shiftregister is

Port ( ip : in STD\_LOGIC;

clk : in STD\_LOGIC;

op : inout STD\_LOGIC\_VECTOR (3 downto 0));

end shiftregister;

architecture Structural of shiftregister is

component dff is

Port ( D : in STD\_LOGIC;

clk : in STD\_LOGIC;

O : out STD\_LOGIC);

end component;

begin

L1:dff port map(ip,clk,op(0));

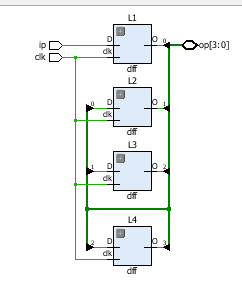
L2:dff port map(op(0),clk,op(1));

L3:dff port map(op(1),clk,op(2));

L4:dff port map(op(2),clk,op(3));

end Structural;

**RTL Diagram**

****

**TBW Code**

entity shiftregister\_tbw is

-- Port ( );

end shiftregister\_tbw;

architecture Behavioral of shiftregister\_tbw is

component shiftregister is

Port ( ip : in STD\_LOGIC;

clk : in STD\_LOGIC;

op : inout STD\_LOGIC\_VECTOR (3 downto 0));

end component;

constant clock\_period:time:=60ns;

Signal ip1,clk1:STD\_LOGIC:='0';

Signal op1:STD\_LOGIC\_VECTOR(3 downto 0);

begin

uut:shiftregister port map(ip=>ip1,clk=>clk1,op=>op1);

clk1<=not clk1 after clock\_period/2;

stim\_proc:process

begin

wait for 100ns;

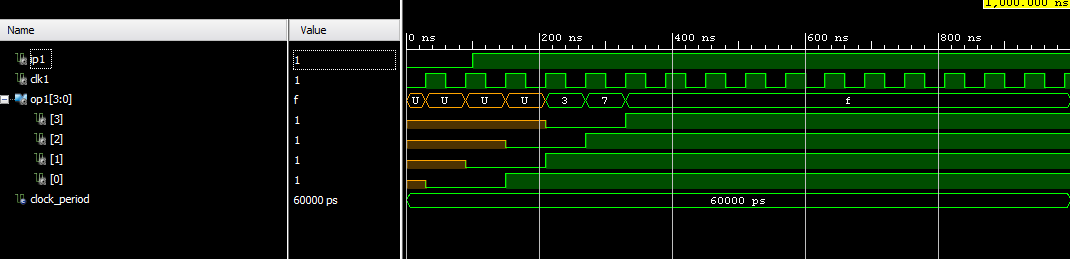
ip1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**DAY 9**

**Up Counter (Behavioral)**

**VHDL Code­**

use IEEE.NUMERIC\_STD.ALL;

entity upcounter is

Port ( clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

q : out unsigned (3 downto 0));

end upcounter;

architecture Behavioral of upcounter is

Signal temp : unsigned (3 downto 0):="0000";

begin

process(clr,clk,temp)

begin

if(clr='1') then

temp<=temp;

elsif(clk'event and clk='1') then

temp<=temp+1;

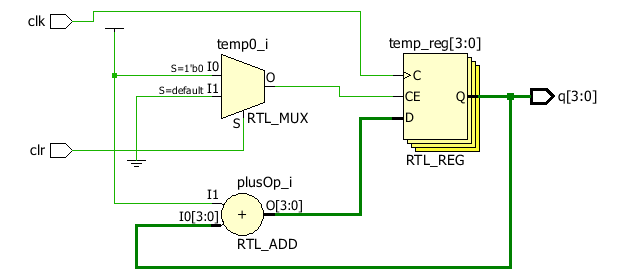
end if;

end process;

q<=temp;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

use IEEE.NUMERIC\_STD.ALL;

entity upcounter\_tbw is

-- Port ( );

end upcounter\_tbw;

architecture Behavioral of upcounter\_tbw is

component upcounter is

Port ( clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

q : out unsigned (3 downto 0));

end component;

constant clock\_period:time:=60ns;

Signal clr1,clk1:STD\_LOGIC:='0';

Signal q1:unsigned (3 downto 0);

begin

uut:upcounter port map(clr=>clr1,clk=>clk1,q=>q1);

clk1<=not clk1 after clock\_period/2;

stim\_proc:process

begin

wait for 100ns;

clr1<='0';

wait for 600ns;

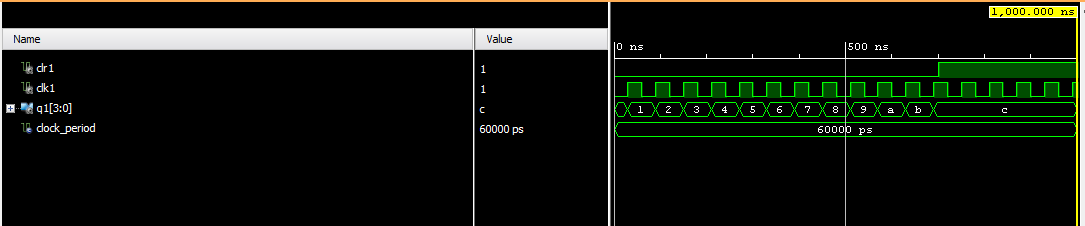
clr1<='1';

wait;

end process;

end Behavioral;

**Waveform**

****

**Down Counter (Behavioral)**

**VHDL Code­**

use IEEE.NUMERIC\_STD.ALL;

entity downcounter is

Port ( clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

q : out unsigned (3 downto 0));

end downcounter;

architecture Behavioral of downcounter is

Signal temp : unsigned (3 downto 0):="1111";

begin

process(clr,clk,temp)

begin

if(clr='1') then

temp<=temp;

elsif(clk'event and clk='1') then

temp<=temp-1;

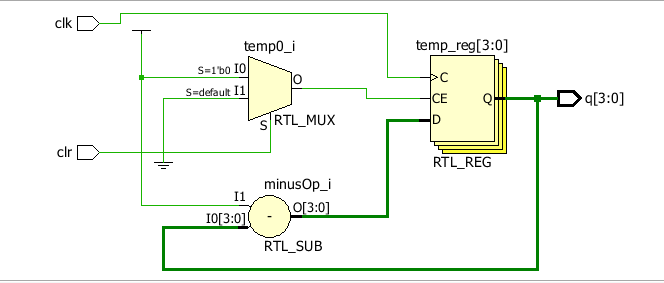
end if;

end process;

q<=temp;

end Behavioral;

**RTL Diagram**

****

**TBW Code**

use IEEE.NUMERIC\_STD.ALL;

entity downcounter\_tbw is

-- Port ( );

end downcounter\_tbw;

architecture Behavioral of downcounter\_tbw is

component downcounter is

Port ( clr : in STD\_LOGIC;

clk : in STD\_LOGIC;

q : out unsigned (3 downto 0));

end component;

constant clock\_period:time:=60ns;

Signal clr1,clk1:STD\_LOGIC:='0';

Signal q1:unsigned (3 downto 0);

begin

uut:downcounter port map(clr=>clr1,clk=>clk1,q=>q1);

clk1<=not clk1 after clock\_period/2;

stim\_proc:process

begin

wait for 100ns;

clr1<='0';

wait for 600ns;

clr1<='1';

wait;

end process;

end Behavioral;

**Waveform**

