# AND GATE DATAFLOW MODEL

**VHD CODE:**

entity and\_df is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end and\_df;

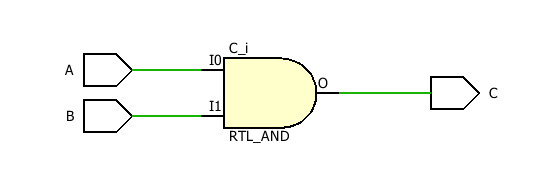
architecture dataflow of and\_df is

begin

C<=A AND B;

end dataflow;

**RTL Diagram:**



**TBW Code:**

entity and\_df\_tbw is

-- Port ( );

end and\_df\_tbw;

architecture dataflow of and\_df\_tbw is

component and\_df 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: and\_df port map(A=>a1, B=>b1, C=>c1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

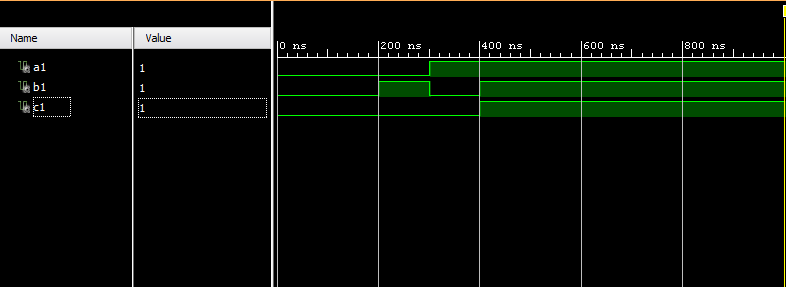
b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**



# OR GATE DATAFLOW MODEL

**VHD CODE:**

entity OR\_Gate\_DF is

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

c : out STD\_LOGIC);

end OR\_Gate\_DF;

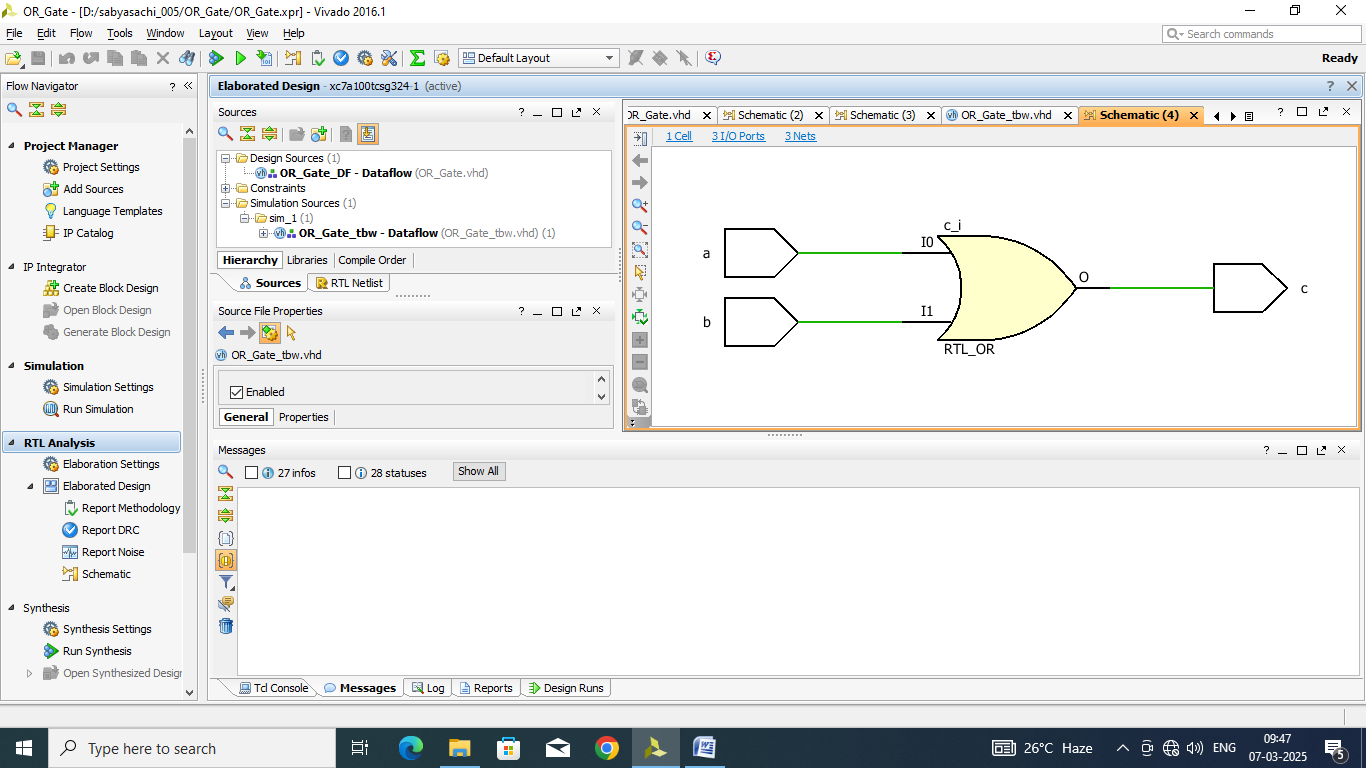
architecture Dataflow of OR\_Gate\_DF is

begin

c<=a or b;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity OR\_Gate\_tbw is

-- Port ( );

end OR\_Gate\_tbw;

architecture Dataflow of OR\_Gate\_tbw is

component OR\_Gate\_DF 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: OR\_Gate\_DF port map(a=>a1, b=>b1, c=>c1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

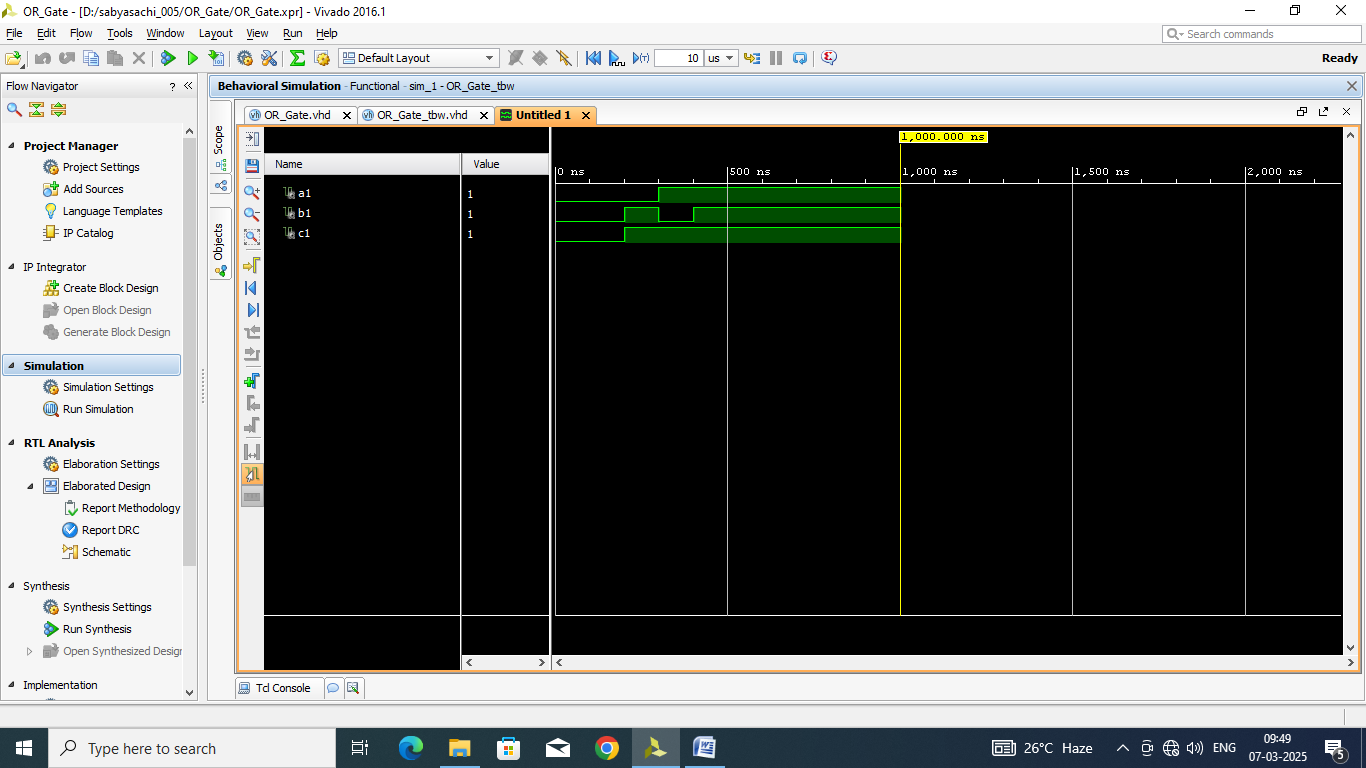
b1<='1';

wait;

end process;

end Dataflow;

**TBW Waveform:**



# NOT GATE DATAFLOW MODEL

**VHD CODE:**

entity NOT\_Gate\_df is

Port ( a : in STD\_LOGIC;

b : out STD\_LOGIC);

end NOT\_Gate\_df;

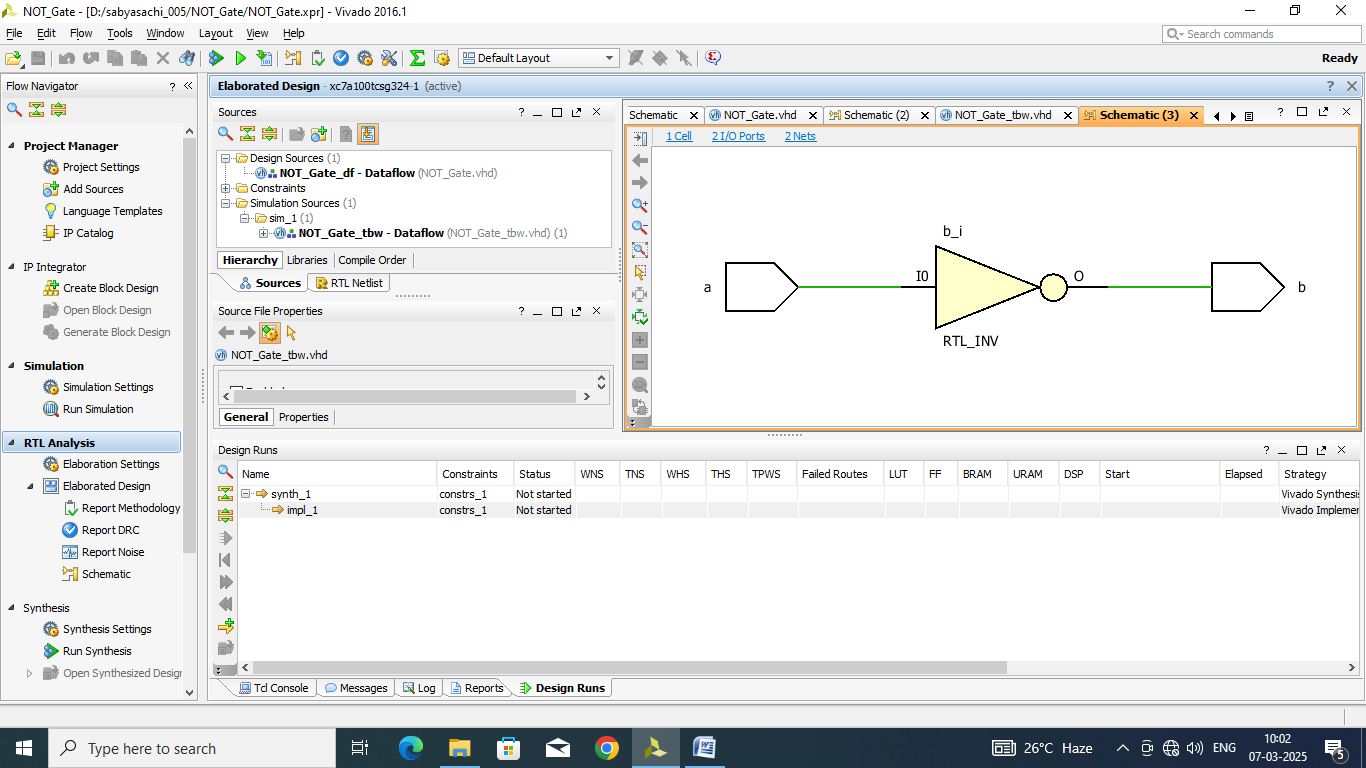
architecture Dataflow of NOT\_Gate\_df is

begin

b<= not a;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity NOT\_Gate\_tbw is

-- Port ( );

end NOT\_Gate\_tbw;

architecture Dataflow of NOT\_Gate\_tbw is

component NOT\_Gate\_df is

Port ( a : in STD\_LOGIC;

b : out STD\_LOGIC);

end component;

signal a1: STD\_LOGIC:='0';

signal b1: STD\_LOGIC;

begin

UUT: NOT\_Gate\_df port map(a=>a1, b=>b1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

wait for 100ns;

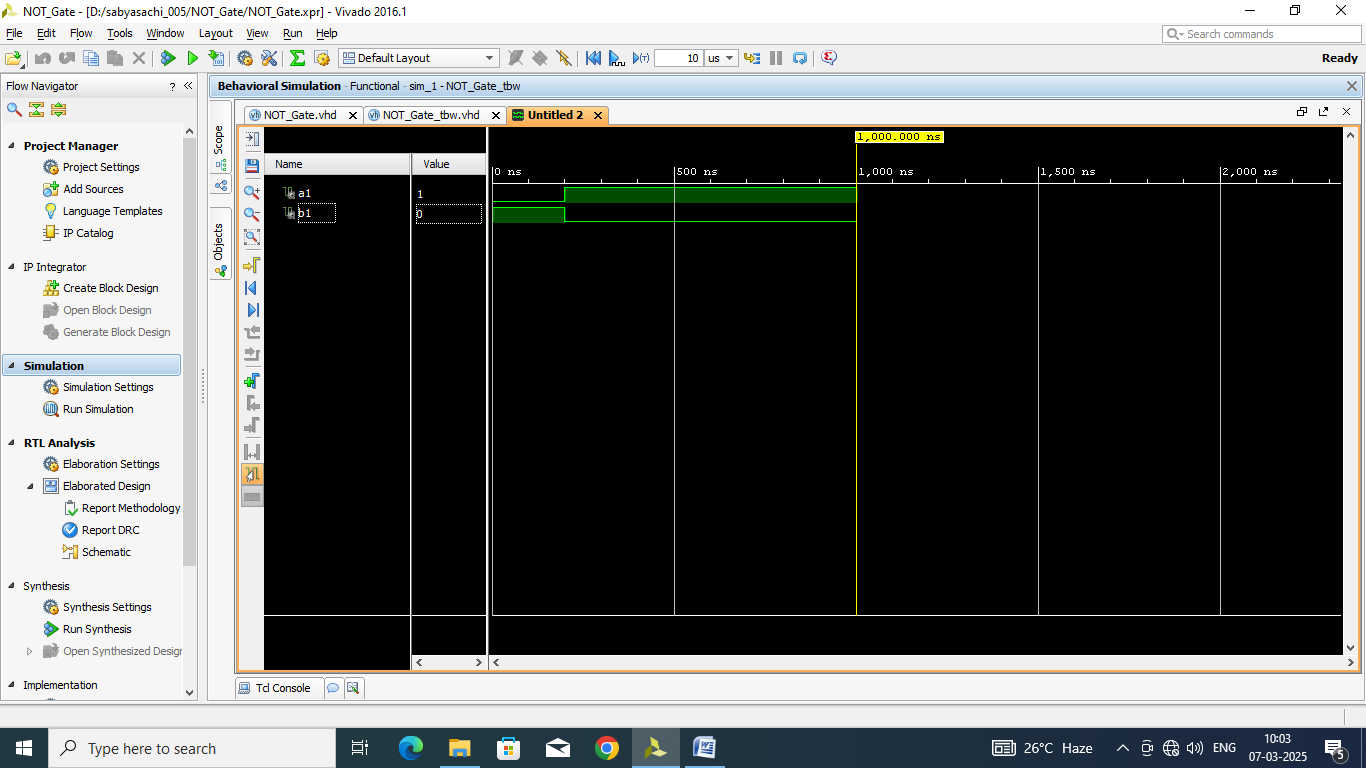
a1<='1';

wait;

end process;

end Dataflow;

**TBW Waveform:**



# NAND GATE DATAFLOW MODEL

**VHD CODE:**

entity NAND\_Gate\_DF is

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

c : out STD\_LOGIC);

end NAND\_Gate\_DF;

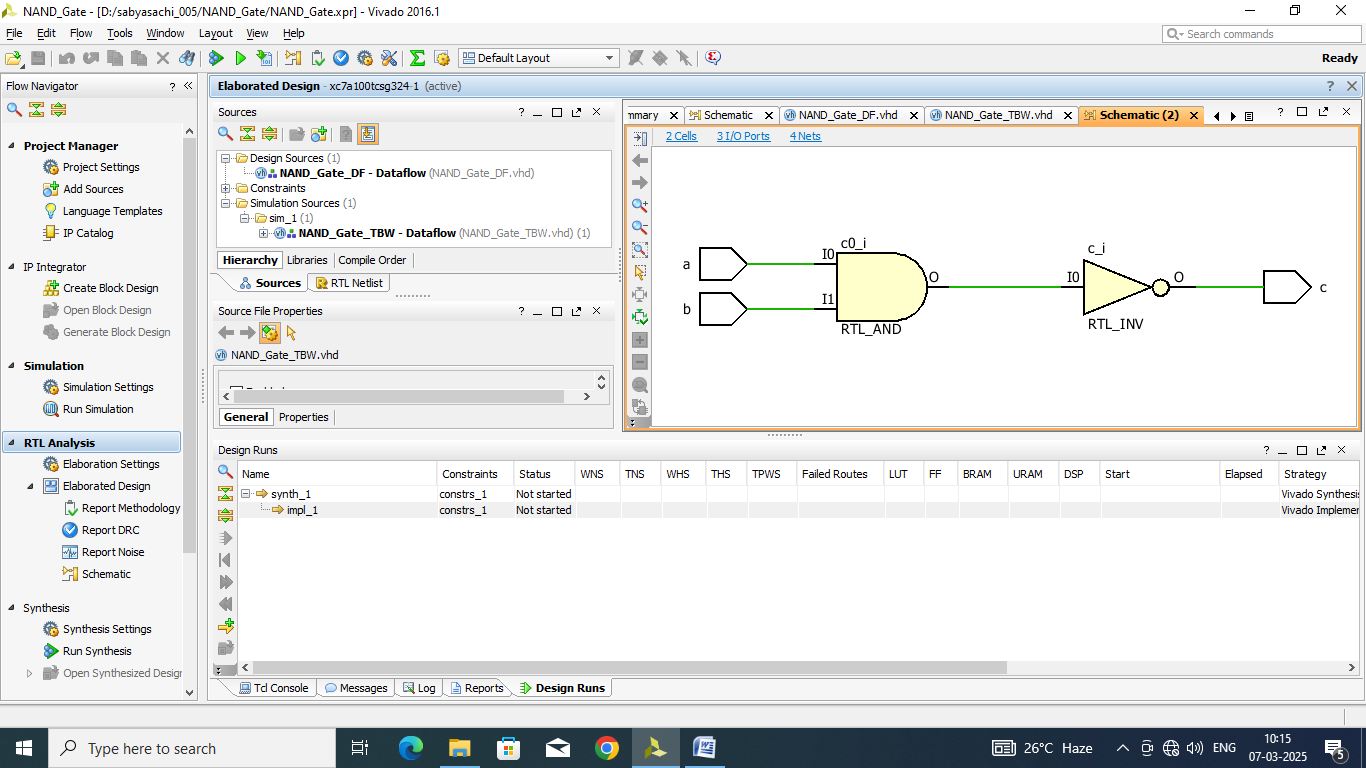
architecture Dataflow of NAND\_Gate\_DF is

begin

c<=a nand b;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity NAND\_Gate\_TBW is

-- Port ( );

end NAND\_Gate\_TBW;

architecture Dataflow of NAND\_Gate\_TBW is

component NAND\_Gate\_DF 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\_Gate\_DF port map(a=>a1, b=>b1,c=>c1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

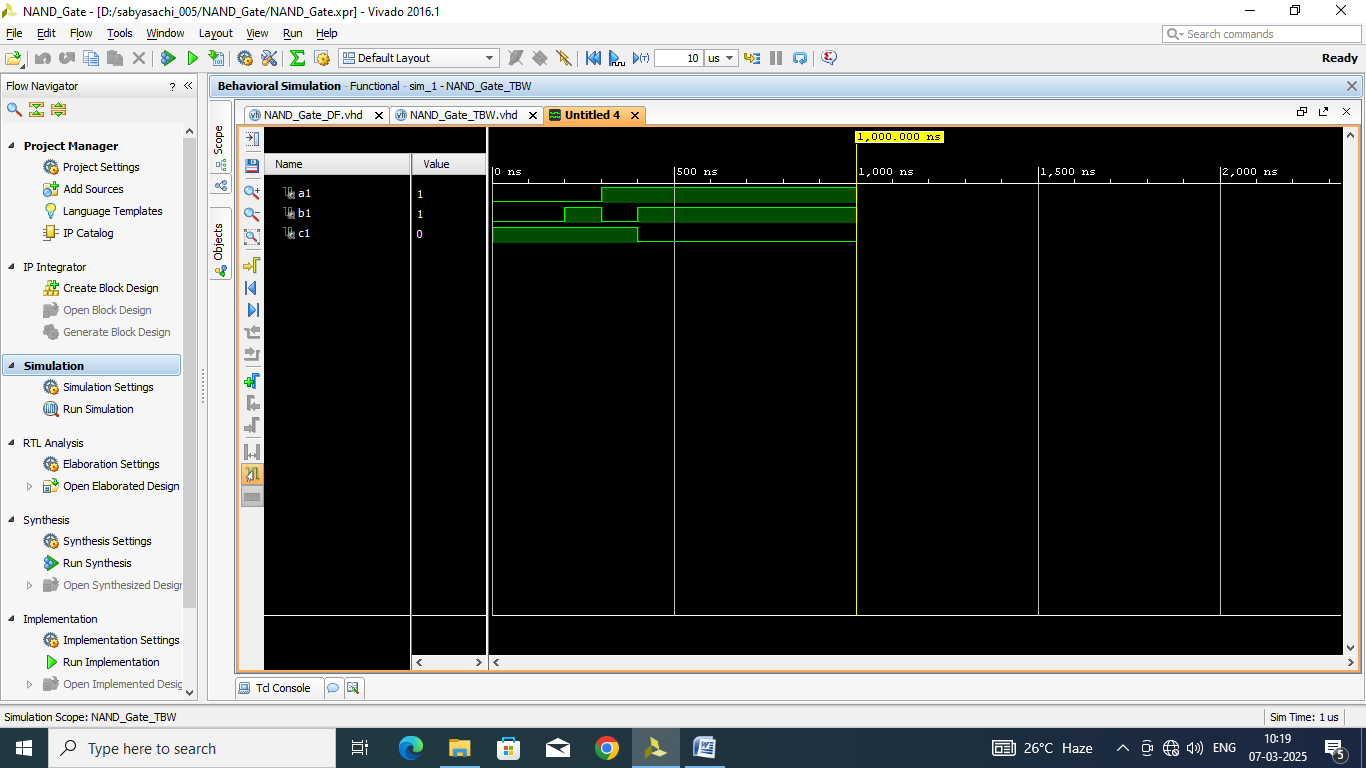
b1<='1';

wait;

end process;

end Dataflow;

**TBW Waveform:**



# NOR GATE DATAFLOW MODEL

**VHD CODE:**

entity NOR\_Gate\_DF is

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

c : out STD\_LOGIC);

end NOR\_Gate\_DF;

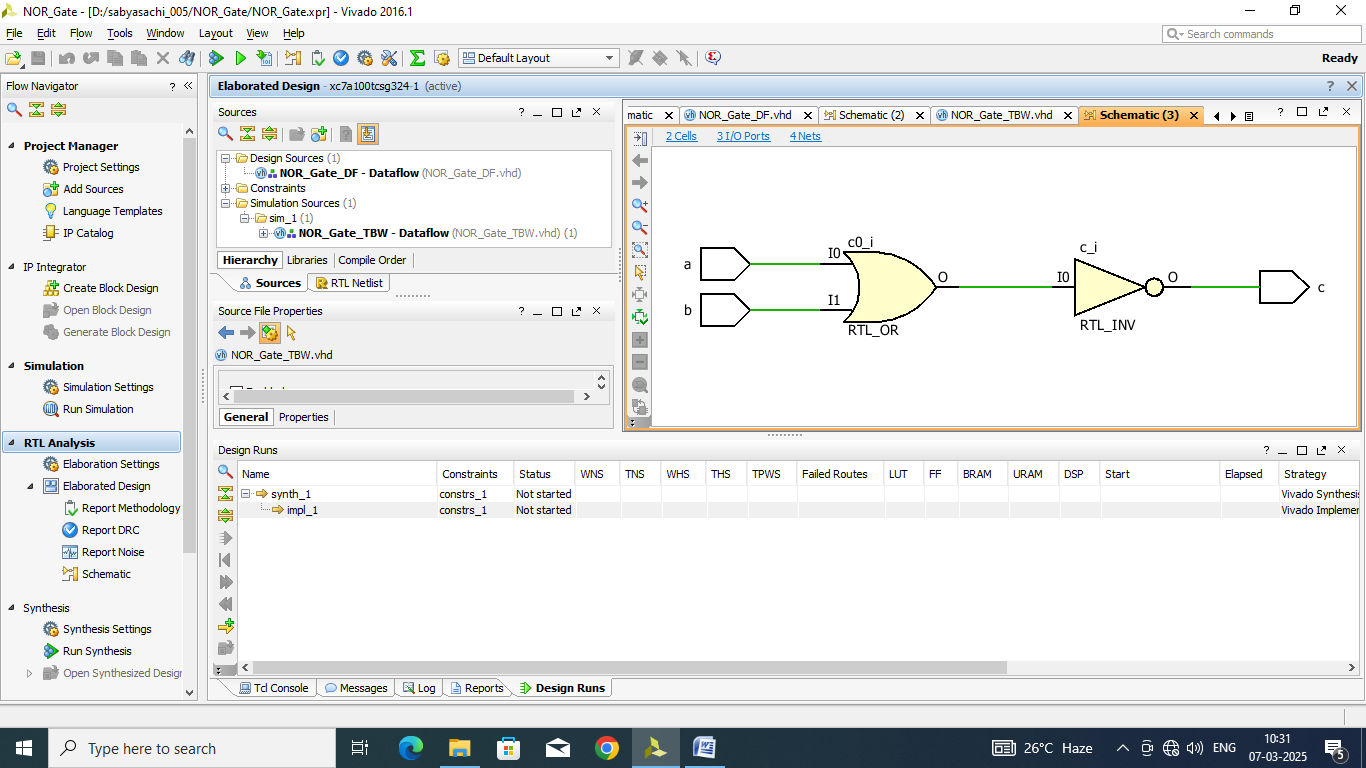
architecture Dataflow of NOR\_Gate\_DF is

begin

c<= a nor b;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity NOR\_Gate\_TBW is

-- Port ( );

end NOR\_Gate\_TBW;

architecture Dataflow of NOR\_Gate\_TBW is

component NOR\_Gate\_DF 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\_Gate\_DF port map(a=>a1, b=>b1,c=>c1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

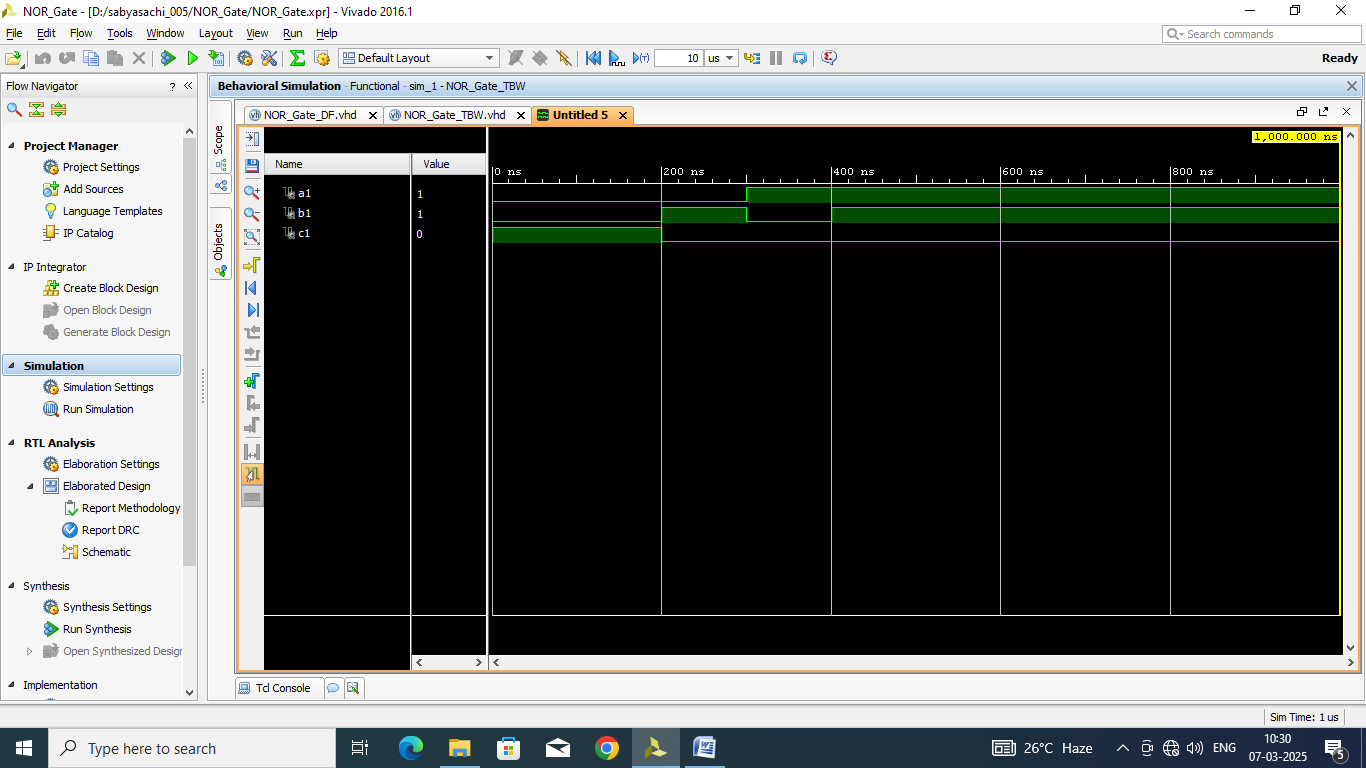
b1<='1';

wait;

end process;

end Dataflow;

**TBW Waveform:**



# AND Gate Using NAND Gate Dataflow

**VHD CODE:**

entity AND\_NAND\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end AND\_NAND\_DF;

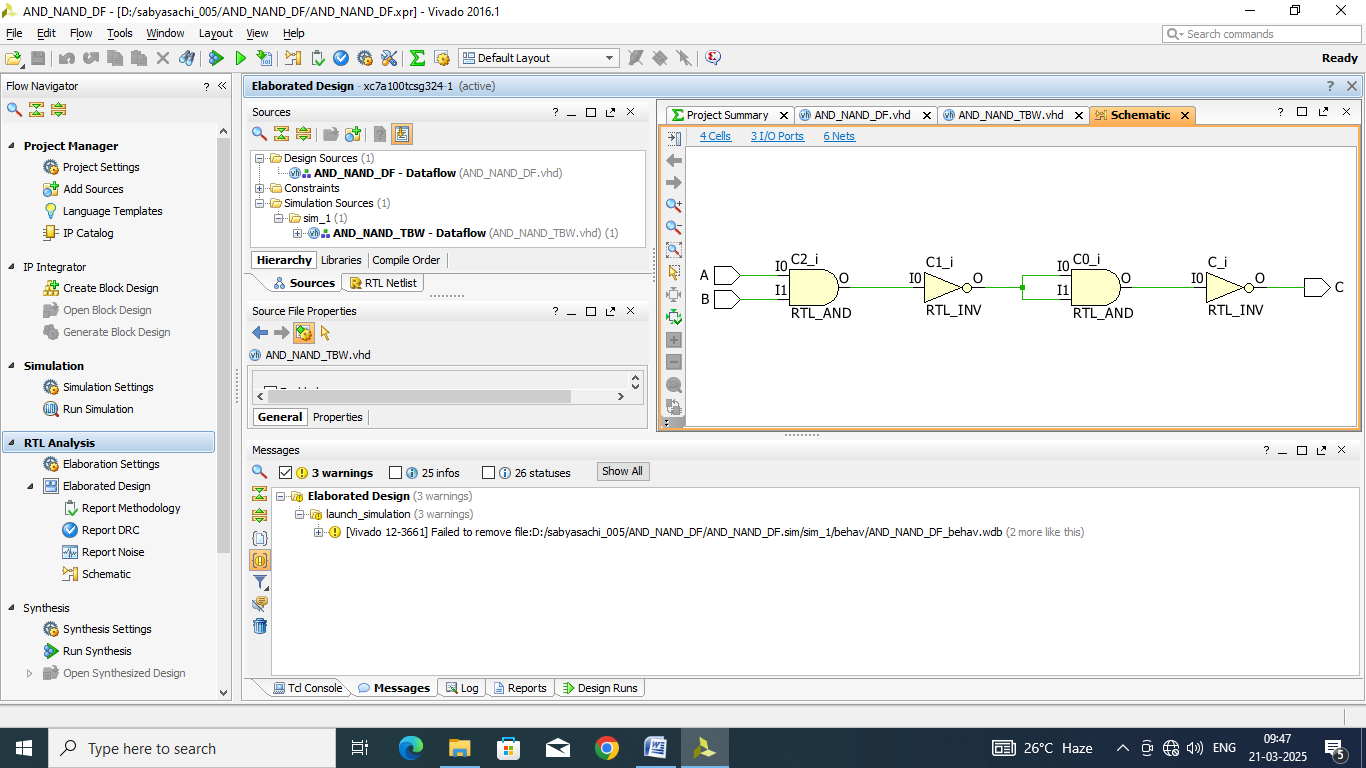
architecture Dataflow of AND\_NAND\_DF is

begin

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

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity AND\_NAND\_TBW is

-- Port ( );

end AND\_NAND\_TBW;

architecture Dataflow of AND\_NAND\_TBW is

component AND\_NAND\_DF 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: AND\_NAND\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

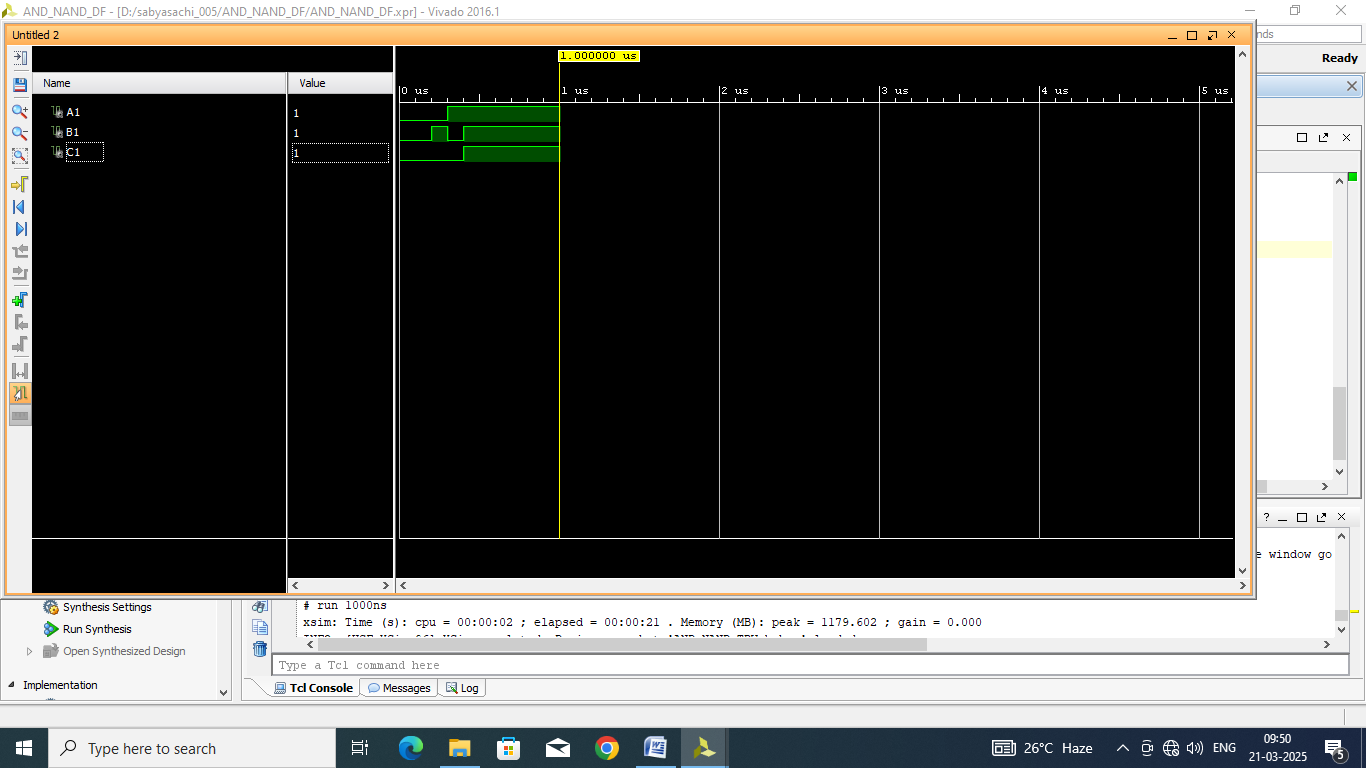
b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**



# OR Gate Using NAND Gate Dataflow

**VHD CODE:**

entity OR\_NAND\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end OR\_NAND\_DF;

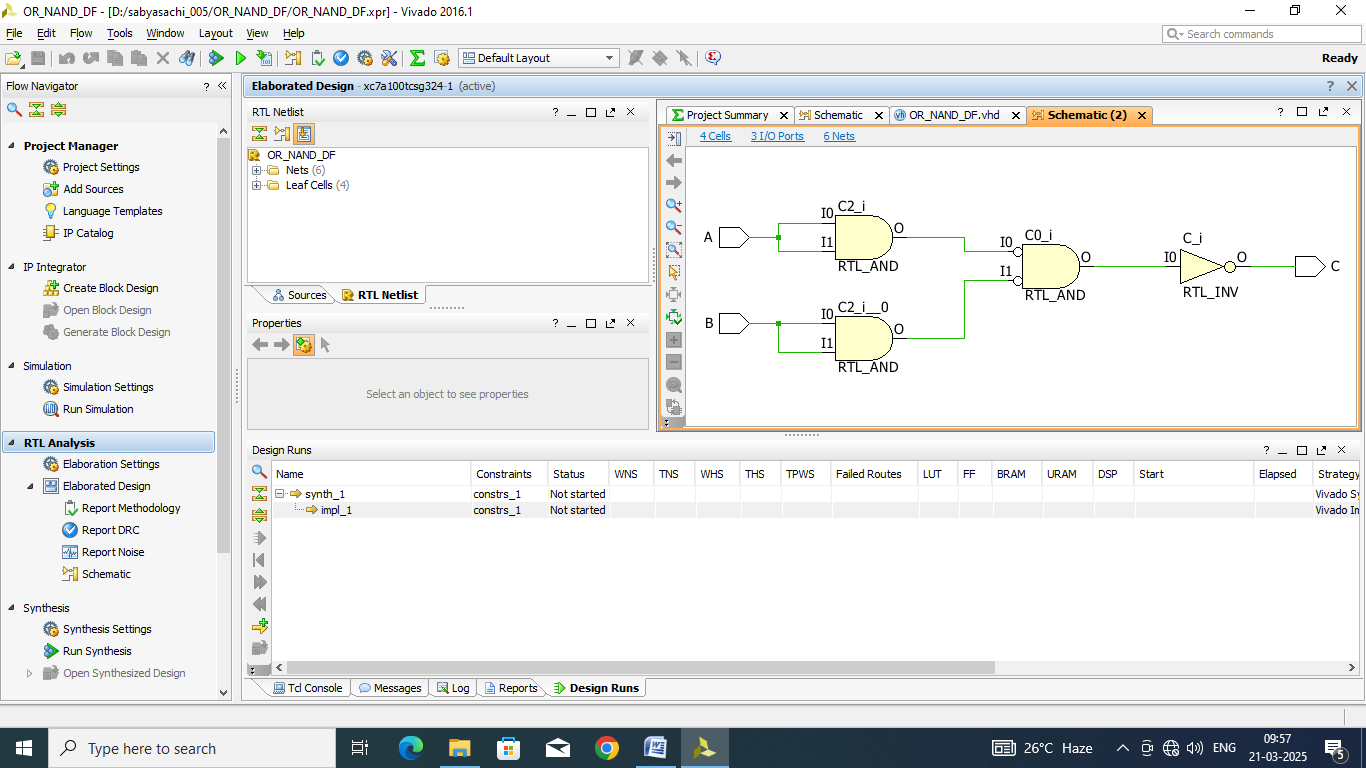
architecture Dataflow of OR\_NAND\_DF is

begin

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

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity OR\_NAND\_TBW is

-- Port ( );

end OR\_NAND\_TBW;

architecture Dataflow of OR\_NAND\_TBW is

component OR\_NAND\_DF 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: OR\_NAND\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

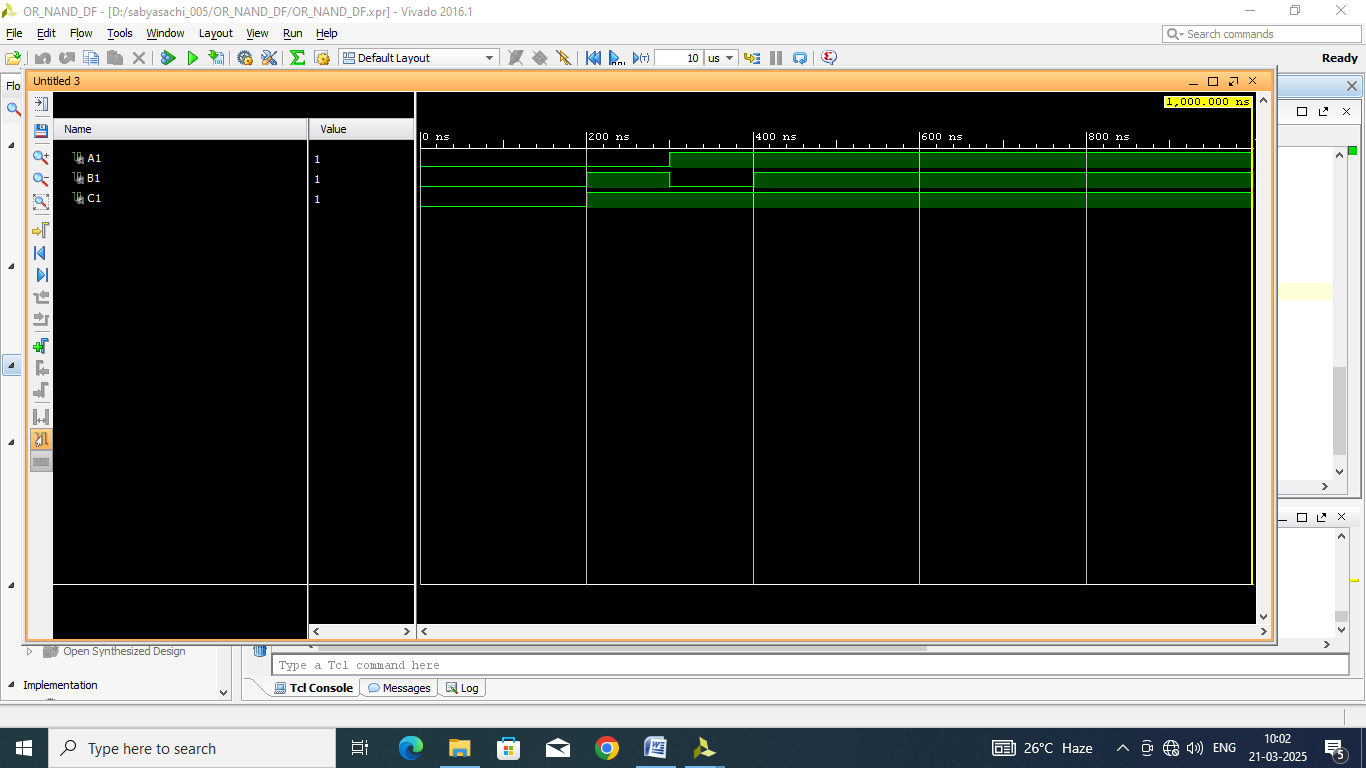
b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**



# not Gate Using NAND Gate Dataflow

**VHD CODE:**

entity NOT\_NAND\_DF is

Port ( A : in STD\_LOGIC;

B : out STD\_LOGIC);

end NOT\_NAND\_DF;

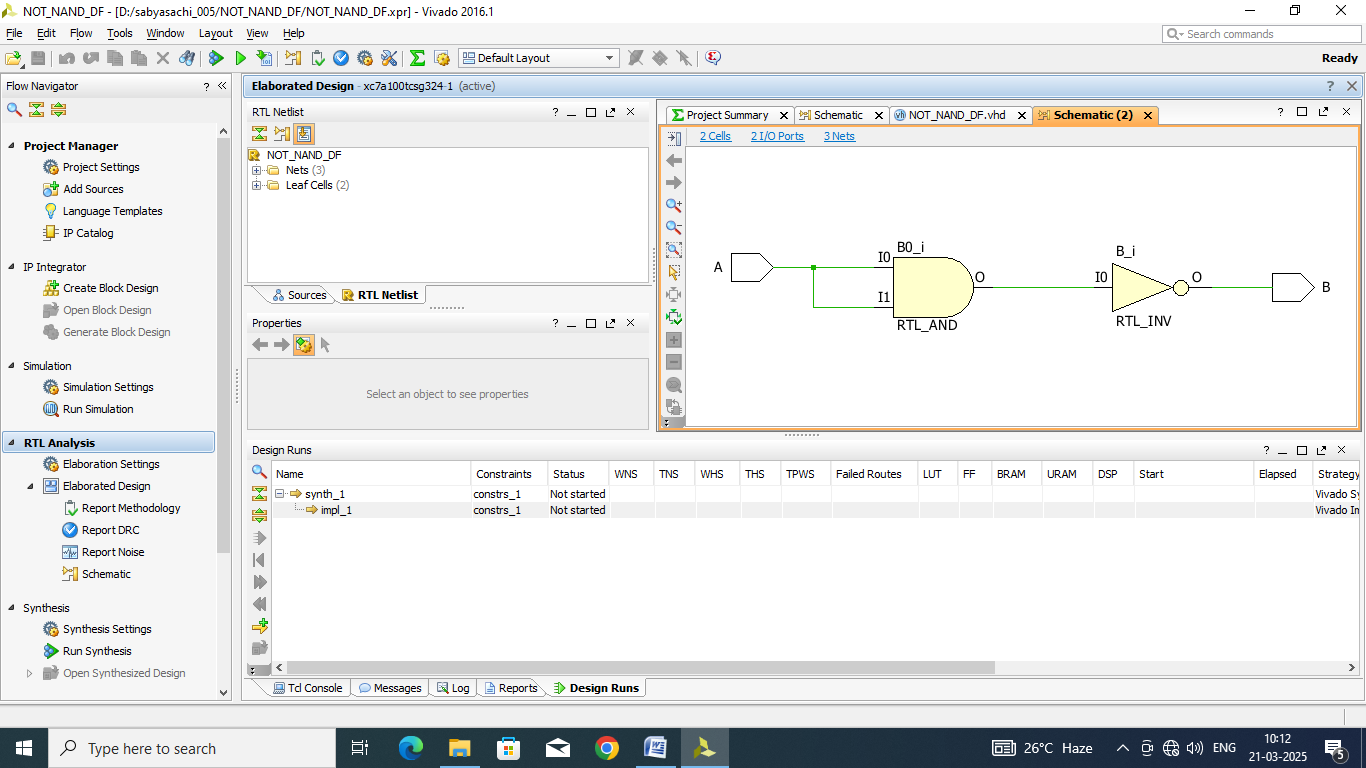
architecture Dataflow of NOT\_NAND\_DF is

begin

B <= A nand A;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity NOT\_NAND\_TBW is

-- Port ( );

end NOT\_NAND\_TBW;

architecture Dataflow of NOT\_NAND\_TBW is

component NOT\_NAND\_DF is

Port ( A : in STD\_LOGIC;

B : out STD\_LOGIC);

end component;

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

signal B1: STD\_LOGIC;

begin

UUT: NOT\_NAND\_DF port map(A=>A1, B=>B1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

wait for 100ns;

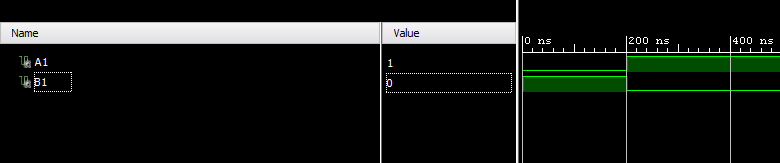
a1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**



# XOR Gate Using NAND Gate Dataflow

**VHD CODE:**

entity XOR\_NAND\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XOR\_NAND\_DF;

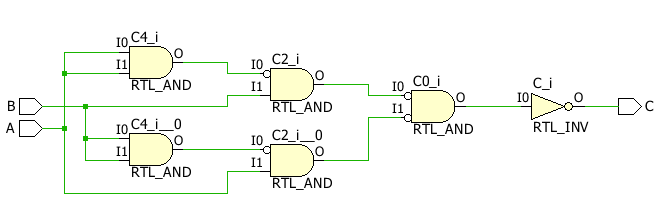
architecture Dataflow of XOR\_NAND\_DF is

begin

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

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity XOR\_NAND\_TBW is

-- Port ( );

end XOR\_NAND\_TBW;

architecture Dataflow of XOR\_NAND\_TBW is

component XOR\_NAND\_DF 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: XOR\_NAND\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# XNOR Gate Using NAND Gate Dataflow

**VHD CODE:**

entity XNOR\_NAND\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XNOR\_NAND\_DF;

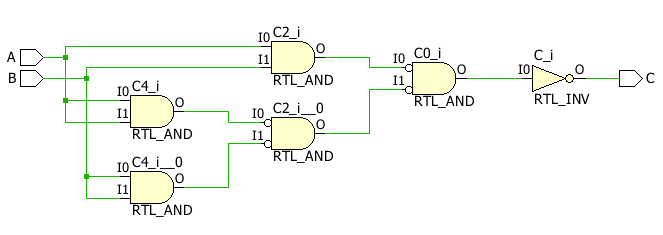
architecture Dataflow of XNOR\_NAND\_DF is

begin

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

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity XNOR\_NAND\_TBW is

-- Port ( );

end XNOR\_NAND\_TBW;

architecture Dataflow of XNOR\_NAND\_TBW is

component XNOR\_NAND\_DF 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: XNOR\_NAND\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# XOR Gate Dataflow

**VHD CODE:**

entity XOR\_GATE\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XOR\_GATE\_DF;

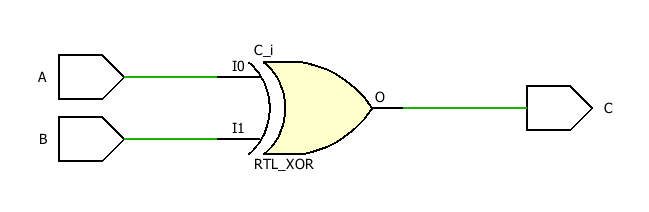
architecture Dataflow of XOR\_GATE\_DF is

begin

C<= A xor B;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity XOR\_GATE\_TBW is

-- Port ( );

end XOR\_GATE\_TBW;

architecture Dataflow of XOR\_GATE\_TBW is

component XOR\_GATE\_DF 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: XOR\_GATE\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# XNOR Gate Dataflow

**VHD CODE:**

entity XNOR\_GATE\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XNOR\_GATE\_DF;

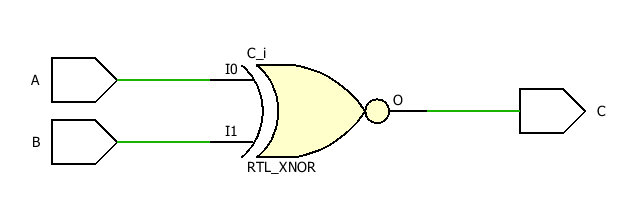
architecture Dataflow of XNOR\_GATE\_DF is

begin

C<= A xnor B;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity XNOR\_GATE\_TBW is

-- Port ( );

end XNOR\_GATE\_TBW;

architecture Dataflow of XNOR\_GATE\_TBW is

component XNOR\_GATE\_DF 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: XNOR\_GATE\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# AND Gate using nor Dataflow

**VHD CODE:**

entity AND\_NOR\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end AND\_NOR\_DF;

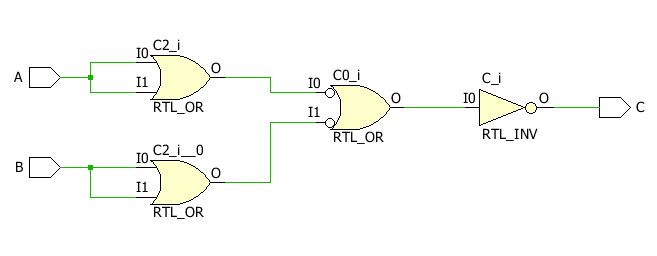
architecture Dataflow of AND\_NOR\_DF is

begin

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

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity AND\_NOR\_TBW is

-- Port ( );

end AND\_NOR\_TBW;

architecture Dataflow of AND\_NOR\_TBW is

component AND\_NOR\_DF 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: AND\_NOR\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# OR Gate using nor Dataflow

**VHD CODE:**

entity OR\_NOR\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end OR\_NOR\_DF;

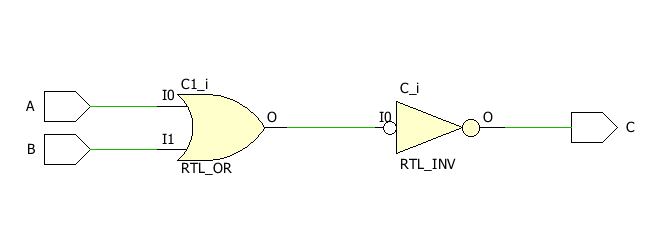
architecture Dataflow of OR\_NOR\_DF is

begin

C<= not(A nor B);

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity OR\_NOR\_TBW is

-- Port ( );

end OR\_NOR\_TBW;

architecture Dataflow of OR\_NOR\_TBW is

component OR\_NOR\_DF 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: OR\_NOR\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# NOT Gate using nor Dataflow

**VHD CODE:**

entity NOT\_NOR\_DF is

Port ( A : in STD\_LOGIC;

B : out STD\_LOGIC);

end NOT\_NOR\_DF;

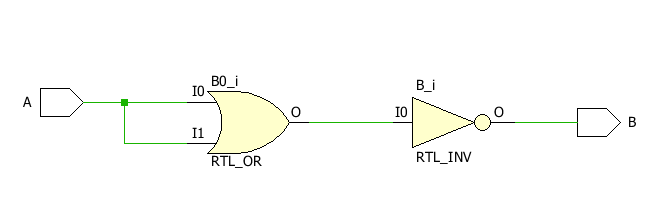
architecture Dataflow of NOT\_NOR\_DF is

begin

B<= A nor A;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity NOT\_NOR\_TBW is

-- Port ( );

end NOT\_NOR\_TBW;

architecture Dataflow of NOT\_NOR\_TBW is

component NOT\_NOR\_DF is

Port ( A : in STD\_LOGIC;

B : out STD\_LOGIC);

end component;

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

signal B1: STD\_LOGIC;

begin

UUT: NOT\_NOR\_DF port map(A=>A1, B=>B1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

wait for 100ns;

a1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# XOR Gate using nor Dataflow

**VHD CODE:**

entity XOR\_NOR\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XOR\_NOR\_DF;

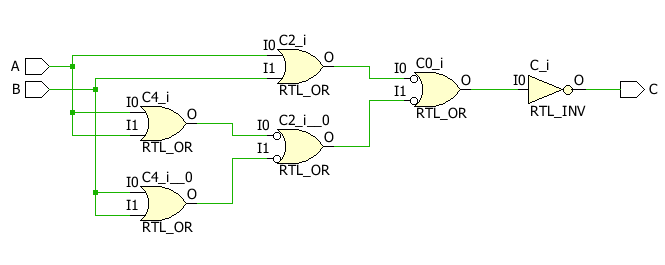
architecture Dataflow of XOR\_NOR\_DF is

begin

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

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity XOR\_NOR\_TBW is

-- Port ( );

end XOR\_NOR\_TBW;

architecture Dataflow of XOR\_NOR\_TBW is

component XOR\_NOR\_DF 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: XOR\_NOR\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='0';

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# XNOR Gate using nor Dataflow

**VHD CODE:**

entity XNOR\_NOR\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XNOR\_NOR\_DF;

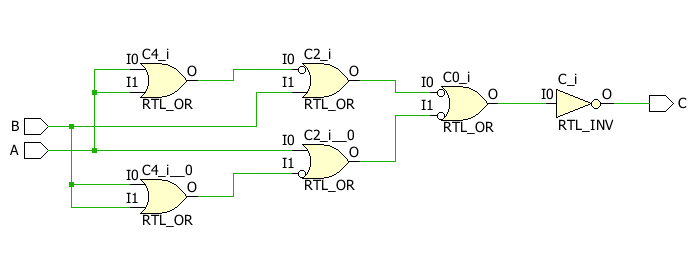
architecture Dataflow of XNOR\_NOR\_DF is

begin

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

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity XNOR\_NOR\_TBW is

-- Port ( );

end XNOR\_NOR\_TBW;

architecture Dataflow of XNOR\_NOR\_TBW is

component XNOR\_NOR\_DF 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: XNOR\_NOR\_DF port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

a1<='0';

b1<='1';

wait for 100ns;

a1<='1';

b1<='0';

wait for 100ns;

a1<='1';

b1<='1';

wait;

end process;

end dataflow;

**TBW Waveform:**

# 

# AND IF Behavioral

**VHD CODE:**

entity AND\_IF\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end AND\_IF\_BH;

architecture Behavioral of AND\_IF\_BH is

begin

process(A,B)

begin

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

C<='1';

else

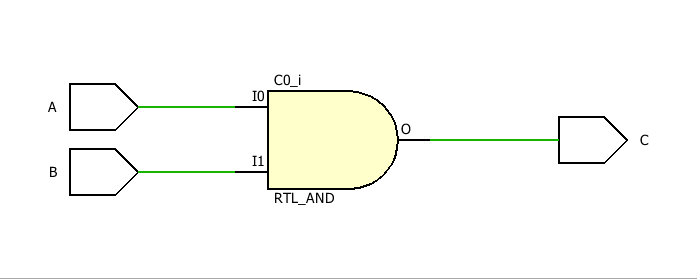
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity AND\_IF\_TBW is

-- Port ( );

end AND\_IF\_TBW;

architecture Behavioral of AND\_IF\_TBW is

component AND\_IF\_BH 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: AND\_IF\_BH port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='1';

wait for 100ns;

A1<='1';

B1<='0';

wait for 100ns;

A1<='1';

B1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# OR IF Behavioral

**VHD CODE:**

entity OR\_IF\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end OR\_IF\_BH;

architecture Behavioral of OR\_IF\_BH is

begin

process(A,B)

begin

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

C<='1';

else

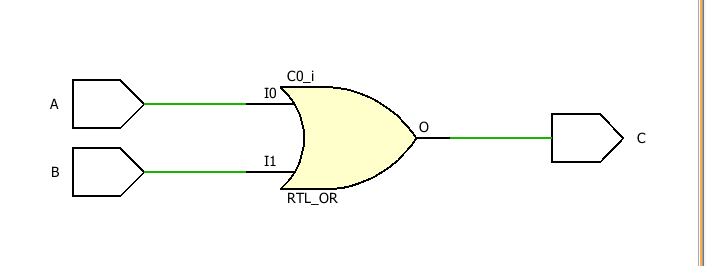
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity OR\_IF\_TBW is

-- Port ( );

end OR\_IF\_TBW;

architecture Behavioral of OR\_IF\_TBW is

component OR\_IF\_BH 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: OR\_IF\_BH port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='1';

wait for 100ns;

A1<='1';

B1<='0';

wait for 100ns;

A1<='1';

B1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# not IF Behavioral

**VHD CODE:**

entity NOT\_IF\_BH is

Port ( A : in STD\_LOGIC;

B : out STD\_LOGIC);

end NOT\_IF\_BH;

architecture Behavioral of NOT\_IF\_BH is

begin

process(A)

begin

if(A='1') then

B<='0';

else

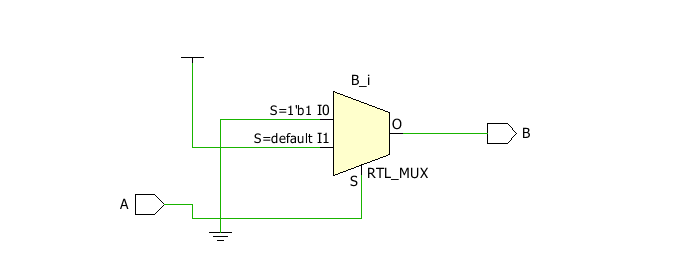
B<='1';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity NOT\_IF\_TBW is

-- Port ( );

end NOT\_IF\_TBW;

architecture Behavioral of NOT\_IF\_TBW is

component NOT\_IF\_BH is

Port ( A : in STD\_LOGIC;

B : out STD\_LOGIC);

end component;

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

signal B1: STD\_LOGIC;

begin

UUT: NOT\_IF\_BH port map(A=>A1, B=>B1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

wait for 100ns;

A1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# NAND IF Behavioral

**VHD CODE:**

entity NAND\_IF\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end NAND\_IF\_BH;

architecture Behavioral of NAND\_IF\_BH is

begin

process(A,B)

begin

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

C<='0';

else

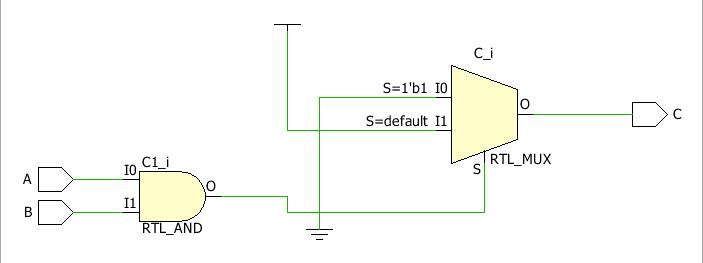
C<='1';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity NAND\_IF\_TBW is

-- Port ( );

end NAND\_IF\_TBW;

architecture Behavioral of NAND\_IF\_TBW is

component NAND\_IF\_BH 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\_IF\_BH port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='1';

wait for 100ns;

A1<='1';

B1<='0';

wait for 100ns;

A1<='1';

B1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# NOR IF Behavioral

**VHD CODE:**

entity NOR\_IF\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end NOR\_IF\_BH;

architecture Behavioral of NOR\_IF\_BH is

begin

process(A,B)

begin

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

C<='1';

else

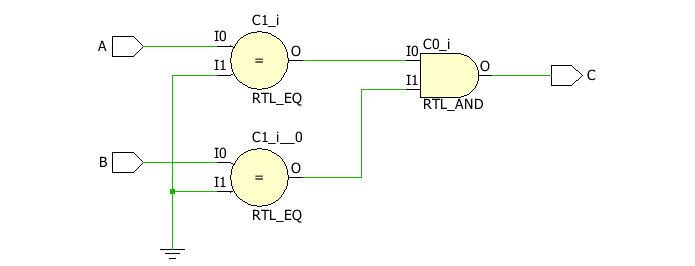
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity NOR\_IF\_TBW is

-- Port ( );

end NOR\_IF\_TBW;

architecture Behavioral of NOR\_IF\_TBW is

component NOR\_IF\_BH 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\_IF\_BH port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='1';

wait for 100ns;

A1<='1';

B1<='0';

wait for 100ns;

A1<='1';

B1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# XOR IF Behavioral

**VHD CODE:**

entity XOR\_IF\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XOR\_IF\_BH;

architecture Behavioral of XOR\_IF\_BH is

begin

process(A,B)

begin

if(A=B) then

C<='0';

else

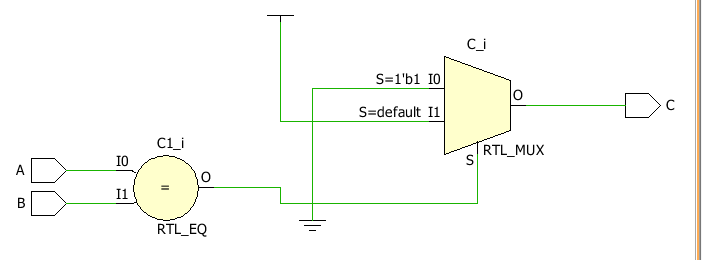
C<='1';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity XOR\_IF\_TBW is

-- Port ( );

end XOR\_IF\_TBW;

architecture Behavioral of XOR\_IF\_TBW is

component XOR\_IF\_BH 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: XOR\_IF\_BH port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='1';

wait for 100ns;

A1<='1';

B1<='0';

wait for 100ns;

A1<='1';

B1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# XNOR IF Behavioral

**VHD CODE:**

entity XNOR\_IF\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end XNOR\_IF\_BH;

architecture Behavioral of XNOR\_IF\_BH is

begin

process(A,B)

begin

if(A=B) then

C<='1';

else

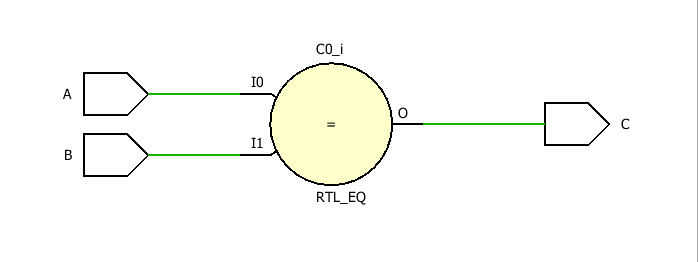
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity XNOR\_IF\_TBW is

-- Port ( );

end XNOR\_IF\_TBW;

architecture Behavioral of XNOR\_IF\_TBW is

component XNOR\_IF\_BH 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: XNOR\_IF\_BH port map(A=>A1, B=>B1, C=>C1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='1';

wait for 100ns;

A1<='1';

B1<='0';

wait for 100ns;

A1<='1';

B1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# HFADD Behavioral

**VHD CODE:**

entity HFADD\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC;

S : out STD\_LOGIC);

end HFADD\_BH;

architecture Behavioral of HFADD\_BH 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

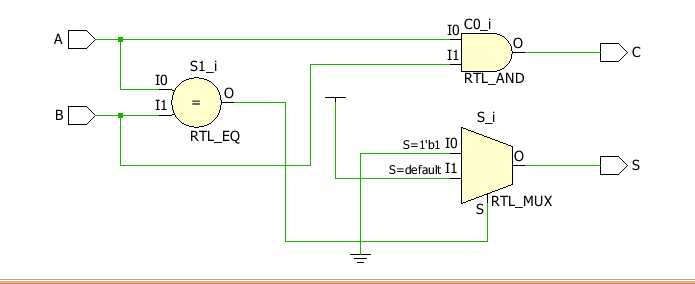
C<='0';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity HFADD\_TBW is

-- Port ( );

end HFADD\_TBW;

architecture Behavioral of HFADD\_TBW is

component HFADD\_BH 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: HFADD\_BH port map(A=>A1, B=>B1, C=>C1, S=>S1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='1';

wait for 100ns;

A1<='1';

B1<='0';

wait for 100ns;

A1<='1';

B1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# HFADD DATAFLOW

**VHD CODE:**

entity HFADD\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end HFADD\_DF;

architecture Dataflow of HFADD\_DF is

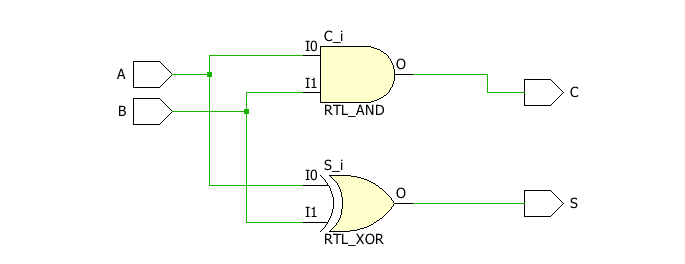
begin

S<= A XOR B;

C<= A AND B;

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity HFADD\_TBW is

-- Port ( );

end HFADD\_TBW;

architecture Dataflow of HFADD\_TBW is

component HFADD\_DF 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: HFADD\_DF port map(A=>A1, B=>B1, C=>C1, S=>S1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='1';

wait for 100ns;

A1<='1';

B1<='0';

wait for 100ns;

A1<='1';

B1<='1';

wait;

end process;

end Dataflow;

**TBW Waveform:**

# 

# FULLADD Behavioral

**VHD CODE:**

entity FULLADD\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end FULLADD\_BH;

architecture Behavioral of FULLADD\_BH is

begin

process(A,B,Cin)

begin

if(A='0') then

if(B=Cin) then

S<='0';

else

S<='1';

end if;

else

if(B=Cin) then

S<='1';

else

S<='0';

end if;

end if;

if((A='1' and B='1') or (B='1' and Cin='1') or (Cin='1' and A='1')) then

Cout<='1';

else

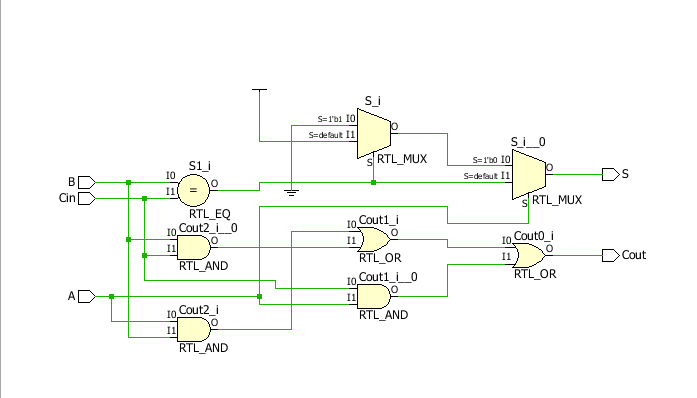
Cout<='0';

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity FULLADD\_TBW is

-- Port ( );

end FULLADD\_TBW;

architecture Behavioral of FULLADD\_TBW is

component FULLADD\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end component;

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

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

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

signal S1: STD\_LOGIC;

signal Cout1: STD\_LOGIC;

begin

UUT: FULLADD\_BH port map(A=>A1, B=>B1, Cin=>Cin1, S=>S1, Cout=>Cout1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='0';

Cin1<='1';

wait for 100ns;

A1<='0';

B1<='1';

Cin1<='0';

wait for 100ns;

A1<='0';

B1<='1';

Cin1<='1';

wait for 100ns;

A1<='1';

B1<='0';

Cin1<='0';

wait for 100ns;

A1<='1';

B1<='0';

Cin1<='1';

wait for 100ns;

A1<='1';

B1<='1';

Cin1<='0';

wait for 100ns;

A1<='1';

B1<='1';

Cin1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**

# 

# fULLADD DATAFLOW

**VHD CODE:**

entity FULLADD\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end FULLADD\_DF;

architecture Dataflow of FULLADD\_DF is

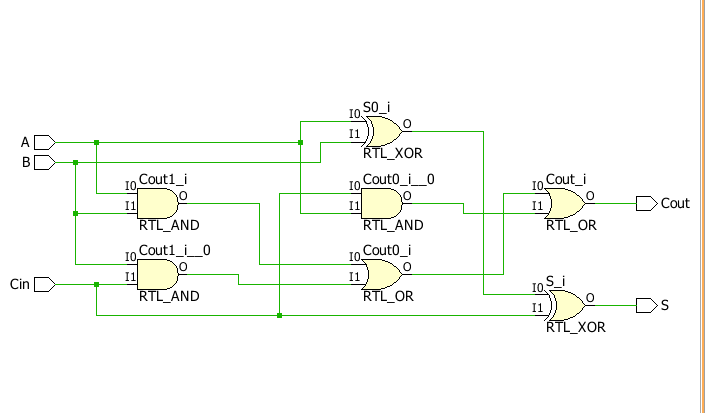
begin

S<= (A XOR B) XOR Cin;

Cout<= (((A AND B) OR (B AND Cin)) OR (Cin AND A));

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity FULLADD\_TBW is

-- Port ( );

end FULLADD\_TBW;

architecture Dataflow of FULLADD\_TBW is

component FULLADD\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end component;

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

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

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

signal S1: STD\_LOGIC;

signal Cout1: STD\_LOGIC;

begin

UUT: FULLADD\_DF port map(A=>A1, B=>B1, Cin=>Cin1, S=>S1, Cout=>Cout1);

Stim\_proc:process

begin

wait for 100ns;

A1<='0';

B1<='0';

Cin1<='1';

wait for 100ns;

A1<='0';

B1<='1';

Cin1<='0';

wait for 100ns;

A1<='0';

B1<='1';

Cin1<='1';

wait for 100ns;

A1<='1';

B1<='0';

Cin1<='0';

wait for 100ns;

A1<='1';

B1<='0';

Cin1<='1';

wait for 100ns;

A1<='1';

B1<='1';

Cin1<='0';

wait for 100ns;

A1<='1';

B1<='1';

Cin1<='1';

wait;

end process;

end Dataflow;

**TBW Waveform:**

# 

# 2:1 MUX DATA FLOW

**VHD CODE:**

entity MUX\_2X1\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end MUX\_2X1\_DF;

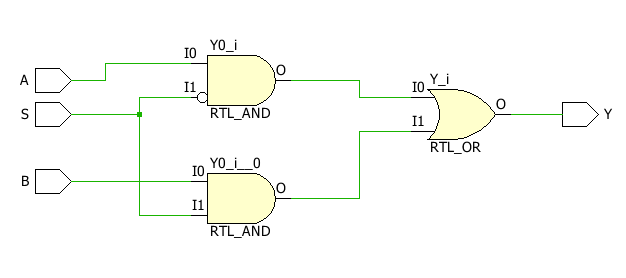
architecture Dataflow of MUX\_2X1\_DF is

begin

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

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity MUX\_2X1\_DF\_TBW is

-- Port ( );

end MUX\_2X1\_DF\_TBW;

architecture Dataflow of MUX\_2X1\_DF\_TBW is

component MUX\_2X1\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

signal a1: STD\_LOGIC :='0';

signal b1: STD\_LOGIC :='0';

signal e1: STD\_LOGIC :='0';

signal y1: STD\_LOGIC ;

begin

uut: MUX\_2X1\_DF port map (A=>a1, B=>b1, S=>e1, Y=>y1);

stim\_proc: process

begin

wait for 100 ns;

a1<='0';

b1<='0';

e1<='0';

wait for 100 ns;

a1<='0';

b1<='0';

e1<='1';

wait for 100 ns;

a1<='0';

b1<='1';

e1<='0';

wait for 100 ns;

a1<='0';

b1<='1';

e1<='1';

wait for 100 ns;

a1<='1';

b1<='0';

e1<='0';

wait for 100 ns;

a1<='1';

b1<='0';

e1<='1';

wait for 100 ns;

a1<='1';

b1<='1';

e1<='0';

wait for 100 ns;

a1<='1';

b1<='1';

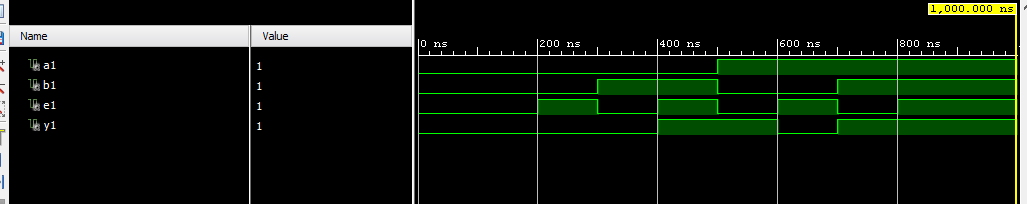
e1<='1';

wait;

end process;

end Dataflow;

**TBW Waveform:**



# 2:1 MUX BEHAVIORIAL

**VHD CODE:**

entity MUX\_2X1\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end MUX\_2X1\_BH;

architecture Behavioral of MUX\_2X1\_BH is

begin

process(A,B)

begin

if (S='0')then

Y<=A;

else

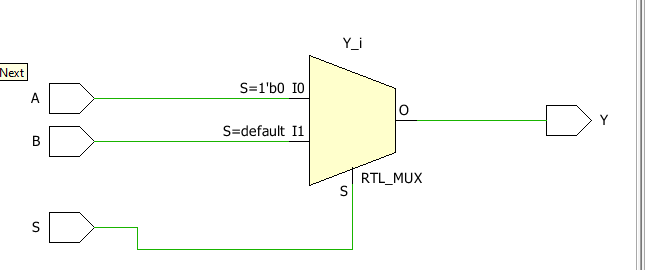
Y<=B;

end if;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity MUX\_2X1\_BH\_TBW is

-- Port ( );

end MUX\_2X1\_BH\_TBW;

architecture Behavioral of MUX\_2X1\_BH\_TBW is

component MUX\_2X1\_BH is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : in STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

signal a1: STD\_LOGIC :='0';

signal b1: STD\_LOGIC :='0';

signal e1: STD\_LOGIC :='0';

signal y1: STD\_LOGIC ;

begin

uut: MUX\_2X1\_BH port map (A=>a1, B=>b1, S=>e1, Y=>y1);

stim\_proc: process

begin

wait for 100 ns;

a1<='0';

b1<='0';

e1<='0';

wait for 100 ns;

a1<='0';

b1<='0';

e1<='1';

wait for 100 ns;

a1<='0';

b1<='1';

e1<='0';

wait for 100 ns;

a1<='0';

b1<='1';

e1<='1';

wait for 100 ns;

a1<='1';

b1<='0';

e1<='0';

wait for 100 ns;

a1<='1';

b1<='0';

e1<='1';

wait for 100 ns;

a1<='1';

b1<='1';

e1<='0';

wait for 100 ns;

a1<='1';

b1<='1';

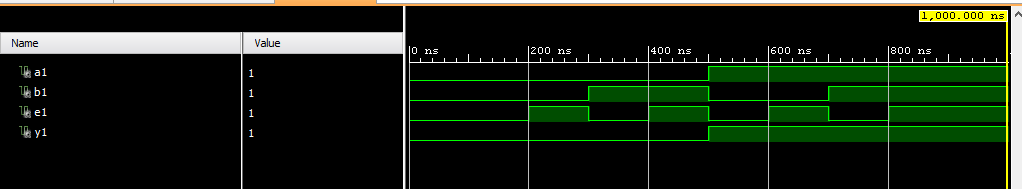
e1<='1';

wait;

end process;

end Behavioral;

**TBW Waveform:**



**4:1 mux dataflow**

**VHD CODE:**

entity multi41 is

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

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

Y : out STD\_LOGIC);

end multi41;

architecture Dataflow of multi41 is

begin

Y<=IP(0) When S="00"

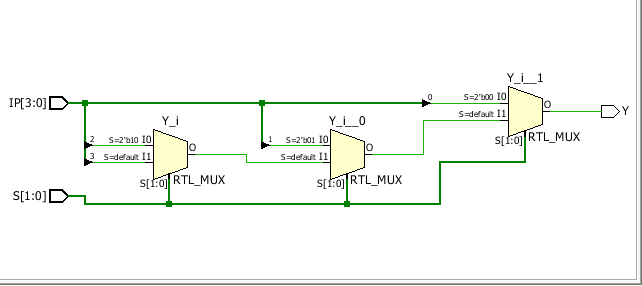
else IP(1) When S="01"

else IP(2) When S="10"

else IP(3);

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity multi41\_tbw is

-- Port ( );

end multi41\_tbw;

architecture Dataflow of multi41\_tbw is

component multi41 is

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

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

Y : out STD\_LOGIC);

end component;

signal IP1:STD\_LOGIC\_VECTOR(3 downto 0):="1010";

signal S1:STD\_LOGIC\_VECTOR(1 downto 0):="00";

signal Y1:STD\_LOGIC;

begin

uut: multi41 port map (IP=>IP1,S=>S1, Y=>Y1);

stim\_proc: process

begin

wait for 100 ns;

S1<="00";

wait for 100 ns;

S1<="01";

wait for 100 ns;

S1<="10";

wait for 100 ns;

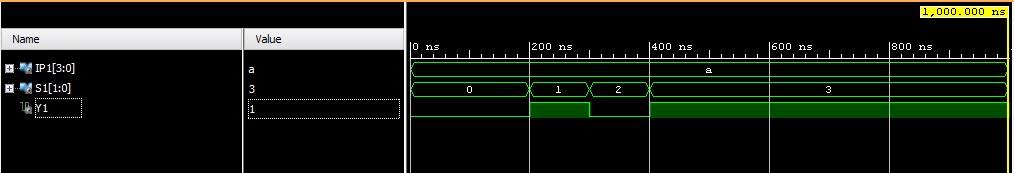
S1<="11";

wait;

end process;

end Behavioral;

**TBW Waveform:**



**BEHAVIARAL 4:1 MULTI**

**VHD CODE:**

entity MULTI41\_BF is

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

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

Y : out STD\_LOGIC);

end MULTI41\_BF;

architecture Behavioral of MULTI41\_BF is

begin

process(IP,S)

begin

case S is

when "00"=>Y<=IP(0);

when "01"=>Y<=IP(1);

when "10"=>Y<=IP(2);

when "11"=>Y<=IP(3);

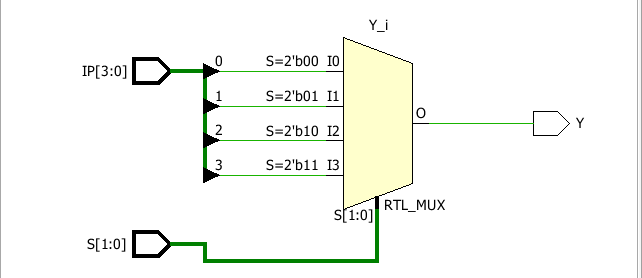
when others=>NULL;

end case;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity MULTI41\_BF\_TBW is

-- Port ( );

end MULTI41\_BF\_TBW;

architecture Behavioral of MULTI41\_BF\_TBW is

component MULTI41\_BF is

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

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

Y : out STD\_LOGIC);

end component;

signal IP1:STD\_LOGIC\_VECTOR(3 downto 0):="1010";

signal S1:STD\_LOGIC\_VECTOR(1 downto 0):="00";

signal Y1:STD\_LOGIC;

begin

uut: MULTI41\_BF port map (IP=>IP1,S=>S1, Y=>Y1);

stim\_proc: process

begin

wait for 100 ns;

S1<="00";

wait for 100 ns;

S1<="01";

wait for 100 ns;

S1<="10";

wait for 100 ns;

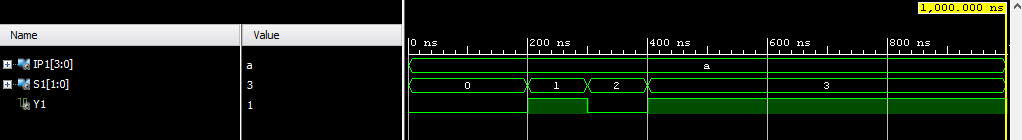
S1<="11";

wait;

end process;

end Behavioral;

**TBW Waveform:**



**3:8 DECODER DATAFLOW**

**VHD CODE:**

entity dec38 is

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

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

end dec38;

architecture dataflow of dec38 is

begin

OP(0)<='1' when IP= "000" else '0';

OP(1)<='1' when IP ="001" else '0';

OP(2)<='1' when IP ="010" else '0';

OP(3)<='1' when IP ="011" else '0';

OP(4)<='1' when IP ="100" else '0';

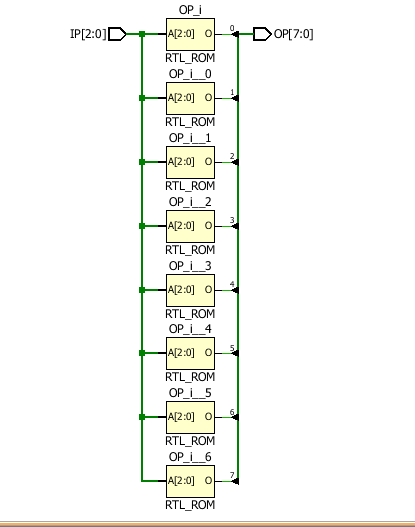
OP(5)<='1' when IP ="101" else '0';

OP(6)<='1' when IP ="110" else '0';

OP(7)<='1' when IP ="111" else '0';

end Dataflow;

**RTL Diagram:**

****

**TBW Code:**

entity dec38\_tbw is

-- Port ( );

end dec38\_tbw;

architecture dataflow of dec38\_tbw is

component dec38 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: dec38 port map (IP=>IP1,OP=>OP1);

stim\_proc: process

begin

wait for 100 ns;

IP1<="000";

wait for 100 ns;

IP1<="001";

wait for 100 ns;

IP1<="010";

wait for 100 ns;

IP1<="011";

wait for 100 ns;

IP1<="100";

wait for 100 ns;

IP1<="101";

wait for 100 ns;

IP1<="110";

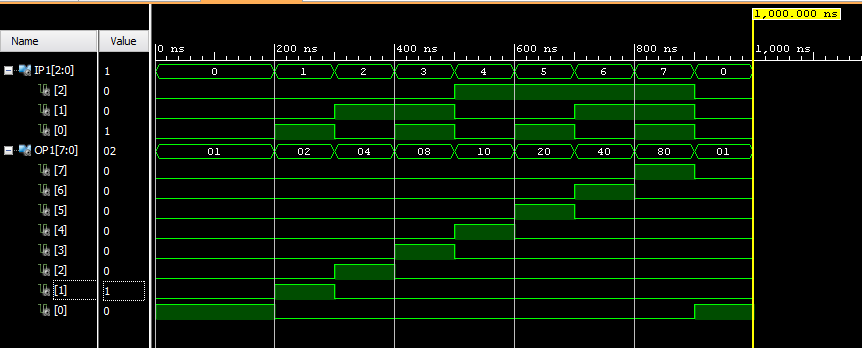
wait for 100 ns;

IP1<="111";

end process;

end dataflow;

**TBW Waveform:**



**3:8 DECODER BEHAVIORAL**

**VHD CODE:**

entity DECODER\_BDF is

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

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

end DECODER\_BDF;

architecture Behavioral of DECODER\_BDF 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';

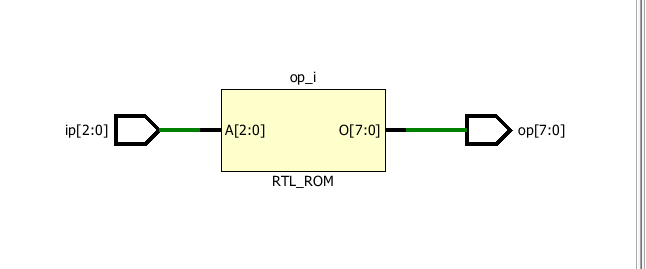
when others=>NULL;

end case;

end process;

end Behavioral;

**RTL Diagram:**

****

**TBW Code:**

entity DECODER\_BDF\_TBW is

-- Port ( );

end DECODER\_BDF\_TBW;

architecture Behavioral of DECODER\_BDF\_TBW is

component DECODER\_BDF 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) :="00000000" ;

begin

uut: DECODER\_BDF port map (ip=>ip1,op=>op1);

stim\_proc: process

begin

wait for 100ns;

ip1<="000";

wait for 100ns;

ip1<="001";

wait for 100ns;

ip1<="010";

wait for 100ns;

ip1<="011";

wait for 100ns;

ip1<="100";

wait for 100ns;

ip1<="101";

wait for 100ns;

ip1<="110";

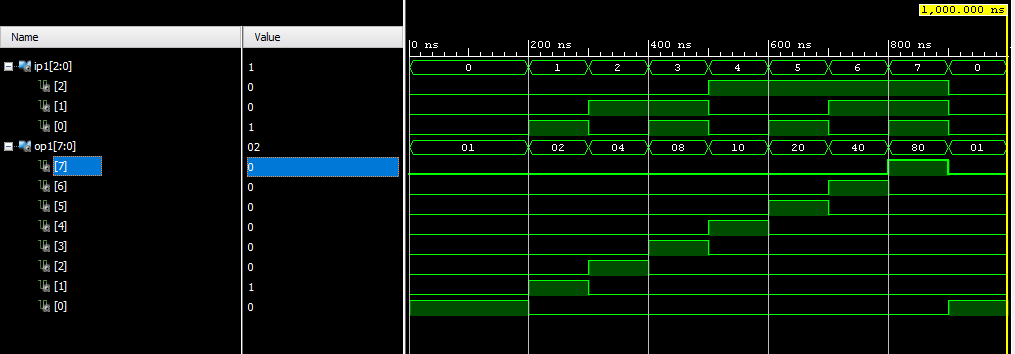
wait for 100ns;

ip1<="111";

end process;

end Behavioral;

**TBW Waveform:**



**FULL ADDER STRUCTURE**

**VHD CODE:**

entity FULL\_ADDER\_STRUCTURAL is

Port ( X : in STD\_LOGIC;

Y : in STD\_LOGIC;

Z : in STD\_LOGIC;

Carry : out STD\_LOGIC;

Sum : out STD\_LOGIC);

end FULL\_ADDER\_STRUCTURAL;

architecture Structural of FULL\_ADDER\_STRUCTURAL is

component HFADD\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end component;

component OR\_Gate\_DF is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC);

end component;

signal S1:STD\_LOGIC;

signal C1:STD\_LOGIC;

signal C2:STD\_LOGIC;

begin

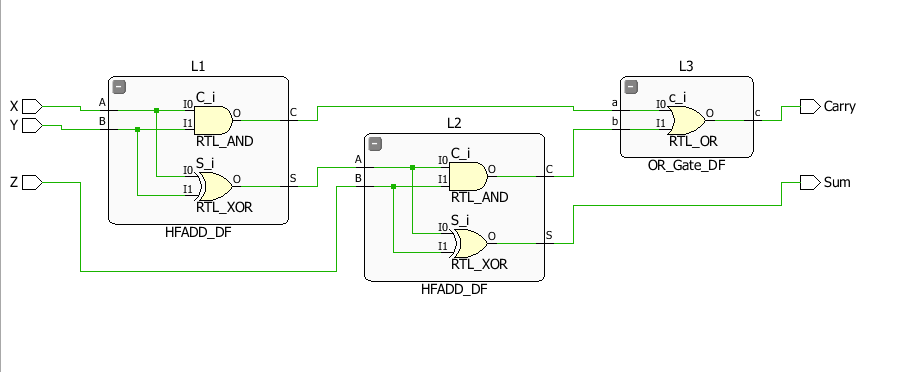
L1:HFADD\_DF Port Map(X,Y,S1,C1);

L2:HFADD\_DF Port Map(S1,Z,SUM,C2);

L3:OR\_Gate\_DF Port Map(C1,C2,CARRY);

end structural;

**RTL Diagram:**

****

**TBW Code:**

entity FULL\_ADDER\_STRUCTURAL\_TBW is

-- Port ( );

end FULL\_ADDER\_STRUCTURAL\_TBW;

architecture structural of FULL\_ADDER\_STRUCTURAL\_TBW is

component FULL\_ADDER\_STRUCTURAL 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 a1: STD\_LOGIC :='0';

signal b1: STD\_LOGIC :='0';

signal e1: STD\_LOGIC :='0';

signal c1: STD\_LOGIC ;

signal d1: STD\_LOGIC ;

begin

uut: FULL\_ADDER\_STRUCTURAL port map (X=>a1, Y=>b1, Z=>e1, Sum=>c1,Carry=>d1);

stim\_proc: process

begin

wait for 100 ns;

a1<='0';

b1<='0';

e1<='0';

wait for 100 ns;

a1<='0';

b1<='0';

e1<='1';

wait for 100 ns;

a1<='0';

b1<='1';

e1<='0';

wait for 100 ns;

a1<='0';

b1<='1';

e1<='1';

wait for 100 ns;

a1<='1';

b1<='0';

e1<='0';

wait for 100 ns;

a1<='1';

b1<='0';

e1<='1';

wait for 100 ns;

a1<='1';

b1<='1';

e1<='0';

wait for 100 ns;

a1<='1';

b1<='1';

e1<='1';

wait;

end process;

end structural;

**TBW Waveform:**

