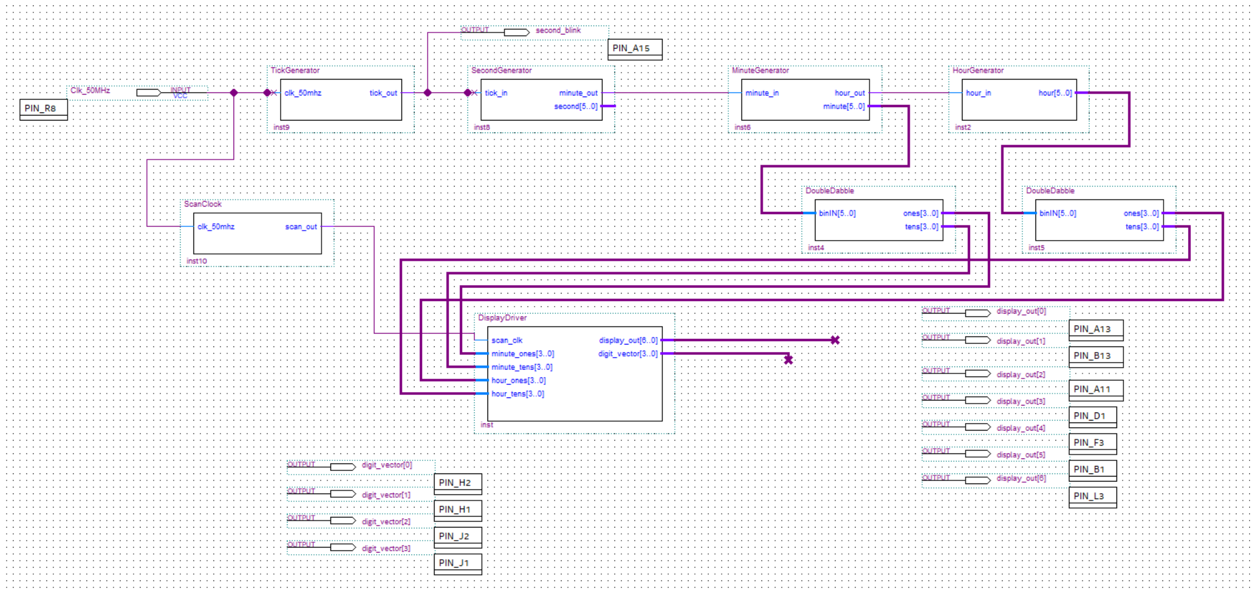
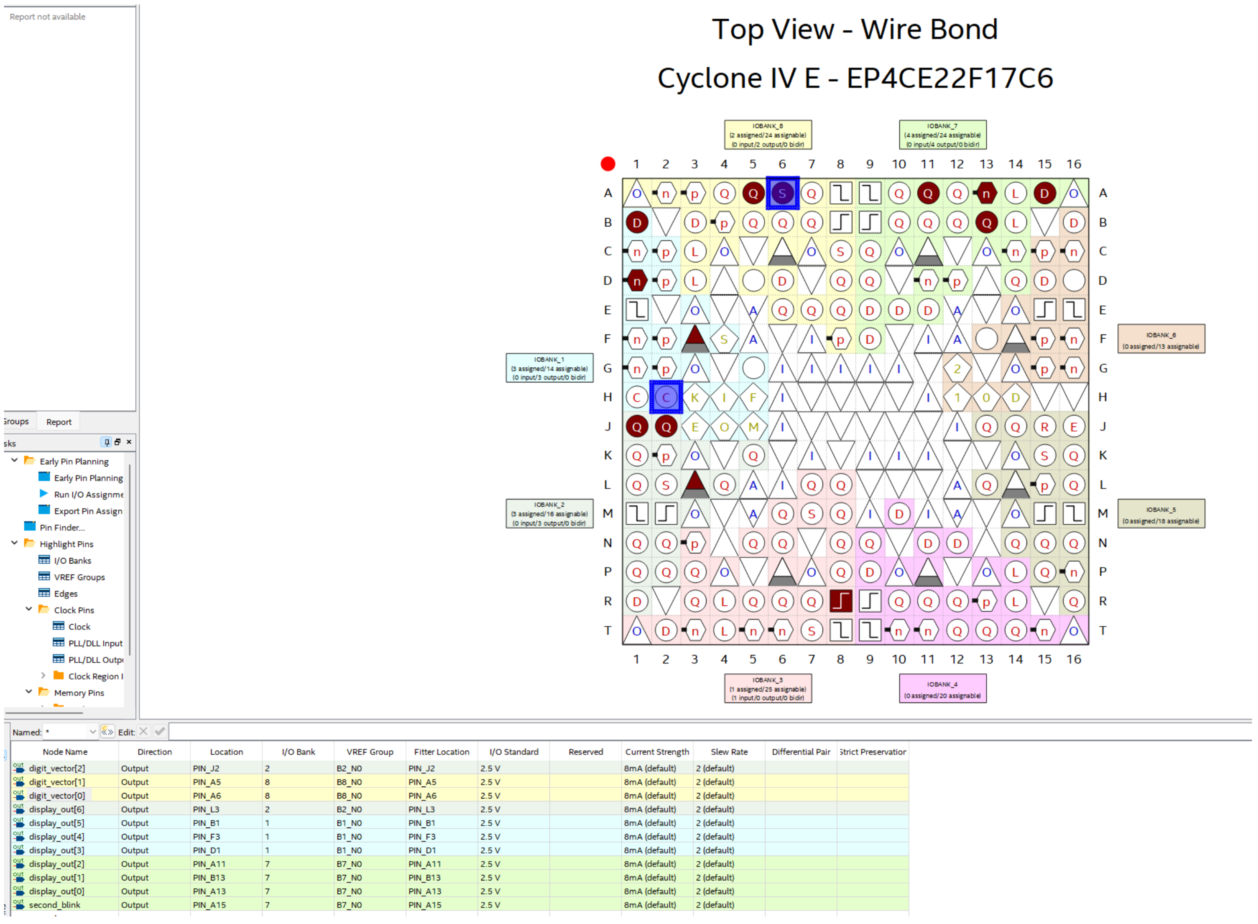
Top Level (Complete_Clock.bdf)

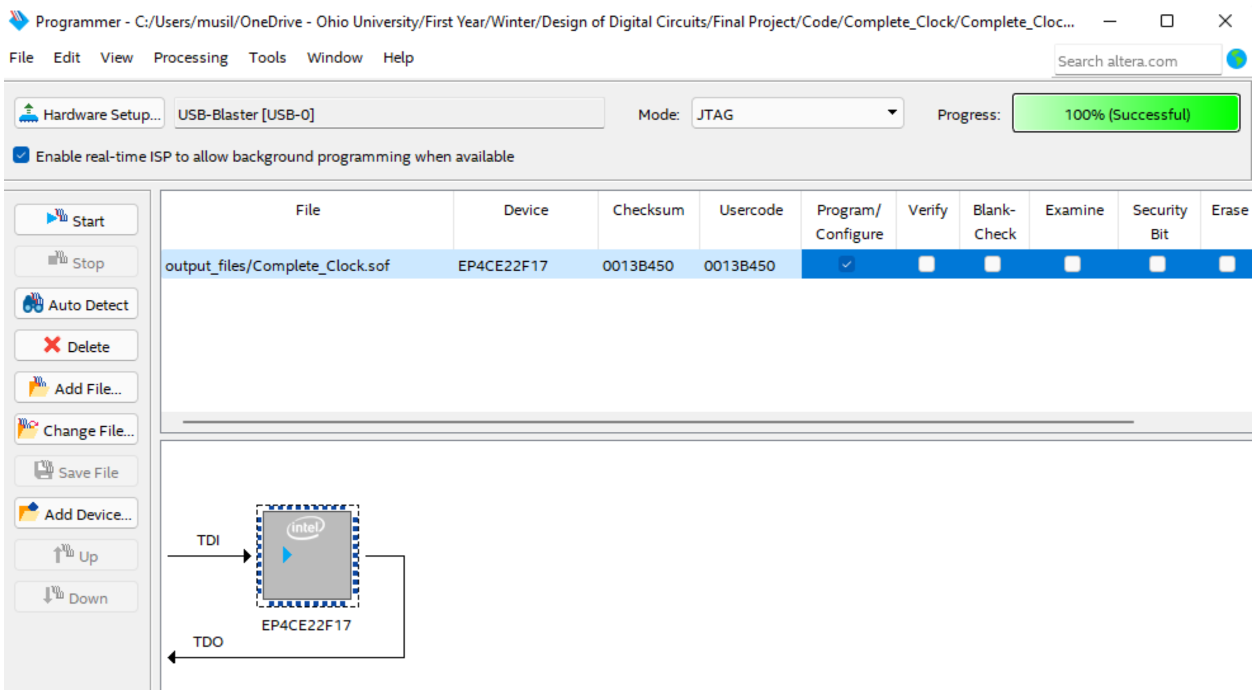


Pin Planner


Report not available
Top View - Wire Bond
Cyclone IV E - EP4CE22F17C6
IOBANK_8
(2 assigned/24 assignable)
(0 input/2 output/0 bidir)
IOBANK_7
(4 assigned/24 assignable)
(0 input/4 output/0 bidir)
IOBANK_1
(3 assigned/14 assignable)
(0 input/3 output/0 bidir)
IOBANK_2
(3 assigned/16 assignable)
(0 input/3 output/0 bidir)
IOBANK_6
(0 assigned/13 assignable)
IOBANK_5
(0 assigned/18 assignable)
IOBANK_3
(1 assigned/25 assignable)
(1 input/0 output/0 bidir)
IOBANK_4
(0 assigned/20 assignable)
Groups
Report
Early Pin Planning
Early Pin Planning
Run I/O Assignme
Export Pin Assign
Pin Finder...
Highlight Pins
I/O Banks
VREF Groups
Edges
Clock Pins
Clock
PLL/DLL Input
PLL/DLL Outpu
Clock Region I
Memory Pins
Named: *
Edit:

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
digit_vector[2]	Output	PIN_J2	2	B2_N0	PIN_J2	2.5 V		8mA (default)	2 (default)		
digit_vector[1]	Output	PIN_A5	8	B8_N0	PIN_A5	2.5 V		8mA (default)	2 (default)		
digit_vector[0]	Output	PIN_A6	8	B8_N0	PIN_A6	2.5 V		8mA (default)	2 (default)		
display_out[6]	Output	PIN_L3	2	B2_N0	PIN_L3	2.5 V		8mA (default)	2 (default)		
display_out[5]	Output	PIN_B1	1	B1_N0	PIN_B1	2.5 V		8mA (default)	2 (default)		
display_out[4]	Output	PIN_F3	1	B1_N0	PIN_F3	2.5 V		8mA (default)	2 (default)		
display_out[3]	Output	PIN_D1	1	B1_N0	PIN_D1	2.5 V		8mA (default)	2 (default)		
display_out[2]	Output	PIN_A11	7	B7_N0	PIN_A11	2.5 V		8mA (default)	2 (default)		
display_out[1]	Output	PIN_B13	7	B7_N0	PIN_B13	2.5 V		8mA (default)	2 (default)		
display_out[0]	Output	PIN_A13	7	B7_N0	PIN_A13	2.5 V		8mA (default)	2 (default)		
second_blink	Output	PIN_A15	7	B7_N0	PIN_A15	2.5 V		8mA (default)	2 (default)		

Programmer



Programmer - C:/Users/musil/OneDrive - Ohio University/First Year/Winter/Design of Digital Circuits/Final Project/Code/Complete_Clock/Complete_Cloc...

USB-Blaster [USB-0]

Mode: JTAG

Progress: 100% (Successful)

✓ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
output_files/Complete_Clock.sof	EP4CE22F17	0013B450	0013B450	✓					

Video of the device blinking

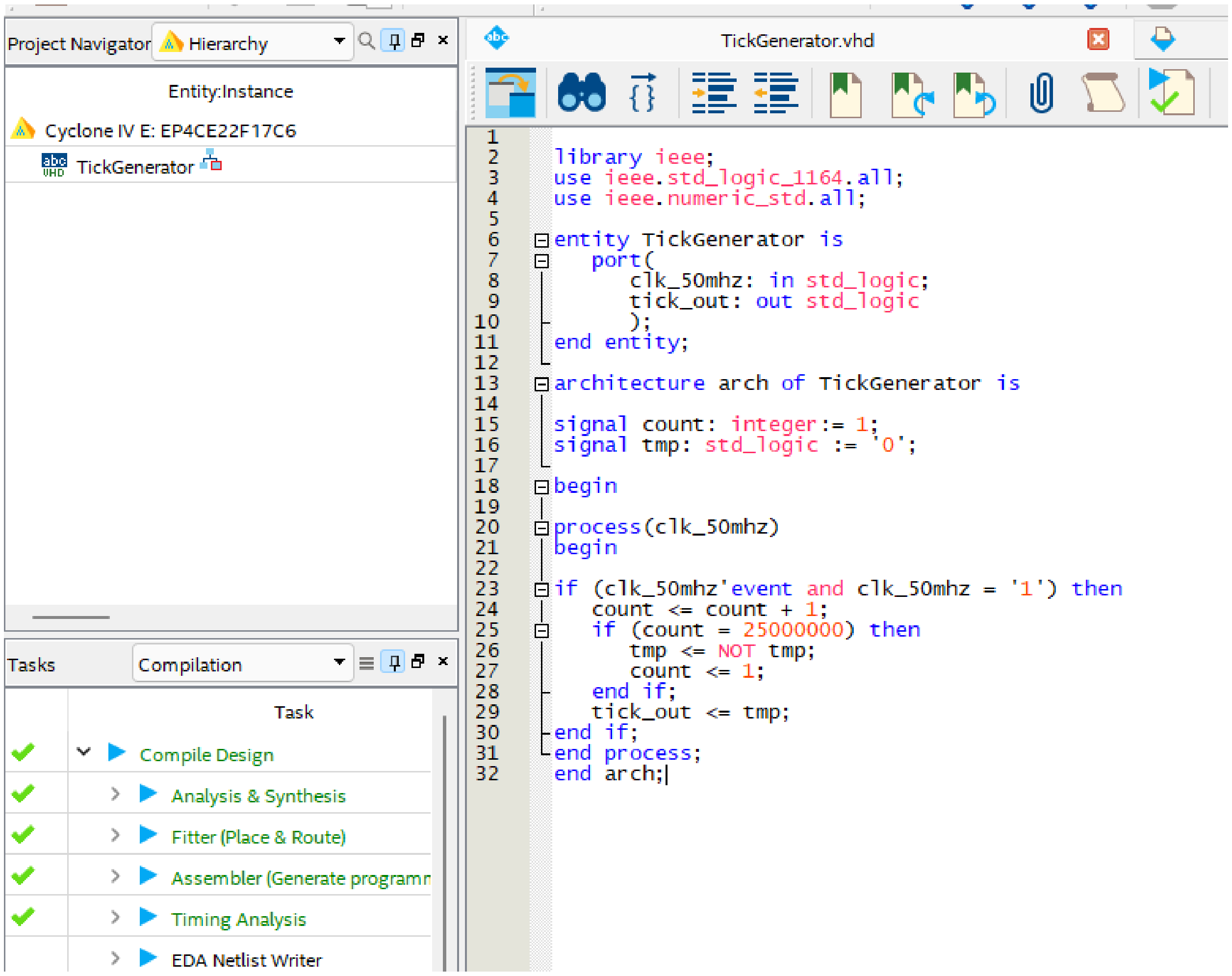
https://youtube.com/shorts/-HyueXFLDLs?feature=share

Screenshots of successfully compiled subblocks

The code and testbenches shown below were compiled using Quartus Lite and ModelSim (ModelSim is included with Quartus Lite)

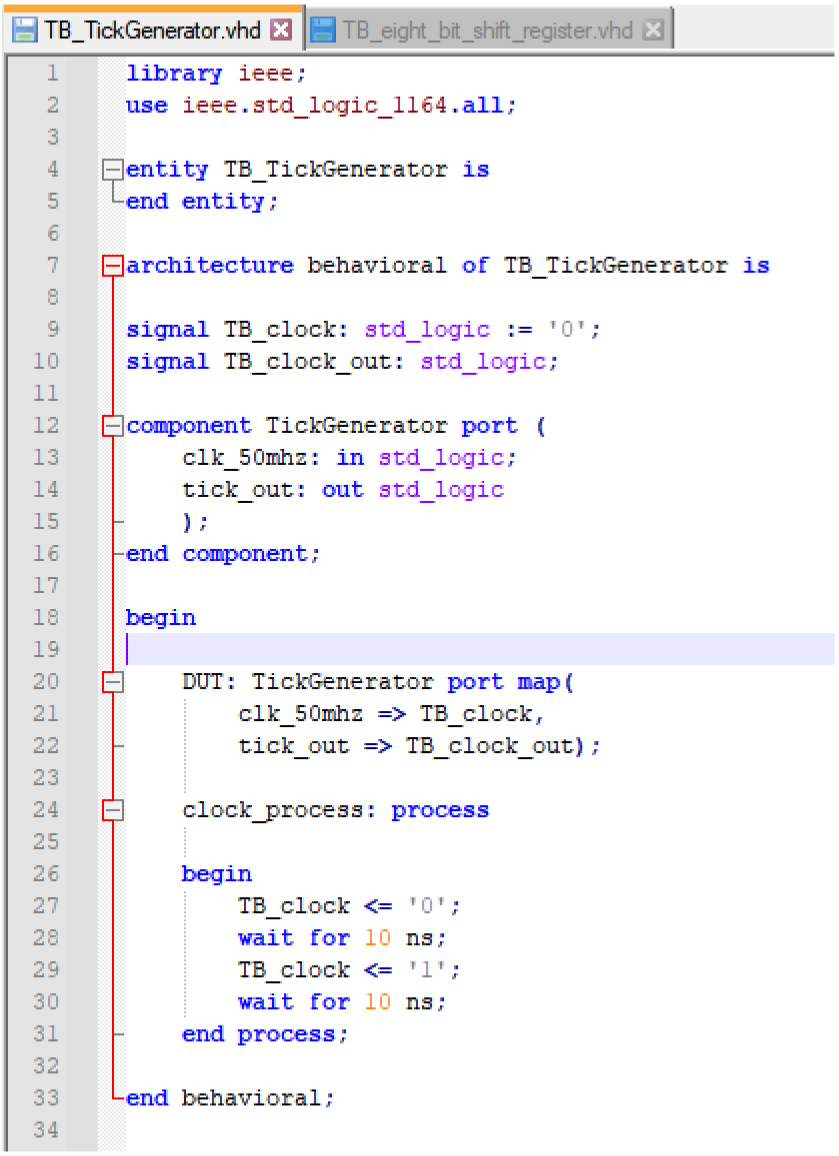
TickGenerator

Code



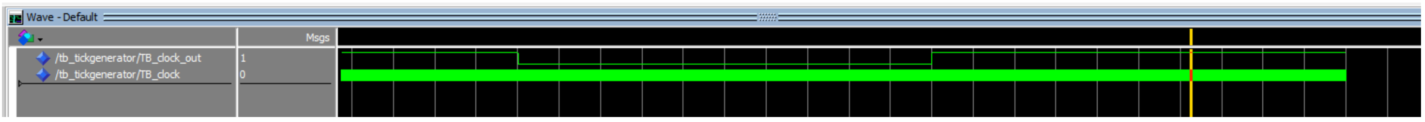
```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity TickGenerator is
    port(
        clk_50mhz: in std_logic;
        tick_out: out std_logic
        );
end entity;

architecture arch of TickGenerator is

signal count: integer:= 1;
signal tmp: std_logic := '0';

begin

process(clk_50mhz)
begin

if (clk_50mhz'event and clk_50mhz = '1') then
    count <= count + 1;
    if (count = 25000000) then
        tmp <= NOT tmp;
        count <= 1;
    end if;
    tick_out <= tmp;
end if;
end process;
end arch;
```

Testbench

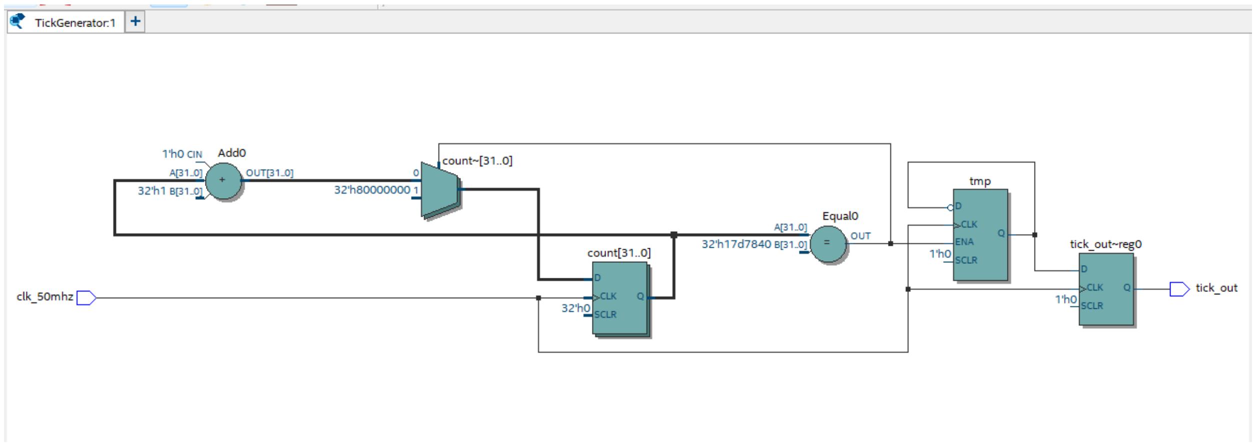


```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity TB_TickGenerator is
end entity;

architecture behavioral of TB_TickGenerator is

signal TB_clock: std_logic := '0';
signal TB_clock_out: std_logic;

component TickGenerator port (
    clk_50mhz: in std_logic;
    tick_out: out std_logic
    );
end component;

begin

    DUT: TickGenerator port map(
        clk_50mhz => TB_clock,
        tick_out => TB_clock_out);

    clock_process: process

    begin
        TB_clock <= '0';
        wait for 10 ns;
        TB_clock <= '1';
        wait for 10 ns;
    end process;

end behavioral;
```

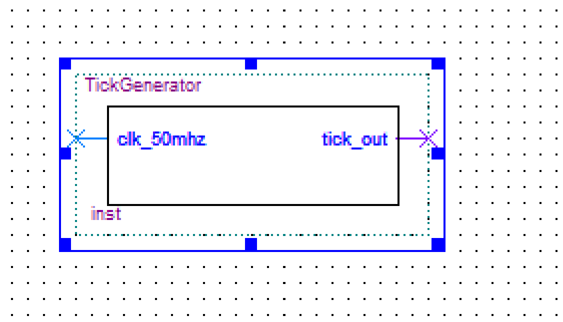
Testbench output



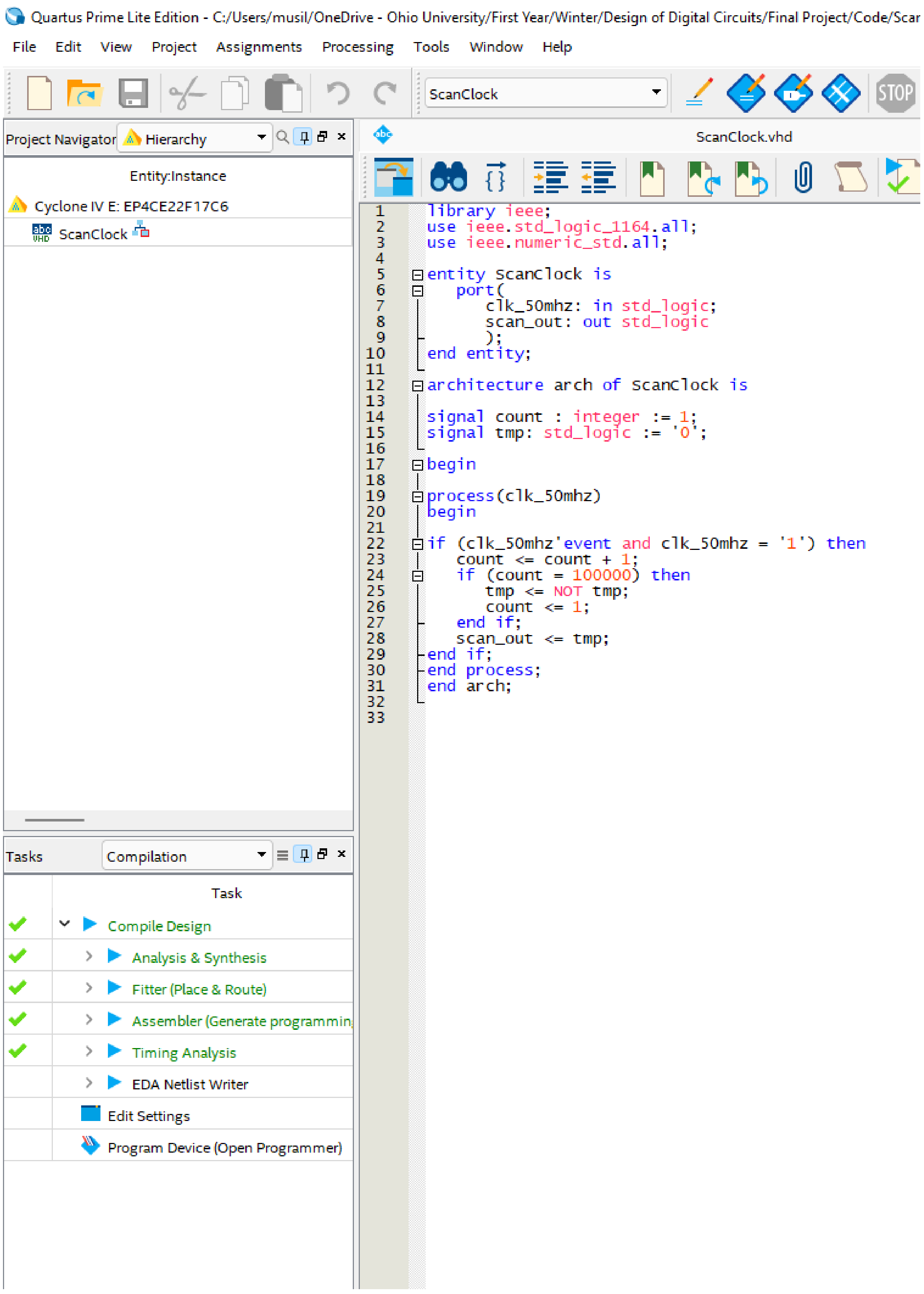
RTL Diagram



Block Design

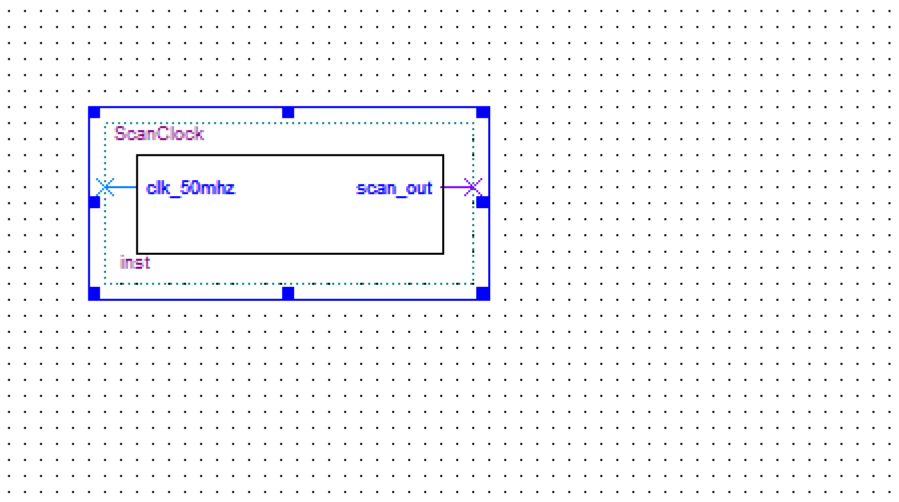


Scan Clock

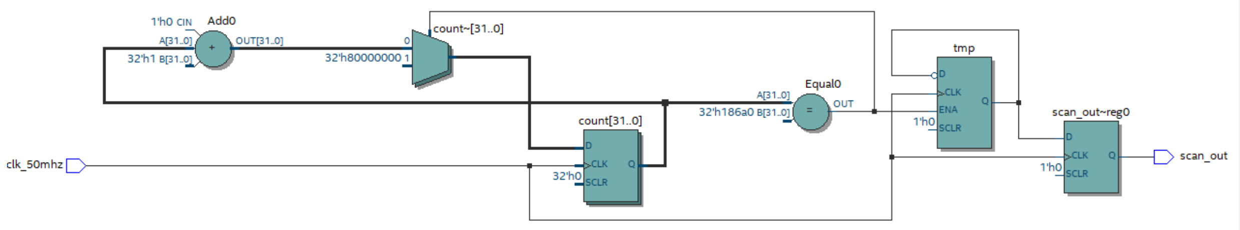

Quartus Prime Lite Edition - C:/Users/musil/OneDrive - Ohio University/First Year/Winter/Design of Digital Circuits/Final Project/Code/Scar
File Edit View Project Assignments Processing Tools Window Help
ScanClock
Project Navigator
Hierarchy
Entity:Instance
Cyclone IV E: EP4CE22F17C6
ScanClock
ScanClock.vhd
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programmin
Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity scanClock is
    port(
        clk_50mhz: in std_logic;
        scan_out: out std_logic
        );
end entity;

architecture arch of scanClock is

signal count : integer := 1;
signal tmp: std_logic := '0';

begin

process(clk_50mhz)
begin

if (clk_50mhz'event and clk_50mhz = '1') then
    count <= count + 1;
    if (count = 100000) then
        tmp <= NOT tmp;
        count <= 1;
    end if;
    scan_out <= tmp;
end if;
end process;
end arch;
```

Block Diagram

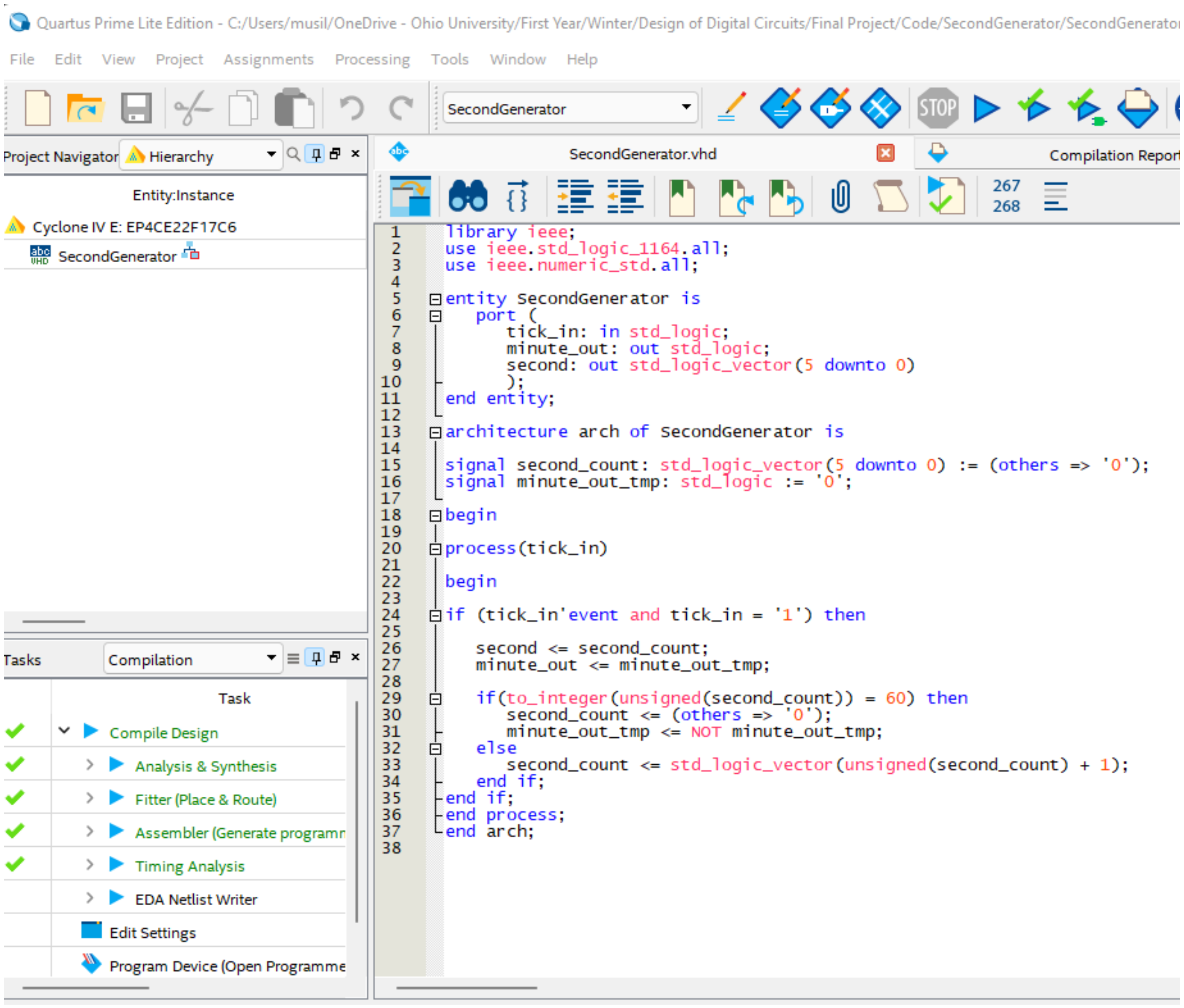


RTL Diagram



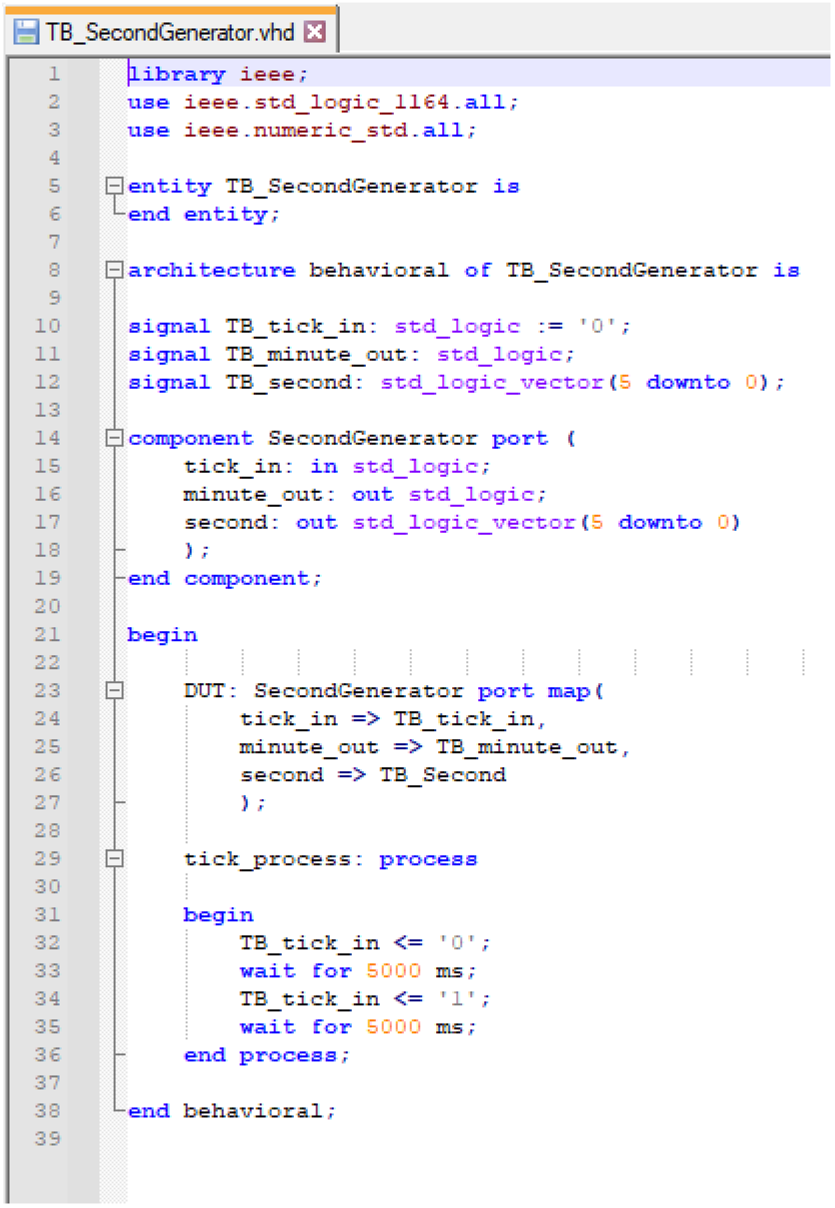
Second Generator

Code


Quartus Prime Lite Edition - C:/Users/musil/OneDrive - Ohio University/First Year/Winter/Design of Digital Circuits/Final Project/Code/SecondGenerator/SecondGenerator
File Edit View Project Assignments Processing Tools Window Help
SecondGenerator
Project Navigator Hierarchy
Entity:Instance
Cyclone IV E: EP4CE22F17C6
SecondGenerator
SecondGenerator.vhd
Compilation Report
Tasks Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programm
Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programme

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity SecondGenerator is
    port (
        tick_in: in std_logic;
        minute_out: out std_logic;
        second: out std_logic_vector(5 downto 0)
        );
end entity;

architecture arch of SecondGenerator is

signal second_count: std_logic_vector(5 downto 0) := (others => '0');
signal minute_out_tmp: std_logic := '0';

begin

process(tick_in)

begin

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

    second <= second_count;
    minute_out <= minute_out_tmp;

    if(to_integer(unsigned(second_count)) = 60) then
        second_count <= (others => '0');
        minute_out_tmp <= NOT minute_out_tmp;
    else
        second_count <= std_logic_vector(unsigned(second_count) + 1);
    end if;
end if;
end process;
end arch;
```

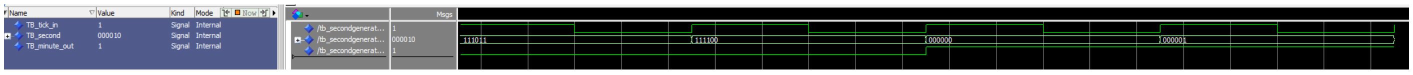
Testbench



TB_SecondGenerator.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity TB_SecondGenerator is
end entity;

architecture behavioral of TB_SecondGenerator is

signal TB_tick_in: std_logic := '0';
signal TB_minute_out: std_logic;
signal TB_second: std_logic_vector(5 downto 0);

component SecondGenerator port (
    tick_in: in std_logic;
    minute_out: out std_logic;
    second: out std_logic_vector(5 downto 0)
    );
end component;

begin

    DUT: SecondGenerator port map(
        tick_in => TB_tick_in,
        minute_out => TB_minute_out,
        second => TB_Second
        );

    tick_process: process

    begin
        TB_tick_in <= '0';
        wait for 5000 ms;
        TB_tick_in <= '1';
        wait for 5000 ms;
    end process;

end behavioral;

```

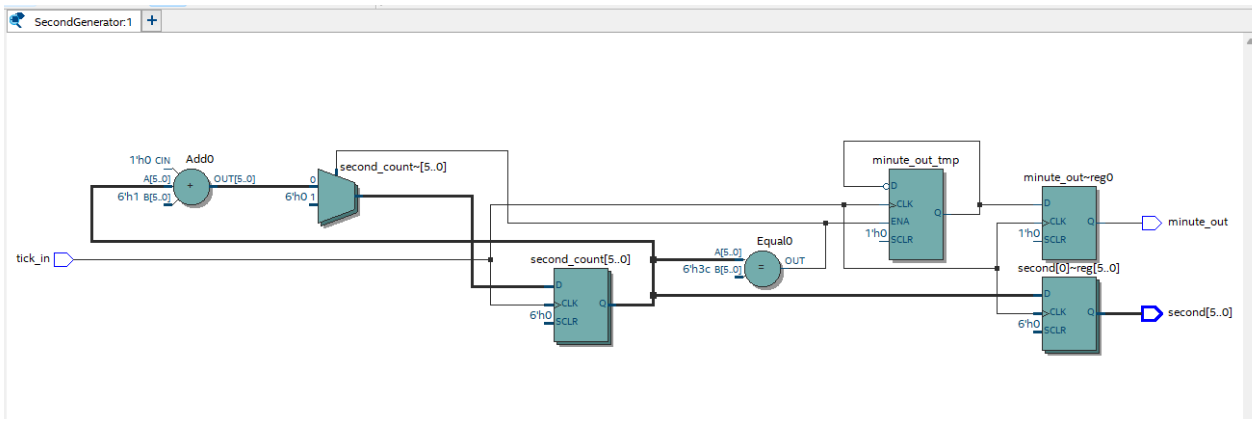
Simulation



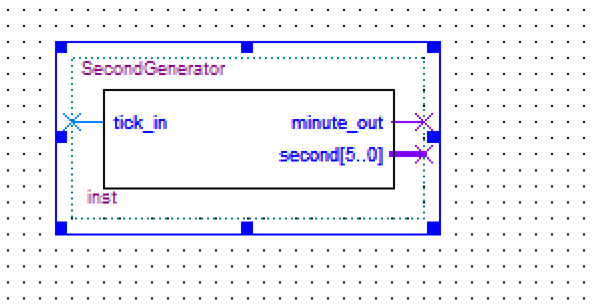
Name	Value	Kind	Mode
TB_tick_in	1	Signal	Internal
TB_second	000010	Signal	Internal
TB_minute_out	1	Signal	Internal

	Msgs
/tb_secondgenerat...	1
/tb_secondgenerat...	000010
/tb_secondgenerat...	1

RTL Diagram

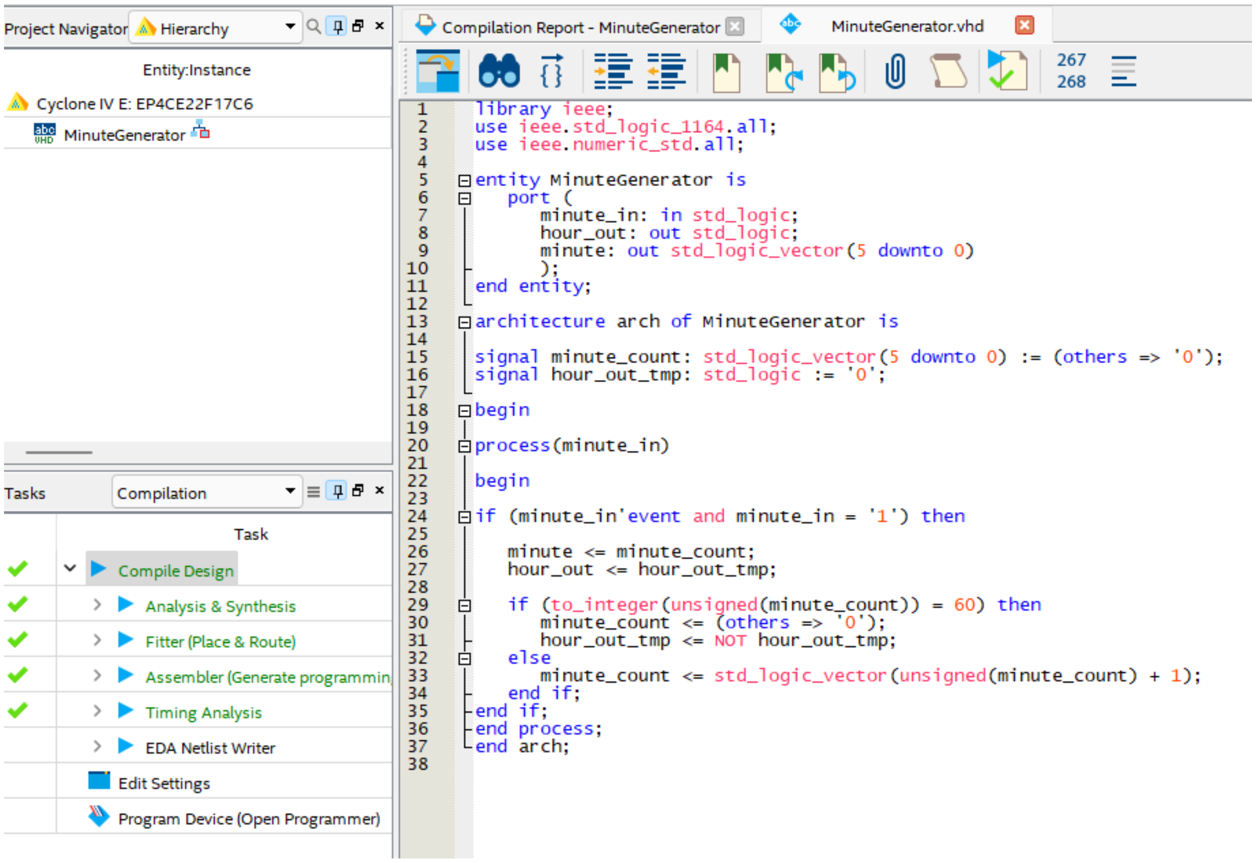


Block Design



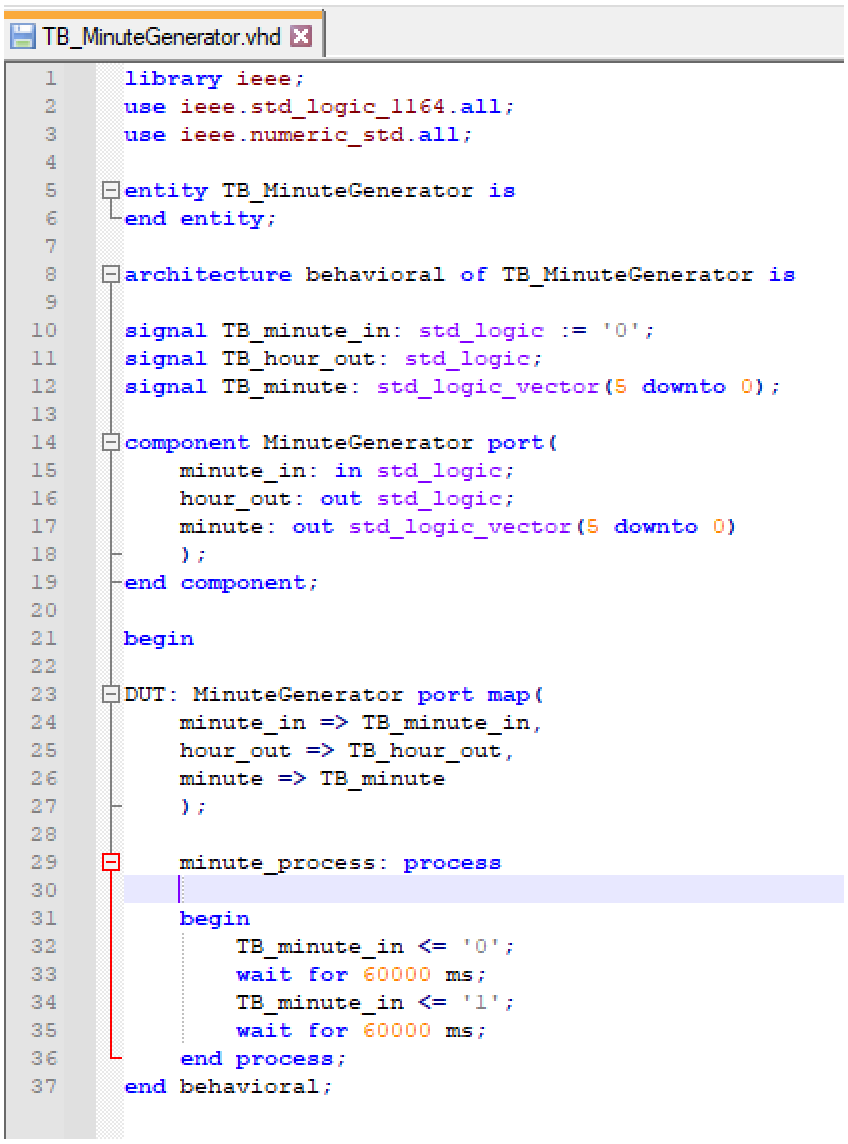
Minute Generator

Code



```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity MinuteGenerator is
    port (
        minute_in: in std_logic;
        hour_out: out std_logic;
        minute: out std_logic_vector(5 downto 0)
        );
end entity;

architecture arch of MinuteGenerator is

signal minute_count: std_logic_vector(5 downto 0) := (others => '0');
signal hour_out_tmp: std_logic := '0';

begin

process(minute_in)

begin

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

    minute <= minute_count;
    hour_out <= hour_out_tmp;

    if (to_integer(unsigned(minute_count)) = 60) then
        minute_count <= (others => '0');
        hour_out_tmp <= NOT hour_out_tmp;
    else
        minute_count <= std_logic_vector(unsigned(minute_count) + 1);
    end if;
end if;
end process;
end arch;
```

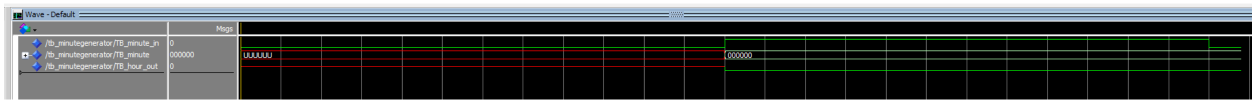
Testbench



TB_MinuteGenerator.vhd

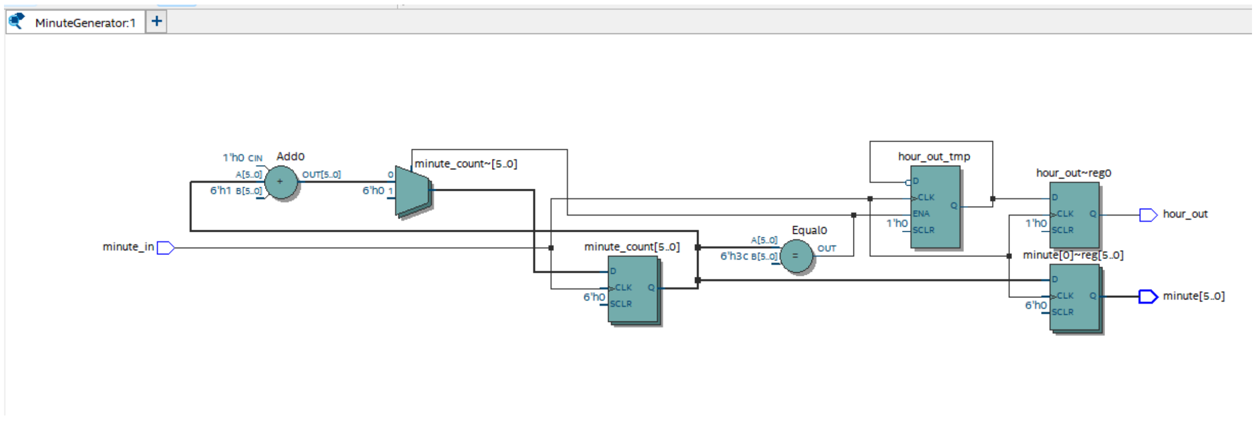
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity TB_MinuteGenerator is
end entity;

architecture behavioral of TB_MinuteGenerator is

signal TB_minute_in: std_logic := '0';
signal TB_hour_out: std_logic;
signal TB_minute: std_logic_vector(5 downto 0);

component MinuteGenerator port(
    minute_in: in std_logic;
    hour_out: out std_logic;
    minute: out std_logic_vector(5 downto 0)
    );
end component;

begin

DUT: MinuteGenerator port map(
    minute_in => TB_minute_in,
    hour_out => TB_hour_out,
    minute => TB_minute
    );

    minute_process: process

    begin
        TB_minute_in <= '0';
        wait for 60000 ms;
        TB_minute_in <= '1';
        wait for 60000 ms;
    end process;
end behavioral;
```

Simulation

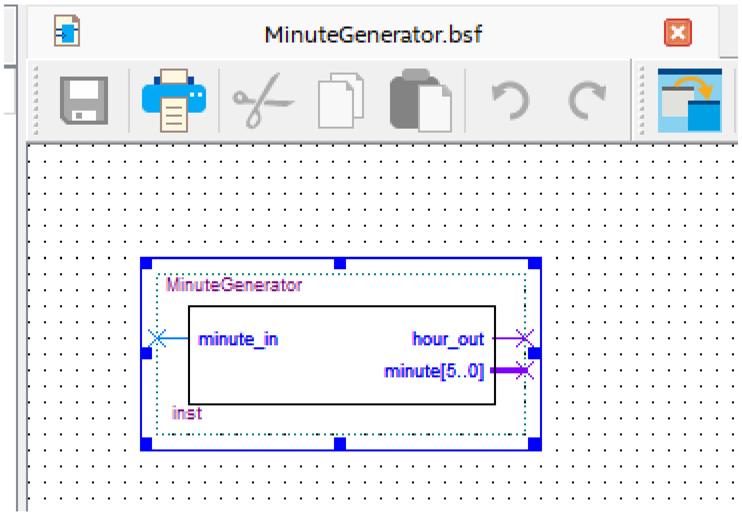


Wave - Default
Msgs
/tb_minutegenerator/TB_minute_in 0
/tb_minutegenerator/TB_minute 000000
/tb_minutegenerator/TB_hour_out 0
UUUUUU
000000

RTL Diagram

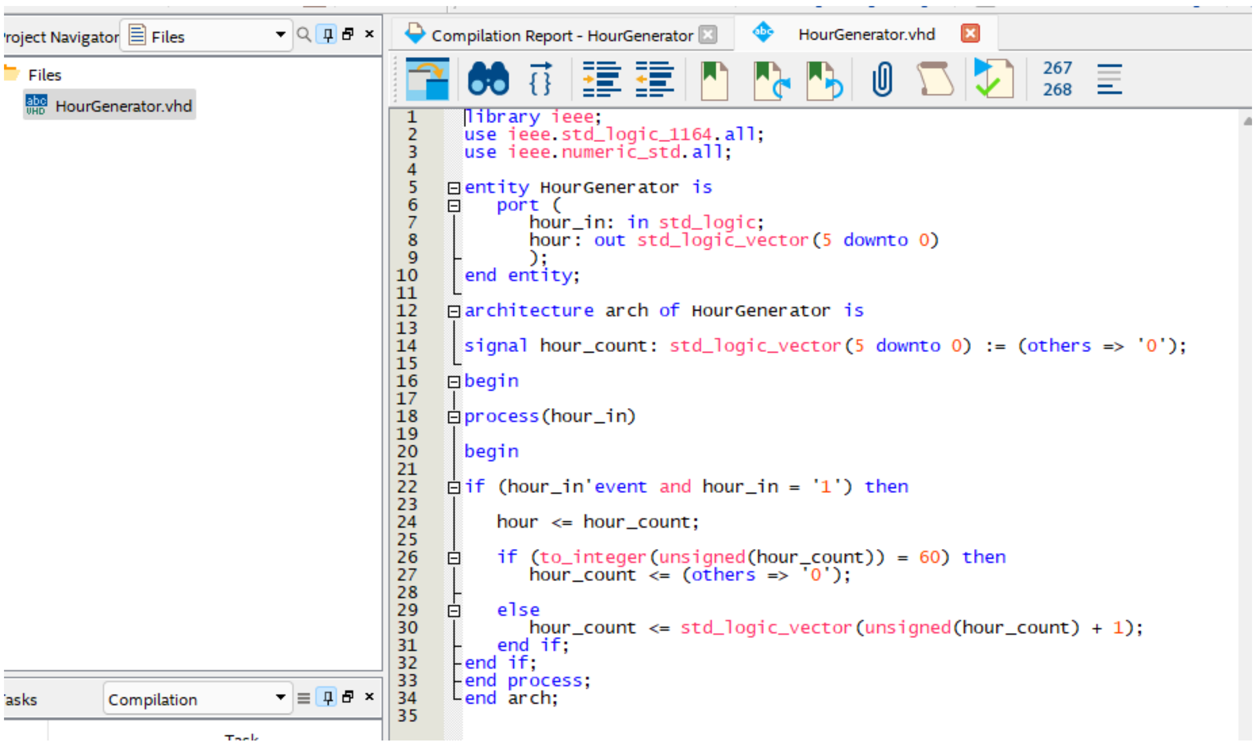


Block Design



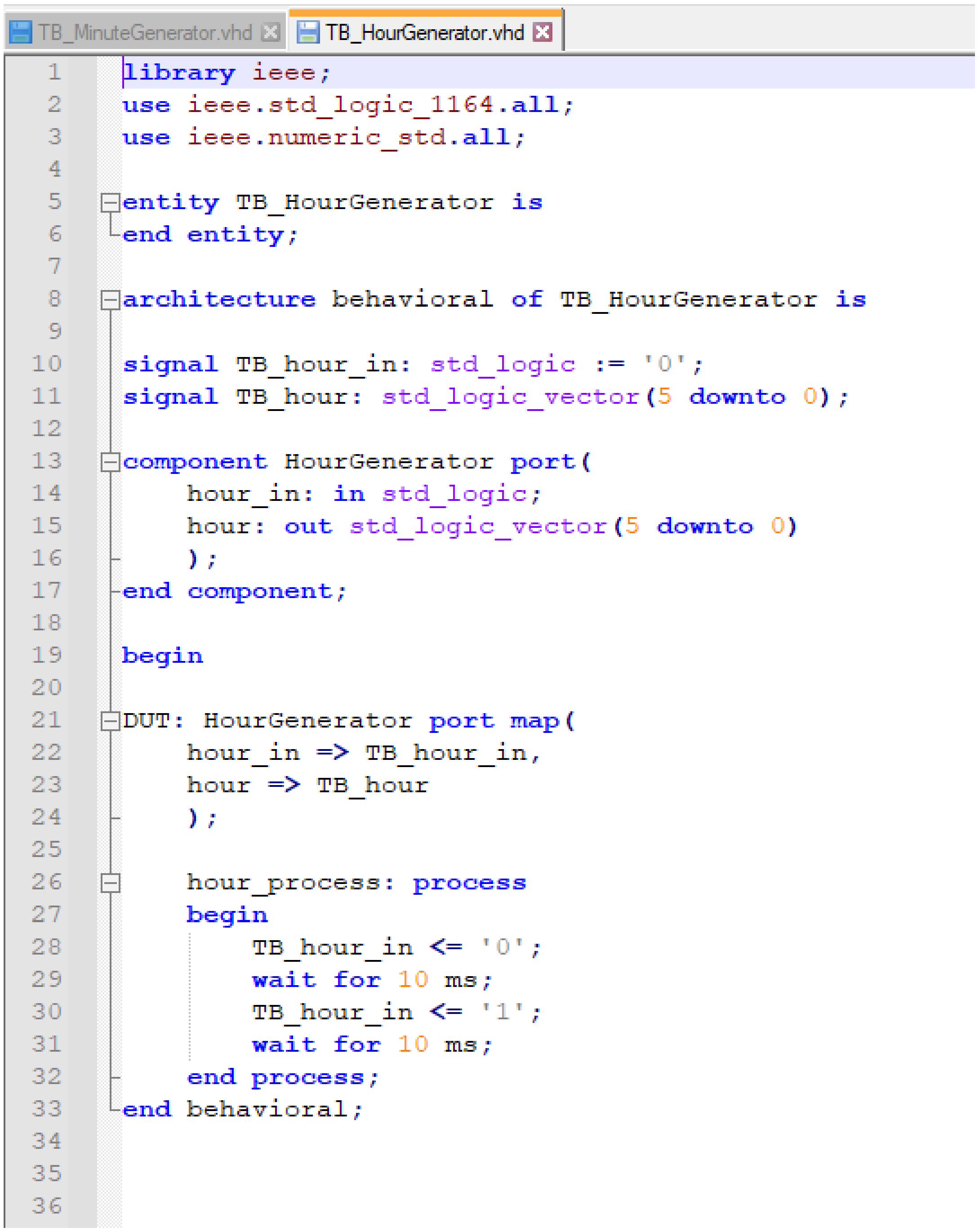
Hour Generator

Code



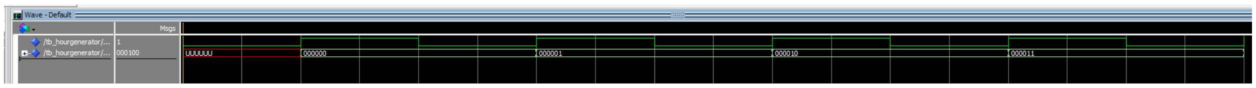
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity HourGenerator is
    port (
        hour_in: in std_logic;
        hour: out std_logic_vector(5 downto 0)
        );
end entity;

architecture arch of HourGenerator is

signal hour_count: std_logic_vector(5 downto 0) := (others => '0');

begin

process(hour_in)

begin

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

    hour <= hour_count;

    if (to_integer(unsigned(hour_count)) = 60) then
        hour_count <= (others => '0');

    else
        hour_count <= std_logic_vector(unsigned(hour_count) + 1);
    end if;
end if;
end process;
end arch;
```

Testbench

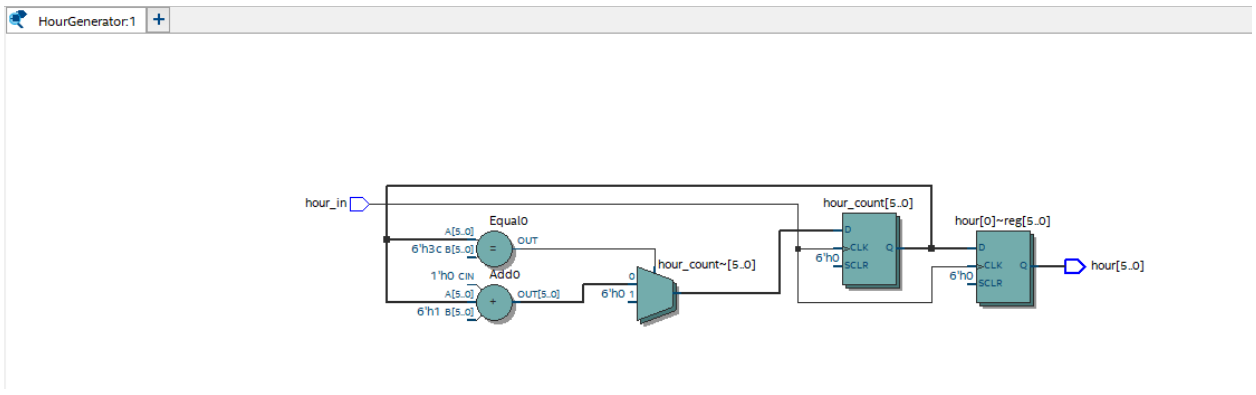

TB_MinuteGenerator.vhd TB_HourGenerator.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity TB_HourGenerator is
end entity;

architecture behavioral of TB_HourGenerator is

signal TB_hour_in: std_logic := '0';
signal TB_hour: std_logic_vector(5 downto 0);

component HourGenerator port(
    hour_in: in std_logic;
    hour: out std_logic_vector(5 downto 0)
    );
end component;

begin

DUT: HourGenerator port map(
    hour_in => TB_hour_in,
    hour => TB_hour
    );

    hour_process: process
    begin
        TB_hour_in <= '0';
        wait for 10 ms;
        TB_hour_in <= '1';
        wait for 10 ms;
    end process;
end behavioral;
```

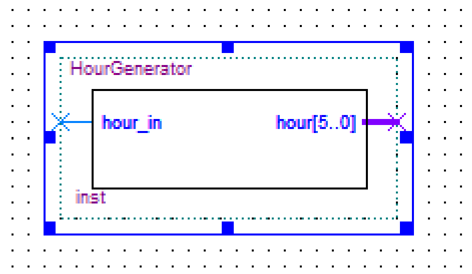
Simulation



RTL Diagram

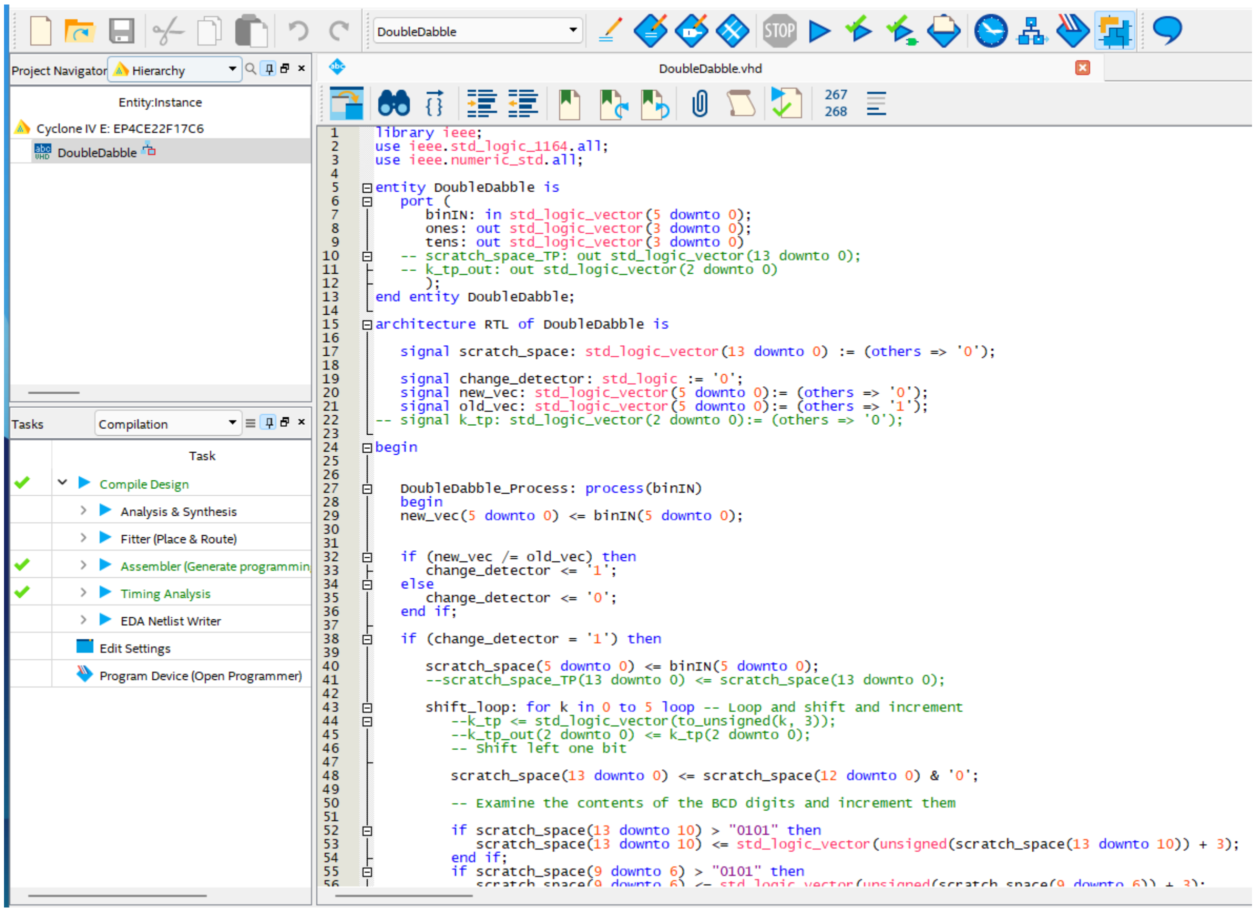


Block Design

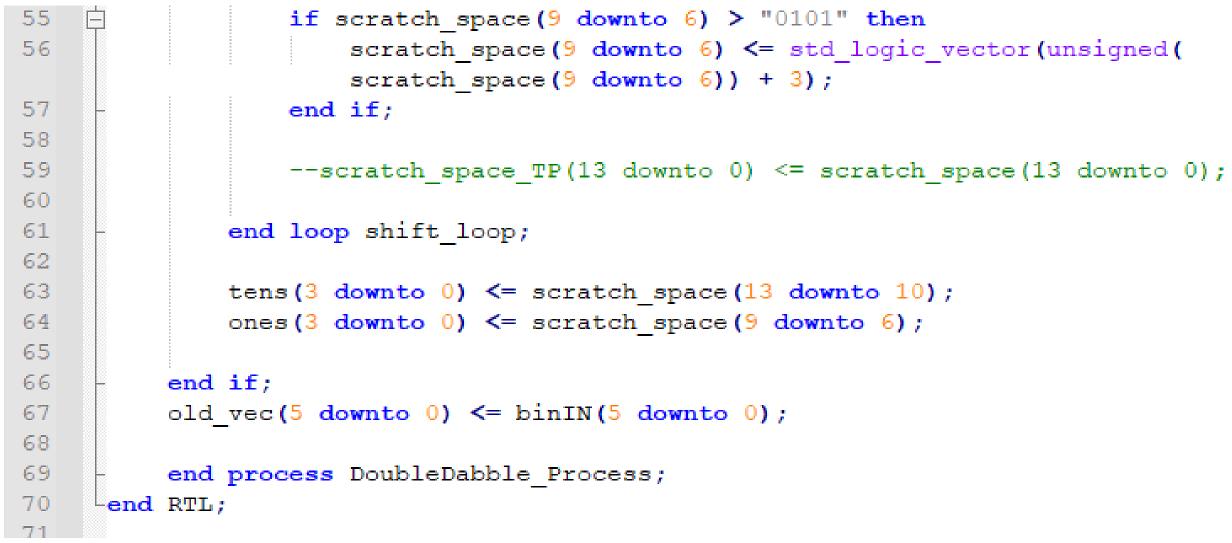


Double Dabble

Code

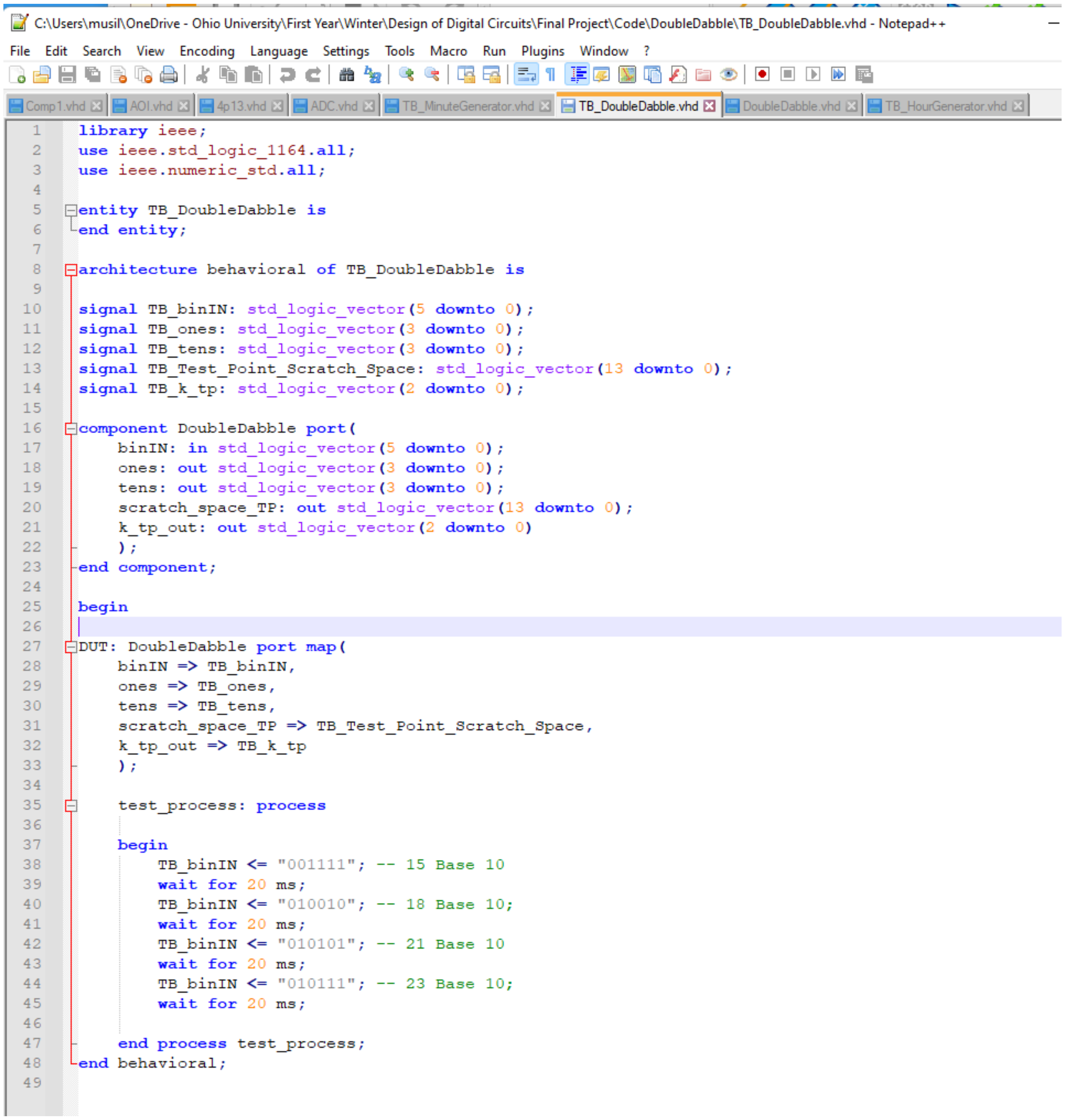


```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity DoubleDabble is
    port (
        binIN: in std_logic_vector(5 downto 0);
        ones: out std_logic_vector(3 downto 0);
        tens: out std_logic_vector(3 downto 0)
    -- scratch_space_TP: out std_logic_vector(13 downto 0);
    -- k_tp_out: out std_logic_vector(2 downto 0)
        );
end entity DoubleDabble;

architecture RTL of DoubleDabble is

    signal scratch_space: std_logic_vector(13 downto 0) := (others => '0');

    signal change_detector: std_logic := '0';
    signal new_vec: std_logic_vector(5 downto 0):= (others => '0');
    signal old_vec: std_logic_vector(5 downto 0):= (others => '1');
-- signal k_tp: std_logic_vector(2 downto 0):= (others => '0');

begin


    DoubleDabble_Process: process(binIN)
    begin
    new_vec(5 downto 0) <= binIN(5 downto 0);


    if (new_vec /= old_vec) then
        change_detector <= '1';
    else
        change_detector <= '0';
    end if;

    if (change_detector = '1') then

        scratch_space(5 downto 0) <= binIN(5 downto 0);
        --scratch_space_TP(13 downto 0) <= scratch_space(13 downto 0);

        shift_loop: for k in 0 to 5 loop -- Loop and shift and increment
            --k_tp <= std_logic_vector(to_unsigned(k, 3));
            --k_tp_out(2 downto 0) <= k_tp(2 downto 0);
            -- Shift left one bit

            scratch_space(13 downto 0) <= scratch_space(12 downto 0) & '0';

            -- Examine the contents of the BCD digits and increment them

            if scratch_space(13 downto 10) > "0101" then
                scratch_space(13 downto 10) <= std_logic_vector(unsigned(scratch_space(13 downto 10)) + 3);
            end if;
            if scratch_space(9 downto 6) > "0101" then
                scratch_space(9 downto 6) <= std_logic_vector(unsigned(scratch_space(9 downto 6)) + 3);
```



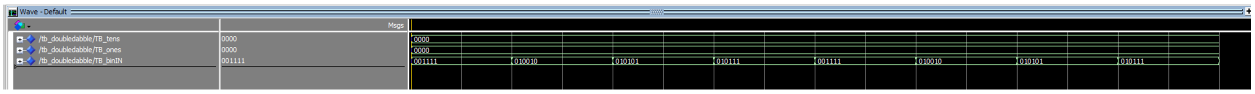
```vhdl
            if scratch_space(9 downto 6) > "0101" then
                scratch_space(9 downto 6) <= std_logic_vector(unsigned(
                scratch_space(9 downto 6)) + 3);
            end if;

            --scratch_space_TP(13 downto 0) <= scratch_space(13 downto 0);

        end loop shift_loop;

        tens(3 downto 0) <= scratch_space(13 downto 10);
        ones(3 downto 0) <= scratch_space(9 downto 6);

    end if;
    old_vec(5 downto 0) <= binIN(5 downto 0);

    end process DoubleDabble_Process;
end RTL;
```

Testbench

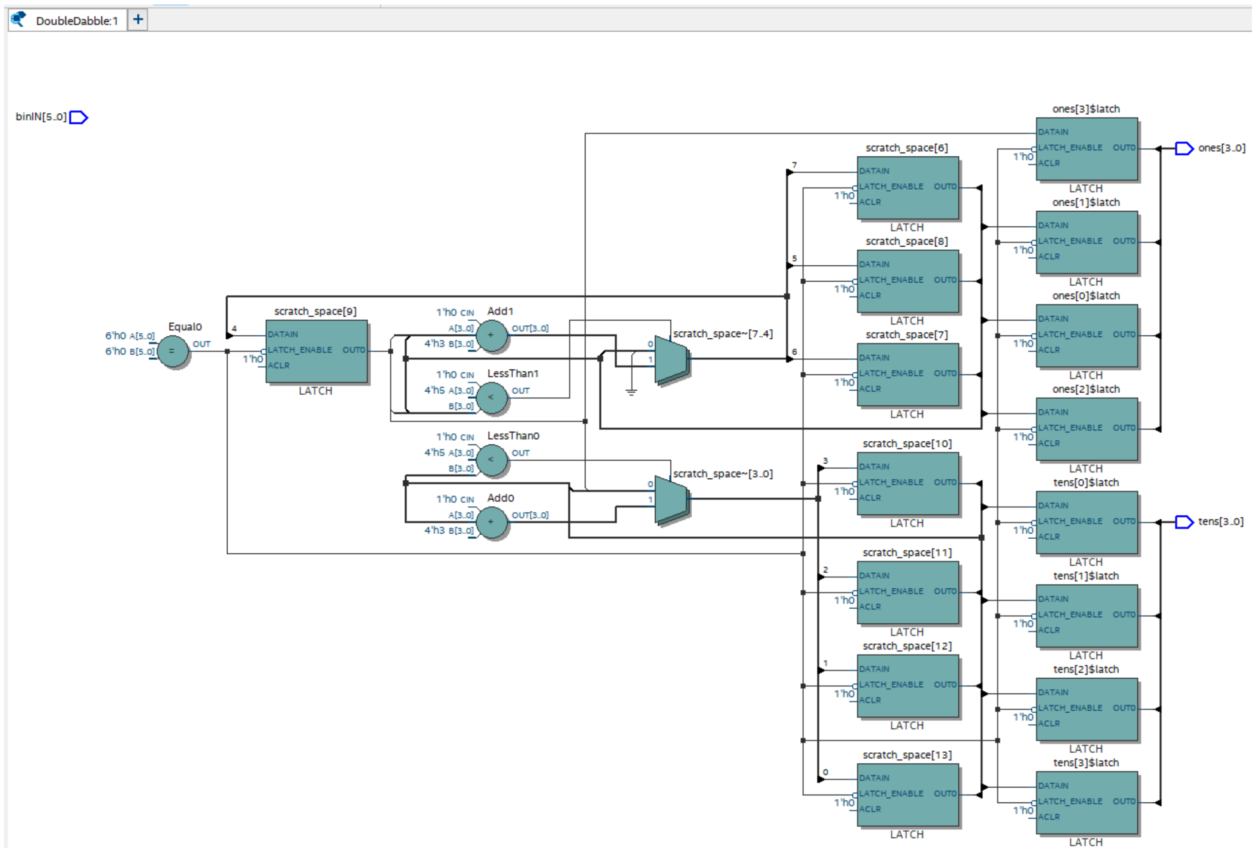

C:\Users\musil\OneDrive - Ohio University\First Year\Winter\Design of Digital Circuits\Final Project\Code\DoubleDabble\TB_DoubleDabble.vhd - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Comp1.vhd
AOI.vhd
4p13.vhd
ADC.vhd
TB_MinuteGenerator.vhd
TB_DoubleDabble.vhd
DoubleDabble.vhd
TB_HourGenerator.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity TB_DoubleDabble is
end entity;

architecture behavioral of TB_DoubleDabble is

signal TB_binIN: std_logic_vector(5 downto 0);
signal TB_ones: std_logic_vector(3 downto 0);
signal TB_tens: std_logic_vector(3 downto 0);
signal TB_Test_Point_Scratch_Space: std_logic_vector(13 downto 0);
signal TB_k_tp: std_logic_vector(2 downto 0);

component DoubleDabble port(
    binIN: in std_logic_vector(5 downto 0);
    ones: out std_logic_vector(3 downto 0);
    tens: out std_logic_vector(3 downto 0);
    scratch_space_TP: out std_logic_vector(13 downto 0);
    k_tp_out: out std_logic_vector(2 downto 0)
    );
end component;

begin

DUT: DoubleDabble port map(
    binIN => TB_binIN,
    ones => TB_ones,
    tens => TB_tens,
    scratch_space_TP => TB_Test_Point_Scratch_Space,
    k_tp_out => TB_k_tp
    );

    test_process: process

    begin
        TB_binIN <= "001111"; -- 15 Base 10
        wait for 20 ms;
        TB_binIN <= "010010"; -- 18 Base 10;
        wait for 20 ms;
        TB_binIN <= "010101"; -- 21 Base 10
        wait for 20 ms;
        TB_binIN <= "010111"; -- 23 Base 10;
        wait for 20 ms;

    end process test_process;
end behavioral;
```

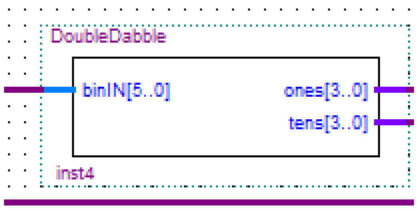
Simulation


Wave - Default
Msgs
/tb_doubledabble/TB_tens 0000
/tb_doubledabble/TB_ones 0000
/tb_doubledabble/TB_binIN 001111
0000
0000
001111
010010
010101
010111
001111
010010
010101
010111

RTL Diagram

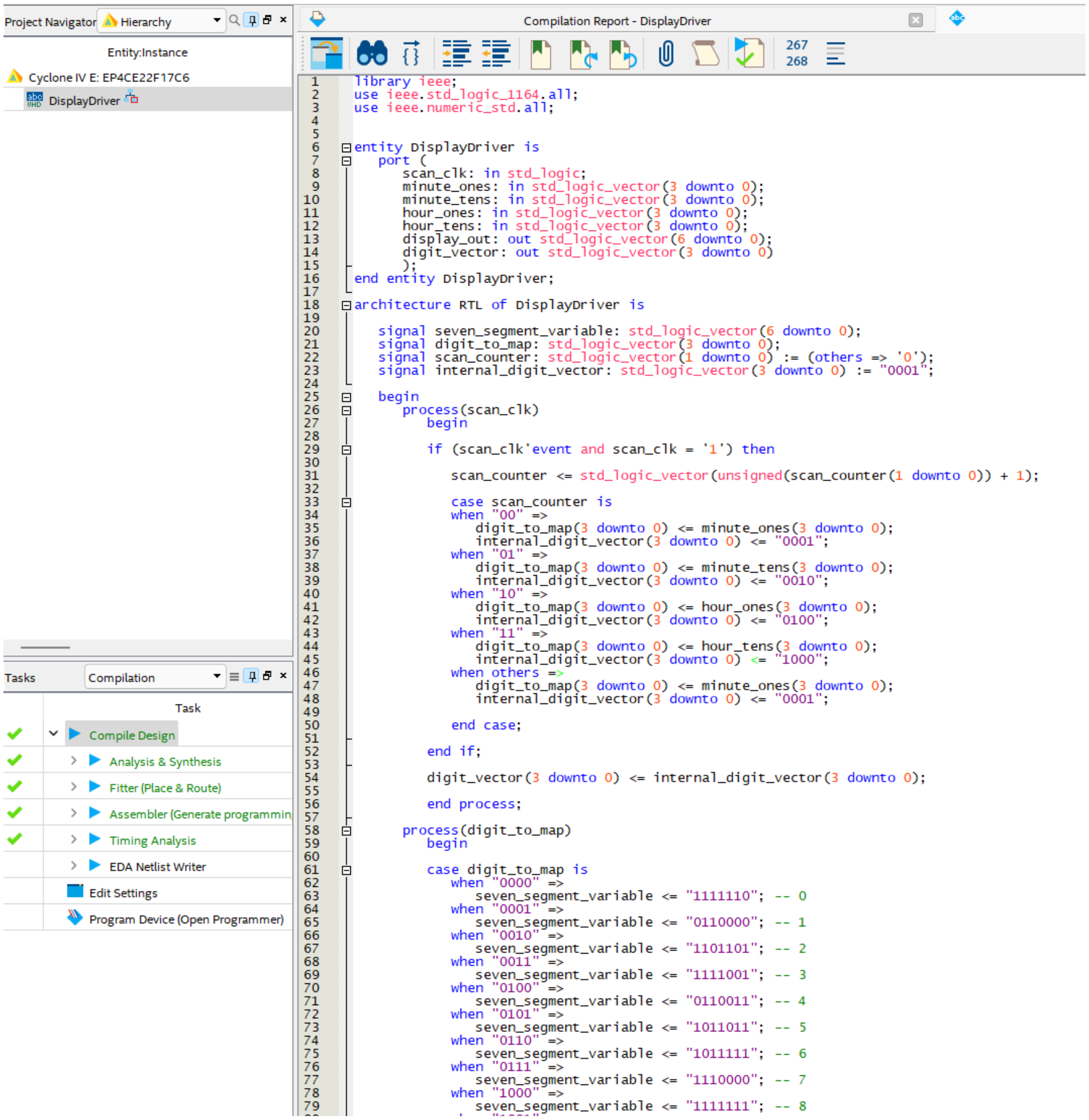


Block Design

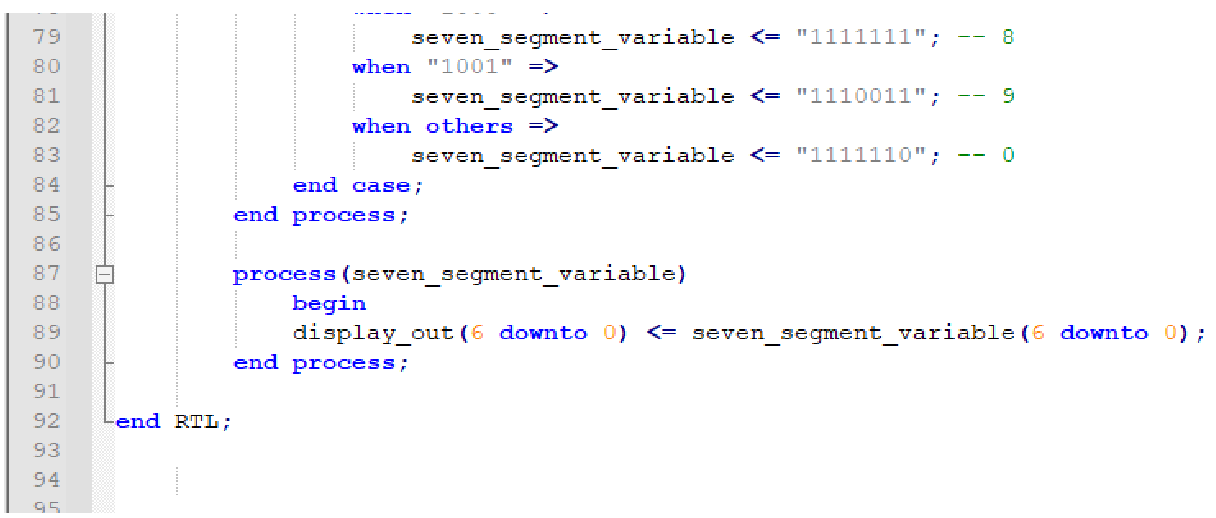


DisplayDriver

Code

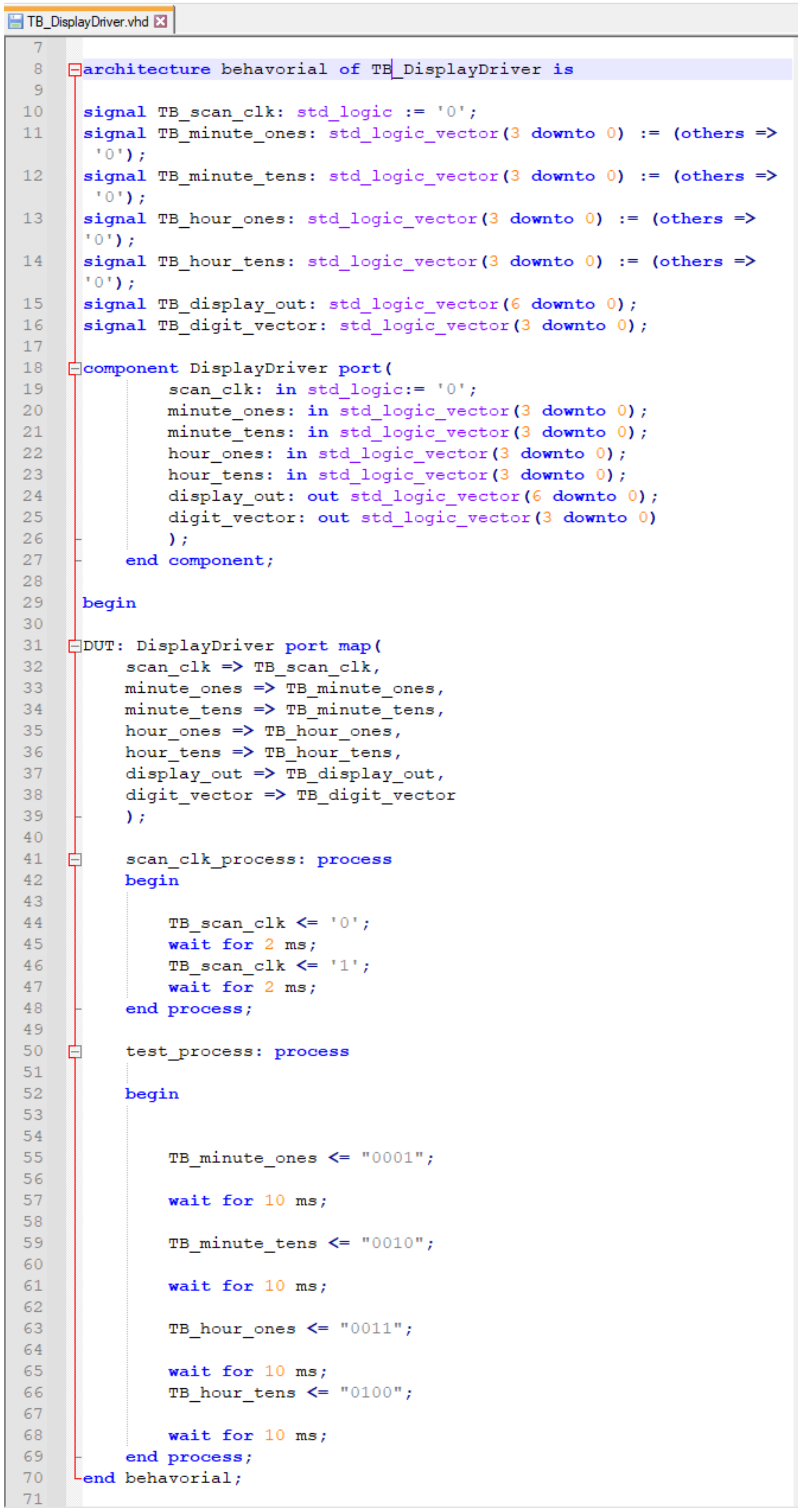


```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity DisplayDriver is
   port (
      scan_clk: in std_logic;
      minute_ones: in std_logic_vector(3 downto 0);
      minute_tens: in std_logic_vector(3 downto 0);
      hour_ones: in std_logic_vector(3 downto 0);
      hour_tens: in std_logic_vector(3 downto 0);
      display_out: out std_logic_vector(6 downto 0);
      digit_vector: out std_logic_vector(3 downto 0)
      );
end entity DisplayDriver;

architecture RTL of DisplayDriver is

   signal seven_segment_variable: std_logic_vector(6 downto 0);
   signal digit_to_map: std_logic_vector(3 downto 0);
   signal scan_counter: std_logic_vector(1 downto 0) := (others => '0');
   signal internal_digit_vector: std_logic_vector(3 downto 0) := "0001";

   begin
      process(scan_clk)
         begin

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

            scan_counter <= std_logic_vector(unsigned(scan_counter(1 downto 0)) + 1);

            case scan_counter is
            when "00" =>
               digit_to_map(3 downto 0) <= minute_ones(3 downto 0);
               internal_digit_vector(3 downto 0) <= "0001";
            when "01" =>
               digit_to_map(3 downto 0) <= minute_tens(3 downto 0);
               internal_digit_vector(3 downto 0) <= "0010";
            when "10" =>
               digit_to_map(3 downto 0) <= hour_ones(3 downto 0);
               internal_digit_vector(3 downto 0) <= "0100";
            when "11" =>
               digit_to_map(3 downto 0) <= hour_tens(3 downto 0);
               internal_digit_vector(3 downto 0) <= "1000";
            when others =>
               digit_to_map(3 downto 0) <= minute_ones(3 downto 0);
               internal_digit_vector(3 downto 0) <= "0001";

            end case;

         end if;

         digit_vector(3 downto 0) <= internal_digit_vector(3 downto 0);

         end process;

      process(digit_to_map)
         begin

         case digit_to_map is
            when "0000" =>
               seven_segment_variable <= "1111110"; -- 0
            when "0001" =>
               seven_segment_variable <= "0110000"; -- 1
            when "0010" =>
               seven_segment_variable <= "1101101"; -- 2
            when "0011" =>
               seven_segment_variable <= "1111001"; -- 3
            when "0100" =>
               seven_segment_variable <= "0110011"; -- 4
            when "0101" =>
               seven_segment_variable <= "1011011"; -- 5
            when "0110" =>
               seven_segment_variable <= "1011111"; -- 6
            when "0111" =>
               seven_segment_variable <= "1110000"; -- 7
            when "1000" =>
               seven_segment_variable <= "1111111"; -- 8
```



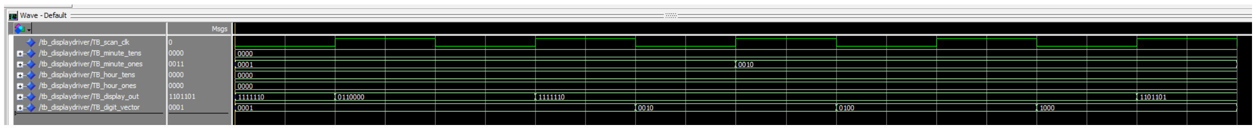
```
                    seven_segment_variable <= "1111111"; -- 8
                when "1001" =>
                    seven_segment_variable <= "1110011"; -- 9
                when others =>
                    seven_segment_variable <= "1111110"; -- 0
            end case;
        end process;

        process(seven_segment_variable)
            begin
            display_out(6 downto 0) <= seven_segment_variable(6 downto 0);
        end process;

end RTL;
```

TestBench

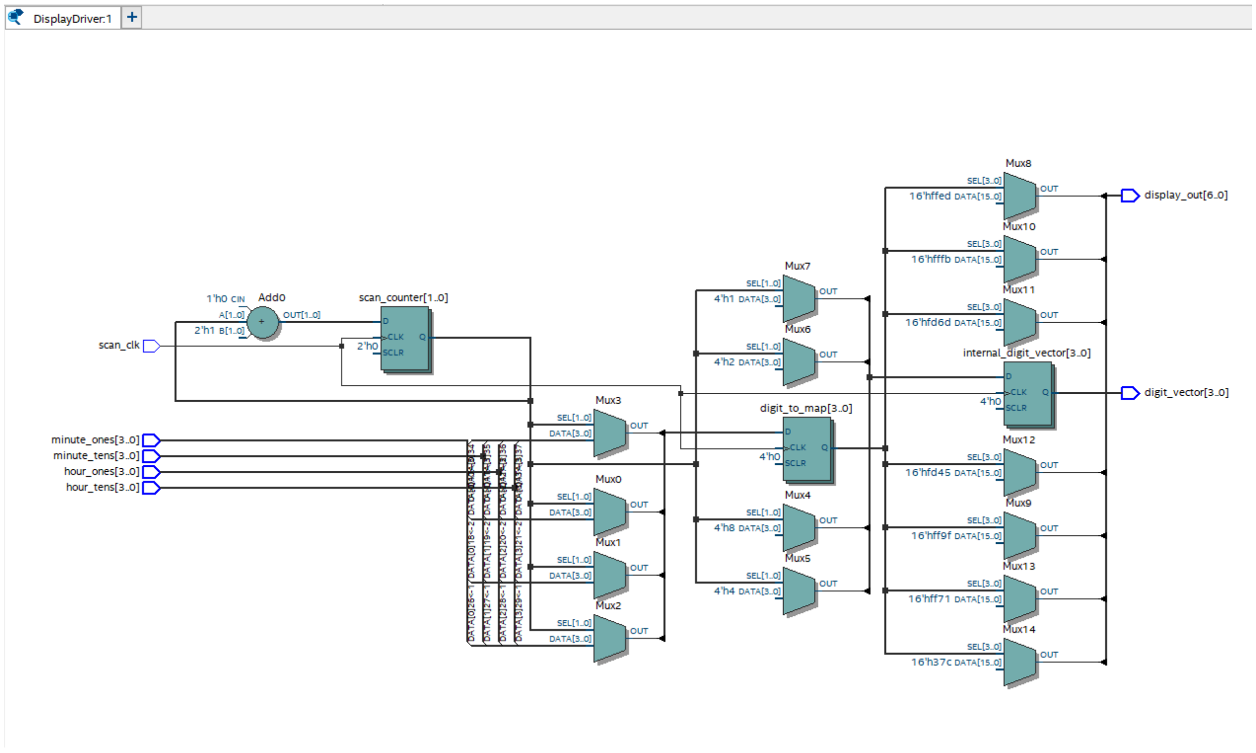

TB_DisplayDriver.vhd

```
architecture behavorial of TB_DisplayDriver is

signal TB_scan_clk: std_logic := '0';
signal TB_minute_ones: std_logic_vector(3 downto 0) := (others =>
 '0');
signal TB_minute_tens: std_logic_vector(3 downto 0) := (others =>
 '0');
signal TB_hour_ones: std_logic_vector(3 downto 0) := (others =>
'0');
signal TB_hour_tens: std_logic_vector(3 downto 0) := (others =>
'0');
signal TB_display_out: std_logic_vector(6 downto 0);
signal TB_digit_vector: std_logic_vector(3 downto 0);

component DisplayDriver port(
        scan_clk: in std_logic:= '0';
        minute_ones: in std_logic_vector(3 downto 0);
        minute_tens: in std_logic_vector(3 downto 0);
        hour_ones: in std_logic_vector(3 downto 0);
        hour_tens: in std_logic_vector(3 downto 0);
        display_out: out std_logic_vector(6 downto 0);
        digit_vector: out std_logic_vector(3 downto 0)
        );
    end component;

begin

DUT: DisplayDriver port map(
    scan_clk => TB_scan_clk,
    minute_ones => TB_minute_ones,
    minute_tens => TB_minute_tens,
    hour_ones => TB_hour_ones,
    hour_tens => TB_hour_tens,
    display_out => TB_display_out,
    digit_vector => TB_digit_vector
    );

    scan_clk_process: process
    begin

        TB_scan_clk <= '0';
        wait for 2 ms;
        TB_scan_clk <= '1';
        wait for 2 ms;
    end process;

    test_process: process

    begin


        TB_minute_ones <= "0001";

        wait for 10 ms;

        TB_minute_tens <= "0010";

        wait for 10 ms;

        TB_hour_ones <= "0011";

        wait for 10 ms;
        TB_hour_tens <= "0100";

        wait for 10 ms;
    end process;
end behavorial;
```

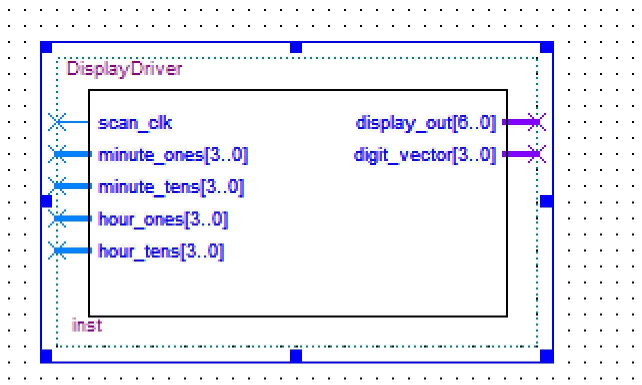
Simulation


Wave - Default
Msgs
/tb_displaydriver/TB_scan_clk 0
/tb_displaydriver/TB_minute_tens 0000
/tb_displaydriver/TB_minute_ones 0011
/tb_displaydriver/TB_hour_tens 0000
/tb_displaydriver/TB_hour_ones 0000
/tb_displaydriver/TB_display_out 1101101
/tb_displaydriver/TB_digit_vector 0001
0000
0001
0010
0000
0000
1111110
0110000
1111110
1101101
0001
0010
0010
0100
1000

RTL Diagram



Block Design


DisplayDriver
scan_clk
minute_ones[3..0]
minute_tens[3..0]
hour_ones[3..0]
hour_tens[3..0]
display_out[6..0]
digit_vector[3..0]
inst