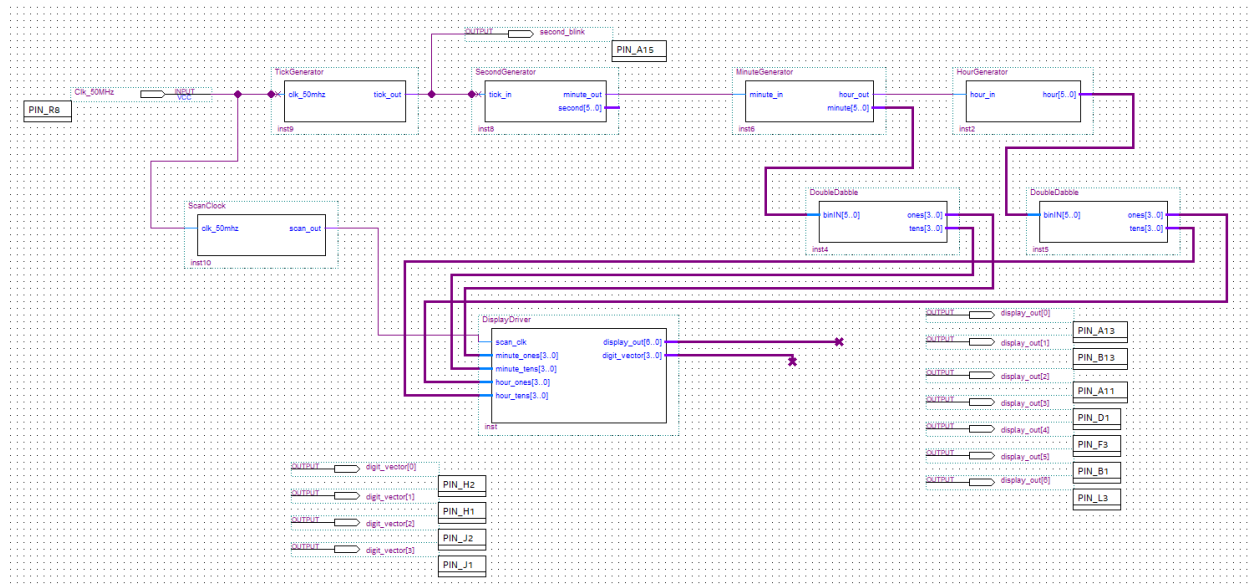


Top Level (Complete_Clock.bdf)



Pin Planner

Report not available

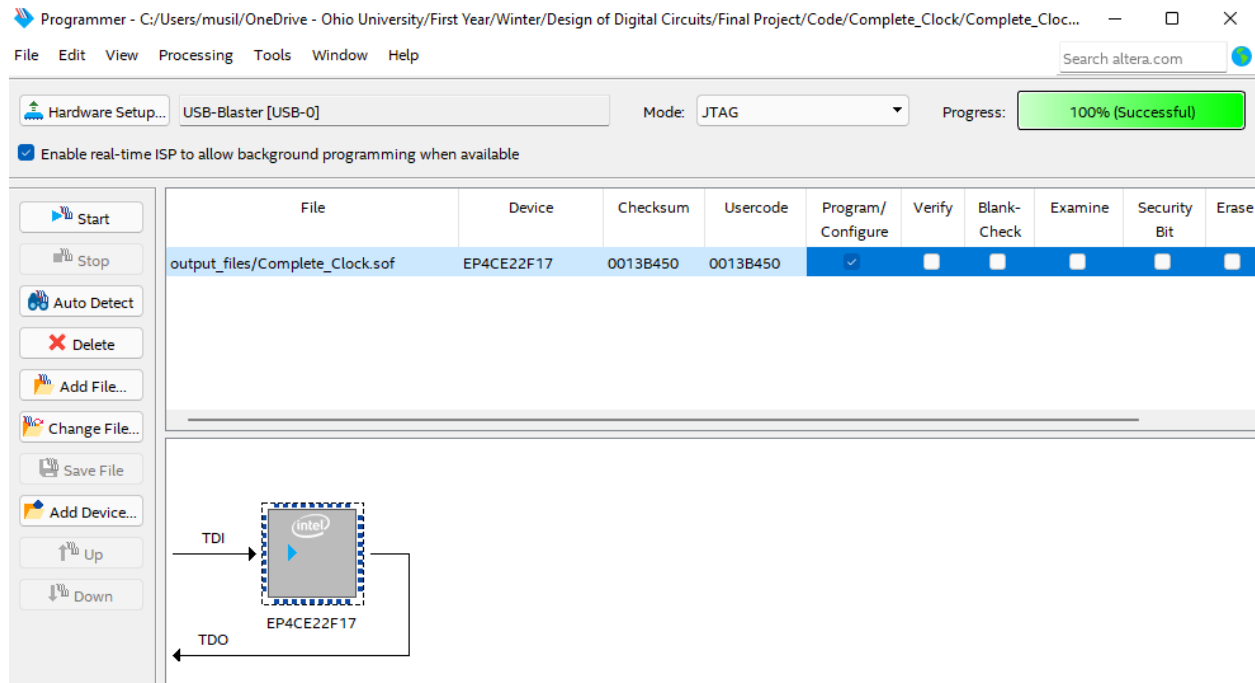
Top View - Wire Bond

Cyclone IV E - EP4CE22F17C6

The Pin Planner interface displays the top view of the Cyclone IV E chip (EP4CE22F17C6) with a wire bond configuration. The chip is a 16x16 grid of pins. The pin assignments are as follows:

Node Name	Direction	Location	I/O Bank	VREF Group	Filter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
digit_vector[2]	Output	PIN_J2	2	B2_NO	PIN_J2	2.5 V		8mA (default)	2 (default)		
digit_vector[1]	Output	PIN_A5	8	B8_NO	PIN_A5	2.5 V		8mA (default)	2 (default)		
digit_vector[0]	Output	PIN_A6	8	B8_NO	PIN_A6	2.5 V		8mA (default)	2 (default)		
display_out[5]	Output	PIN_L3	2	B2_NO	PIN_L3	2.5 V		8mA (default)	2 (default)		
display_out[4]	Output	PIN_B1	1	B1_NO	PIN_B1	2.5 V		8mA (default)	2 (default)		
display_out[3]	Output	PIN_F3	1	B1_NO	PIN_F3	2.5 V		8mA (default)	2 (default)		
display_out[2]	Output	PIN_D1	1	B1_NO	PIN_D1	2.5 V		8mA (default)	2 (default)		
display_out[1]	Output	PIN_A11	7	B7_NO	PIN_A11	2.5 V		8mA (default)	2 (default)		
display_out[0]	Output	PIN_B13	7	B7_NO	PIN_B13	2.5 V		8mA (default)	2 (default)		
second_blink	Output	PIN_A15	7	B7_NO	PIN_A15	2.5 V		8mA (default)	2 (default)		

Programmer



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

The screenshot displays the Quartus Lite IDE interface. On the left, the Project Navigator shows the hierarchy for 'Entity:Instance' with 'Cyclone IV E: EP4CE22F17C6' and 'TickGenerator'. Below this, the Tasks window is open, showing a list of compilation tasks with green checkmarks indicating successful completion:

- Compile Design
- Analysis & Synthesis
- Fitter (Place & Route)
- Assembler (Generate program)
- Timing Analysis
- EDA Netlist Writer

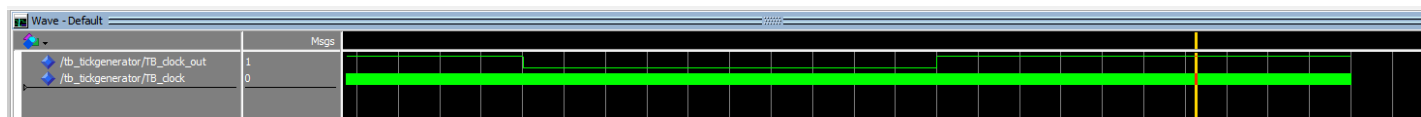
The main editor window displays the VHDL code for 'TickGenerator.vhd'.

```
1
2  library ieee;
3  use ieee.std_logic_1164.all;
4  use ieee.numeric_std.all;
5
6  entity TickGenerator is
7  port(
8      clk_50mhz: in std_logic;
9      tick_out: out std_logic
10 );
11 end entity;
12
13 architecture arch of TickGenerator is
14     signal count: integer := 1;
15     signal tmp: std_logic := '0';
16
17 begin
18     process(clk_50mhz)
19     begin
20         if (clk_50mhz'event and clk_50mhz = '1') then
21             count <= count + 1;
22             if (count = 25000000) then
23                 tmp <= NOT tmp;
24                 count <= 1;
25             end if;
26             tick_out <= tmp;
27         end if;
28     end process;
29 end arch;
```

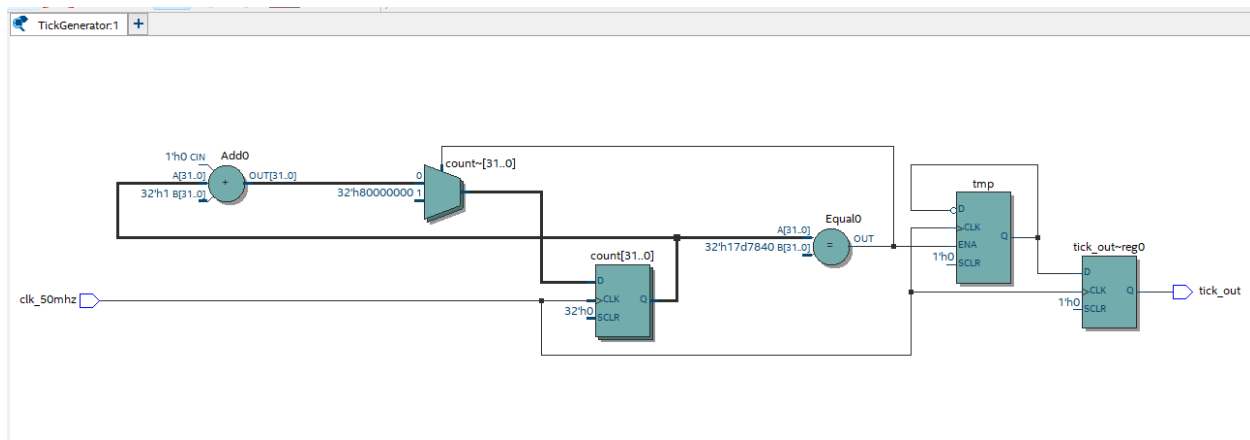
Testbench

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity TB_TickGenerator is
5  end entity;
6
7  architecture behavioral of TB_TickGenerator is
8
9      signal TB_clock: std_logic := '0';
10     signal TB_clock_out: std_logic;
11
12     component TickGenerator port (
13         clk_50mhz: in std_logic;
14         tick_out: out std_logic
15     );
16 end component;
17
18 begin
19
20     DUT: TickGenerator port map(
21         clk_50mhz => TB_clock,
22         tick_out => TB_clock_out);
23
24     clock_process: process
25
26     begin
27         TB_clock <= '0';
28         wait for 10 ns;
29         TB_clock <= '1';
30         wait for 10 ns;
31     end process;
32
33 end behavioral;
34
```

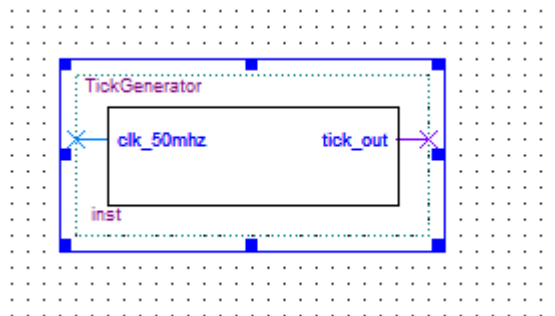
Testbench output



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/Scan

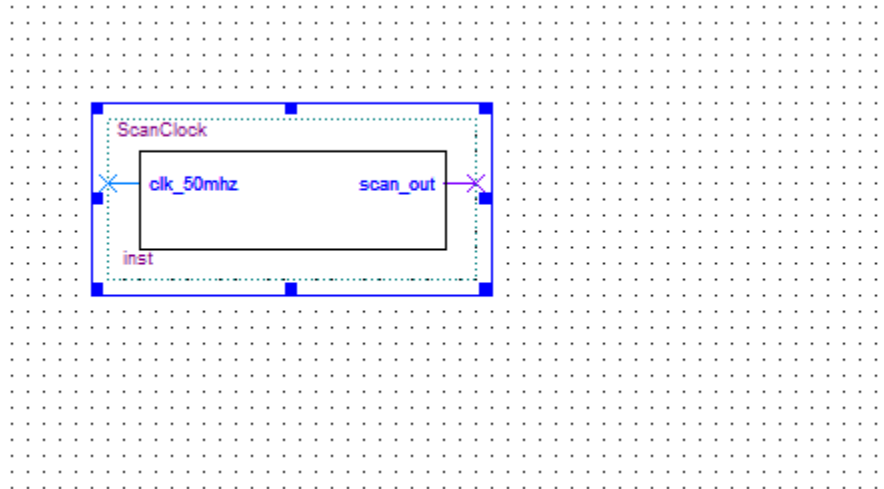
File Edit View Project Assignments Processing Tools Window Help

The screenshot displays the Quartus Prime Lite Edition interface. The top toolbar includes icons for file operations, a search bar with 'ScanClock' entered, and a 'STOP' button. The Project Navigator on the left shows the hierarchy: Entity:Instance, Cyclone IV E: EP4CE22F17C6, and ScanClock.vhd. The main editor window shows the VHDL code for ScanClock.vhd, which defines an entity with a clock input and a scan output, and an architecture that implements a counter and a scan output toggle. The bottom Tasks window shows a list of compilation tasks, all of which are marked as completed with green checkmarks.

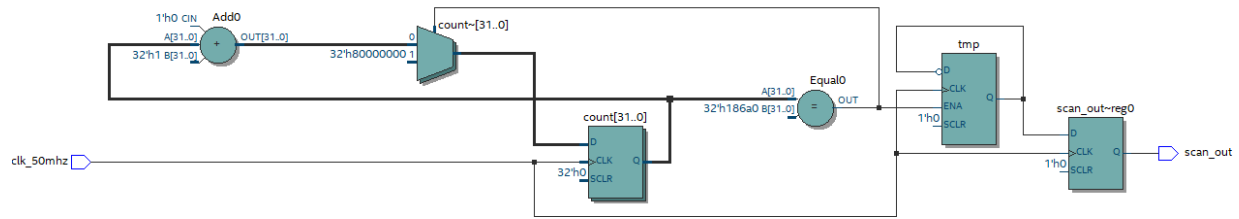
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity scanClock is
6 port(
7     clk_50mhz: in std_logic;
8     scan_out: out std_logic
9 );
10 end entity;
11
12 architecture arch of scanClock is
13
14     signal count : integer := 1;
15     signal tmp: std_logic := '0';
16
17 begin
18
19     process(clk_50mhz)
20     begin
21
22         if (clk_50mhz'event and clk_50mhz = '1') then
23             count <= count + 1;
24             if (count = 100000) then
25                 tmp <= NOT tmp;
26                 count <= 1;
27             end if;
28             scan_out <= tmp;
29         end if;
30     end process;
31 end arch;
```

Task
✓ Compile Design
✓ > Analysis & Synthesis
✓ > Fitter (Place & Route)
✓ > Assembler (Generate programming file)
✓ > Timing Analysis
> EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)

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

267
268

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity SecondGenerator is
6 port (
7     tick_in: in std_logic;
8     minute_out: out std_logic;
9     second: out std_logic_vector(5 downto 0)
10 );
11 end entity;
12
13 architecture arch of SecondGenerator is
14     signal second_count: std_logic_vector(5 downto 0) := (others => '0');
15     signal minute_out_tmp: std_logic := '0';
16
17 begin
18     process(tick_in)
19     begin
20         if (tick_in'event and tick_in = '1') then
21             second_count <= second_count;
22             minute_out <= minute_out_tmp;
23
24             if(to_integer(unsigned(second_count)) = 60) then
25                 second_count <= (others => '0');
26                 minute_out_tmp <= NOT minute_out_tmp;
27             else
28                 second_count <= std_logic_vector(unsigned(second_count) + 1);
29             end if;
30         end if;
31     end process;
32 end arch;
```

Tasks

Compilation

	Task
✓	Compile Design
✓	Analysis & Synthesis
✓	Fitter (Place & Route)
✓	Assembler (Generate programr
✓	Timing Analysis
	EDA Netlist Writer
	Edit Settings
	Program Device (Open Programme

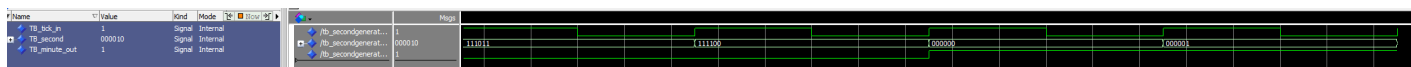
Testbench

```

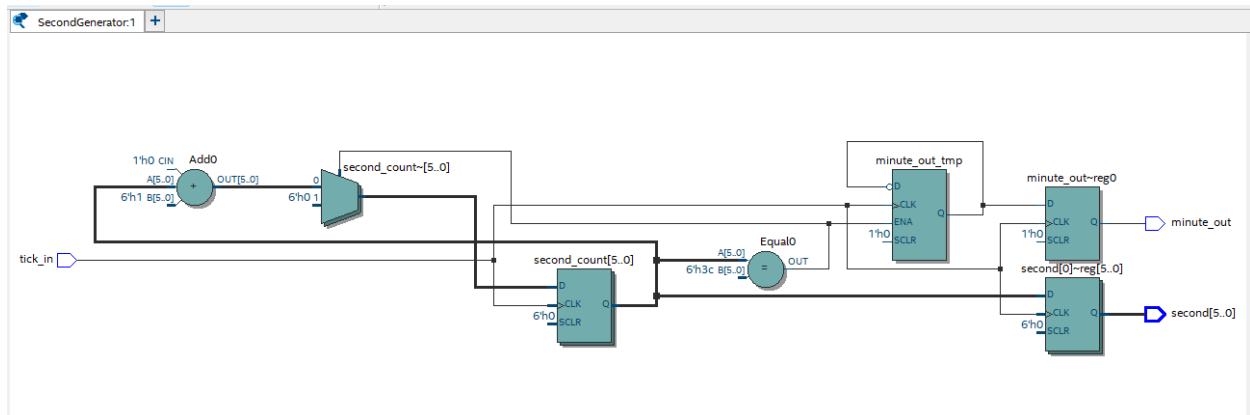
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity TB_SecondGenerator is
6  end entity;
7
8  architecture behavioral of TB_SecondGenerator is
9
10     signal TB_tick_in: std_logic := '0';
11     signal TB_minute_out: std_logic;
12     signal TB_second: std_logic_vector(5 downto 0);
13
14     component SecondGenerator port (
15         tick_in: in std_logic;
16         minute_out: out std_logic;
17         second: out std_logic_vector(5 downto 0)
18     );
19 end component;
20
21 begin
22
23     DUT: SecondGenerator port map(
24         tick_in => TB_tick_in,
25         minute_out => TB_minute_out,
26         second => TB_Second
27     );
28
29     tick_process: process
30
31     begin
32         TB_tick_in <= '0';
33         wait for 5000 ms;
34         TB_tick_in <= '1';
35         wait for 5000 ms;
36     end process;
37
38 end behavioral;
39

```

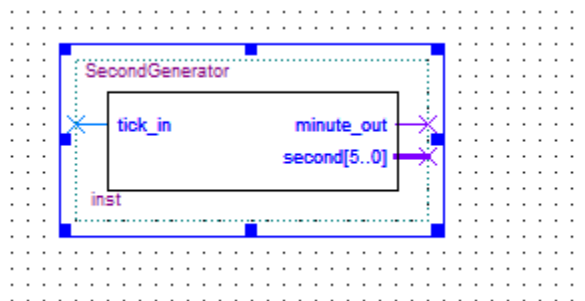
Simulation



RTL Diagram



Block Design



Minute Generator

Code

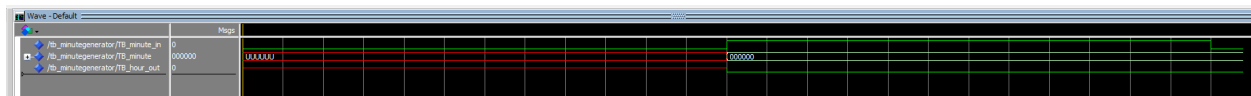
The screenshot displays the Quartus II IDE interface. The top toolbar includes icons for Project Navigator, Hierarchy, and various design tools. The top status bar shows the file name 'MinuteGenerator.vhd' and line counts 267 and 268. The left pane shows the Project Navigator with 'Entity: Instance' and 'Cyclone IV E: EP4CE22F17C6'. The main editor window shows the VHDL code for the 'MinuteGenerator' entity. The code defines the entity with inputs 'minute_in' and outputs 'hour_out' and 'minute'. It includes a process 'process(minute_in)' that updates the 'minute_count' and 'hour_out_tmp' signals based on the 'minute_in' input. The bottom pane shows the 'Tasks' window with a list of tasks including 'Compile Design', 'Analysis & Synthesis', 'Fitter (Place & Route)', 'Assembler (Generate programmin', 'Timing Analysis', 'EDA Netlist Writer', 'Edit Settings', and 'Program Device (Open Programmer)'. The 'Compile Design' task is currently selected and highlighted.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity MinuteGenerator is
6  port (
7      minute_in: in std_logic;
8      hour_out: out std_logic;
9      minute: out std_logic_vector(5 downto 0)
10 );
11 end entity;
12
13 architecture arch of MinuteGenerator is
14     signal minute_count: std_logic_vector(5 downto 0) := (others => '0');
15     signal hour_out_tmp: std_logic := '0';
16
17 begin
18     process(minute_in)
19     begin
20         if (minute_in'event and minute_in = '1') then
21             minute_count <= (others => '0');
22             hour_out_tmp <= NOT hour_out_tmp;
23
24             if (to_integer(unsigned(minute_count)) = 60) then
25                 minute_count <= (others => '0');
26                 hour_out_tmp <= NOT hour_out_tmp;
27             else
28                 minute_count <= std_logic_vector(unsigned(minute_count) + 1);
29             end if;
30         end if;
31     end process;
32 end arch;
```

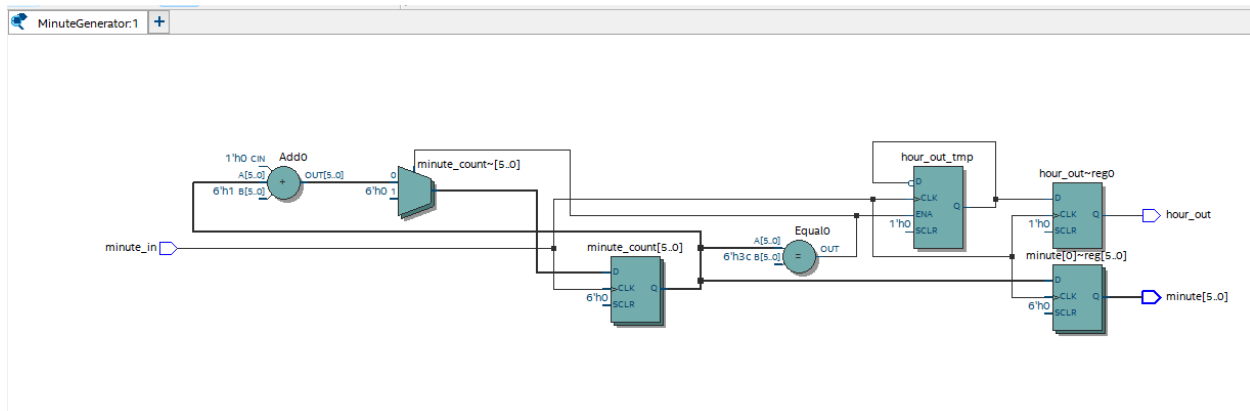
Testbench

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity TB_MinuteGenerator is
6  end entity;
7
8  architecture behavioral of TB_MinuteGenerator is
9
10     signal TB_minute_in: std_logic := '0';
11     signal TB_hour_out: std_logic;
12     signal TB_minute: std_logic_vector(5 downto 0);
13
14     component MinuteGenerator port(
15         minute_in: in std_logic;
16         hour_out: out std_logic;
17         minute: out std_logic_vector(5 downto 0)
18     );
19 end component;
20
21 begin
22
23     DUT: MinuteGenerator port map(
24         minute_in => TB_minute_in,
25         hour_out => TB_hour_out,
26         minute => TB_minute
27     );
28
29     minute_process: process
30     begin
31         TB_minute_in <= '0';
32         wait for 60000 ms;
33         TB_minute_in <= '1';
34         wait for 60000 ms;
35     end process;
36 end behavioral;
```

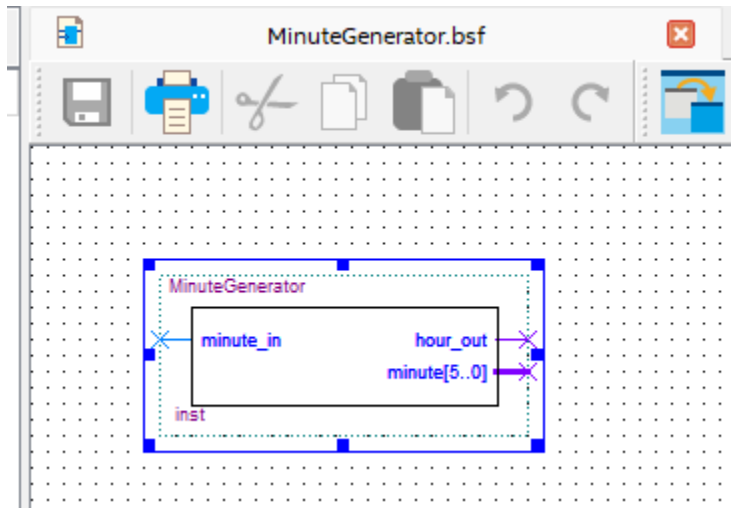
Simulation



RTL Diagram

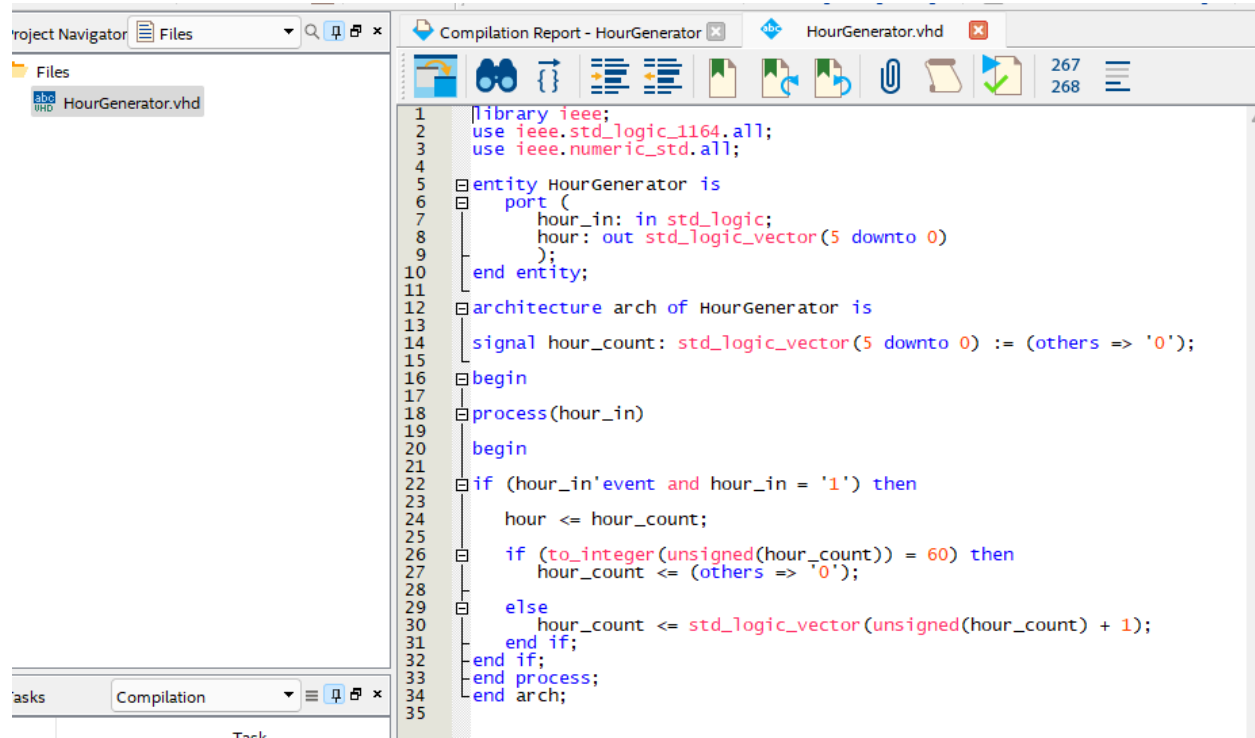


Block Design



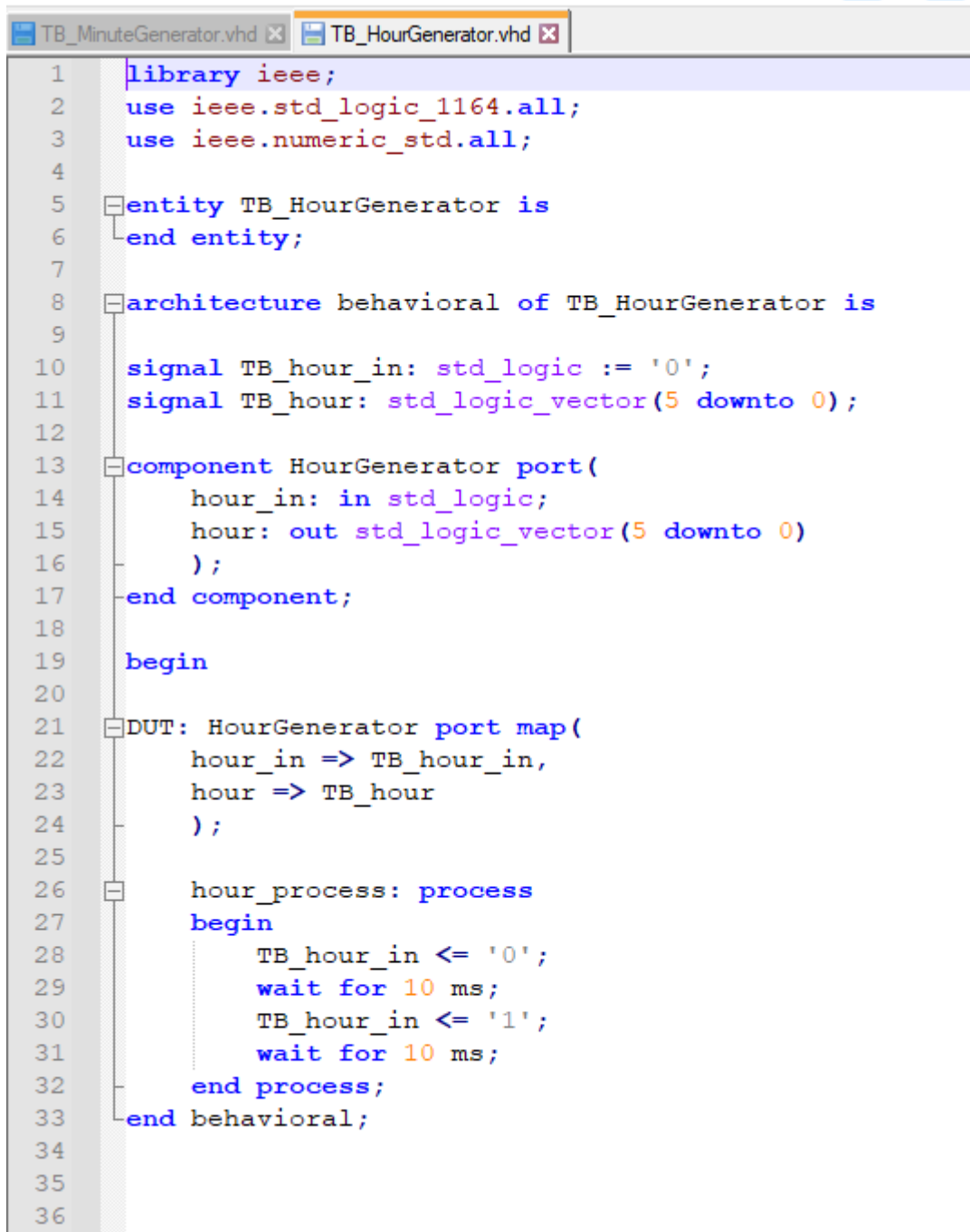
Hour Generator

Code



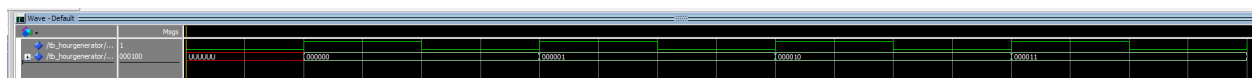
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity HourGenerator is
6 port (
7     hour_in: in std_logic;
8     hour: out std_logic_vector(5 downto 0)
9 );
10 end entity;
11
12 architecture arch of HourGenerator is
13     signal hour_count: std_logic_vector(5 downto 0) := (others => '0');
14
15 begin
16     process(hour_in)
17     begin
18         if (hour_in'event and hour_in = '1') then
19             hour_count <= (others => '0');
20         else
21             hour_count <= std_logic_vector(unsigned(hour_count) + 1);
22         end if;
23     end process;
24 end arch;
```


Testbench



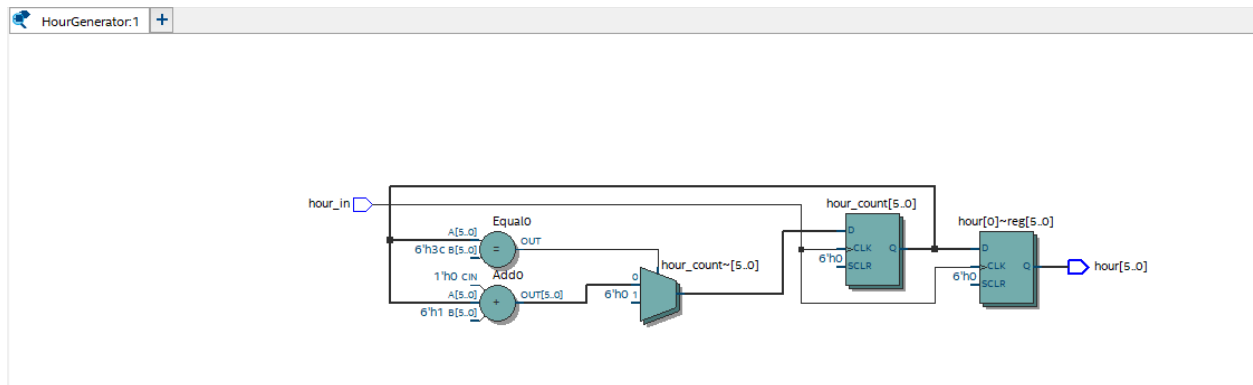
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity TB_HourGenerator is
6  end entity;
7
8  architecture behavioral of TB_HourGenerator is
9
10     signal TB_hour_in: std_logic := '0';
11     signal TB_hour: std_logic_vector(5 downto 0);
12
13     component HourGenerator port(
14         hour_in: in std_logic;
15         hour: out std_logic_vector(5 downto 0)
16     );
17 end component;
18
19 begin
20
21     DUT: HourGenerator port map(
22         hour_in => TB_hour_in,
23         hour => TB_hour
24     );
25
26     hour_process: process
27     begin
28         TB_hour_in <= '0';
29         wait for 10 ms;
30         TB_hour_in <= '1';
31         wait for 10 ms;
32     end process;
33 end behavioral;
34
35
36
```

Simulation

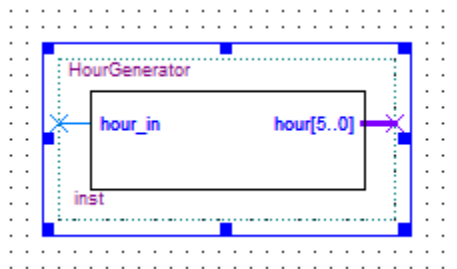


Wave - Default	Time	Signal	Value
1	00:00:00	hour_in	0
2	00:00:00	hour	000000
3	00:00:00	hour_in	1
4	00:00:00	hour	000000
5	00:00:00	hour_in	0
6	00:00:00	hour	000000
7	00:00:00	hour_in	1
8	00:00:00	hour	000000
9	00:00:00	hour_in	0
10	00:00:00	hour	000000
11	00:00:00	hour_in	1
12	00:00:00	hour	000000
13	00:00:00	hour_in	0
14	00:00:00	hour	000000
15	00:00:00	hour_in	1
16	00:00:00	hour	000000
17	00:00:00	hour_in	0
18	00:00:00	hour	000000
19	00:00:00	hour_in	1
20	00:00:00	hour	000000
21	00:00:00	hour_in	0
22	00:00:00	hour	000000
23	00:00:00	hour_in	1
24	00:00:00	hour	000000
25	00:00:00	hour_in	0
26	00:00:00	hour	000000
27	00:00:00	hour_in	1
28	00:00:00	hour	000000
29	00:00:00	hour_in	0
30	00:00:00	hour	000000
31	00:00:00	hour_in	1
32	00:00:00	hour	000000
33	00:00:00	hour_in	0
34	00:00:00	hour	000000
35	00:00:00	hour_in	1
36	00:00:00	hour	000000

RTL Diagram

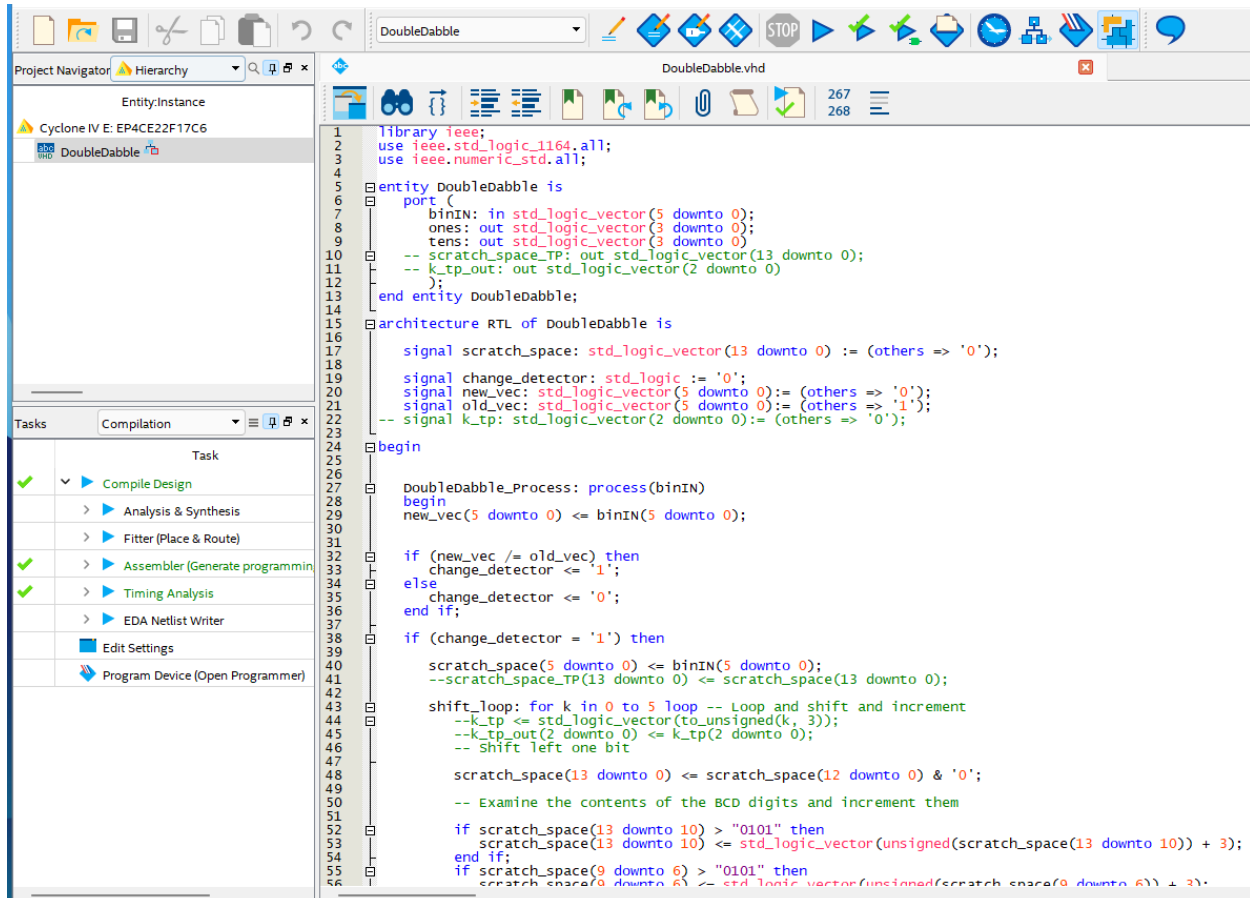


Block Design



Double Dabble

Code



```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity DoubleDabble is
6 port (
7     binIN: in std_logic_vector(5 downto 0);
8     ones: out std_logic_vector(3 downto 0);
9     tens: out std_logic_vector(3 downto 0);
10    -- scratch_space_TP: out std_logic_vector(13 downto 0);
11    -- k_tp_out: out std_logic_vector(2 downto 0)
12 );
13 end entity DoubleDabble;
14
15 architecture RTL of DoubleDabble is
16
17     signal scratch_space: std_logic_vector(13 downto 0) := (others => '0');
18
19     signal change_detector: std_logic := '0';
20     signal new_vec: std_logic_vector(5 downto 0) := (others => '0');
21     signal old_vec: std_logic_vector(5 downto 0) := (others => '1');
22     -- signal k_tp: std_logic_vector(2 downto 0) := (others => '0');
23
24 begin
25
26     DoubleDabble_Process: process(binIN)
27     begin
28         new_vec(5 downto 0) <= binIN(5 downto 0);
29
30
31         if (new_vec /= old_vec) then
32             change_detector <= '1';
33         else
34             change_detector <= '0';
35         end if;
36
37         if (change_detector = '1') then
38
39             scratch_space(5 downto 0) <= binIN(5 downto 0);
40             --scratch_space_TP(13 downto 0) <= scratch_space(13 downto 0);
41
42             shift_loop: for k in 0 to 5 loop -- Loop and shift and increment
43                 --k_tp <= std_logic_vector(to_unsigned(k, 3));
44                 --k_tp_out(2 downto 0) <= k_tp(2 downto 0);
45                 -- Shift left one bit
46
47                 scratch_space(13 downto 0) <= scratch_space(12 downto 0) & '0';
48
49                 -- Examine the contents of the BCD digits and increment them
50
51                 if scratch_space(13 downto 10) > "0101" then
52                     scratch_space(13 downto 10) <= std_logic_vector(unsigned(scratch_space(13 downto 10)) + 3);
53                 end if;
54                 if scratch_space(9 downto 6) > "0101" then
55                     scratch_space(9 downto 6) <= std_logic_vector(unsigned(scratch_space(9 downto 6)) + 3);
56                 end if;
```

```
55     if scratch_space(9 downto 6) > "0101" then
56         scratch_space(9 downto 6) <= std_logic_vector(unsigned(
           scratch_space(9 downto 6)) + 3);
57     end if;
58
59     --scratch_space_TP(13 downto 0) <= scratch_space(13 downto 0);
60
61     end loop shift_loop;
62
63     tens(3 downto 0) <= scratch_space(13 downto 10);
64     ones(3 downto 0) <= scratch_space(9 downto 6);
65
66 end if;
67 old_vec(5 downto 0) <= binIN(5 downto 0);
68
69 end process DoubleDabble_Process;
70 end RTL;
71
```

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 ?

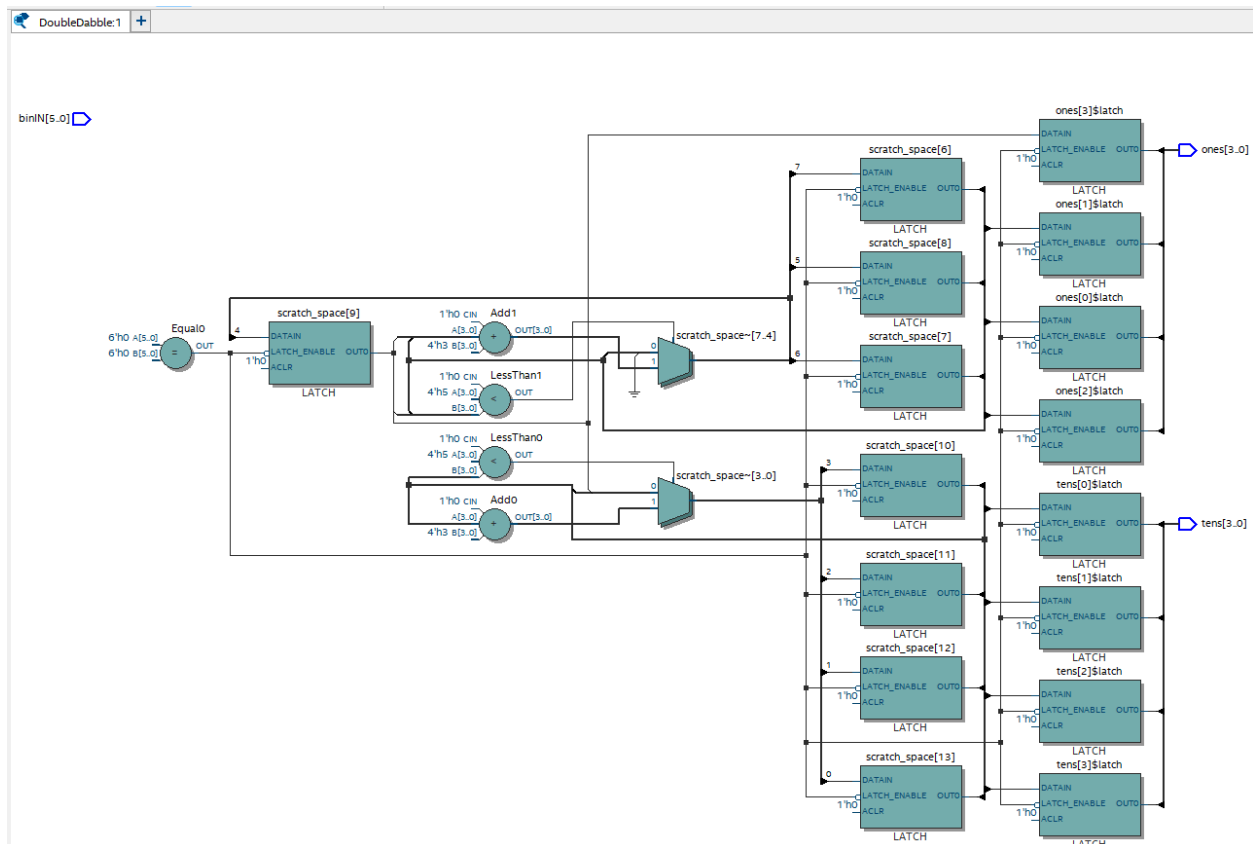
Comp1.vhd x AOI.vhd x 4p13.vhd x ADC.vhd x TB_MinuteGenerator.vhd x TB_DoubleDabble.vhd x DoubleDabble.vhd x TB_HourGenerator.vhd x

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity TB_DoubleDabble is
6  end entity;
7
8  architecture behavioral of TB_DoubleDabble is
9
10     signal TB_binIN: std_logic_vector(5 downto 0);
11     signal TB_ones: std_logic_vector(3 downto 0);
12     signal TB_tens: std_logic_vector(3 downto 0);
13     signal TB_Test_Point_Scratch_Space: std_logic_vector(13 downto 0);
14     signal TB_k_tp: std_logic_vector(2 downto 0);
15
16     component DoubleDabble port(
17         binIN: in std_logic_vector(5 downto 0);
18         ones: out std_logic_vector(3 downto 0);
19         tens: out std_logic_vector(3 downto 0);
20         scratch_space_TP: out std_logic_vector(13 downto 0);
21         k_tp_out: out std_logic_vector(2 downto 0)
22     );
23 end component;
24
25 begin
26
27     DUT: DoubleDabble port map(
28         binIN => TB_binIN,
29         ones => TB_ones,
30         tens => TB_tens,
31         scratch_space_TP => TB_Test_Point_Scratch_Space,
32         k_tp_out => TB_k_tp
33     );
34
35     test_process: process
36     begin
37         TB_binIN <= "001111"; -- 15 Base 10
38         wait for 20 ms;
39         TB_binIN <= "010010"; -- 18 Base 10;
40         wait for 20 ms;
41         TB_binIN <= "010101"; -- 21 Base 10
42         wait for 20 ms;
43         TB_binIN <= "010111"; -- 23 Base 10;
44         wait for 20 ms;
45
46     end process test_process;
47 end behavioral;
48
49
```

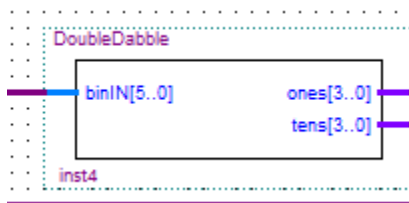
Simulation

Wave - Default		Hops									
TB_DoubleDabble_TB_ones	0000	0000									
	0000	0000									
	001111	001111	010010	010101	010111	001111	010010	010101	010111		

RTL Diagram



Block Design



DisplayDriver

Code

The screenshot displays the Quartus II IDE interface. On the left, the Project Navigator shows the hierarchy for 'Cyclone IV E: EP4CE22F17C6' with 'DisplayDriver' selected. The main window shows the VHDL code for 'DisplayDriver'. The code defines an entity with ports for scan clock, minute/ten digits, hour/ten digits, display output, and digit vector. It includes an RTL architecture with signals for seven-segment variable, digit-to-map, scan counter, and internal digit vector. The logic processes the scan clock to increment the scan counter and map the digit values to the seven-segment variable. The tasks window at the bottom left shows a list of compilation tasks, with 'Compile Design' highlighted.

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5
6 entity DisplayDriver is
7 port (
8     scan_clk: in std_logic;
9     minute_ones: in std_logic_vector(3 downto 0);
10    minute_tens: in std_logic_vector(3 downto 0);
11    hour_ones: in std_logic_vector(3 downto 0);
12    hour_tens: in std_logic_vector(3 downto 0);
13    display_out: out std_logic_vector(6 downto 0);
14    digit_vector: out std_logic_vector(3 downto 0);
15 );
16 end entity DisplayDriver;
17
18 architecture RTL of DisplayDriver is
19
20     signal seven_segment_variable: std_logic_vector(6 downto 0);
21     signal digit_to_map: std_logic_vector(3 downto 0);
22     signal scan_counter: std_logic_vector(1 downto 0) := (others => '0');
23     signal internal_digit_vector: std_logic_vector(3 downto 0) := "0001";
24
25     begin
26     process(scan_clk)
27     begin
28
29         if (scan_clk'event and scan_clk = '1') then
30
31             scan_counter <= std_logic_vector(unsigned(scan_counter(1 downto 0)) + 1);
32
33             case scan_counter is
34             when "00" =>
35                 digit_to_map(3 downto 0) <= minute_ones(3 downto 0);
36                 internal_digit_vector(3 downto 0) <= "0001";
37             when "01" =>
38                 digit_to_map(3 downto 0) <= minute_tens(3 downto 0);
39                 internal_digit_vector(3 downto 0) <= "0010";
40             when "10" =>
41                 digit_to_map(3 downto 0) <= hour_ones(3 downto 0);
42                 internal_digit_vector(3 downto 0) <= "0100";
43             when "11" =>
44                 digit_to_map(3 downto 0) <= hour_tens(3 downto 0);
45                 internal_digit_vector(3 downto 0) <= "1000";
46             when others =>
47                 digit_to_map(3 downto 0) <= minute_ones(3 downto 0);
48                 internal_digit_vector(3 downto 0) <= "0001";
49             end case;
50
51         end if;
52
53         digit_vector(3 downto 0) <= internal_digit_vector(3 downto 0);
54
55     end process;
56
57     process(digit_to_map)
58     begin
59         case digit_to_map is
60         when "0000" =>
61             seven_segment_variable <= "1111110"; -- 0
62         when "0001" =>
63             seven_segment_variable <= "0110000"; -- 1
64         when "0010" =>
65             seven_segment_variable <= "1101101"; -- 2
66         when "0011" =>
67             seven_segment_variable <= "1111001"; -- 3
68         when "0100" =>
69             seven_segment_variable <= "0110011"; -- 4
70         when "0101" =>
71             seven_segment_variable <= "1011011"; -- 5
72         when "0110" =>
73             seven_segment_variable <= "1011111"; -- 6
74         when "0111" =>
75             seven_segment_variable <= "1110000"; -- 7
76         when "1000" =>
77             seven_segment_variable <= "1111111"; -- 8
78         when "1001" =>
79             seven_segment_variable <= "1111111"; -- 9
80         end case;
81     end process;
82 end architecture RTL;
```

Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming file)
Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)

```
79         seven_segment_variable <= "1111111"; -- 8
80     when "1001" =>
81         seven_segment_variable <= "1110011"; -- 9
82     when others =>
83         seven_segment_variable <= "1111110"; -- 0
84     end case;
85 end process;
86
87 process(seven_segment_variable)
88     begin
89         display_out(6 downto 0) <= seven_segment_variable(6 downto 0);
90     end process;
91
92 end RTL;
```


TestBench

```
TB_DisplayDriver.vhd
7
8 architecture behavioral of TB_DisplayDriver is
9
10 signal TB_scan_clk: std_logic := '0';
11 signal TB_minute_ones: std_logic_vector(3 downto 0) := (others =>
    '0');
12 signal TB_minute_tens: std_logic_vector(3 downto 0) := (others =>
    '0');
13 signal TB_hour_ones: std_logic_vector(3 downto 0) := (others =>
    '0');
14 signal TB_hour_tens: std_logic_vector(3 downto 0) := (others =>
    '0');
15 signal TB_display_out: std_logic_vector(6 downto 0);
16 signal TB_digit_vector: std_logic_vector(3 downto 0);
17
18 component DisplayDriver port(
19     scan_clk: in std_logic:= '0';
20     minute_ones: in std_logic_vector(3 downto 0);
21     minute_tens: in std_logic_vector(3 downto 0);
22     hour_ones: in std_logic_vector(3 downto 0);
23     hour_tens: in std_logic_vector(3 downto 0);
24     display_out: out std_logic_vector(6 downto 0);
25     digit_vector: out std_logic_vector(3 downto 0)
26 );
27 end component;
28
29 begin
30
31 DUT: DisplayDriver port map(
32     scan_clk => TB_scan_clk,
33     minute_ones => TB_minute_ones,
34     minute_tens => TB_minute_tens,
35     hour_ones => TB_hour_ones,
36     hour_tens => TB_hour_tens,
37     display_out => TB_display_out,
38     digit_vector => TB_digit_vector
39 );
40
41 scan_clk_process: process
42 begin
43
44     TB_scan_clk <= '0';
45     wait for 2 ms;
46     TB_scan_clk <= '1';
47     wait for 2 ms;
48 end process;
49
50 test_process: process
51 begin
52
53
54
55     TB_minute_ones <= "0001";
56
57     wait for 10 ms;
58
59     TB_minute_tens <= "0010";
60
61     wait for 10 ms;
62
63     TB_hour_ones <= "0011";
64
65     wait for 10 ms;
66     TB_hour_tens <= "0100";
67
68     wait for 10 ms;
69 end process;
70 end behavioral;
71
```

