



# ARTYLAB

ARTY FPGA VHDL/ASSEMBLY EXAMPLES

Embedded Systems Project

Agnese Salutari, [agnese.salutari@student.univaq.it](mailto:agnese.salutari@student.univaq.it)

# ARTYLAB PROJECT OVERVIEW

In this project I performed some VHDL/Assembly exercises.

There are different sections, each one corresponding to a folder (named as the relative section).

Every section contains a different exercise type:

- Section 1: Simulation and Synthesis of simple combinatorial circuits
- Section 2: Sequential circuits, I/O and constraints
- Section 3: PicoBlaze
- Section 4: Section 4: Serial Communication

# ENVIRONMENT

In order to perform the exercises proposed in the various sections, I used the following tools:

- For design, synthesis and simulation:
  - Vivado Web Package: [https://www.xilinx.com/products/design\\_tools/vivado.html](https://www.xilinx.com/products/design_tools/vivado.html), using Xilinx Artix-35T FPGA (xc7a35ticsg324-IL) model
- To run the resulting code on the real hardware:
  - Arty Board: <https://reference.digilentinc.com/reference/programmable-logic/arty/reference-manual?redirect=1>
- PicoBlaze material:
  - <https://www.xilinx.com/products/intellectual-property/picoblaze.html>



# SECTION I

SIMULATION AND SYNTHESIS OF SIMPLE COMBINATORIAL CIRCUITS

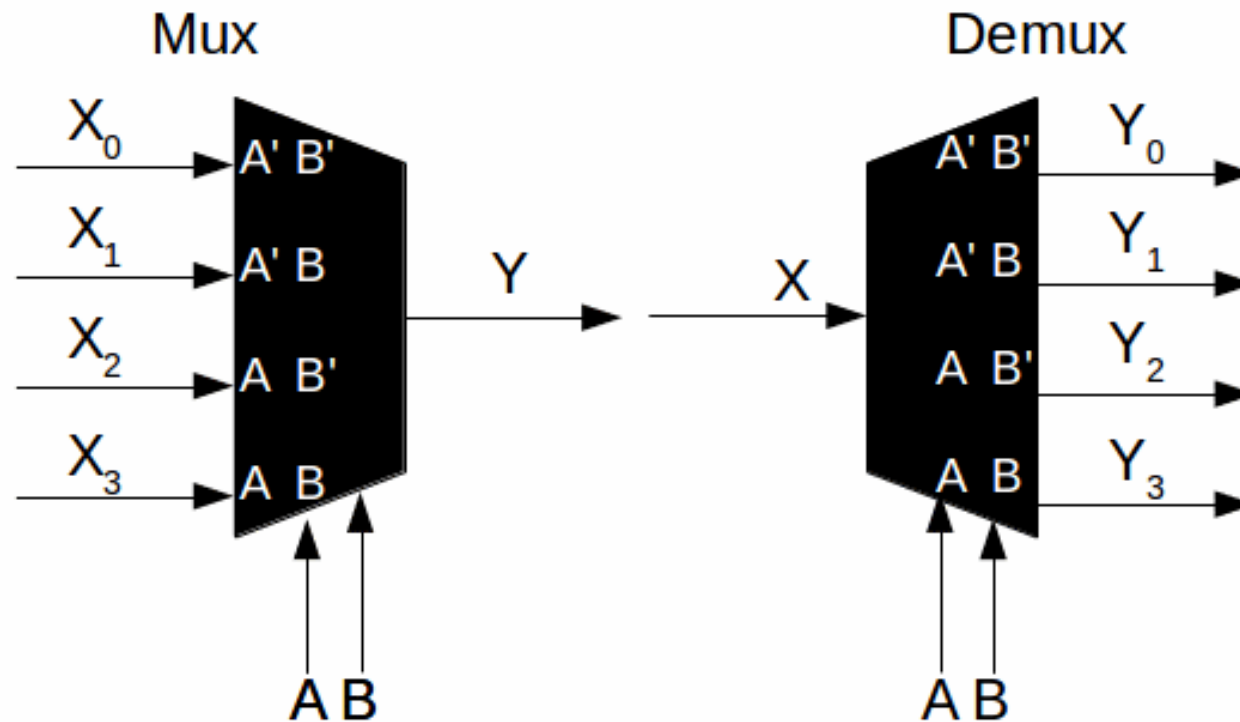


# SECTION I – TO DO

In this first section, I had to:

- Develop a VHDL description of the following combinatorial circuits:
  - Multiplexer
  - Demultiplexer
- Verify that the circuits behavior was correct by means of behavioral simulation.
- Verify if the obtained circuits was synthesizable and implementable.

## SECTION I – MULTIPLEXER AND DEMULTIPLEXER



# SECTION I – WORKSPACE

I organized this section in two sub-sections:

- Multiplexer

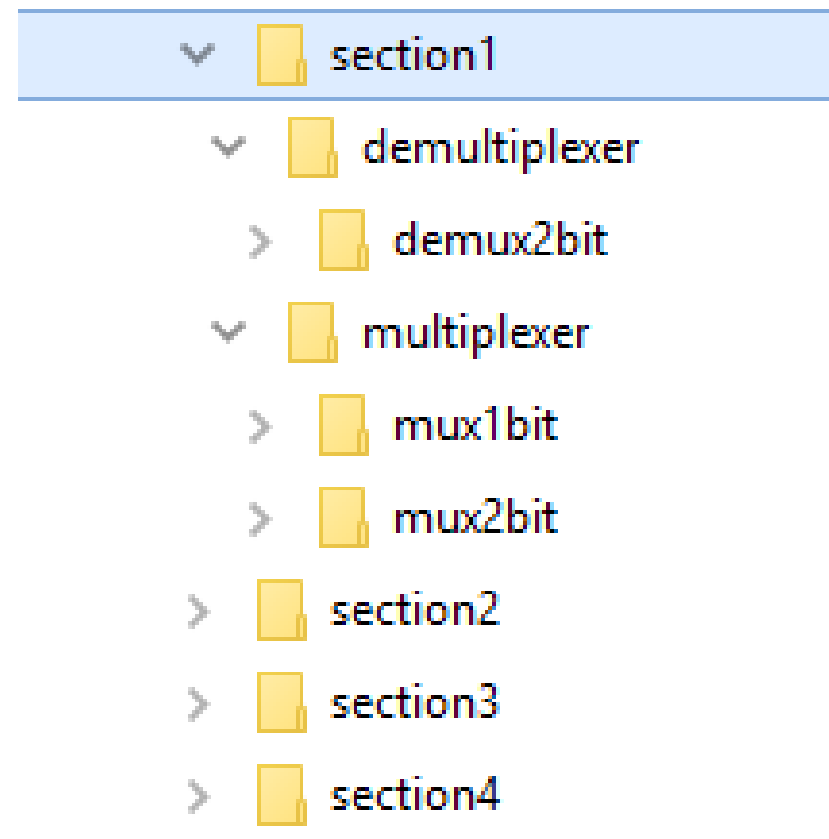
That consist of the following Vivado Projects:

- 1 bit Multiplexer
- 2 bit Multiplexer

- Demultiplexer

That consist of the following Vivado Project:

- 2 bit Demultiplexer



## SECTION I – I BIT MULTIPLEXER VHDL CODE

```
30 -----
31
32
33 library IEEE;
34 use IEEE.STD_LOGIC_1164.ALL;
35
36 -- Uncomment the following library declaration if using
37 -- arithmetic functions with Signed or Unsigned values
38 --use IEEE.NUMERIC_STD.ALL;
39
40 -- Uncomment the following library declaration if instantiating
41 -- any Xilinx leaf cells in this code.
42 --library UNISIM;
43 --use UNISIM.VComponents.all;
44
45 entity mux1bit is
46     Port ( A : in STD_LOGIC;
47           B : in STD_LOGIC;
48           SEL : in STD_LOGIC;
49           Y : out STD_LOGIC);
50 end mux1bit;
51
52 architecture Behavioral of mux1bit is
53
54 begin
55     with SEL select
56     Y <= A when '0',
57         B when others;
58 end Behavioral;
59
```

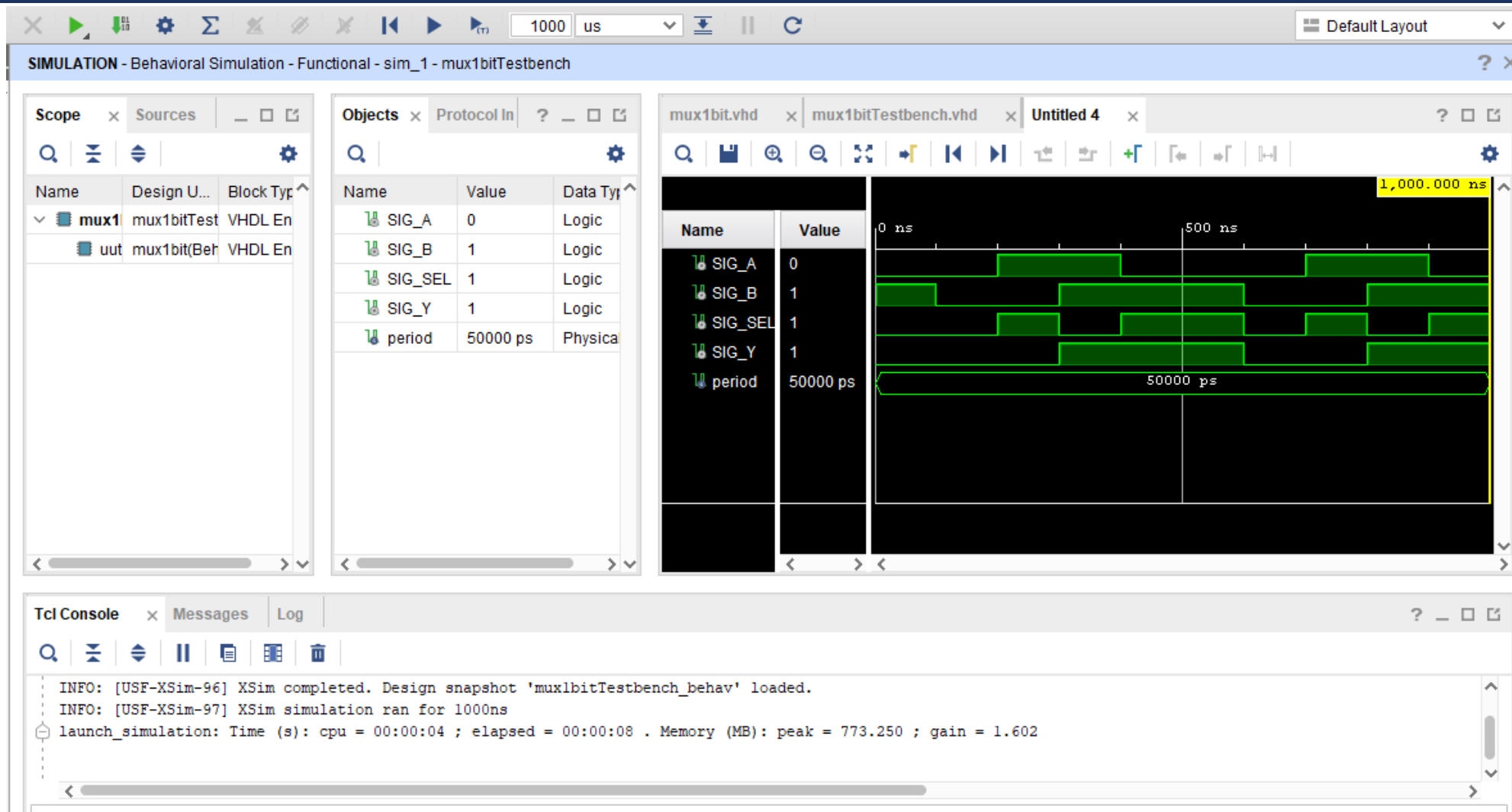


# SECTION I – 1 BIT MULTIPLEXER TESTBENCH VHDL CODE

```
30 -----
31
32
33 library IEEE;
34 use IEEE.STD_LOGIC_1164.ALL;
35
36 -- Uncomment the following library declaration if using
37 -- arithmetic functions with Signed or Unsigned values
38 --use IEEE.NUMERIC_STD.ALL;
39
40 -- Uncomment the following library declaration if instantiating
41 -- any Xilinx leaf cells in this code.
42 --library UNISIM;
43 --use UNISIM.VComponents.all;
44
45 entity mux1bitTestbench is
46 -- Port ( );
47 end mux1bitTestbench;
48
49 ▼ architecture Behavioral of mux1bitTestbench is
50     -- Component Declaration for the Unit Under Test (UUT):
51     COMPONENT mux1bit
52     ▼ PORT(
53         A : IN std_logic;
54         B : IN std_logic;
55         SEL: IN std_logic;
56         Y: OUT std_logic
57     );
58     END COMPONENT;
59     -- Input Signals:
60     signal SIG_A : std_logic := '0';
61     signal SIG_B : std_logic := '1';
62     signal SIG_SEL : std_logic := '0';
63     -- Output Signals:
64     signal SIG_Y : std_logic;
65     -- Delta Time:
66     constant period : time := 50 ns;
67 ▼ begin
68     -- Unit Under Test (UUT) Instantiation:
69 ▼     uut: mux1bit PORT MAP (
70         A => SIG_A,
71         B => SIG_B,
72         SEL => SIG_SEL,
73         Y => SIG_Y
74     );
75     -- Stimulus processes:
```

```
74     );
75     -- Stimulus processes:
76     stimSEL: process
77 ▼ begin
78         -- hold reset state for 100 ns.
79         wait for 100 ns;
80         -- Stimulus:
81         SIG_SEL <= '0';
82         wait for period * 2;
83         SIG_SEL <= '1';
84         wait for period * 2;
85         SIG_SEL <= '0';
86         wait for period * 2;
87         SIG_SEL <= '1';
88         wait for period * 2;
89     end process;
90     stimA: process
91     ▼ begin
92         -- hold reset state for 100 ns.
93         wait for 100 ns;
94         -- Stimulus:
95         SIG_A <= '0';
96         wait for period * 2;
97         SIG_A <= '1';
98         wait for period * 4;
99         SIG_A <= '0';
100        wait for period * 2;
101    end process;
102    stimB: process
103    ▼ begin
104        -- hold reset state for 100 ns.
105        wait for 100 ns;
106        -- Stimulus:
107        SIG_B <= '0';
108        wait for period * 4;
109        SIG_B <= '1';
110        wait for period * 4;
111    end process;
112
113 end Behavioral;
```

# SECTION I – 1 BIT MULTIPLEXER BEHAVIOURAL SIMULATION



# SECTION I – 1 BIT MULTIPLEXER SYNTHESIS

Implementation Complete ✓

Default Layout

IMPLEMENTED DESIGN - xc7a35ticsg324-1L

Sources Netlist

- mux1bit
  - Nets (8)
  - Leaf Cells (5)

Properties

Select an object to see properties

Project Summary Device mux1bit.vhd mux1bitTestbench.vhd

X0Y2 X1Y2

X0Y1 X1Y1

X0Y0 X1Y0

Tcl Console Messages Log Reports Design Runs Power DRC Timing

Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): NA	Worst Hold Slack (WHS): NA	Worst Pulse Width Slack (WPWS): NA
Total Negative Slack (TNS): NA	Total Hold Slack (THS): NA	Total Pulse Width Negative Slack (TPWS): NA
Number of Failing Endpoints: NA	Number of Failing Endpoints: NA	Number of Failing Endpoints: NA
Total Number of Endpoints: NA	Total Number of Endpoints: NA	Total Number of Endpoints: NA

There are no user specified timing constraints.

Timing Summary - impl\_1 (saved)

## SECTION 1 – 1 BIT MULTIPLEXER RESULTS

I obtained the following results:

- Behavioural Simulation worked as expected by design.
- Synthesis and Implementation worked too.

## SECTION 1 – 2 BIT MULTIPLEXER VHDL CODE

```
30 -----
31
32
33 library IEEE;
34 use IEEE.STD_LOGIC_1164.ALL;
35
36 -- Uncomment the following library declaration if using
37 -- arithmetic functions with Signed or Unsigned values
38 --use IEEE.NUMERIC_STD.ALL;
39
40 -- Uncomment the following library declaration if instantiating
41 -- any Xilinx leaf cells in this code.
42 --library UNISIM;
43 --use UNISIM.VComponents.all;
44
45 entity mux2bit is
46     Port (
47         A : IN std_logic;
48         B : IN std_logic;
49         C : IN std_logic;
50         D : IN std_logic;
51         SEL : IN std_logic_vector (1 downto 0);
52         Y : OUT std_logic
53     );
54 end mux2bit;
55
56 architecture Behavioral of mux2bit is
57
58 begin
59     with SEL select
60         Y <= A when "00",
61             B when "01",
62             C when "10",
63             D when others;
64 end Behavioral;
65
```

## SECTION 1 – 2 BIT MULTIPLEXER TESTBENCH VHDL CODE

```
43 --use UNISIM.VComponents.all;
44
45 entity mux2bitTestbench is
46 -- Port ( );
47 end mux2bitTestbench;
48
49 ▼ architecture Behavioral of mux2bitTestbench is
50 -- Component Declaration for the Unit Under Test (UUT):
51 COMPONENT mux2bit
52 ▼ PORT(
53     A : IN std_logic;
54     B : IN std_logic;
55     C : IN std_logic;
56     D : IN std_logic;
57     SEL : IN std_logic_vector (1 downto 0);
58     Y : OUT std_logic
59 );
60 END COMPONENT;
61 -- Input Signals:
62 signal SIG_A : std_logic := '0';
63 signal SIG_B : std_logic := '0';
64 signal SIG_C : std_logic := '0';
65 signal SIG_D : std_logic := '0';
66 signal SIG_SEL : std_logic_vector (1 downto 0) := "00";
67 -- Output Signals:
68 signal SIG_Y : std_logic := '0';
69 -- Delta Time:
70 constant period : time := 50 ns;
71 -- Time for reset:
72 constant reset_time : time := 100 ns;
73 ▼ begin
74 -- Unit Under Test (UUT) Instantiation:
75 ▼ uut: mux2bit PORT MAP(
76     A => SIG_A,
77     B => SIG_B,
78     C => SIG_C,
79     D => SIG_D,
80     SEL => SIG_SEL,
81     Y => SIG_Y
82 );
83 -- Stimulus Processes:
84 stimSEL : process
85 ▼ begin
86     wait for reset_time;
87     SIG_SEL <= "11";
88     wait for period * 2;
89     SIG_SEL <= "10";
90     wait for period * 2;
91     SIG_SEL <= "01";
92     wait for period * 2;
93     SIG_SEL <= "00";
94     wait for period * 2;
95 end process;
96 stimA : process
97 ▼ begin
98     wait for reset_time;
99     SIG_A <= '1';
100    wait for period * 5;
101    SIG_A <= '0';
102    wait for period * 3;
103
104    SIG_A <= '0';
105    wait for period * 3;
106    SIG_B <= '0';
107    wait for period * 4;
108    SIG_B <= '1';
109    wait for period * 4;
110 end process;
111 stimC : process
112 ▼ begin
113     wait for reset_time;
114     SIG_C <= '1';
115     wait for period * 2;
116     SIG_C <= '0';
117     wait for period * 2;
118 end process;
119 stimD : process
120 ▼ begin
121     wait for reset_time;
122     SIG_D <= '0';
123     wait for period * 2;
124     SIG_D <= '1';
125     wait for period * 2;
126 end process;
127 end Behavioral;
128
129
```

# SECTION 1 – 2 BIT MULTIPLEXER BEHAVIOURAL SIMULATION

The screenshot displays the Xilinx Vivado Behavioral Simulation interface for a 2-bit multiplexer testbench. The main window shows the simulation results for the testbench, including the Scope, Objects, and Waveform views.

**Scope View:**

Name	Design U...	Block
mux2	mux2bitTest	VHDL
uut	mux2bit(Beh	VHDL

**Objects View:**

Name	Value	Data Type
SIG_A	0	Logic
SIG_B	1	Logic
SIG_C	1	Logic
SIG_D	0	Logic
SIG_SEL[0]	0	Array
[1]	0	Logic
[0]	0	Logic
SIG_Y	0	Logic
period	50000 ps	Physical Typ
reset_time	100000 ps	Physical Typ

**Waveform View:**

The waveform shows the signals SIG\_A, SIG\_B, SIG\_C, SIG\_D, SIG\_Y, period, and reset\_time over time. The time scale is 1,000.000 ns. The signals are plotted as digital waveforms. The SIG\_Y signal is highlighted in green.

**Tcl Console:**

```
# run 1000ns
INFO: [USF-XSim-96] XSim completed. Design snapshot 'mux2bitTestbench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:08 . Memory (MB): peak = 772.941 ; gain = 0.000
```

# SECTION I – 2 BIT MULTIPLEXER SYNTHESIS

Tools Reports Window Layout View Help Q- Quick Access Implementation Complete ✓

Default Layout

IMPLEMENTED DESIGN - xc7a35ticsg324-1L

Sources Netlist x ? \_ □ □

mux2bit

- > Nets (14)
- > Leaf Cells (8)

Properties ? \_ □ □ x

Select an object to see properties

Project Summary x Device x mux2bitTestbench.vhd x ? \_ □ □

X0Y2 X1Y2

X0Y1 X1Y1

X0Y0 X1Y0

Tcl Console Messages Log Reports Design Runs Power DRC Timing x ? \_ □ □

Design Timing Summary

General Information

Timer Settings

Design Timing Summary

- > Check Timing (0)
- User Ignored Paths
- Unconstrained Paths

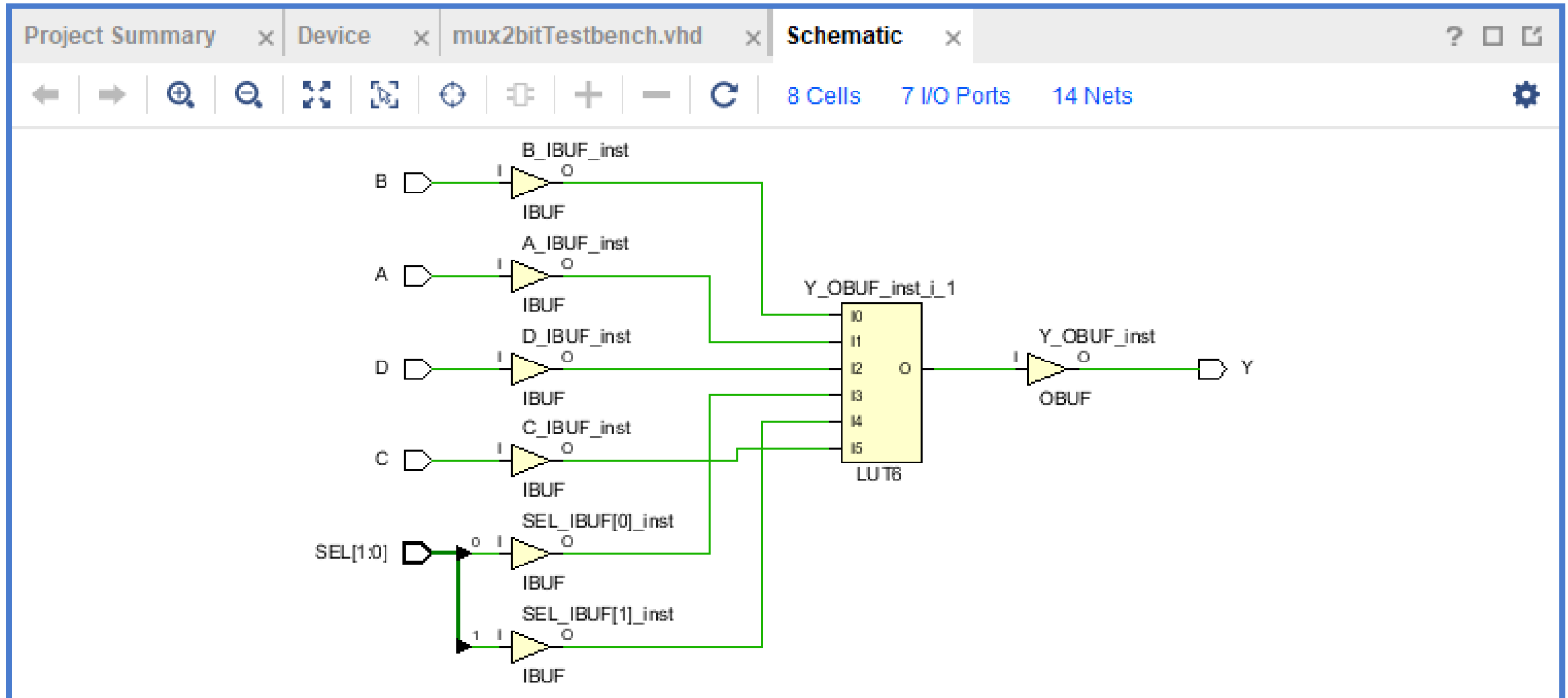
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): NA	Worst Hold Slack (WHS): NA	Worst Pulse Width Slack (WPWS): NA
Total Negative Slack (TNS): NA	Total Hold Slack (THS): NA	Total Pulse Width Negative Slack (TPWS): NA
Number of Failing Endpoints: NA	Number of Failing Endpoints: NA	Number of Failing Endpoints: NA
Total Number of Endpoints: NA	Total Number of Endpoints: NA	Total Number of Endpoints: NA

There are no user specified timing constraints.

Timing Summary - impl\_1 (saved)



# SECTION 1 – 2 BIT MULTIPLEXER SCHEMATIC



## SECTION 1 – 2 BIT MULTIPLEXER RESULTS

I obtained the following results:

- Behavioural Simulation worked as expected by design.
- Synthesis and Implementation worked too.

## SECTION 1 – 2 BIT DEMULTIPLEXER VHDL CODE

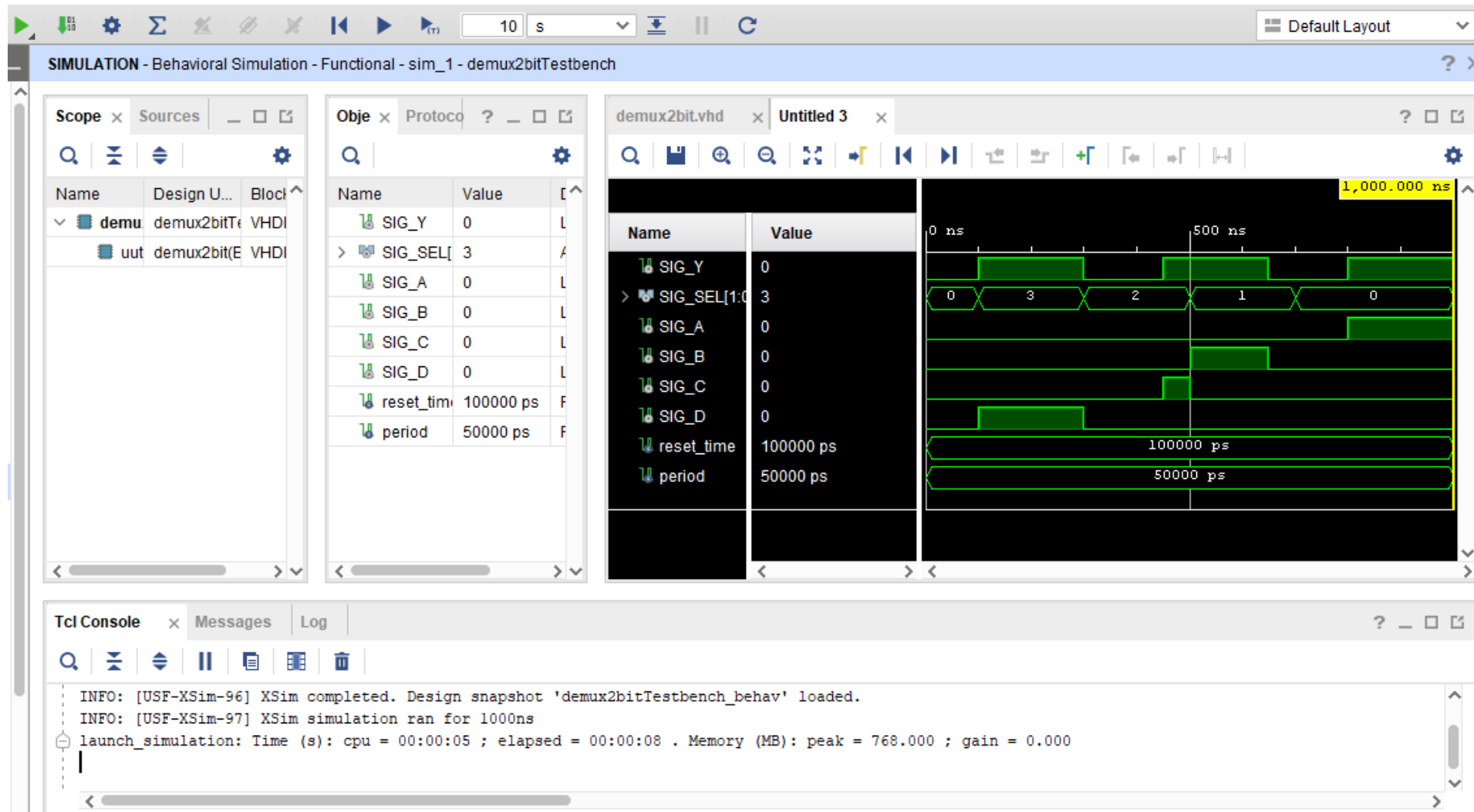
```
43 --use UNISIM.VComponents.all;
44
45 entity demux2bit is
46   Port ( Y : in STD_LOGIC;
47         SEL : in STD_LOGIC_VECTOR (1 DOWNTO 0);
48         A : out STD_LOGIC;
49         B : out STD_LOGIC;
50         C : out STD_LOGIC;
51         D : out STD_LOGIC);
52 end demux2bit;
53
54 architecture Behavioral of demux2bit is
55
56 begin
57   selection : process(SEL, Y) is
58   begin
59     if (SEL = "00") then
60       A <= Y;
61       B <= '0';
62       C <= '0';
63       D <= '0';
64     elsif (SEL = "01") then
65       A <= '0';
66       B <= Y;
67       C <= '0';
68       D <= '0';
69     elsif (SEL = "10") then
70       A <= '0';
71       B <= '0';
72       C <= Y;
73       D <= '0';
74     else
75       A <= '0';
76       B <= '0';
77       C <= '0';
78       D <= Y;
79     end if;
80   end process;
81
82 end Behavioral;
83
```

# SECTION 1 – 2 BIT DEMULTIPLEXER TESTBENCH VHDL CODE

```
32 library IEEE;
33 use IEEE.STD_LOGIC_1164.ALL;
34
35 -- Uncomment the following library declaration if using
36 -- arithmetic functions with Signed or Unsigned values
37 --use IEEE.NUMERIC_STD.ALL;
38
39 -- Uncomment the following library declaration if instantiating
40 -- any Xilinx leaf cells in this code.
41 --library UNISIM;
42 --use UNISIM.VComponents.all;
43
44 entity demux2bitTestbench is
45 -- Port ( );
46 end demux2bitTestbench;
47
48 architecture Behavioral of demux2bitTestbench is
49 -- Component Declaration for the Unit Under Test (UUT):
50 component demux2bit
51 PORT(
52     Y : in STD_LOGIC;
53     SEL : in STD_LOGIC_VECTOR (1 DOWNTO 0);
54     A : out STD_LOGIC;
55     B : out STD_LOGIC;
56     C : out STD_LOGIC;
57     D : out STD_LOGIC
58 );
59 end component;
60 -- Input Signals:
61 signal SIG_Y : STD_LOGIC := '0';
62 signal SIG_SEL : STD_LOGIC_VECTOR (1 DOWNTO 0) := "00";
63 -- Output Signals:
64 signal SIG_A : STD_LOGIC := '0';
65 signal SIG_B : STD_LOGIC := '0';
66 signal SIG_C : STD_LOGIC := '0';
67 signal SIG_D : STD_LOGIC := '0';
68 -- Reset Time:
69 constant reset_time : time := 100 ns;
70 -- Delta Time:
71 constant period : time := 50 ns;
72
73 begin
74 -- Under Unit Test (UUT) Instantiation:
75 uut: demux2bit PORT MAP(
76     Y => SIG_Y,
```

```
74 -- Under Unit Test (UUT) Instantiation:
75 uut: demux2bit PORT MAP(
76     Y => SIG_Y,
77     SEL => SIG_SEL,
78     A => SIG_A,
79     B => SIG_B,
80     C => SIG_C,
81     D => SIG_D
82 );
83 -- Stimulus Processes:
84 stimY : process
85 begin
86     wait for reset_time;
87     SIG_Y <= '1';
88     wait for period * 4;
89     SIG_Y <= '0';
90     wait for period;
91 end process;
92 stimSEL : process
93 begin
94     wait for reset_time;
95     SIG_SEL <= "11";
96     wait for period * 4;
97     SIG_SEL <= "10";
98     wait for period * 4;
99     SIG_SEL <= "01";
100    wait for period * 4;
101    SIG_SEL <= "00";
102    wait for period * 4;
103 end process;
104 end Behavioral;
105
```

# SECTION I – 2 BIT DEMULTIPLEXER BEHAVIOURAL SIMULATION



# SECTION I – 2 BIT DEMULTIPLEXER SYNTHESIS

Implementation Complete ✓

Default Layout

IMPLEMENTED DESIGN - xc7a35ticsg324-1L

Sources Netlist x

demux2bit

- > Nets (14)
- > Leaf Cells (11)

Properties

Select an object to see properties

Project Summary x Device x demux2bit.vhd x

X0Y2 X1Y2

X0Y1 X1Y1

X0Y0 X1Y0

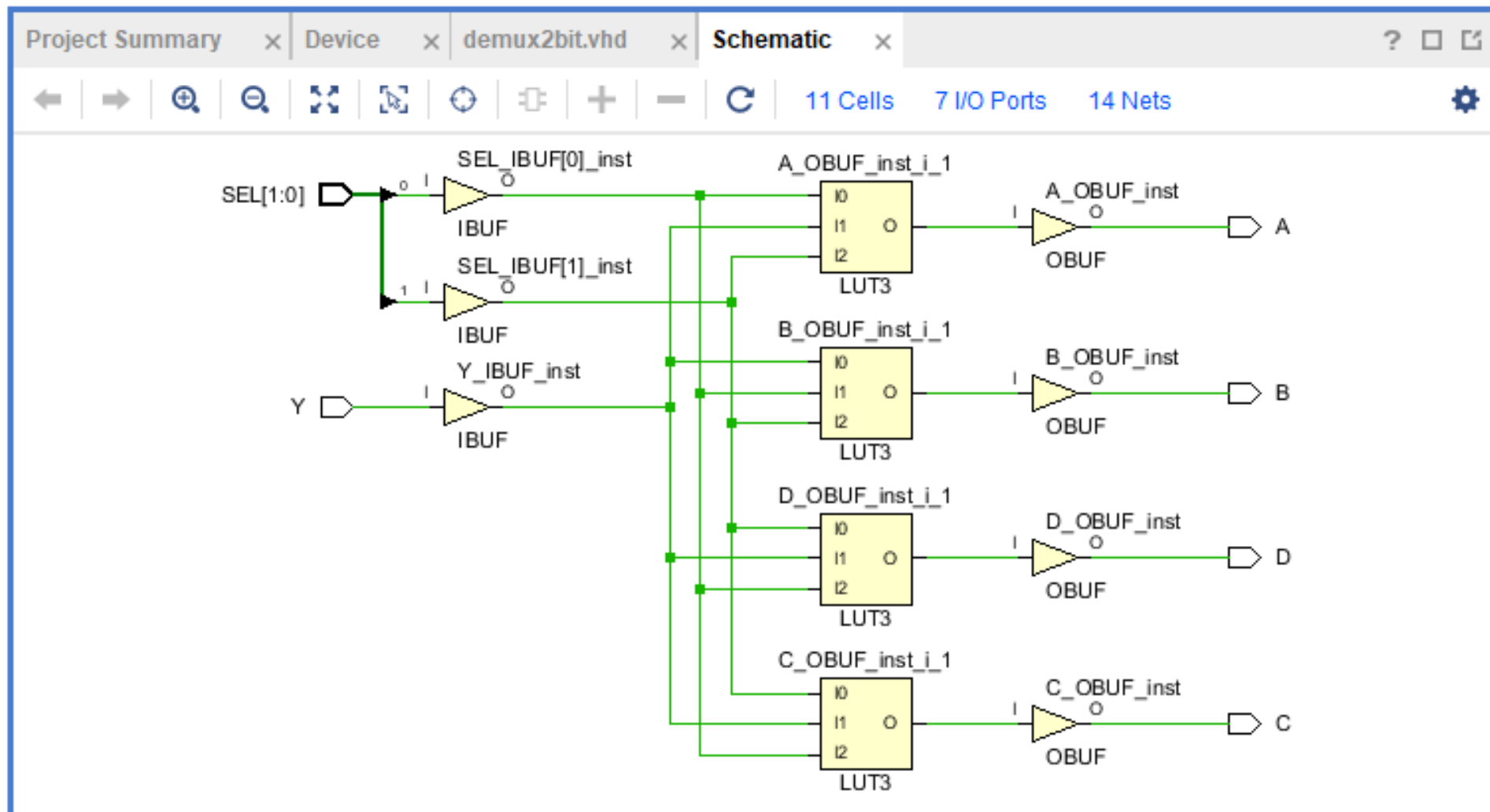
Tcl Console Messages Log Reports Design Runs Power DRC Timing x

Design Timing Summary

General Information	Setup	Hold	Pulse Width
Timer Settings	Worst Negative Slack (WNS):	Worst Hold Slack (WHS):	Worst Pulse Width Slack (WPWS):
Design Timing Summary	NA	NA	NA
Check Timing (0)	Total Negative Slack (TNS):	Total Hold Slack (THS):	Total Pulse Width Negative Slack (TPWS):
	NA	NA	NA

Timing Summary - impl\_1 (saved)

## SECTION 1 – 2 BIT DEMULTIPLEXER SCHEMATIC



## SECTION 1 – 2 BIT DEMULTIPLEXER RESULTS

I obtained the following results:

- Behavioural Simulation worked as expected by design.
- Synthesis and Implementation worked too.





# SECTION 2

SEQUENTIAL CIRCUITS, I/O AND CONSTRAINTS



## SECTION 2 – TO DO

In this second section, I had to:

- Given Counter VHDL code
  - Modify it, in order to perform Vivado Behavioural Simulation (a faster clock was needed)
  - Create a Testbench for that Counter to simulate it on Vivado
  - Create Constraints (on XDC file) and generate the Bitstream (using the given code), in order to run it on Arty board
- Given Sequence Detector VHDL code
  - Modify XDC file to specify the right connection between sequence detector, board LEDs and board switches
- Simulate the given GCD VHDL codes, one for each different kind of hardware description (behavioral, RTL with FSM and RTL with FSM and datapath).

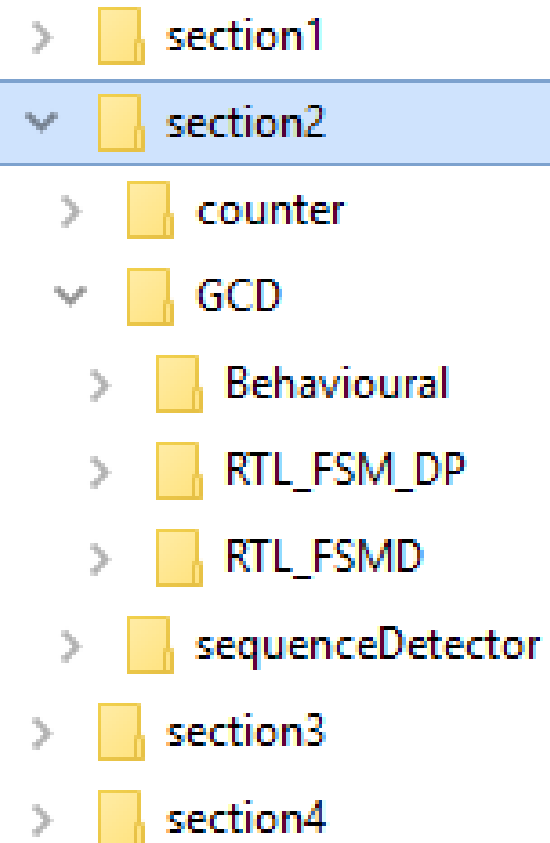
## SECTION 2 – WORKSPACE

I organized this section in two sub-sections:

- Counter
- GCD

That consist of the following Vivado Project:

- Behavioural
- RTL\_FSM\_DP
- RTL\_FSMD
- SequenceDetector



## SECTION 2 – MODIFIED COUNTER CODE

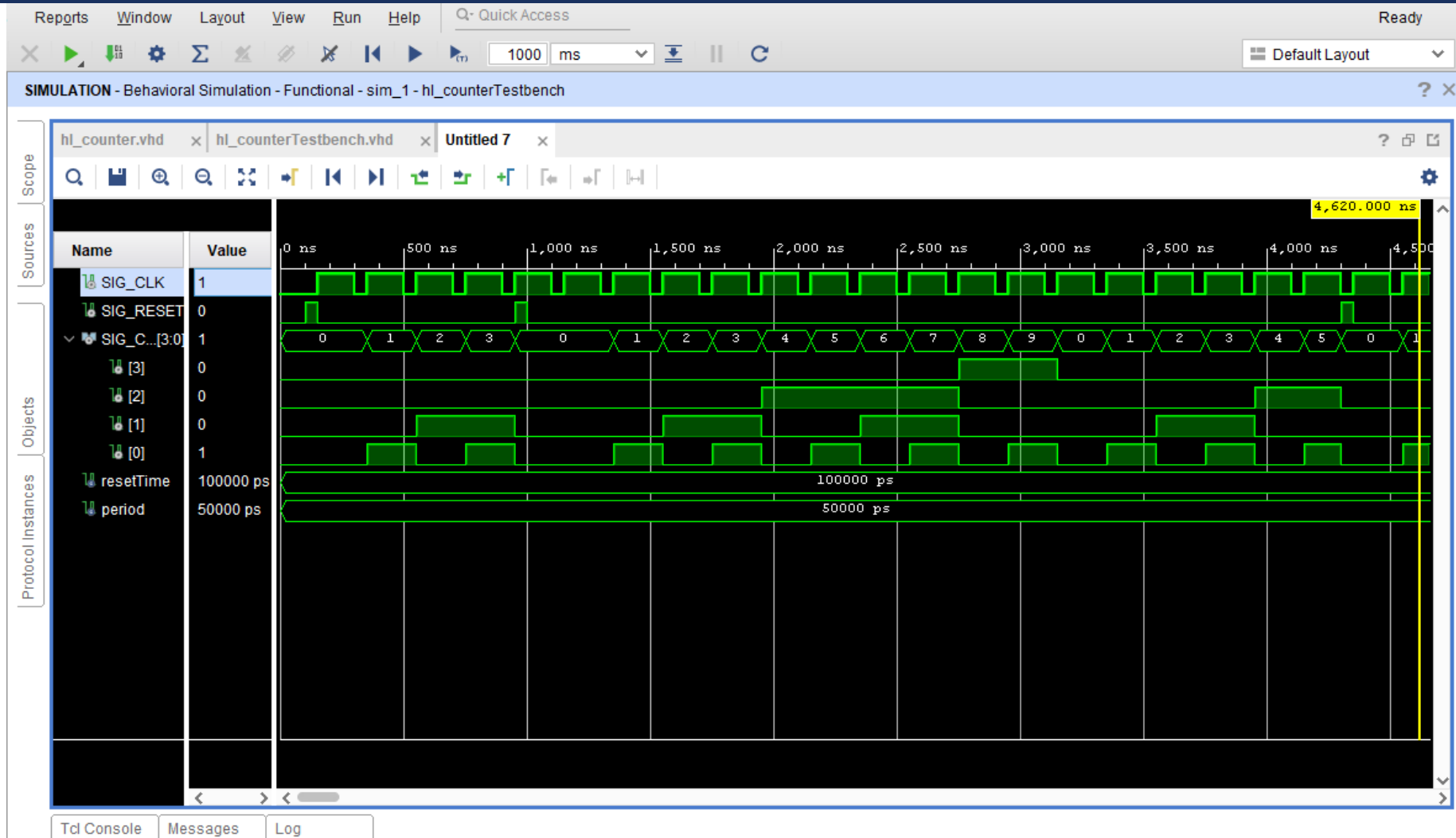
```
66      temp_count <= std_logic_vector(temp_count_internal);
67      --      END IF;
68      --      END IF;
69
70      --      count_out <= temp_count;
71  --END PROCESS;
72
73  countingFastForBehavSim : process(reset, clk, temp_count)
74  begin
75      if reset = '1' then
76          temp_count_internal <= "0000";
77          temp_count <= std_logic_vector(temp_count_internal);
78      elsif clk' event and clk = '1' then
79          if temp_count_internal < 9 then
80              temp_count_internal <= temp_count_internal + 1;
81              temp_count <= std_logic_vector(temp_count_internal);
82          else
83              temp_count_internal <= "0000";
84              temp_count <= std_logic_vector(temp_count_internal);
85          end if;
86      end if;
87      count_out <= temp_count;
88  end process;
89
90  END behav;
91
```

## SECTION 2 – COUNTER TESTBENCH VHDL CODE

```
30 -----
31
32
33 library IEEE;
34 use IEEE.STD_LOGIC_1164.ALL;
35
36 -- Uncomment the following library declaration if using
37 -- arithmetic functions with Signed or Unsigned values
38 --use IEEE.NUMERIC_STD.ALL;
39
40 -- Uncomment the following library declaration if instantiating
41 -- any Xilinx leaf cells in this code.
42 --library UNISIM;
43 --use UNISIM.VComponents.all;
44
45 entity hl_counterTestbench is
46 -- Port ( );
47 end hl_counterTestbench;
48
49 architecture Behavioral of hl_counterTestbench is
50 -- Component Declaration for the Unit Under Test (UUT):
51 component hl_counter
52     PORT(
53         clk : in STD_LOGIC;
54         reset : in STD_LOGIC;
55         count_out : out STD_LOGIC_VECTOR (3 downto 0)
56     );
57 end component;
58 -- Input Signals:
59 signal SIG_CLK : STD_LOGIC := '0';
60 signal SIG_RESET : STD_LOGIC := '0';
61 -- Output Signals:
62 signal SIG_COUNT_OUT : STD_LOGIC_VECTOR (3 downto 0) := "0000";
63 -- Reset Time:
64 constant resetTime : time := 100 ns;
65 -- Delta Time:
66 constant period : time := 50 ns;
67 begin
68 -- Under Unit Test (UUT) Instantiation:
69 uut : hl_counter PORT MAP(
70     clk => SIG_CLK,
71     reset => SIG_RESET,
72     count_out => SIG_COUNT_OUT
73 );
74 -- Stimulus Processes:
75 stimCLK : process
```

```
59 signal SIG_CLK : STD_LOGIC := '0';
60 signal SIG_RESET : STD_LOGIC := '0';
61 -- Output Signals:
62 signal SIG_COUNT_OUT : STD_LOGIC_VECTOR (3 downto 0) := "0000";
63 -- Reset Time:
64 constant resetTime : time := 100 ns;
65 -- Delta Time:
66 constant period : time := 50 ns;
67 begin
68 -- Under Unit Test (UUT) Instantiation:
69 uut : hl_counter PORT MAP(
70     clk => SIG_CLK,
71     reset => SIG_RESET,
72     count_out => SIG_COUNT_OUT
73 );
74 -- Stimulus Processes:
75 stimCLK : process
76 begin
77     wait for resetTime;
78     SIG_CLK <= '0';
79     wait for period;
80     SIG_CLK <= '1';
81     wait for period;
82 end process;
83 stimRESET : process
84 begin
85     wait for resetTime;
86     SIG_RESET <= '1';
87     wait for period;
88     SIG_RESET <= '0';
89     wait for period * 16;
90     SIG_RESET <= '1';
91     wait for period;
92     SIG_RESET <= '0';
93     wait for period * 64;
94 end process;
95 end Behavioral;
96
```

# SECTION 2 – COUNTER BEHAVIOURAL SIMULATION



# SECTION 2 – COUNTER SYNTHESIS

The screenshot displays the Xilinx Vivado IDE interface for a project named "IMPLEMENTED DESIGN - xc7a35ticsg324-1L". The top status bar indicates "Implementation Complete" with a green checkmark. The main workspace is divided into several panes:

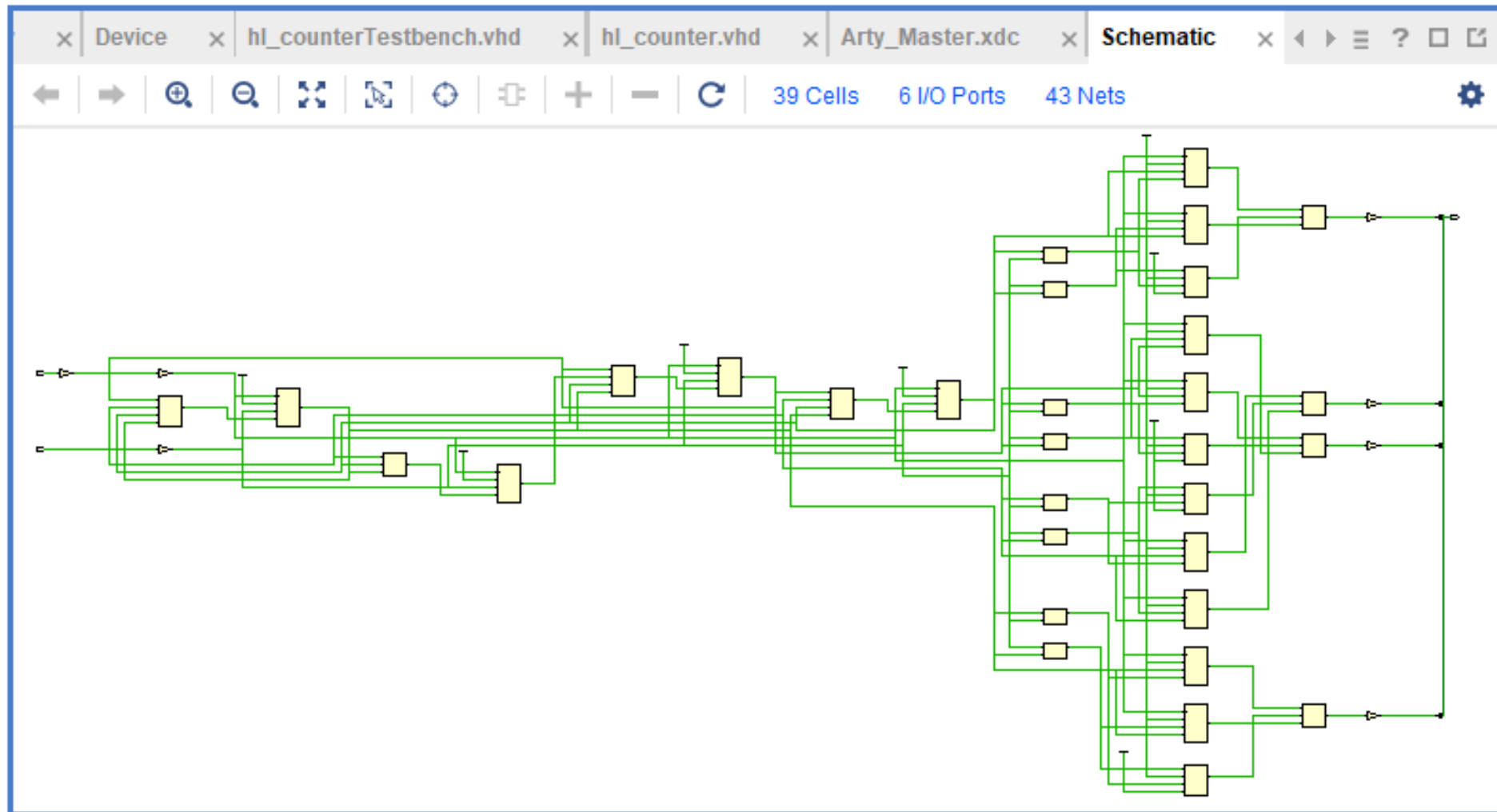
- Sources:** Shows the project hierarchy with "hl\_counter" containing "Nets (43)" and "Leaf Cells (41)".
- Source File Properties:** Displays properties for "Arty\_Master.xdc".
- Project Summary:** Shows the device "xc7a35ticsg324-1L" and the selected file "hl\_counterTestbench.vhd".
- Timing:** Displays the "Design Timing Summary" table.

The central pane shows a logic diagram with four flip-flops labeled X0Y2, X1Y2, X0Y1, and X1Y1, connected in a counter configuration. The bottom pane shows the "Design Timing Summary" table, which indicates that there are no user-specified timing constraints.

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): NA	Worst Hold Slack (WHS): NA	Worst Pulse Width Slack (WPWS): NA
Total Negative Slack (TNS): NA	Total Hold Slack (THS): NA	Total Pulse Width Negative Slack (TPWS): NA
Number of Failing Endpoints: NA	Number of Failing Endpoints: NA	Number of Failing Endpoints: NA
Total Number of Endpoints: NA	Total Number of Endpoints: NA	Total Number of Endpoints: NA

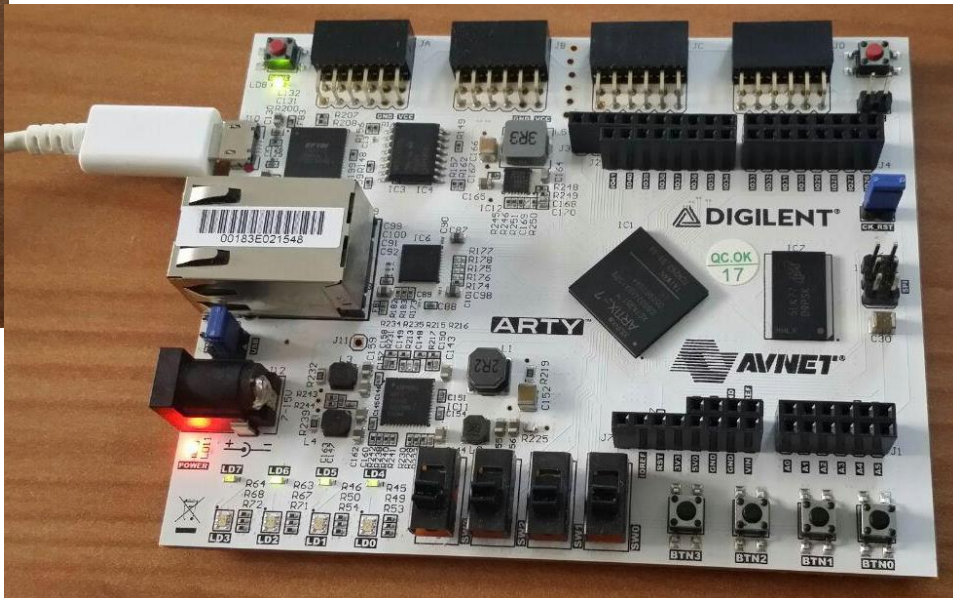
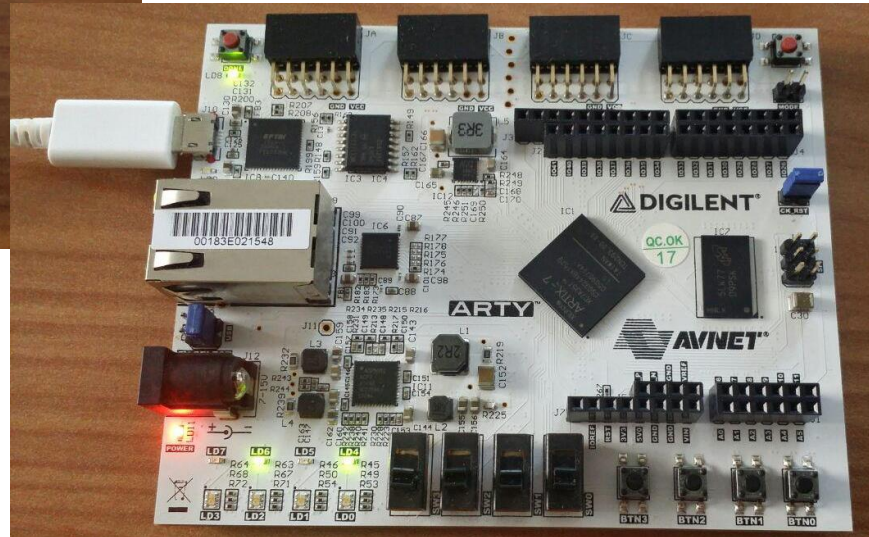
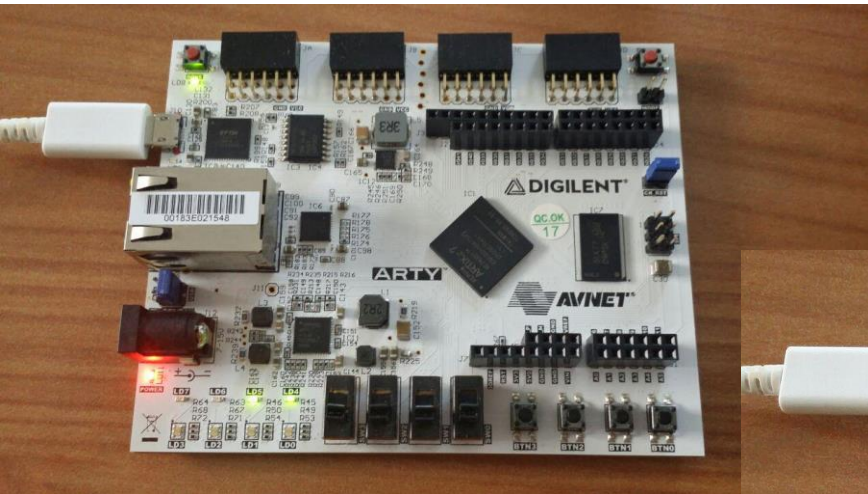
There are no user specified timing constraints.

## SECTION 2 – COUNTER SCHEMATIC





## SECTION 2 – COUNTER ON ARTY BOARD

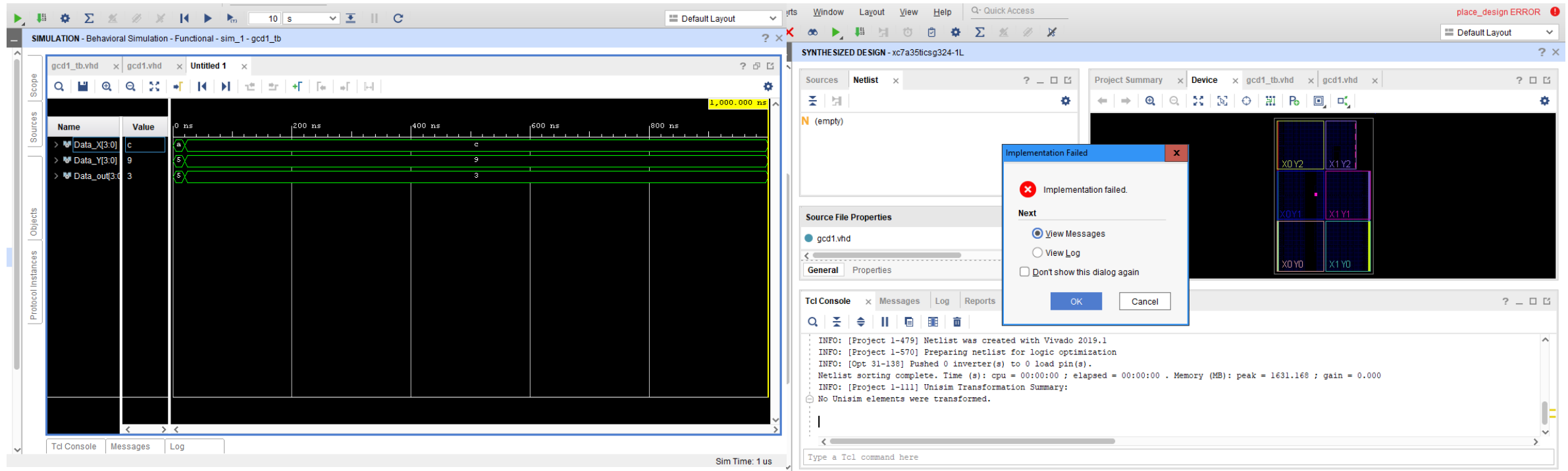


## SECTION 2 – SEQUENCE DETECTOR XDC FILE

```
5
6  ## Clock signal
7
8  set_property -dict { PACKAGE_PIN E3      IOSTANDARD LVCMOS33 } [get_ports { clock }]; #IO_L12P_T1_MRCC_35 Sch=gclk[100]
9  #create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports { CLK100MHZ }];
10
11  ##Switches
```

```
32
33  ##LEDs
34
35  set_property -dict { PACKAGE_PIN H5      IOSTANDARD LVCMOS33 } [get_ports { y }]; #IO_L24N_T3_35 Sch=led[4]
36  #set_property -dict { PACKAGE_PIN J5      IOSTANDARD LVCMOS33 } [get_ports { count_out[1] }]; #IO_25_35 Sch=led[5]
37  #set_property -dict { PACKAGE_PIN T9      IOSTANDARD LVCMOS33 } [get_ports { count_out[2] }]; #IO_L24P_T3_A01_D17_14 Sch=led[6]
38  #set_property -dict { PACKAGE_PIN T10     IOSTANDARD LVCMOS33 } [get_ports { count_out[3] }]; #IO_L24N_T3_A00_D16_14 Sch=led[7]
39
40  ##Buttons
41
42  set_property -dict { PACKAGE_PIN D9      IOSTANDARD LVCMOS33 } [get_ports { reset }]; #IO_L6N_T0_VREF_16 Sch=btn[0]
43  #set_property -dict { PACKAGE_PIN C9      IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #IO_L11P_T1_SRCC_16 Sch=btn[1]
44  #set_property -dict { PACKAGE_PIN B9      IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #IO_L11N_T1_SRCC_16 Sch=btn[2]
45  set_property -dict { PACKAGE_PIN B8      IOSTANDARD LVCMOS33 } [get_ports { x }]; #IO_L12P_T1_MRCC_16 Sch=btn[3]
46
47  ##Pmod Header JA
48
```

## SECTION 2 – BEHAVIOURAL GCD SIMULATION RESULTS

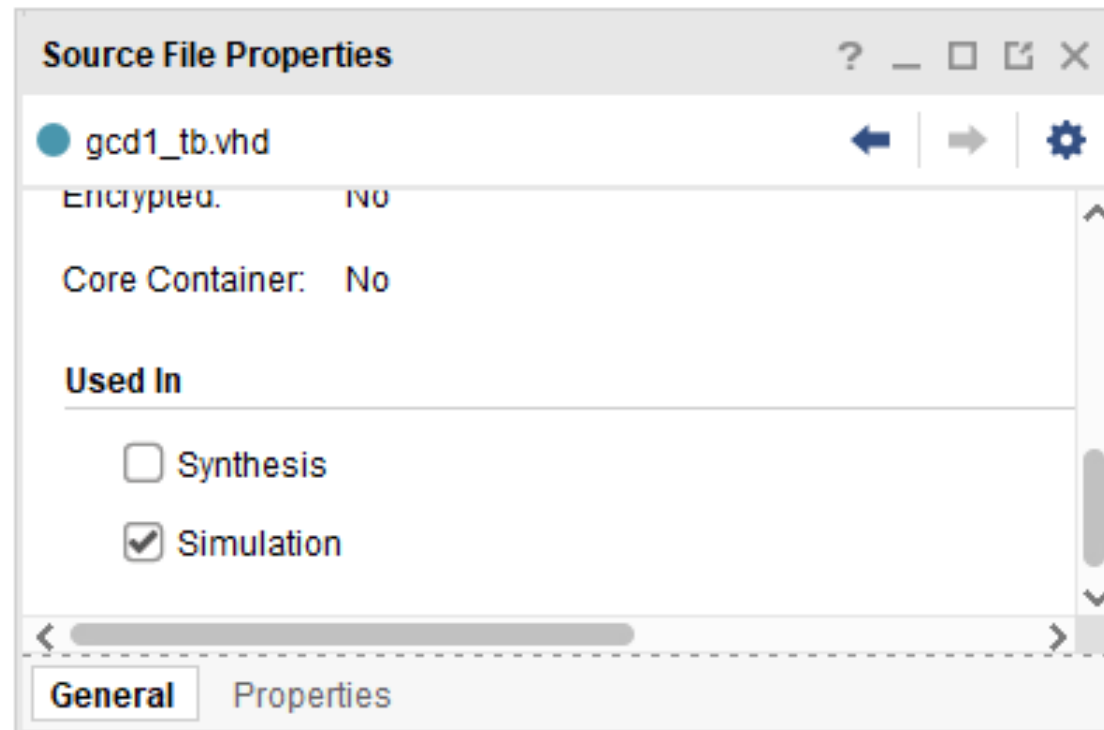


In a first time I tried to implement the material I had, I get an error so I thought that it was not implementable.

## SECTION 2 – BEHAVIOURAL GCD ERROR SOLUTION

I finally found out the solution to make it implementable:

- Telling Vivado not to get the Testbench file as a source for Synthesis (Pay Attention!!! It acts like this by default!)



## SECTION 2 – BEHAVIOURAL GCD SIMULATION RESULTS

You can even add Constraints (via XDC file) to select input and output peripherals, in order to run the Project on the Arty board, like that:

```
10
11  ##Switches
12
13  set_property -dict { PACKAGE_PIN A8      IOSTANDARD LVCMOS33 } [get_ports { Data_X[0] }]; #IO_L12N_T1_MRCC_16 Sch=sw[0]
14  set_property -dict { PACKAGE_PIN C11    IOSTANDARD LVCMOS33 } [get_ports { Data_X[1] }]; #IO_L13P_T2_MRCC_16 Sch=sw[1]
15  set_property -dict { PACKAGE_PIN C10    IOSTANDARD LVCMOS33 } [get_ports { Data_X[2] }]; #IO_L13N_T2_MRCC_16 Sch=sw[2]
16  set_property -dict { PACKAGE_PIN A10    IOSTANDARD LVCMOS33 } [get_ports { Data_X[3] }]; #IO_L14P_T2_SRCC_16 Sch=sw[3]
17
18  ##RGB LEDs
```

```
32
33  ##LEDs
34
35  set_property -dict { PACKAGE_PIN H5      IOSTANDARD LVCMOS33 } [get_ports { Data_out[0] }]; #IO_L24N_T3_35 Sch=led[4]
36  set_property -dict { PACKAGE_PIN J5      IOSTANDARD LVCMOS33 } [get_ports { Data_out[1] }]; #IO_25_35 Sch=led[5]
37  set_property -dict { PACKAGE_PIN T9      IOSTANDARD LVCMOS33 } [get_ports { Data_out[2] }]; #IO_L24P_T3_A01_D17_14 Sch=led[6]
38  set_property -dict { PACKAGE_PIN T10     IOSTANDARD LVCMOS33 } [get_ports { Data_out[3] }]; #IO_L24N_T3_A00_D16_14 Sch=led[7]
39
40  ##Buttons
41
42  set_property -dict { PACKAGE_PIN D9      IOSTANDARD LVCMOS33 } [get_ports { Data_Y[0] }]; #IO_L6N_T0_VREF_16 Sch=btn[0]
43  set_property -dict { PACKAGE_PIN C9      IOSTANDARD LVCMOS33 } [get_ports { Data_Y[1] }]; #IO_L11P_T1_SRCC_16 Sch=btn[1]
44  set_property -dict { PACKAGE_PIN B9      IOSTANDARD LVCMOS33 } [get_ports { Data_Y[2] }]; #IO_L11N_T1_SRCC_16 Sch=btn[2]
45  set_property -dict { PACKAGE_PIN B8      IOSTANDARD LVCMOS33 } [get_ports { Data_Y[3] }]; #IO_L12P_T1_MRCC_16 Sch=btn[3]
46
47  ##Dmod Header 2A
```

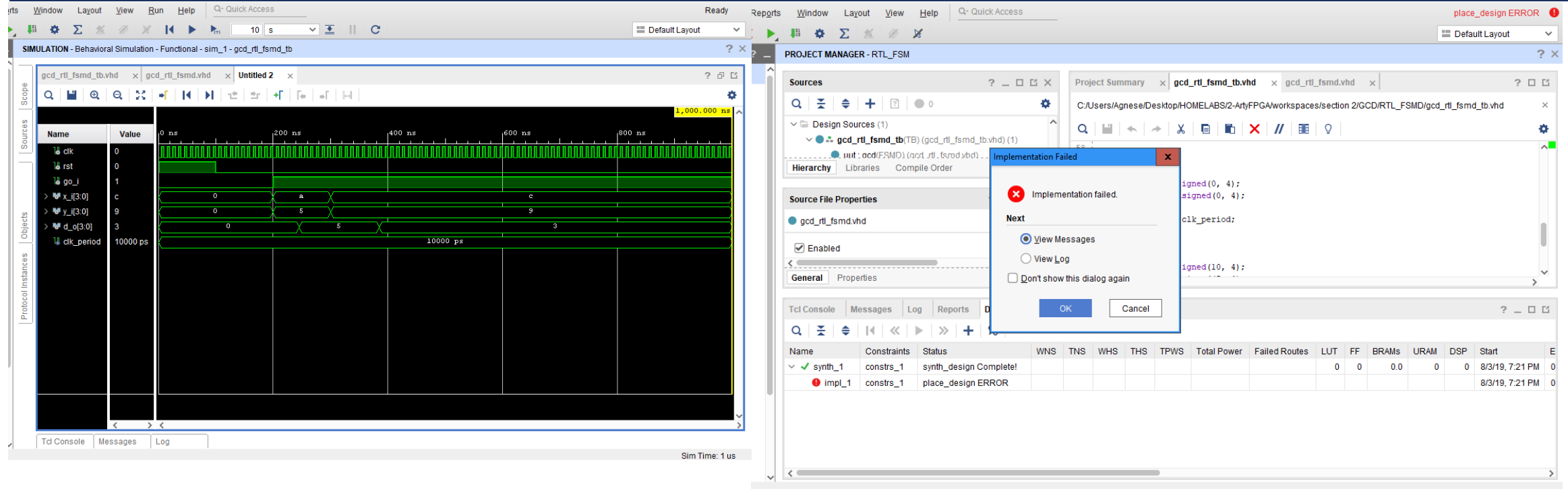


## SECTION 2 – BEHAVIOURAL GCD NEW IMPLEMENTATION

The screenshot displays the Xilinx Vivado IDE interface. The top menu bar includes 'ports', 'Window', 'Layout', 'View', and 'Help'. The 'Quick Access' toolbar is visible. The main window is titled 'SYNTHESIZED DESIGN - xc7a35ticsg324-1L'. The 'Netlist' tab is active, showing a hierarchy with 'gcd1' containing 'Nets (48)' and 'Leaf Cells (40)'. The 'Schematic' tab is also visible, showing a circuit diagram with 38 Cells, 12 I/O Ports, and 48 Nets. A dialog box titled 'Implementation Completed' is open, indicating 'Implementation successfully completed.' and offering options: 'Open Implemented Design' (selected), 'Generate Bitstream', 'View Reports', and 'Don't show this dialog again'. The 'Properties' panel is empty. The 'Tcl Console' panel shows the following table:

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start
✓ synth_1	constrs_1	synth_design Complete!								14	0	0.0	0	0	8/8/19, 5:16 P
✓ impl_1	constrs_1	route_design Complete!	NA	NA	NA	NA	NA	0.607	0	14	0	0.0	0	0	8/8/19, 5:18 P

## SECTION 2 – RTL WITH FSM GCD SIMULATION RESULTS

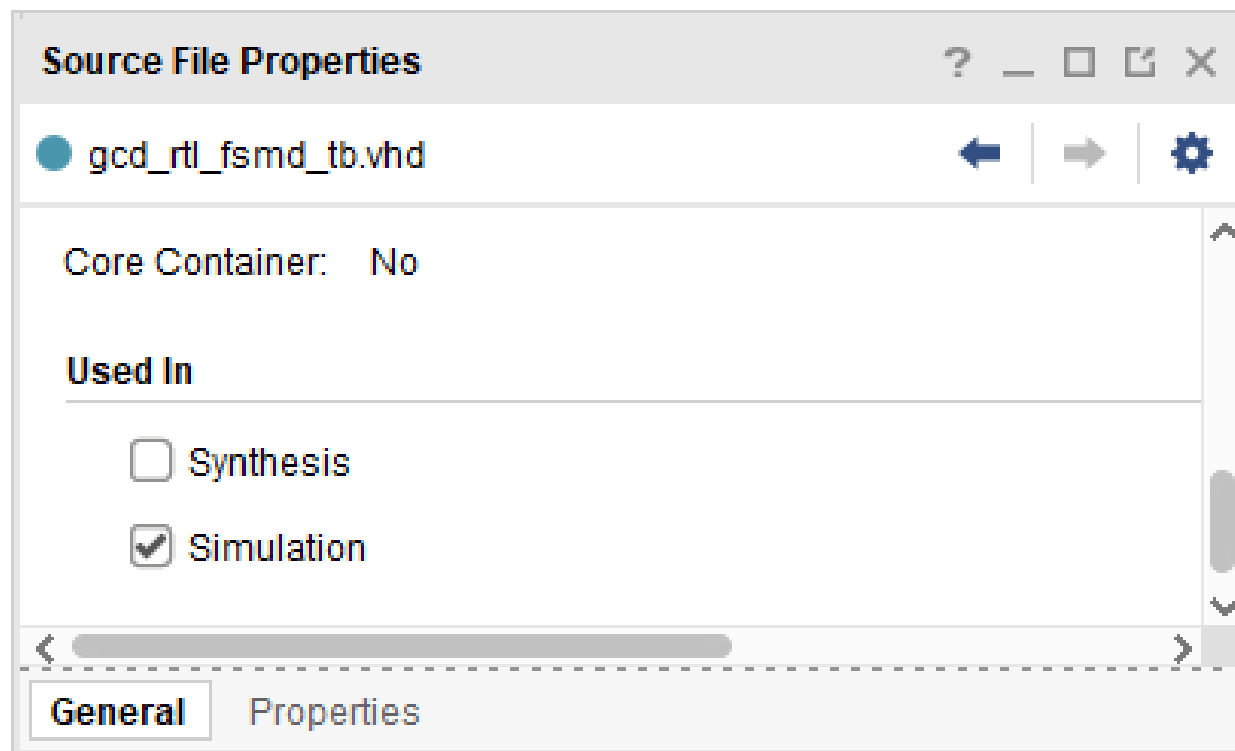


In a first time I tried to implement the material I had, I get an error so I thought that it was not implementable.

## SECTION 2 – RTL WITH FSM GCD ERROR SOLUTION

I finally found out the solution (like in the previous case):

- Telling Vivado not to get the Testbench file as a source for Synthesis (Pay Attention!!! It acts like this by default!)



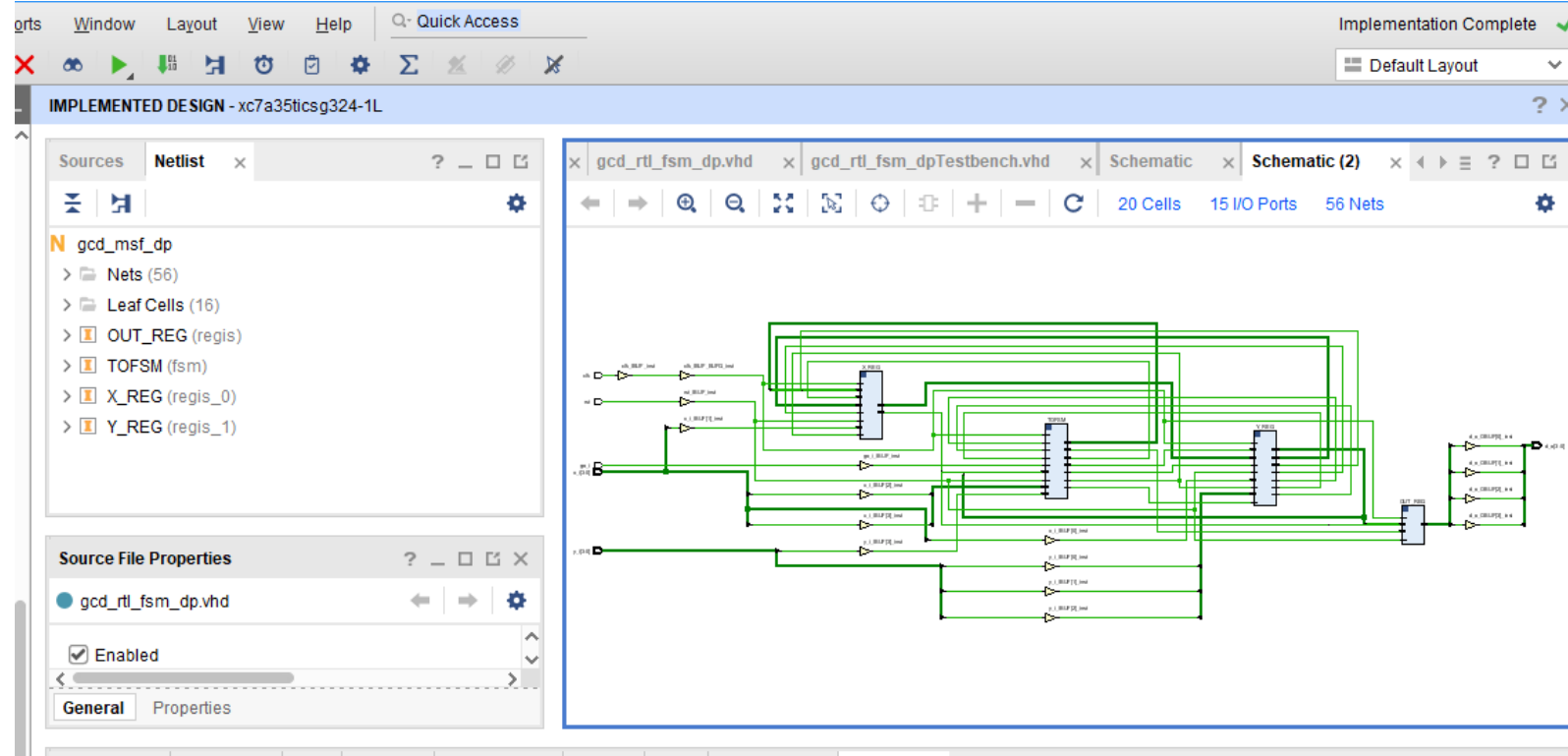


## SECTION 2 – RTL WITH FSM GCD NEW IMPLEMENTATION

The screenshot displays the Vivado IDE interface during the implementation phase. The top status bar indicates "Implementation Complete" with a green checkmark. The main window is titled "SYNTHESIZED DESIGN - xc7a35ticsg324-1L". The "Sources" pane on the left shows the "Netlist" tab. A dialog box titled "Implementation Completed" is open, displaying the message "Implementation successfully completed." and offering three options: "Open Implemented Design" (selected), "Generate Bitstream", and "View Reports". There is also a checkbox for "Don't show this dialog again". The "Schematic" tab on the right shows a complex netlist diagram with green lines representing connections between components. The bottom console window shows the following log messages:

```
INFO: [Project 1-479] Netlist was created with Vivado 2019.1
INFO: [Project 1-570] Preparing netlist for logic optimization
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 1836.602 ; gain = 0.000
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.
```

# SECTION 2 – RTL WITH FSM AND DATAPATH GCD SIMULATION RESULTS



It's implementable.



# SECTION 3

PICOBLAZE



## SECTION 3 – TO DO

In this section, I had to:

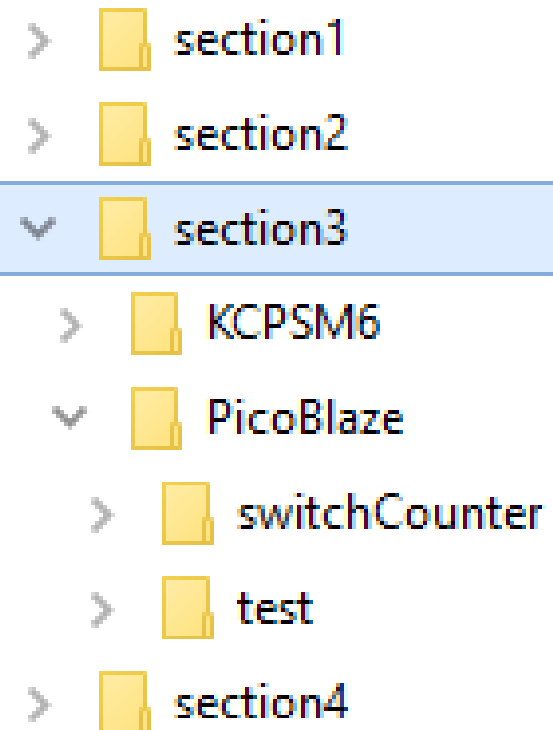
- Perform a test:
  - Write a given Assembly program on a psm file
  - Generate the ROM VHD file, using the kcpasm6.exe (downloaded with PicoBlaze material) and import it in a Vivado Project
  - Add given embedded\_pico.vhd file
  - Add Constraints (on XDC file) as needed
  - Synthesize and run the project on the Arty Board
- Do the same steps with an Assembly program written by me:
  - Switch Counter Project: it counts the number of switches that are switched on (the 1 bits in input register) and shows the output via leds

## SECTION 3 – WORKSPACE

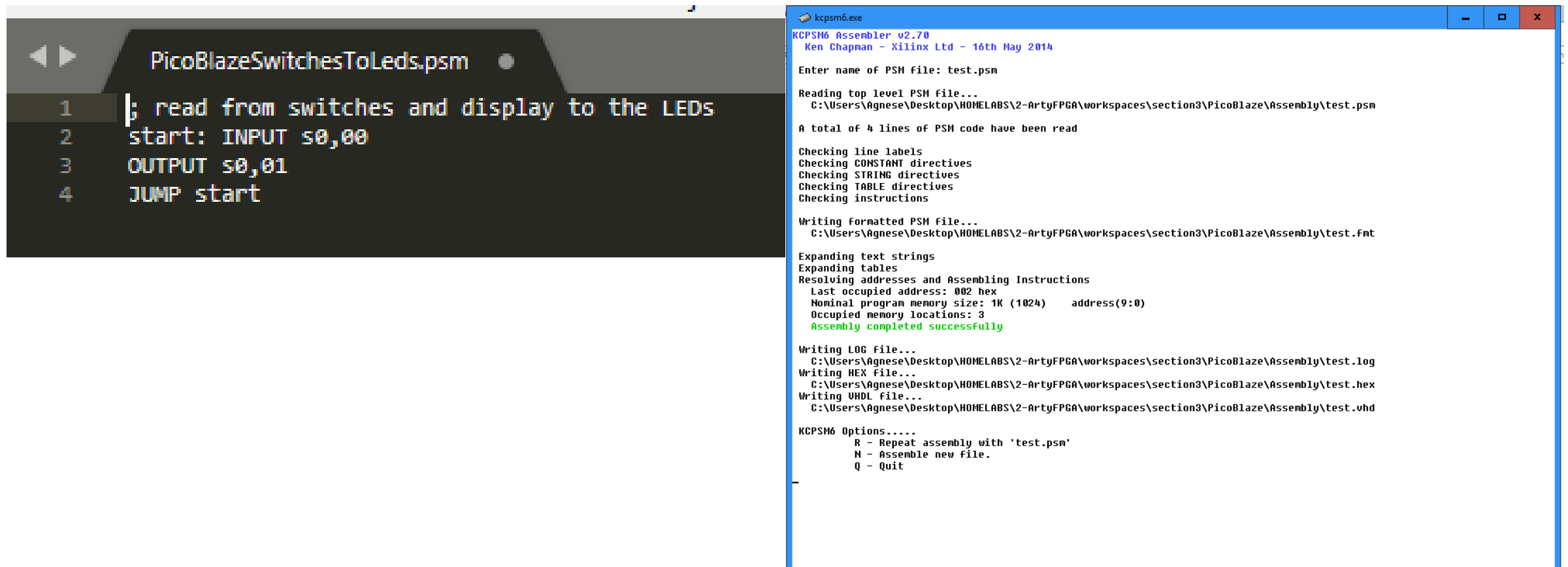
I organized this section in two sub-sections:

- Test
- Switch Counter

KPCSM6 contains downloaded PicoBlaze material.



## SECTION 3 – TEST ASSEMBLY CODE AND KCPSM6.EXE OUTPUT



The image shows two side-by-side windows. The left window displays the assembly code for 'PicoBlazeSwitchesToLeds.psm'. The right window shows the output of the 'kcpsm6.exe' assembler.

```
PicoBlazeSwitchesToLeds.psm
1  ; read from switches and display to the LEDs
2  start: INPUT s0,00
3  OUTPUT s0,01
4  JUMP start
```

```
kcpsm6.exe
KCPSM6 Assembler v2.70
Ken Chapman - Xilinx Ltd - 16th May 2014

Enter name of PSM File: test.psm

Reading top level PSM file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\Assembly\test.psm

A total of 4 lines of PSM code have been read

Checking line labels
Checking CONSTANT directives
Checking STRING directives
Checking TABLE directives
Checking instructions

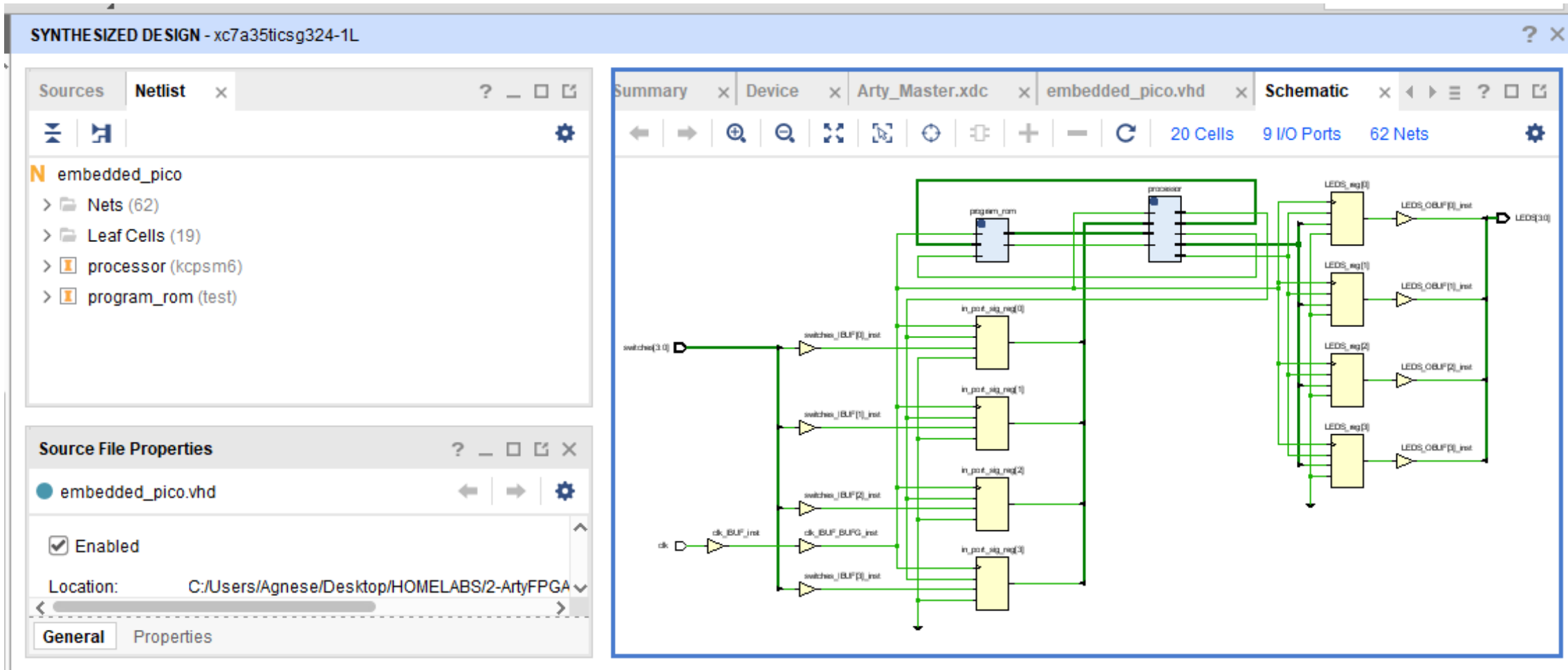
Writing formatted PSM file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\Assembly\test.fmt

Expanding text strings
Expanding tables
Resolving addresses and Assembling Instructions
Last occupied address: 002 hex
Nominal program memory size: 1K (1024) address(9:0)
Occupied memory locations: 3
Assembly completed successfully

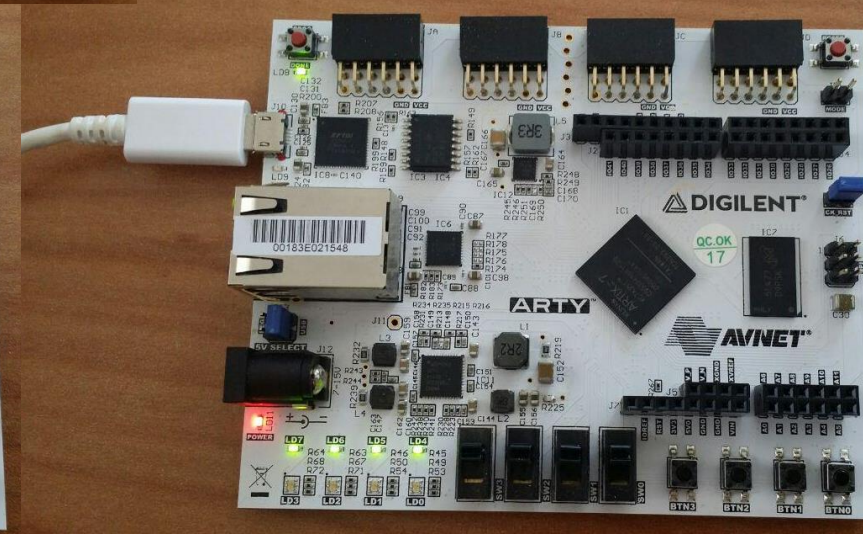
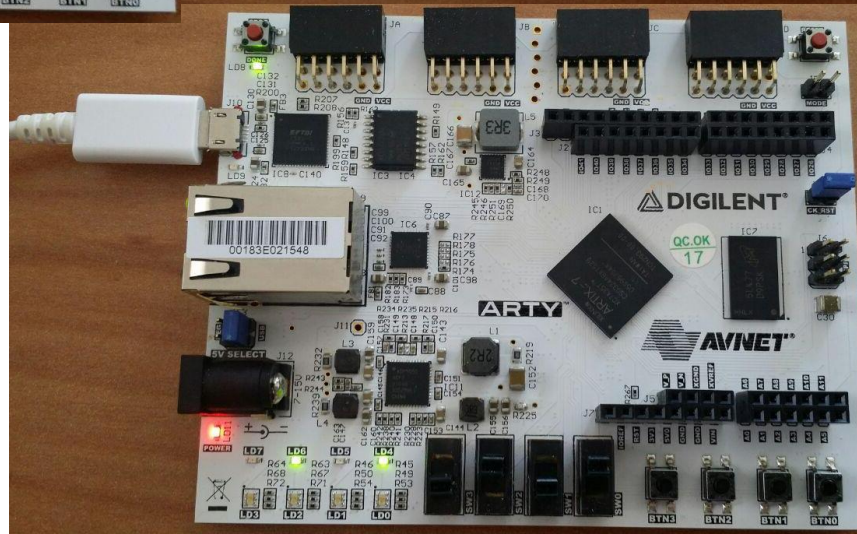
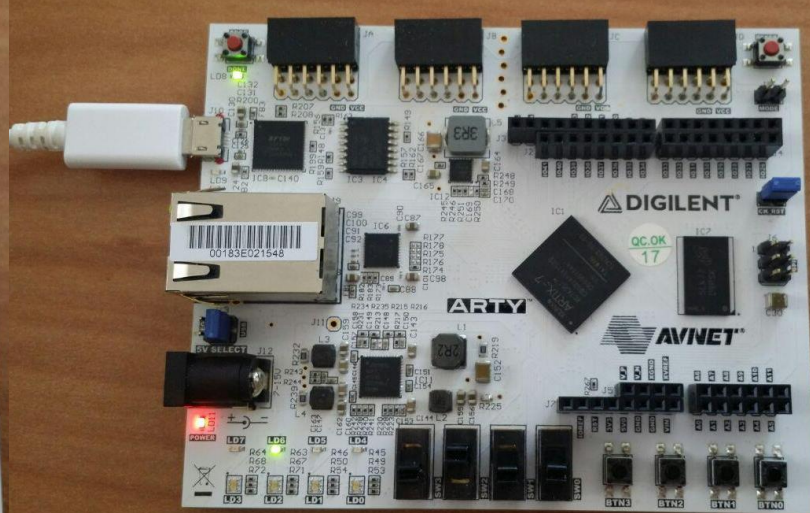
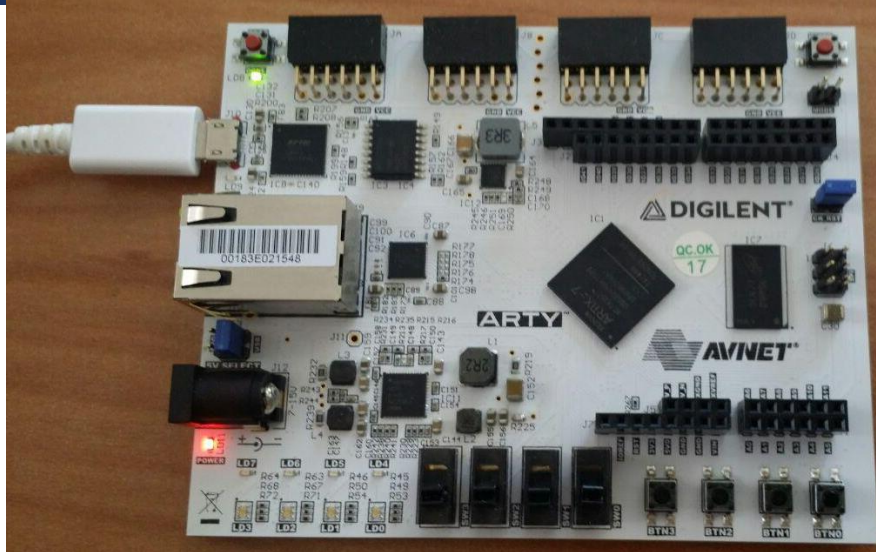
Writing LOG file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\Assembly\test.log
Writing HEX file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\Assembly\test.hex
Writing VHDL file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\Assembly\test.vhd

KCPSM6 Options.....
R - Repeat assembly with 'test.psm'
N - Assemble new file.
Q - Quit
```

## SECTION 3 – TEST SCHEMATIC



## SECTION 3 – TEST ON ARTY BOARD





# SECTION 3 – SWITCH COUNTER ASSEMBLY CODE AND KCPSM6.EXE OUTPUT

```
38
39 ;=====
40 ;== MAIN APPLICATION CODE
41 ; Count the number of the switches that are switched on and display the result to the LEDs
42 main:
43     ; Prepare arguments passed through registers
44     INPUT s0, 00 ; Input Register
45     load s1, 00 ; 1 bits number
46     do_while:
47         OUTPUT s1, 01 ; Output Register
48         compare s0, 00 ; See if there are still 1 bits to count in the Input Register
49         jump Z, main ; There are no more 1 bits to count
50         sl0 s0
51         jump C, one_detected
52         ; There is no 0S to return to, so the main program typically loops over itself
53         jump do_while
54     one_detected:
55         add s1, 01 ; Increment 1 bits number
56         jump do_while
57
58 ;=====
```

```
kcpsm6.exe
KCPSM6 Assembler v2.70
Ken Chapman - Xilinx Ltd - 16th May 2014

Enter name of PSM file: switchCounter.psm

Reading top level PSM file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\switchCounter\Assembly\switchCounter.psm

A total of 56 lines of PSM code have been read

Checking line labels
Checking CONSTANT directives
Checking STRING directives
Checking TABLE directives
Checking instructions

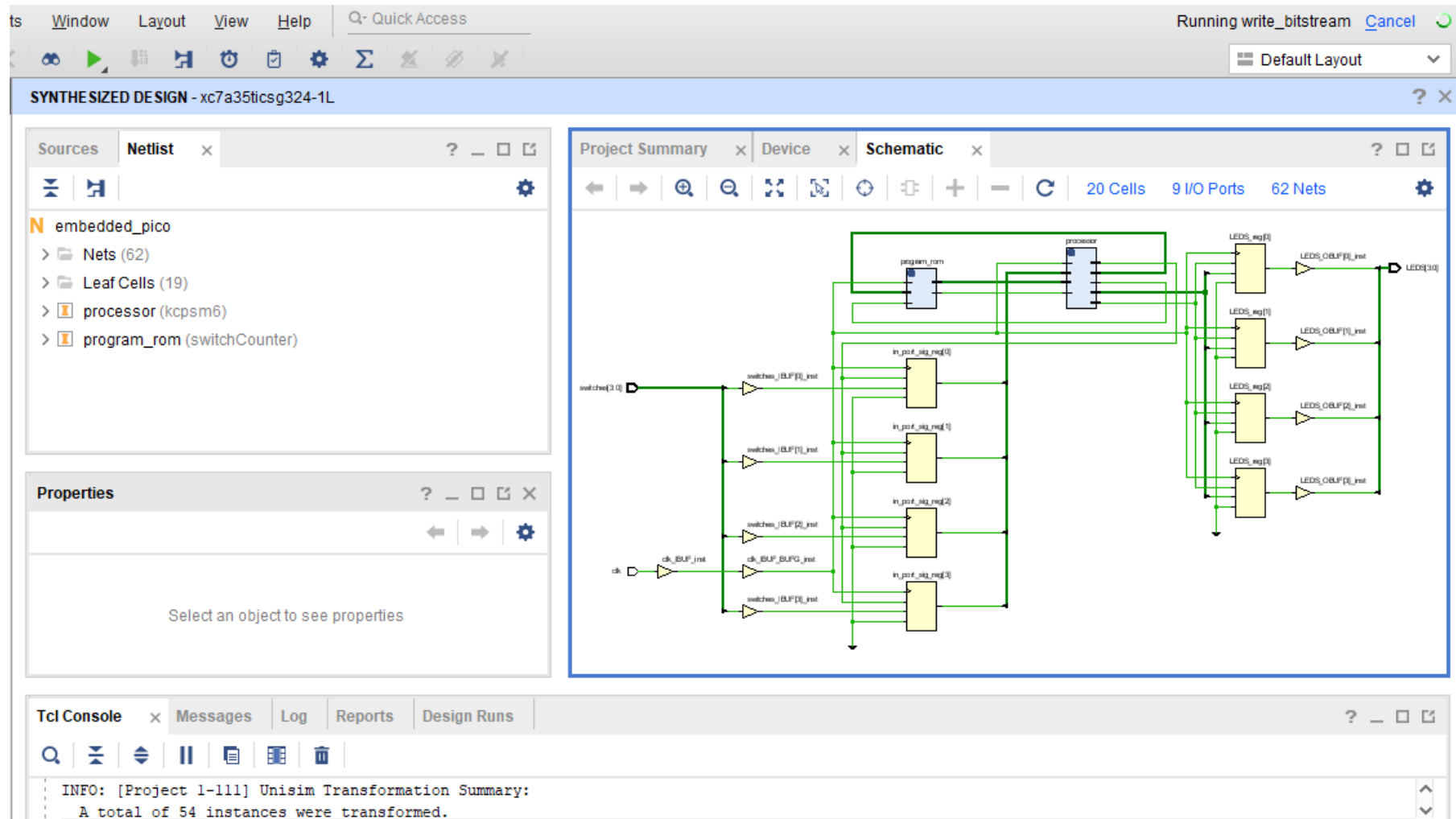
Writing formatted PSM file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\switchCounter\Assembly\switchCounter.fmt

Expanding text strings
Expanding tables
Resolving addresses and Assembling Instructions
Last occupied address: 009 hex
Nominal program memory size: 1K (1024) address(9:0)
Occupied memory locations: 10
Assembly completed successfully

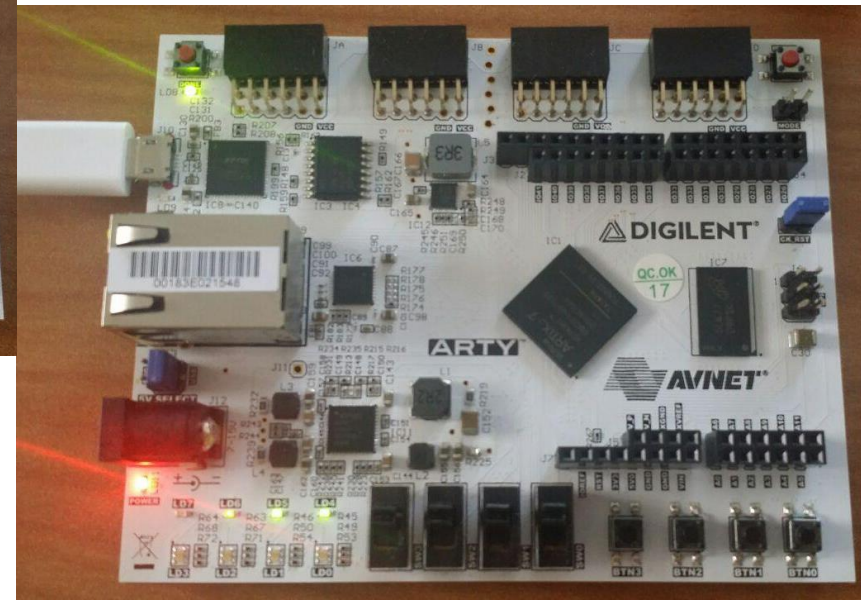
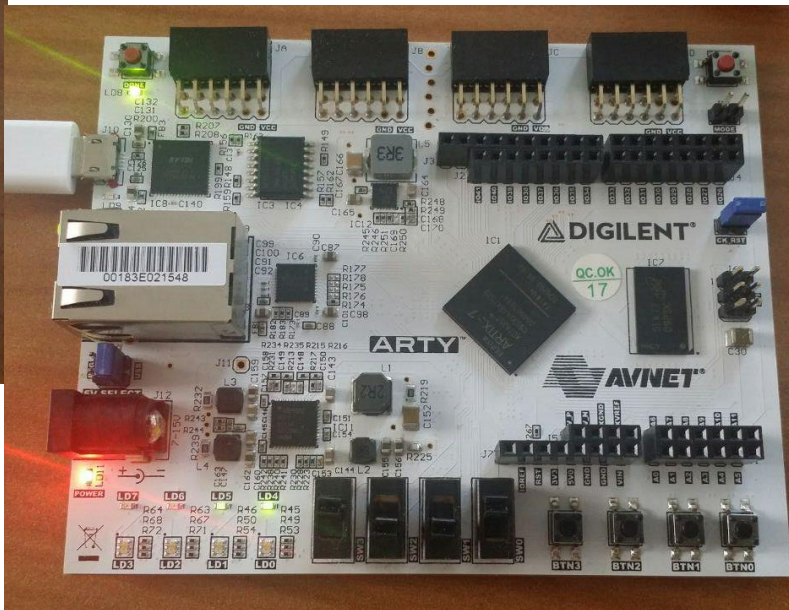
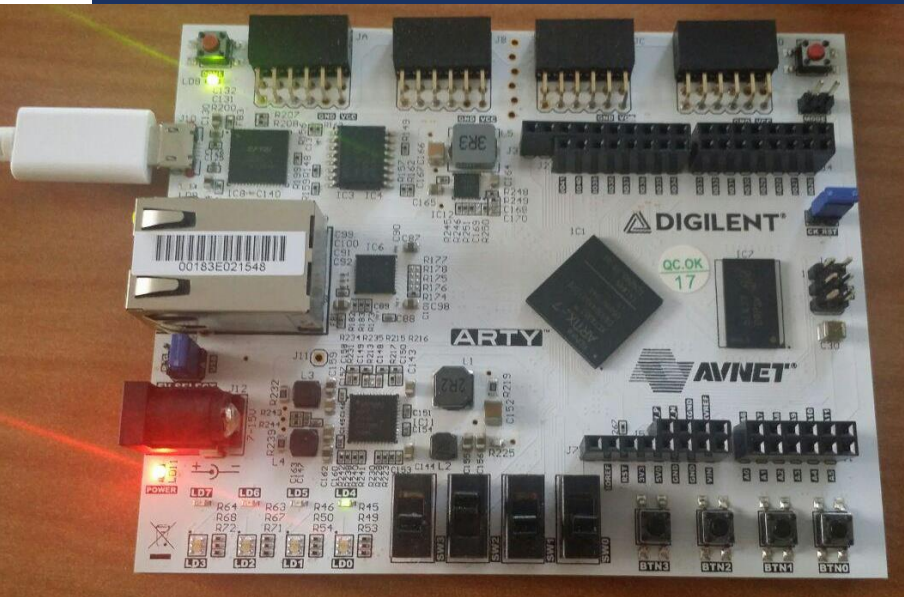
Writing LOG file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\switchCounter\Assembly\switchCounter.log
Writing HEX file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\switchCounter\Assembly\switchCounter.hex
Writing VHDL file...
C:\Users\Agnese\Desktop\HOMELABS\2-ArtyFPGA\workspaces\section3\PicoBlaze\switchCounter\Assembly\switchCounter.vhd

KCPSM6 Options....
R - Repeat assembly with 'switchCounter.psm'
N - Assemble new file.
Q - Quit
```

# SECTION 3 – SWITCH COUNTER SCHEMATIC



## SECTION 3 – SWITCH COUNTER ON ARTY BOARD





# SECTION 4

SERIAL COMMUNICATION



## SECTION 4 – TO DO

In this last section, I had to:

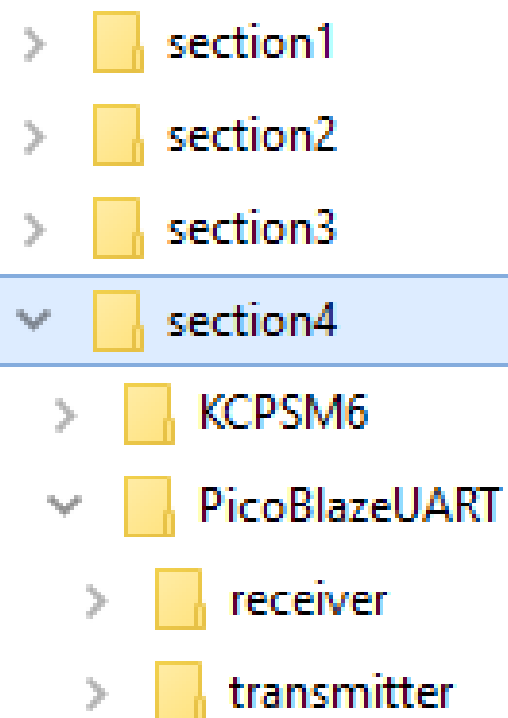
- Given the a test Transmitter:
  - Add proper Constraints (in XDC file)
  - Run it on the Arty Board
- Implement a Receiver and run it on the Arty Board

## SECTION 4 – WORKSPACE

I organized this section in two sub-sections:

- Transmitter
- Receiver

KPCSM6 contains downloaded PicoBlaze material.



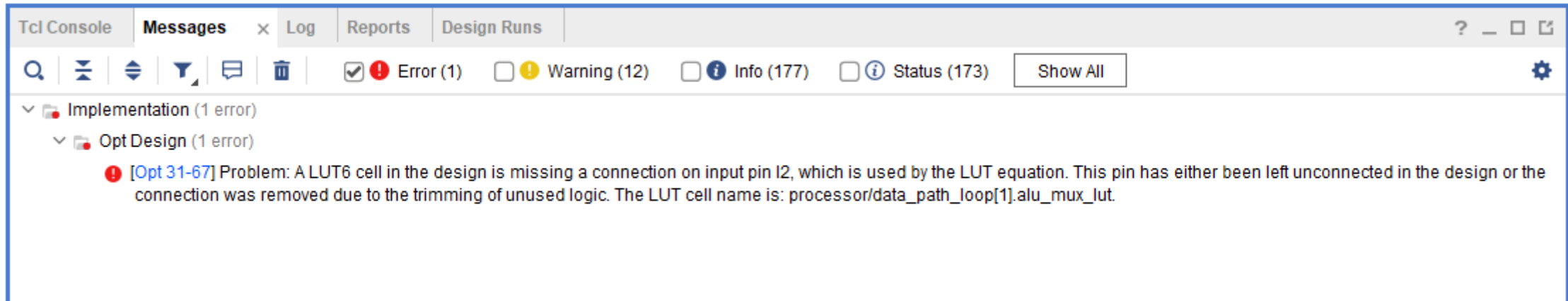
## SECTION 4 – TRANSMITTER XDC FILE

```
5
6  ## Clock signal
7
8  set_property -dict { PACKAGE_PIN E3      IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L12P_T1_MRCC_35 Sch=gclk[100]
9  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports { clk }];
10 #set_property DONT_TOUCH TRUE [get_cells processor/data_path_loop[1].alu_mux_lut];
11
12 ##Switches
```

```
91
92  ##USB-UART Interface
93
94  set_property -dict { PACKAGE_PIN D10     IOSTANDARD LVCMOS33 } [get_ports { txd }]; #IO_L19N_T3_VREF_16 Sch=uart_rxd_out
95  #set_property -dict { PACKAGE_PIN A9      IOSTANDARD LVCMOS33 } [get_ports { txd }]; #IO_L14N_T2_SRCC_16 Sch=uart_txd_in
96
97  ##ChirpKit Single Ended Analog Inputs
```

## SECTION 4 – TRANSMITTER VIVADO ERROR

I synthesized the given transmitter correctly, but during implementation some error occurred:



I tried to use different constraints for clock and txd port in XDC file, but error didn't change.

I tried to forbid Vivado to change logic components (for optimization), but it was the same.



## SECTION 4 – RECEIVER ASSEMBLY CODE

```
1 ; Software implemented UART Receiver
2 ;
3 ; Receives one character into 's5' at 115200 baud with 1 start bit, 1 stop bit
4 ; parity. All timing is based on a 50MHz clock where each bit period is equivalent
5 ; instructions. A valid character is signified by Z=0 and C=0. A timeout (~51us)
6 ; character is invalid then Z=0 and C=1.
7 ;
8 ; Registers used s0, s1, s2 and s5.
9 ;
10
11 CONSTANT UART_input_port, 04 ; Receive serial data
12 CONSTANT UART_output_port, 10 ; Transmit serial data
13 CONSTANT serial_data, 00000001'b ; bit0 - serial data
14 LOAD s0, serial_data
15 OUTPUT s0, UART_output_port ; initialise serial output
16
17 UART_RX:
18     LOAD s1, 255'd ; Detect beginning of start bit (0) or timeout
19 rx_timeout:
20     INPUT s0, UART_input_port ; 255 x 5 = 1275 instructions or ~51us.
21     TEST s0, serial_data ; test serial input for change to '0'
22     JUMP Z, start_bit
23     SUB s1, 1'd
24     JUMP NZ, rx_timeout
25     RETURN ; Timeout returns with Z=1 and C=0
26 ;
27
28 RETURN ; Timeout returns with Z=1 and C=0
29
30 ;
31 start_bit:
32     LOAD s1, 51'd ; Wait until middle of start bit
33 mid_start_delay: SUB s1, 1'd ; 51 x 2 = 102 instruction delay
34     JUMP NZ, mid_start_delay
35     INPUT s0, UART_input_port ; test for start bit = '0'
36     SR1 s0 ; shift start bit into carry flag and force Z=0
37     RETURN C ; Will abort with C=1 and Z=0 if start bit was High
38 ;
39 LOAD s2, 08 ; 8 bits to receive
40 RX_loop:
41     LOAD s1, 105'd ; Loop delay is (105 x 2) + 6 = 216 instructions
42 rx_bit_delay:
43     SUB s1, 1'd
44     JUMP NZ, rx_bit_delay
45     INPUT s0, UART_input_port ; sample data bit at mid-point
46     SR0 s0 ; move data bit into carry flag
47     SRA s5 ; Shift data bit into 's5' LSB first
48     SUB s2, 1'd ; count 8 bits
49     JUMP NZ, RX_loop
50 ;
51 ; Finally wait one more bit period and sample the stop bit which should be High.
52 ; If it is Low then set carry flag to indicate error. But if it is High the
53 ; character is good and the return must be made with Z=0.
54 ;
55 stop_bit:
56     LOAD s1, 106'd ; Wait until middle of stop bit
57 stop_bit_delay: SUB s1, 1'd ; (106 x 2) + 5 = 217 instructions
58     JUMP NZ, stop_bit_delay
59     INPUT s0, UART_input_port ; test for stop bit = '1'
60     XOR s0, serial_data ; invert bit so that correct value for carry flag
61     SR1 s0 ; shift inverted bit into carry flag force Z=0
62     RETURN ; For good character return with Z=0 and C=0
```

## SECTION 4 – RECEIVER EMBEDDED\_PICO.VHD

```
8  entity embedded_pico is
9  port (
10     clk: in std_logic;
11     rxd: in std_logic
12 );
13
14 end embedded_pico;
15
16
17 architecture behavioral of embedded_pico is
18
19     component kcpsm6
20     generic(
21         hwbuild : std_logic_vector(7 downto 0) := X"00";
22         interrupt_vector : std_logic_vector(11 downto 0) := X"3FF";
23         scratch_pad_memory_size : integer := 64;
24     port (
25         address : out std_logic_vector(11 downto 0);
26         instruction : in std_logic_vector(17 downto 0);
27         bram_enable : out std_logic;
28         in_port : in std_logic_vector(7 downto 0);
29         out_port : out std_logic_vector(7 downto 0);
30         port_id : out std_logic_vector(7 downto 0);
31         write_strobe : out std_logic;
32         k_write_strobe : out std_logic;
33         read_strobe : out std_logic;
34         interrupt : in std_logic;
35         interrupt_ack : out std_logic;
36         sleep : in std_logic;
37         reset : in std_logic;
38         clk : in std_logic;
39     end component;
40
41     component receiver
42     generic(
43         C_FAMILY : string := "S6";
44         C_RAM_SIZE_KWORDS : integer := 1;
45         C_JTAG_LOADER_ENABLE : integer := 0;
46     port (
47         address : in std_logic_vector(11 downto 0);
48         instruction : out std_logic_vector(17 downto 0);
49         enable : in std_logic;
```

```
47     instruction : out std_logic_vector(17 downto 0);
48     enable : in std_logic;
49     rdl : out std_logic;
50     clk : in std_logic;
51 end component;
52
53     component uart_rx6
54     port (
55         serial_in : in std_logic;
56         en_16_x_baud : in std_logic;
57         data_out : out std_logic_vector(7 downto 0);
58         buffer_read : in std_logic;
59         buffer_data_present : out std_logic;
60         buffer_half_full : out std_logic;
61         buffer_full : out std_logic;
62         buffer_reset : in std_logic;
63         clk : in std_logic
64     );
65 end component;
66
67     signal address : std_logic_vector(11 downto 0);
68     signal instruction : std_logic_vector(17 downto 0);
69     signal bram_enable : std_logic;
70     signal k_write_strobe : std_logic;
71     signal kcpsm6_sleep : std_logic;
72     signal kcpsm6_reset : std_logic;
73     signal en_16_x_baud : std_logic;
74     signal interrupt_ack : std_logic;
75     signal interrupt : std_logic;
76     signal pb_in_data : std_logic_vector(7 downto 0);
77     signal pb_out_data : std_logic_vector(7 downto 0);
78     signal uart_write : std_logic;
79     signal baud_count : integer range 0 to 162 := 0;
80     signal uart_rx : std_logic;
81     signal uart_rx_data_out : std_logic_vector(7 downto 0);
82     signal read_from_uart_rx : std_logic;
83     signal uart_rx_data_present : std_logic;
84
85     --
86
87
88     begin
89
90
91     processor: kcpsm6
92     generic map (
93         hwbuild => X"00",
```

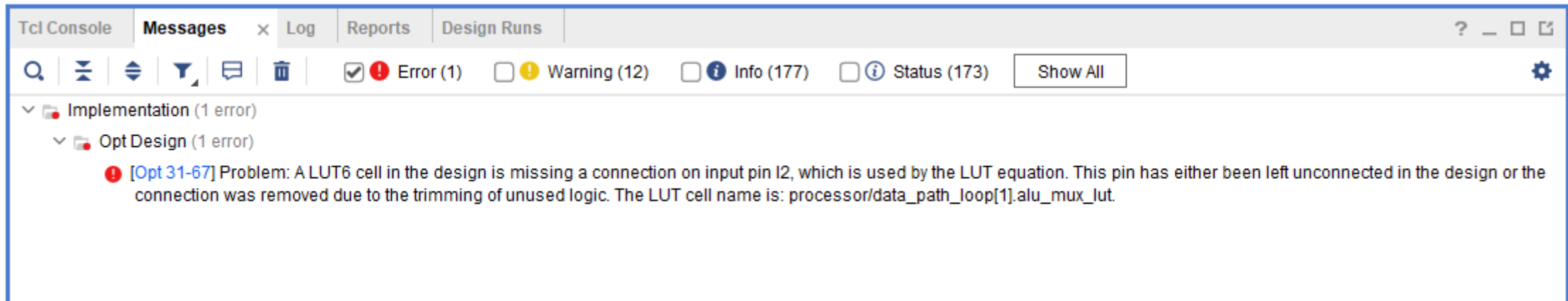
## SECTION 4 – RECEIVER EMBEDDED\_PICO.VHD

```
86 --
87
88
89 ▼ begin
90
91
92 ▼ processor: kcpsm6
93     generic map (
94         hwbuild => X"00",
95         interrupt_vector => X"3FF",
96         scratch_pad_memory_size => 64)
97     port map(
98         address => address,
99         instruction => instruction,
100         bram_enable => bram_enable,
101         port_id => open,--
102         write_strobe => uart_write,
103         k_write_strobe => k_write_strobe,
104         out_port => pb_out_data,
105         read_strobe => read_from_uart_rx,
106         in_port=> pb_in_data,
107         interrupt => '0',
108         interrupt_ack => open,
109         sleep => kcpsm6_sleep,
110         reset => '0',--
111         clk => clk);
112
113 kcpsm6_sleep <= '0';
114
115
116 ▼ program_rom: receiver
117     generic map(
118         C_FAMILY => "7S",
119         C_RAM_SIZE_KWORDS => 2,
120         C_JTAG_LOADER_ENABLE => 1)
121     port map(
122         address => address,
123         instruction => instruction,
124         enable => bram_enable,
125         rd1 => kcpsm6_reset,
126         clk => clk);
```

```
123         rd1 => kcpsm6_reset,
124         clk => clk);
125
126
127 rx: uart_rx6
128 port map (
129     serial_in => rxd,
130     en_16_x_baud => en_16_x_baud,
131     data_out => uart_rx_data_out ,
132     buffer_read => read_from_uart_rx,
133     buffer_data_present => uart_rx_data_present,
134     buffer_half_full => pb_in_data(0),
135     buffer_full => pb_in_data(7),
136     buffer_reset => kcpsm6_reset,
137     clk => clk
138 );
139
140
141 -- UART Baudrate timer
142 ▼ baud_timer: PROCESS (clk)
143 BEGIN
144     IF clk'event and clk= '1' THEN
145         IF baud_count= 162 THEN
146             baud_count<= 0;
147             en_16_x_baud <= '1';
148         ELSE
149             baud_count<= baud_count+ 1;
150             en_16_x_baud <= '0';
151         END IF;
152     END IF;
153 END PROCESS baud_timer;
154
155
156
157 end behavioral;
158
159
```

## SECTION 4 – RECEIVER VIVADO ERROR

I tried to analyze the previous error by using Xilinx receiver code, but the problem was just the same:



I tried again to use different constraints for clock and txd port in XDC file and to avoid Vivado to change logic components, but error didn't change.





THANKS FOR  
YOUR  
ATTENTION