ARTYLAB

ARTY FPGA VHDL/ASSEMBLY EXAMPLES

Embedded Systems Project

Agnese Salutari, 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, using Xilinx Artix-35T FPGA (xc7a35ticsg324-1L) 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 1

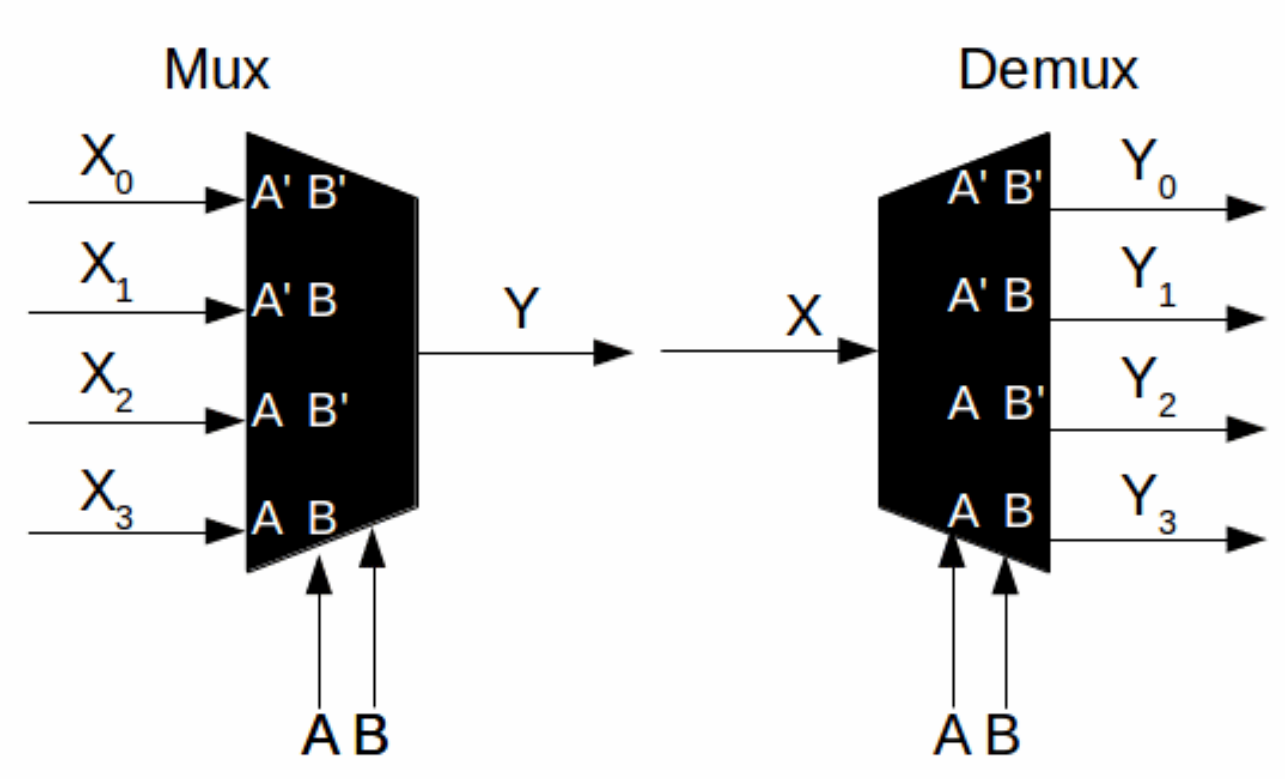
SIMULATION AND SYNTHESIS OF SIMPLE COMBINATORIAL CIRCUITS

SECTION 1 – 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 1 – MULTIPLEXER AND DEMULTIPLEXER



SECTION 1 – WORKSPACE

I organized this section in two sub-sections:

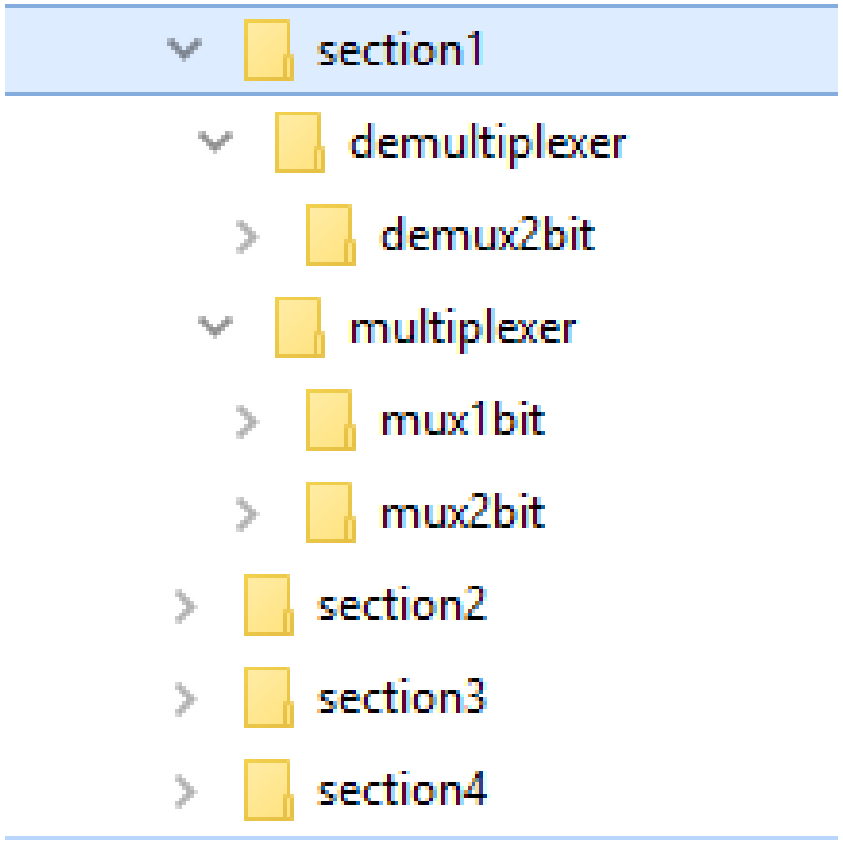
- Multiplexer

  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 1 – 1 BIT MULTIPLEXER VHDL CODE

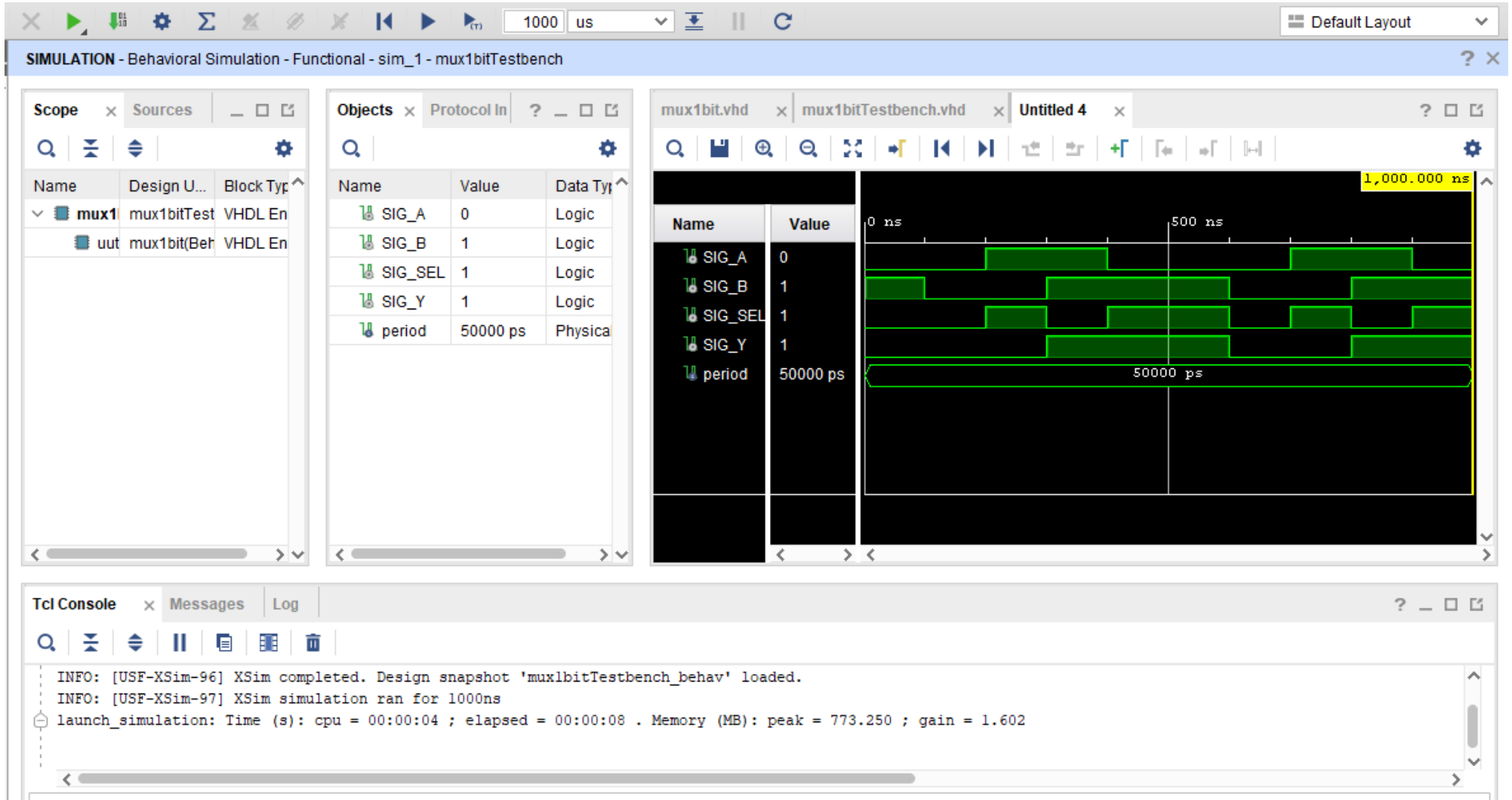
```
30  ----------------------------------------------------------------------------------
31
32
33  library IEEE;
34  use IEEE.STD_LOGIC_1164.ALL;
35
36  -- Uncomment the following library declaration if using
37  -- arithmetic functions with Signed or Unsigned values
38  --use IEEE.NUMERIC_STD.ALL;
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 1 – 1 BIT MULTIPLEXER TESTBENCH VHDL CODE

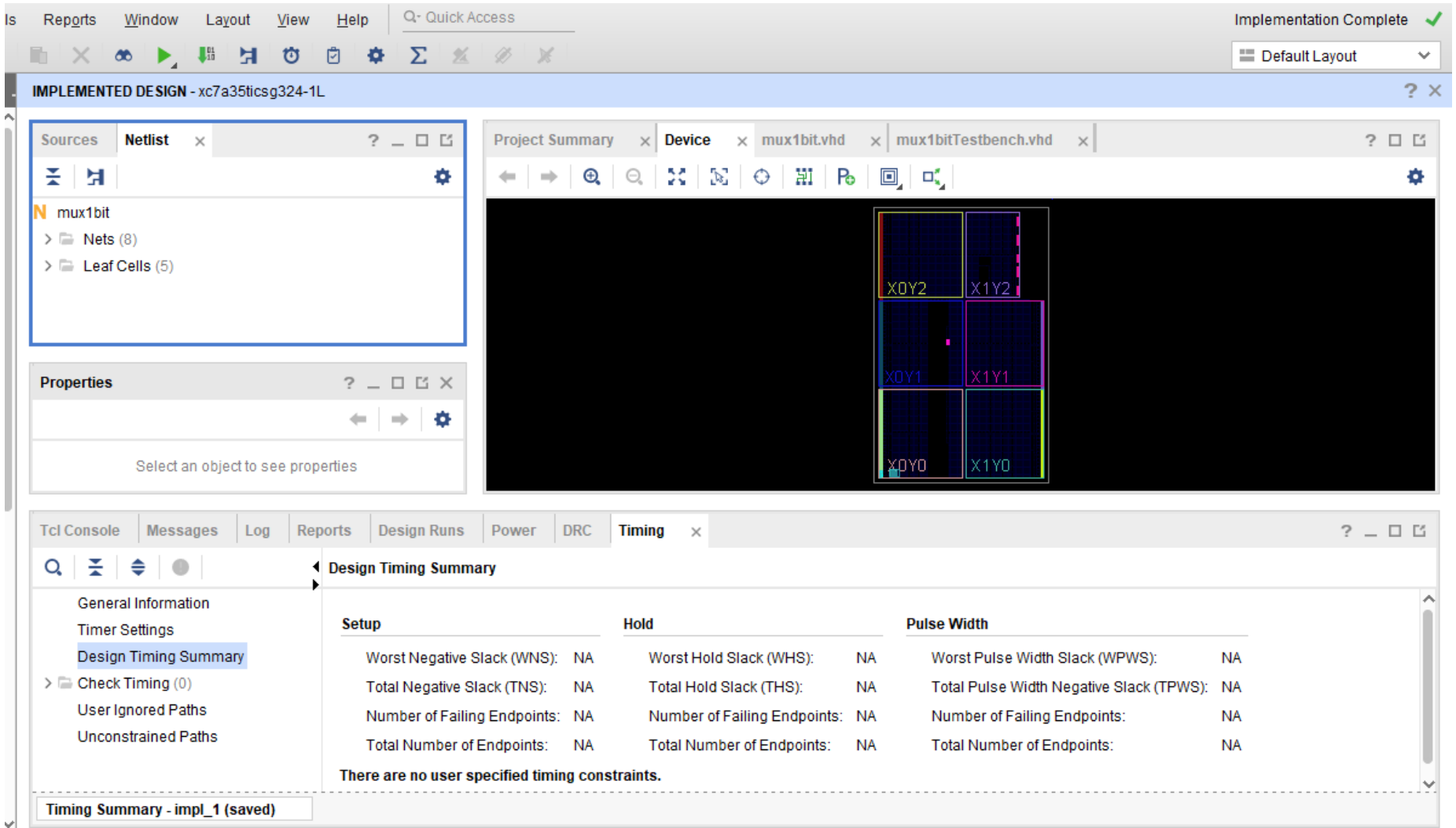
```
30  ----------------------------------------------------------------
31
32
33  library IEEE;
34  use IEEE.STD_LOGIC_1164.ALL;
35
36  -- Uncomment the following library declaration if using
37  -- arithmetic functions with Signed or Unsigned values
38  --use IEEE.NUMERIC_STD.ALL;
39
40  -- Uncomment 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 1 – 1 BIT MULTIPLEXER BEHAVIOURAL SIMULATION



SECTION 1 – 1 BIT MULTIPLEXER SYNTHESIS



SECTION 1 – 1 BIT MULTIPLEXER RESULTS

I obtained the following results:

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

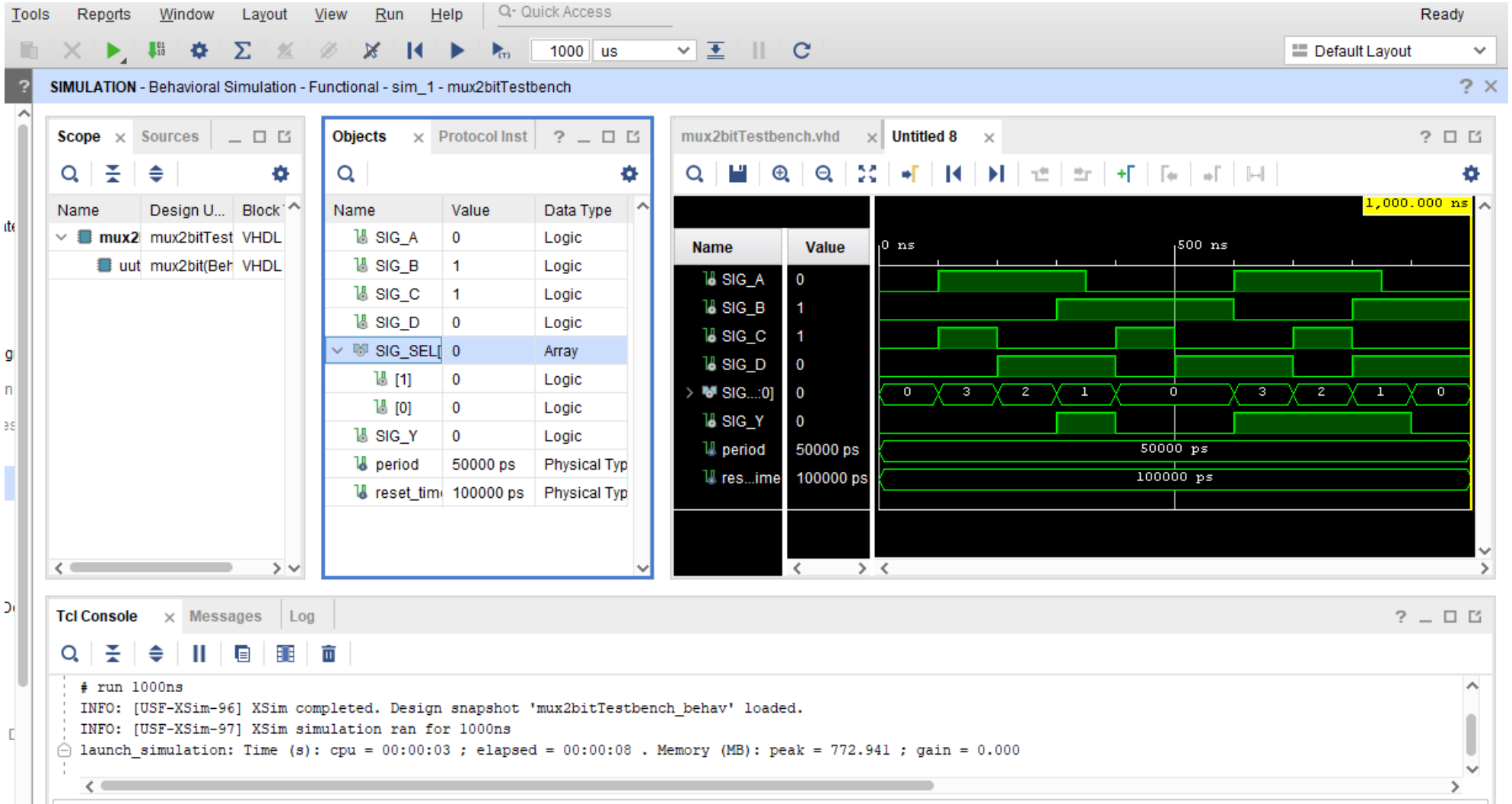
SECTION 1 – 2 BIT MULTIPLEXER VHDL CODE

```
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
```

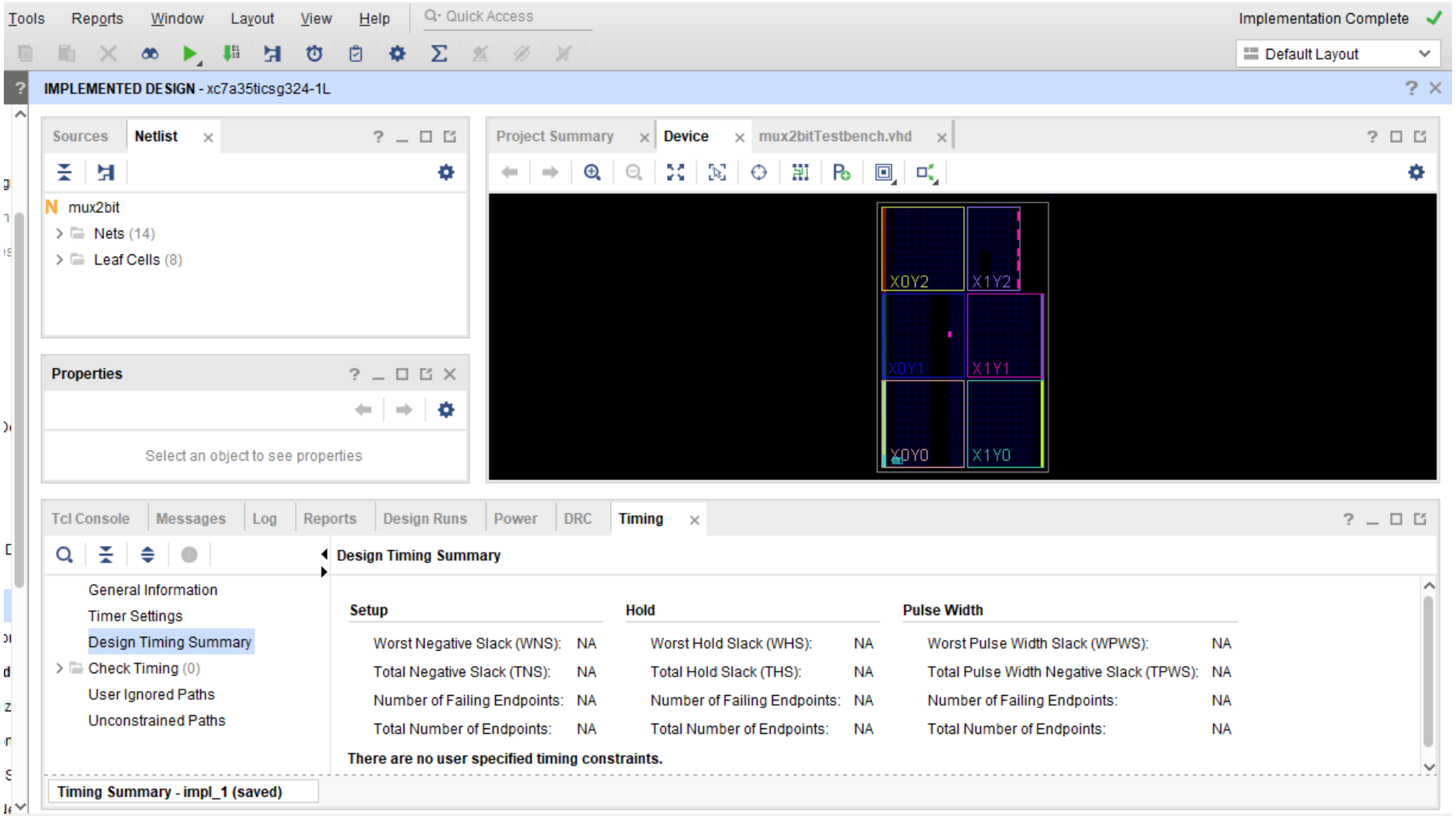
SECTION 1 – 2 BIT MULTIPLEXER TESTBENCH VHDL CODE

```
--use UNISIM.VComponents.all;

entity mux2bitTestbench is
--  Port ( );
end mux2bitTestbench;

architecture Behavioral of mux2bitTestbench is
    -- Component Declaration for the Unit Under Test (UUT):
    COMPONENT mux2bit
    PORT(
        A : IN std_logic;
        B : IN std_logic;
        C : IN std_logic;
        D : IN std_logic;
        SEL : IN std_logic_vector (1 downto 0);
        Y : OUT std_logic
    );
    END COMPONENT;
    -- Input Signals:
    signal SIG_A : std_logic := '0';
    signal SIG_B : std_logic := '0';
    signal SIG_C : std_logic := '0';
    signal SIG_D : std_logic := '0';
    signal SIG_SEL : std_logic_vector (1 downto 0) := "00";
    -- Output Signals:
    signal SIG_Y : std_logic := '0';
    -- Delta Time:
    constant period : time := 50 ns;
    -- Time for reset:
    constant reset_time : time := 100 ns;
begin
    -- Unit Under Test (UUT) Instantiation:
    uut: mux2bit PORT MAP(
        A => SIG_A,
        B => SIG_B,
        C => SIG_C,
        D => SIG_D,
        SEL => SIG_SEL,
        Y => SIG_Y
    );
    -- Stimulus Processes:
    stimSEL : process
    begin
        wait for reset_time;
        SIG_SEL <= "11";
        wait for period * 2;
        SIG_SEL <= "10";
        wait for period * 2;
        SIG_SEL <= "01";
        wait for period * 2;
        SIG_SEL <= "00";
        wait for period * 2;
    end process;
    stimA : process
    begin
        wait for reset_time;
        SIG_A <= '1';
        wait for period * 5;
        SIG_A <= '0';
        wait for period * 3;
        SIG_A <= '0';
        wait for period * 3;
    end process;
    stimB : process
    begin
        wait for reset_time;
        SIG_B <= '0';
        wait for period * 4;
        SIG_B <= '1';
        wait for period * 4;
    end process;
    stimC : process
    begin
        wait for reset_time;
        SIG_C <= '1';
        wait for period * 2;
        SIG_C <= '0';
        wait for period * 2;
    end process;
    stimD : process
    begin
        wait for reset_time;
        SIG_D <= '0';
        wait for period * 2;
        SIG_D <= '1';
        wait for period * 2;
    end process;
end Behavioral;
```

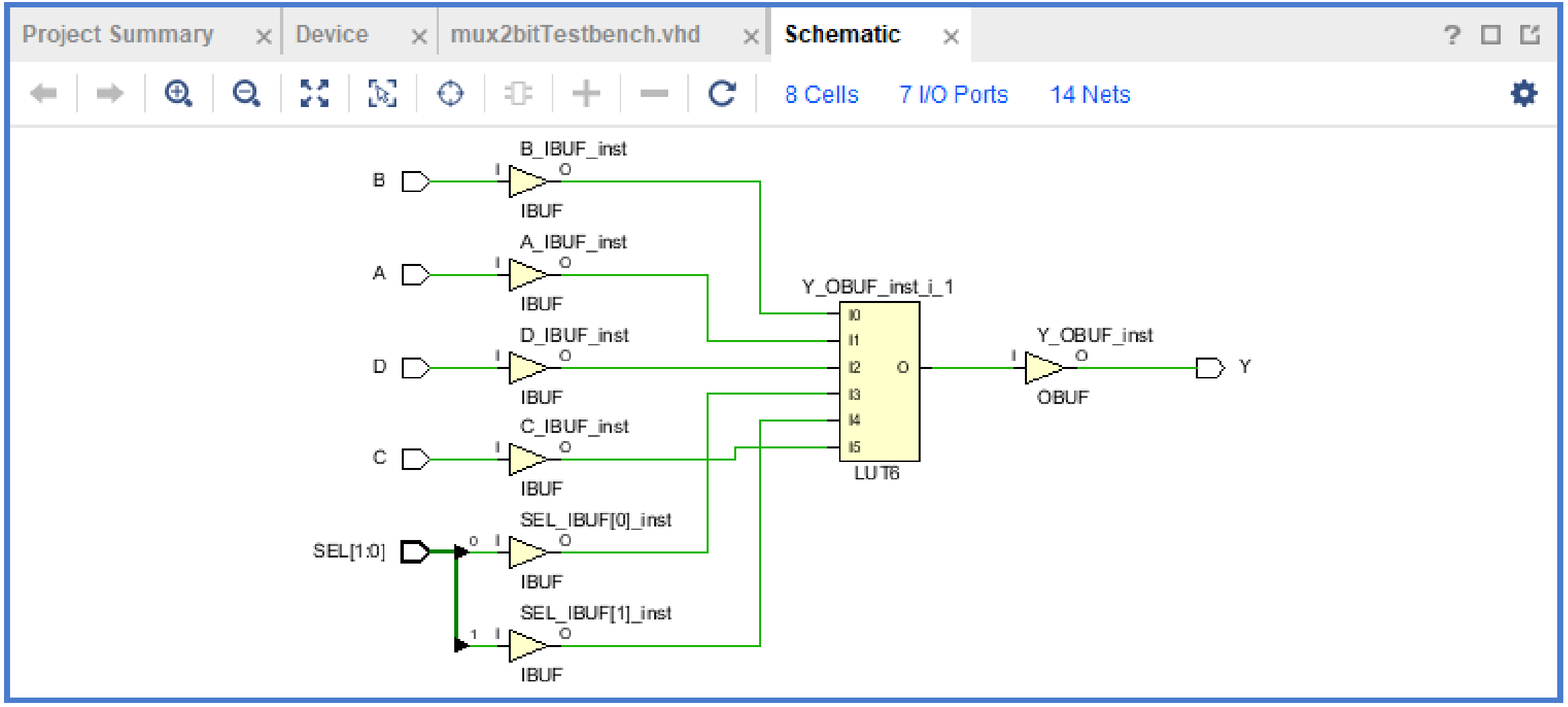
SECTION 1 – 2 BIT MULTIPLEXER BEHAVIOURAL SIMULATION



SECTION 1 – 2 BIT MULTIPLEXER SYNTHESIS



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

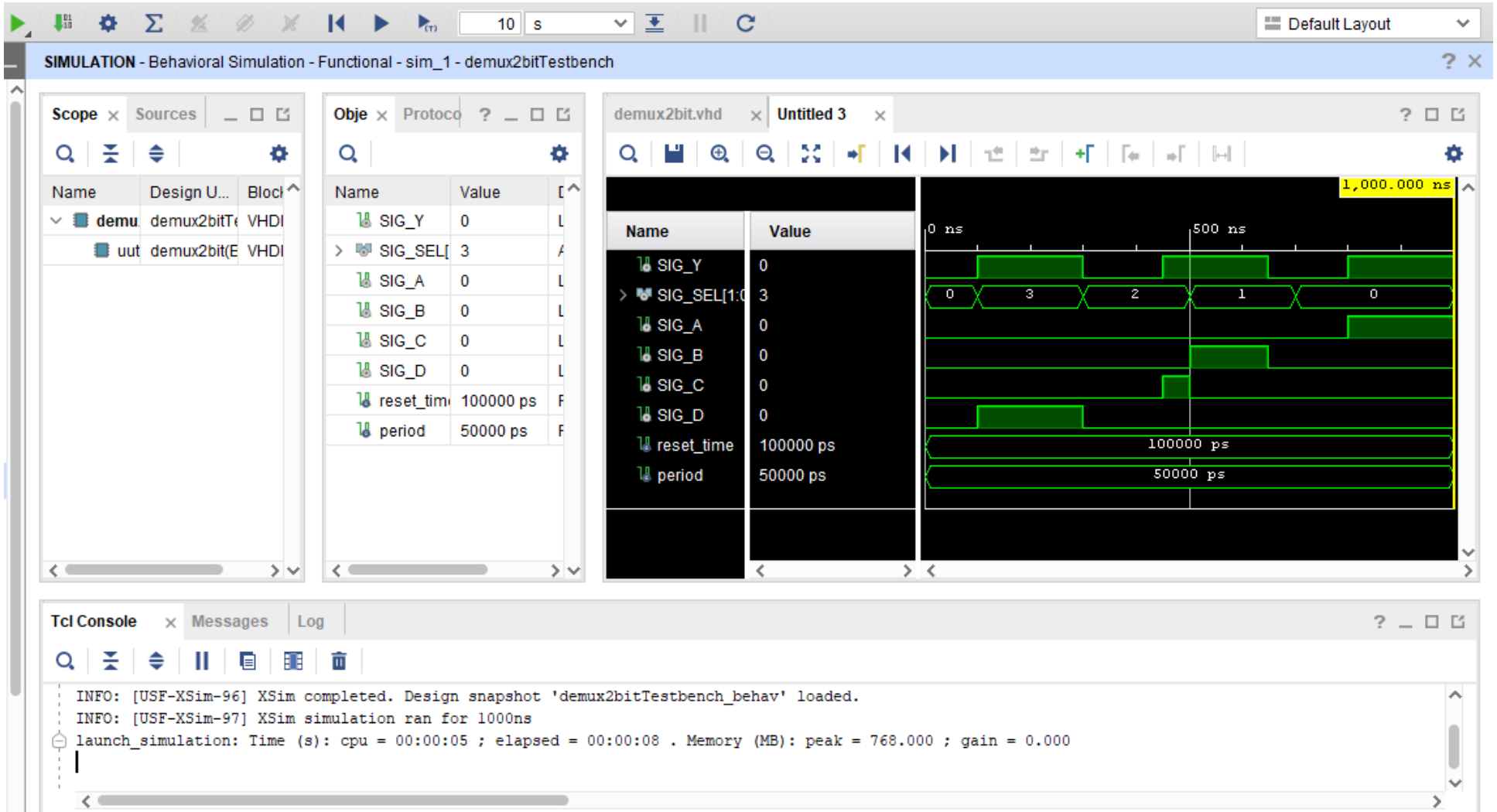
```
43  --use UNISIM.VComponents.all;
44
45  entity demux2bit is
46      Port ( Y : in STD_LOGIC;
47             SEL : in STD_LOGIC_VECTOR (1 DOWNTO 0);
48             A : out STD_LOGIC;
49             B : out STD_LOGIC;
50             C : out STD_LOGIC;
51             D : out STD_LOGIC);
52  end demux2bit;
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

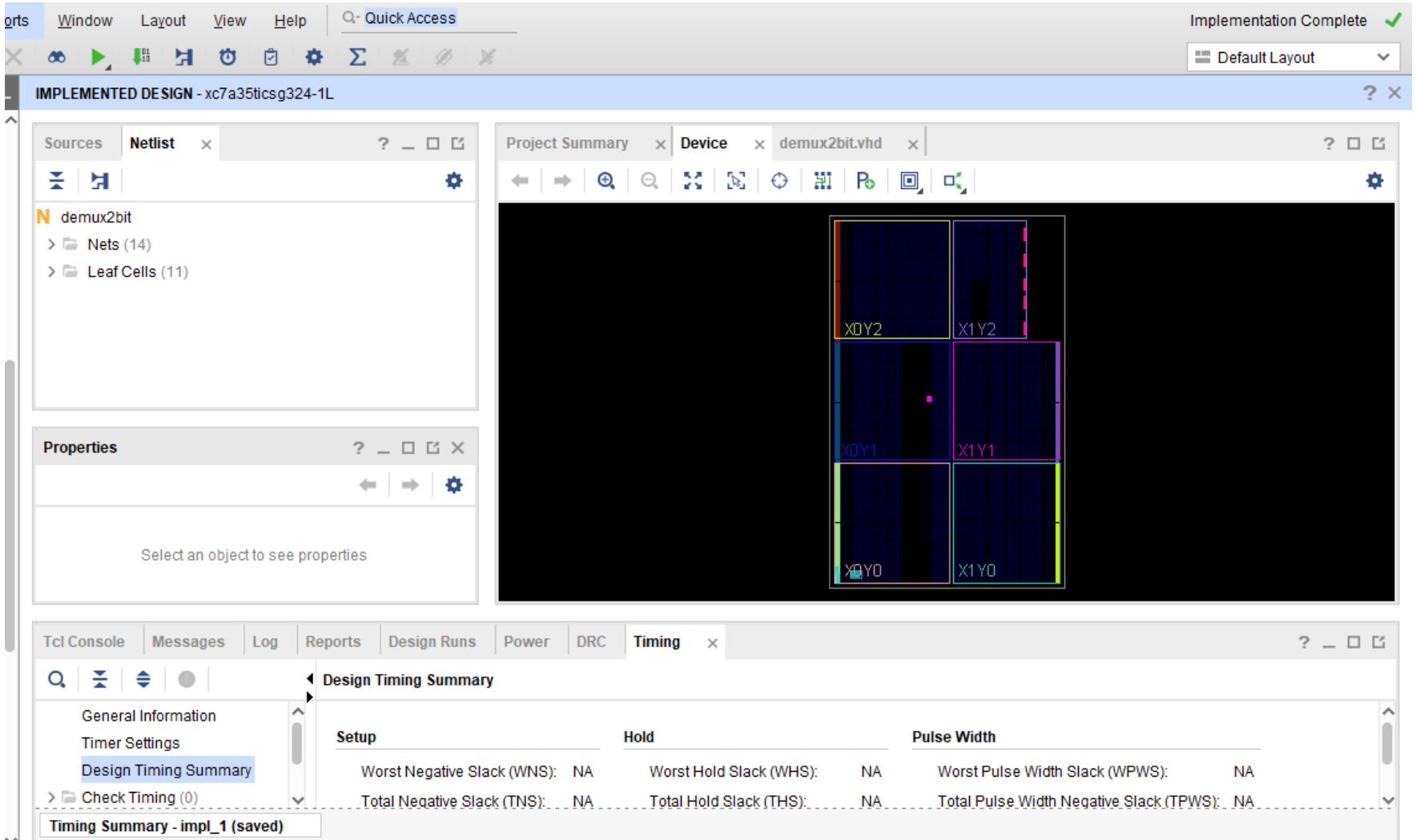
```
32
33  library IEEE;
34  use IEEE.STD_LOGIC_1164.ALL;
35
36  -- Uncomment the following library declaration if using
37  -- arithmetic functions with Signed or Unsigned values
38  --use IEEE.NUMERIC_STD.ALL;
39
40  -- Uncomment the following library declaration if instantiating
41  -- any Xilinx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

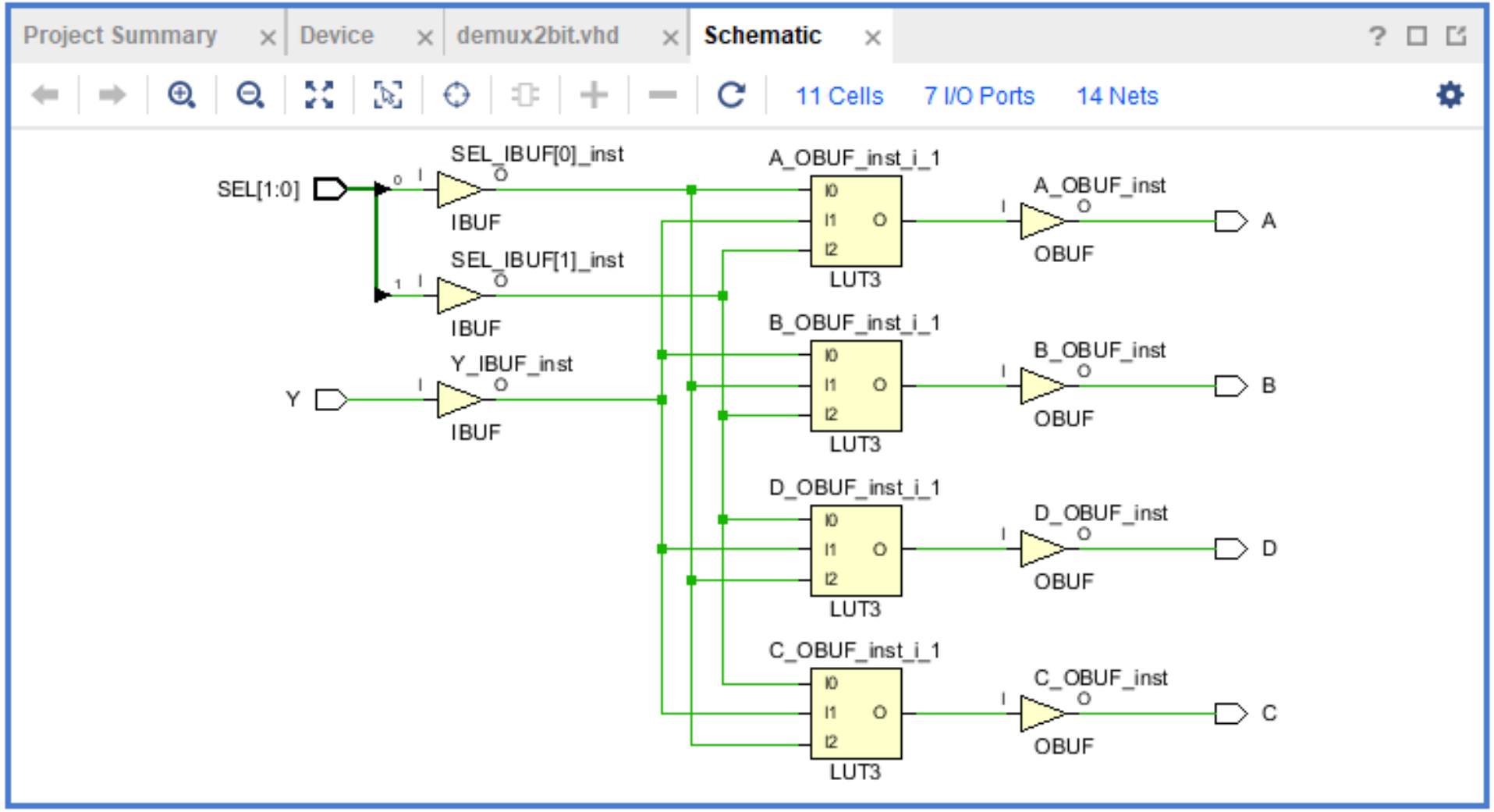
SECTION 1 – 2 BIT DEMULTIPLEXER BEHAVIOURAL SIMULATION



SECTION 1 – 2 BIT DEMULTIPLEXER SYNTHESIS



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 FSMD and RTL with FSM and datapath).

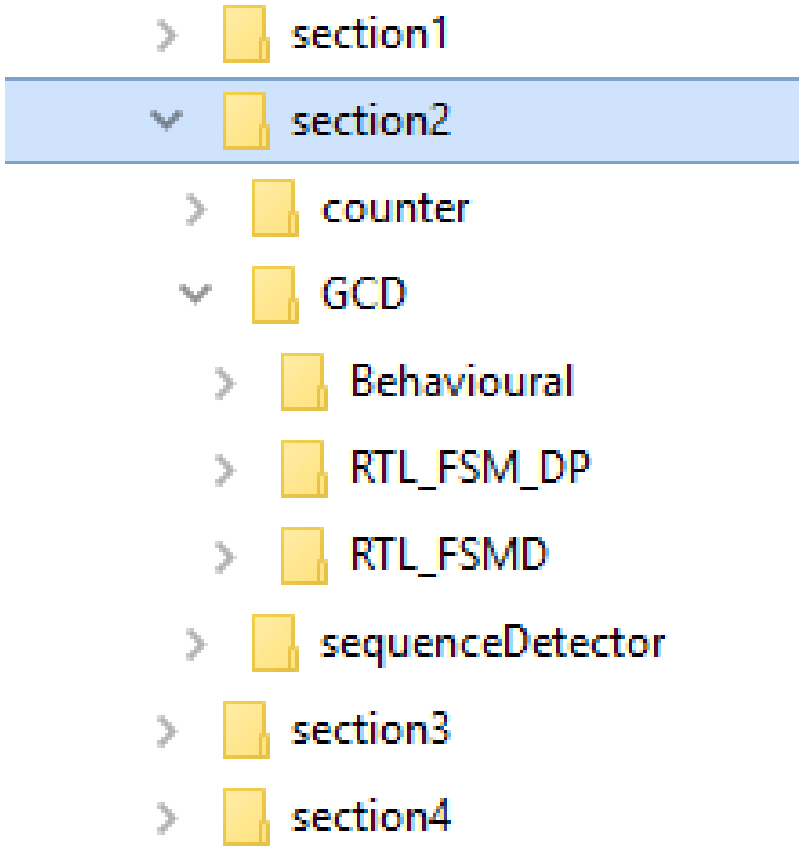
SECTION 2 – WORKSPACE

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

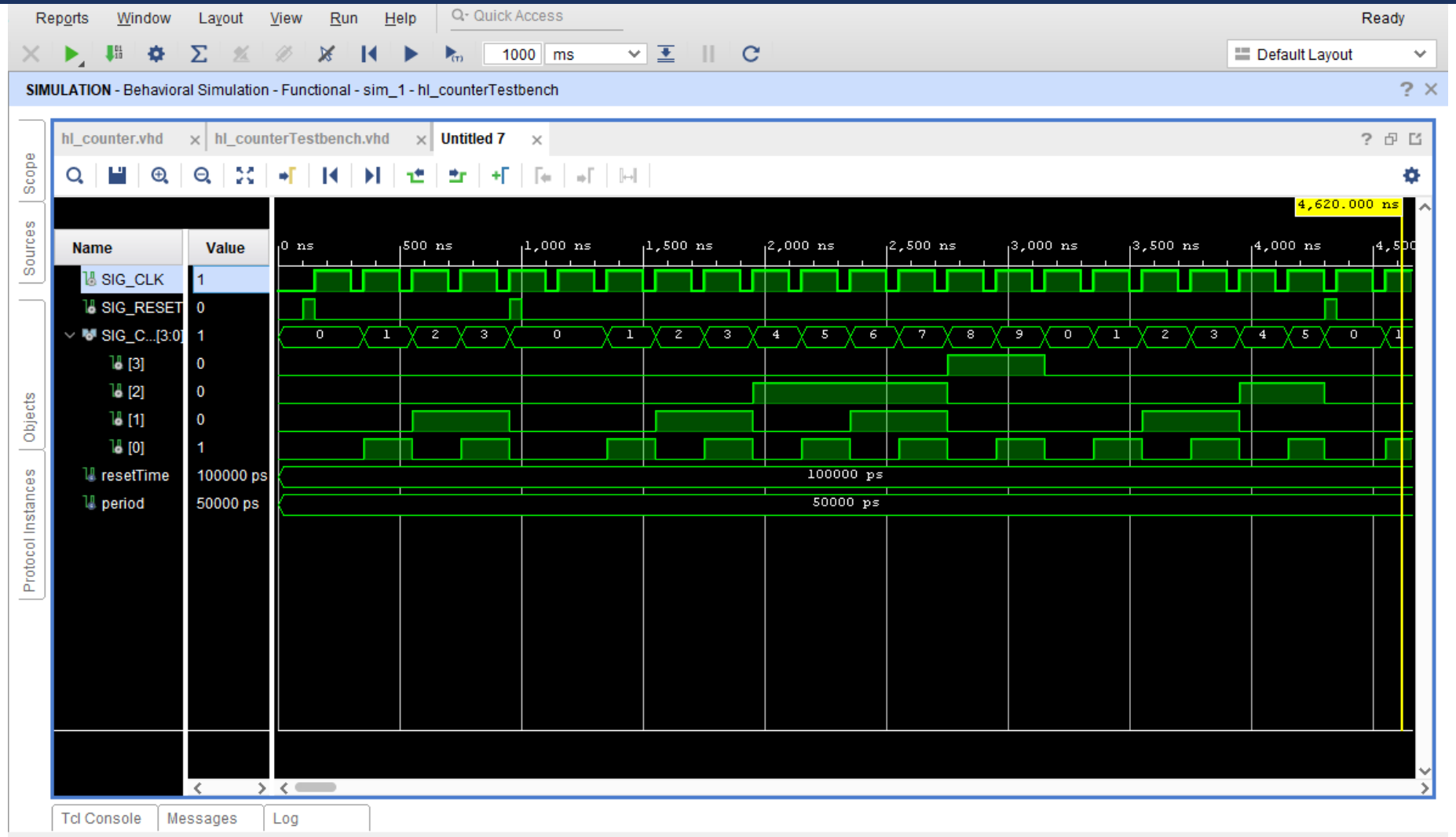
```
67      --          END IF;
68      --      END IF;
69
70      --      count_out <= temp_count;
71      --END PROCESS;
72
73          countingFastForBehavSim : process(reset, clk, temp_count)
74              begin
75                  if reset = '1' then
76                      temp_count_internal <= "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

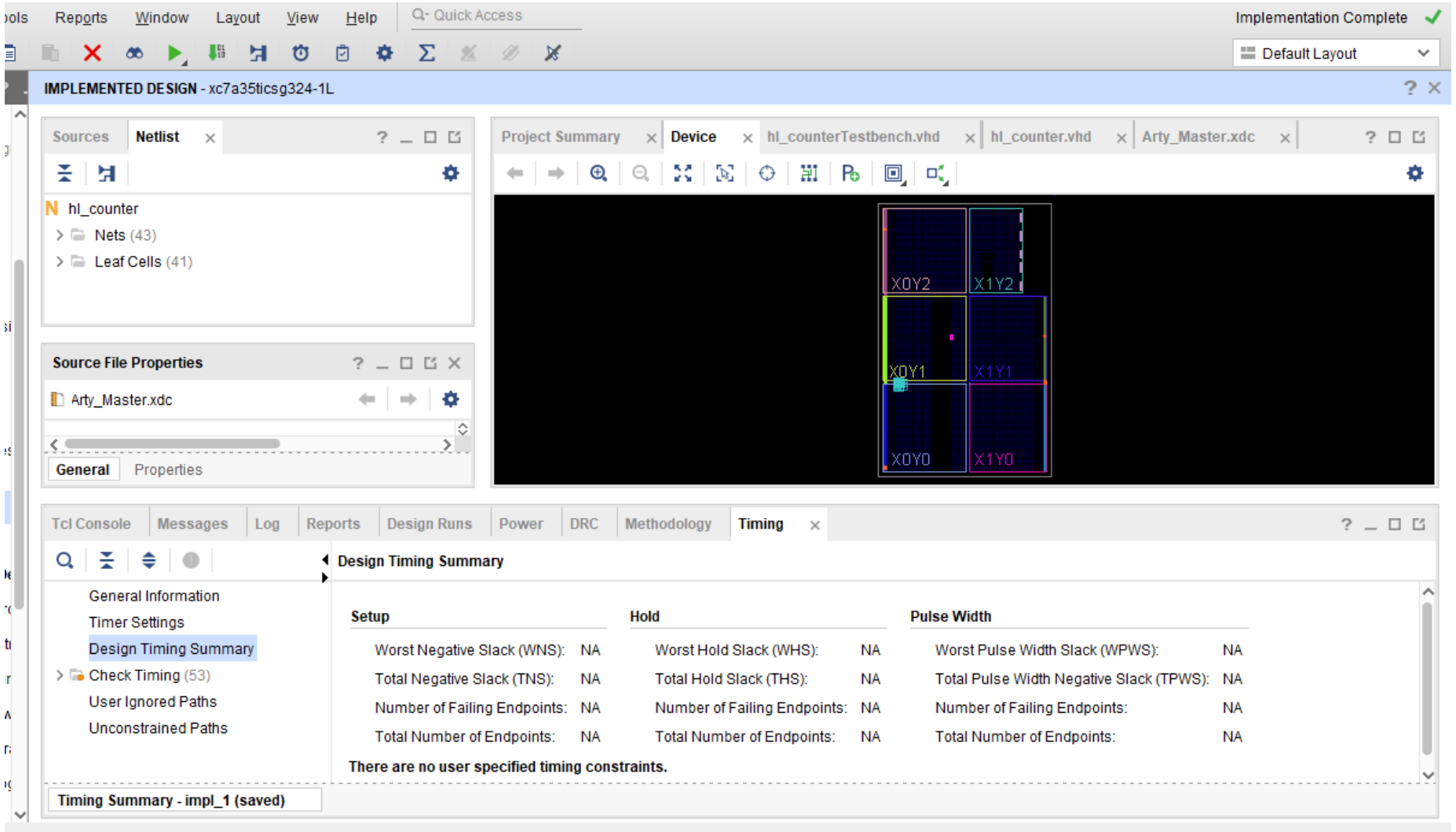
```
30  ----------------------------------------------------------------------------
31
32
33  library IEEE;
34  use IEEE.STD_LOGIC_1164.ALL;
35
36  -- Uncomment the following library declaration if using
37  -- arithmetic functions with Signed or Unsigned values
38  --use IEEE.NUMERIC_STD.ALL;
39
40  -- 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
```

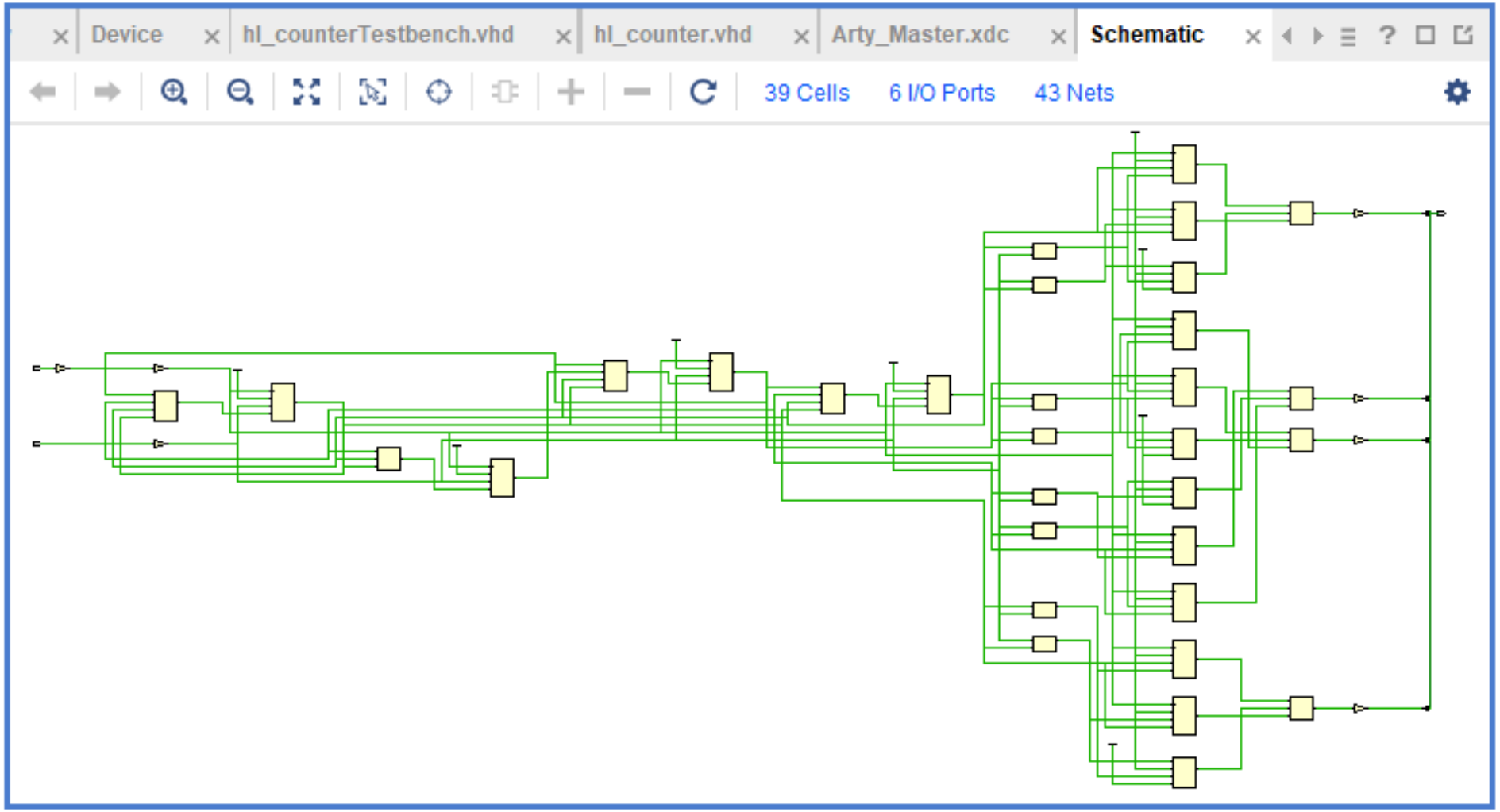
SECTION 2 – COUNTER BEHAVIOURAL SIMULATION



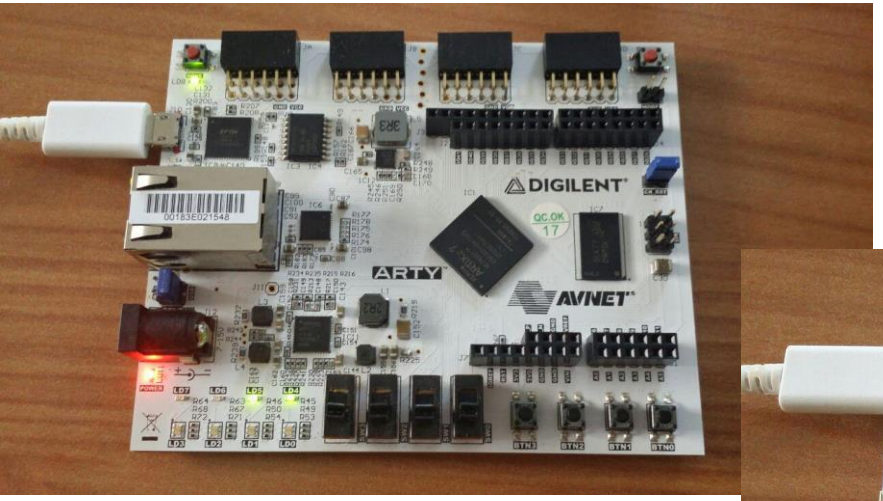
SECTION 2 – COUNTER SYNTHESIS

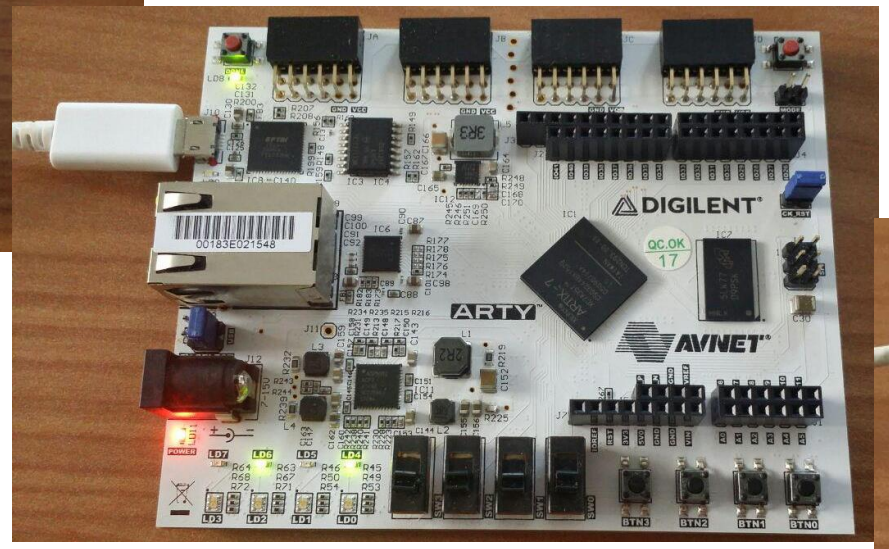


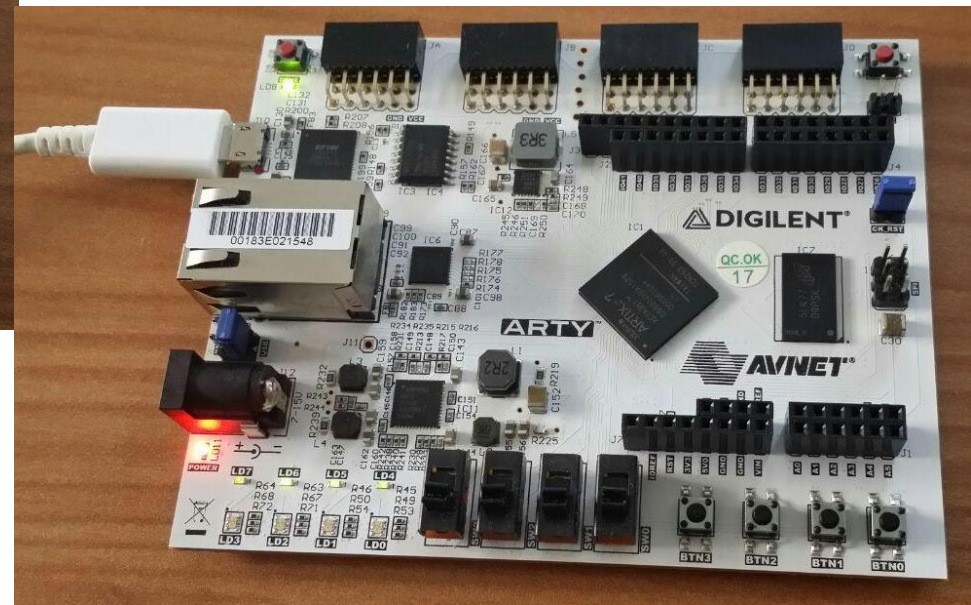
SECTION 2 – COUNTER SCHEMATIC



SECTION 2 – COUNTER ON ARTY BOARD





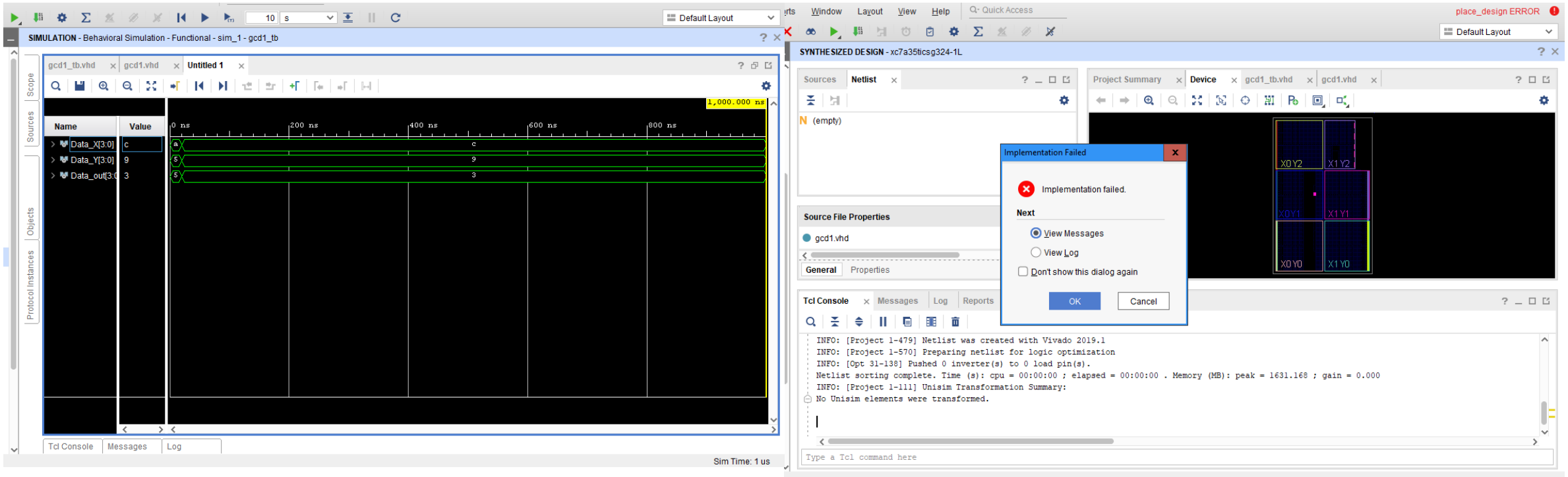


SECTION 2 – SEQUENCE DETECTOR XDC FILE

```
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
```

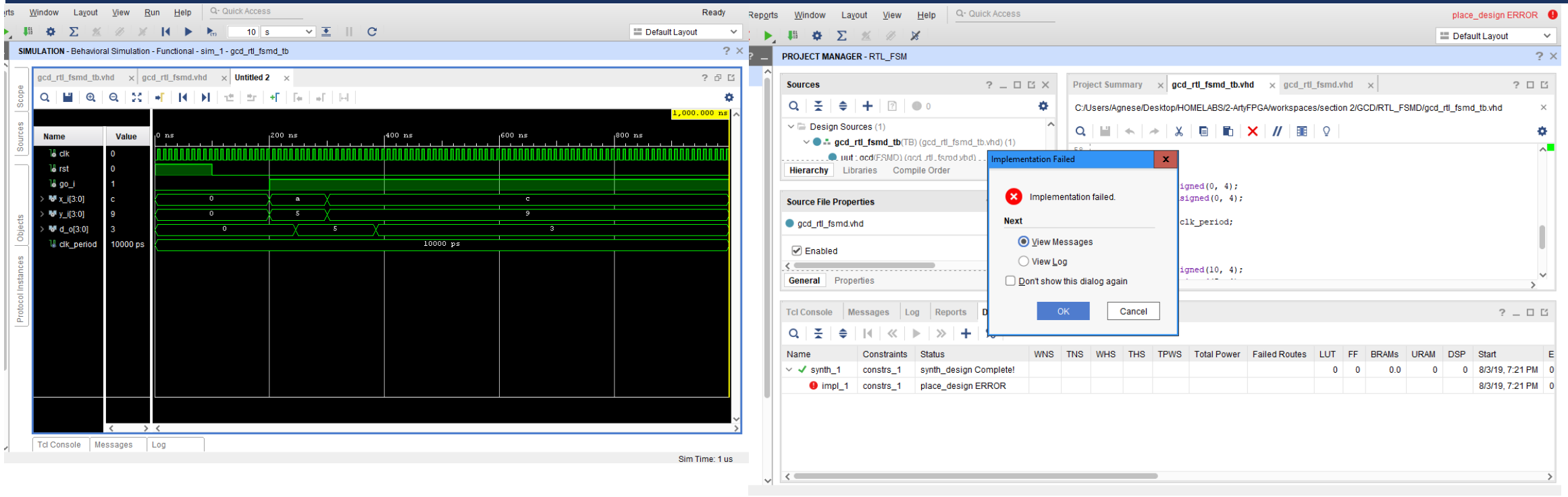
```
32
33  ##LEDs
34
35  set_property -dict { PACKAGE_PIN H5    IOSTANDARD LVCMOS33 } [get_ports { y }]; #IO_L24N_T3_35 Sch=led[4]
36  #set_property -dict { PACKAGE_PIN J5    IOSTANDARD LVCMOS33 } [get_ports { count_out[1] }]; #IO_25_35 Sch=led[5]
37  #set_property -dict { PACKAGE_PIN T9    IOSTANDARD 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
```

SECTION 2 – BEHAVIOURAL GDC SIMULATION RESULTS



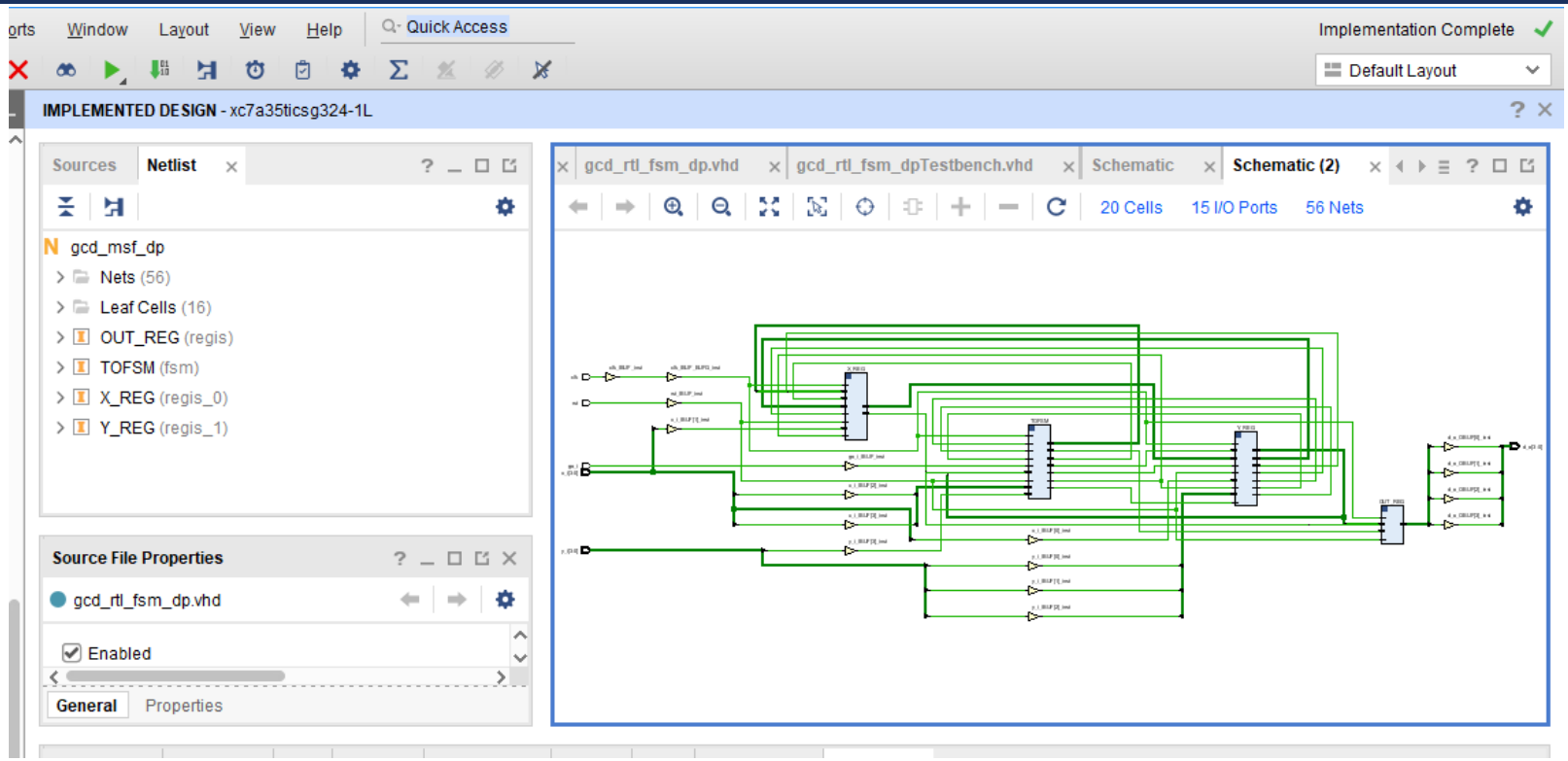
It's not implementable.

SECTION 2 – RTL WITH FSM GDC SIMULATION RESULTS



It's not implementable.

SECTION 2 – RTL WITH FSM AND DATAPATH GDC SIMULATION RESULTS



It's the only implementable one.

SECTION 3

PICOBLAZE

SECTION 3 – TO DO

In this section, I had to:

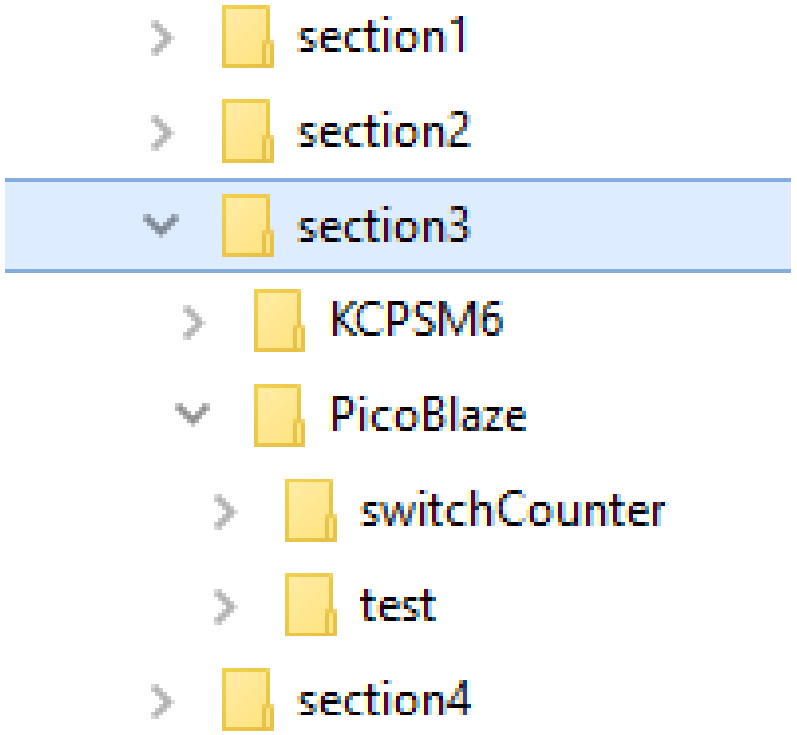
- Perform a test:
  - Write a given Assembly program on a psm file
  - Generate the ROM VHD file, using the kcpsm6.exe (downloaded with PicoBlaze material) and import it in a Vivado Project
  - Add given embedded_pico.vhd file
  - Add Constraints (on XDC file) as needed
  - Synthesize and run the project 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

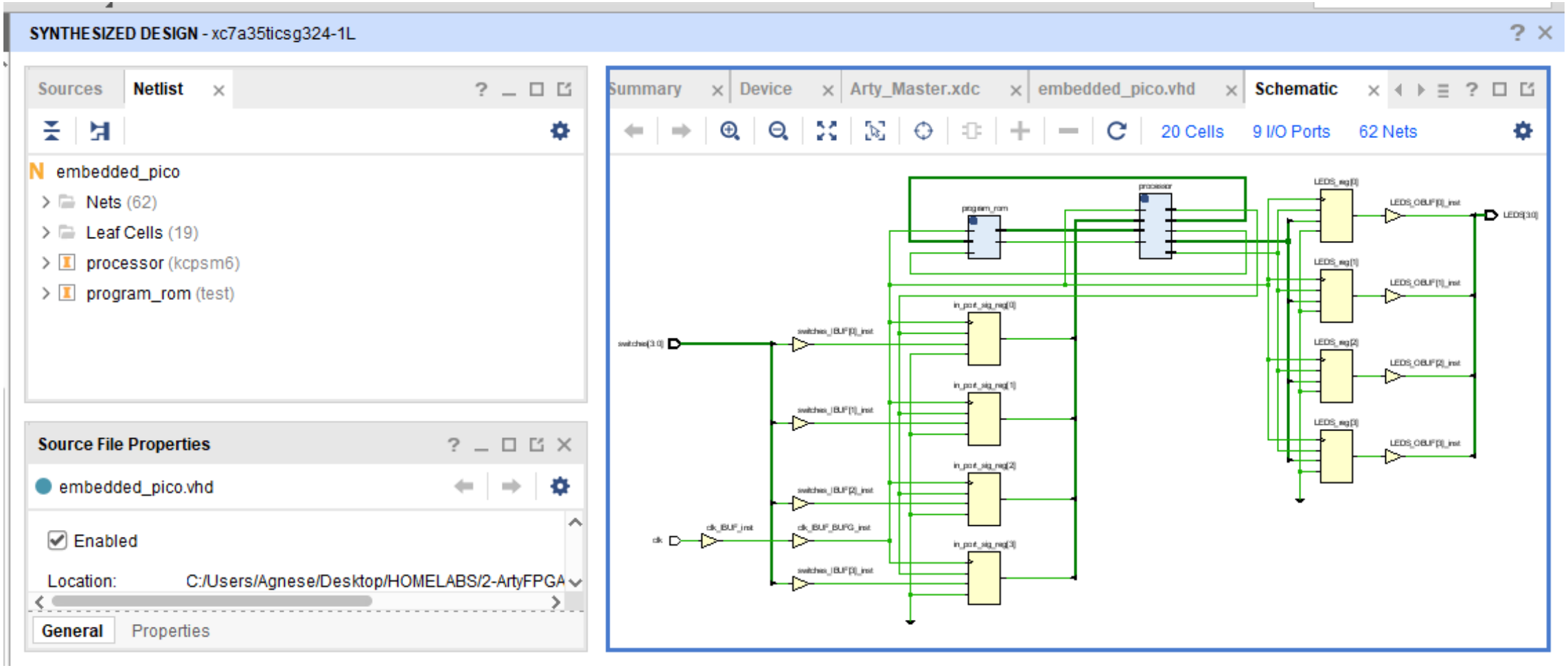
PicoBlazeSwitchesToLeds.psm

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

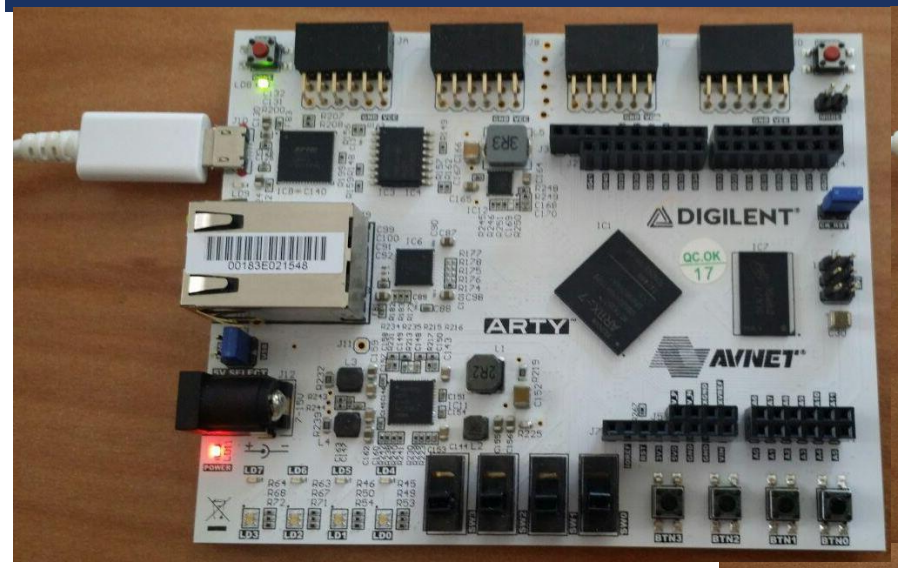
kcpsm6.exe

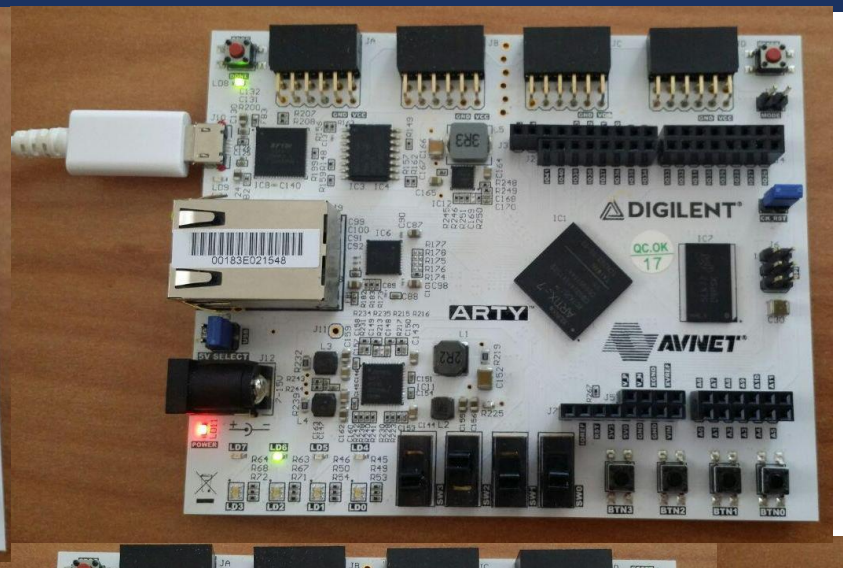
```
KCPSM6 Assembler v2.70
  Ken Chapman - Xilinx Ltd - 16th May 2014

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
```

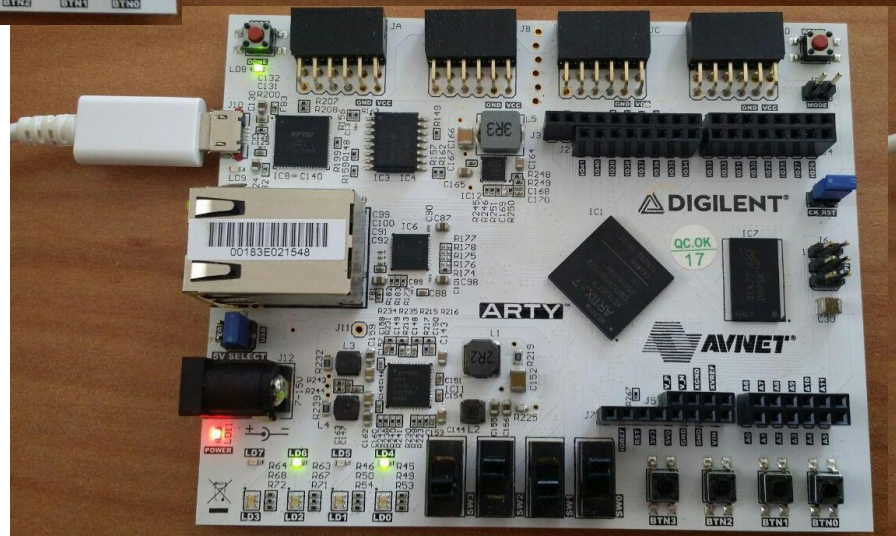
SECTION 3 – TEST SCHEMATIC

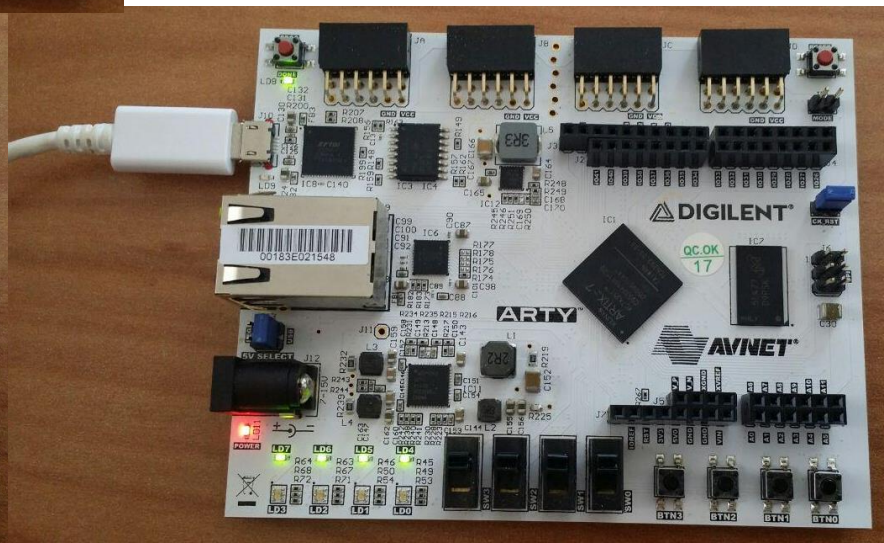


SECTION 3 – TEST ON ARTY BOARD







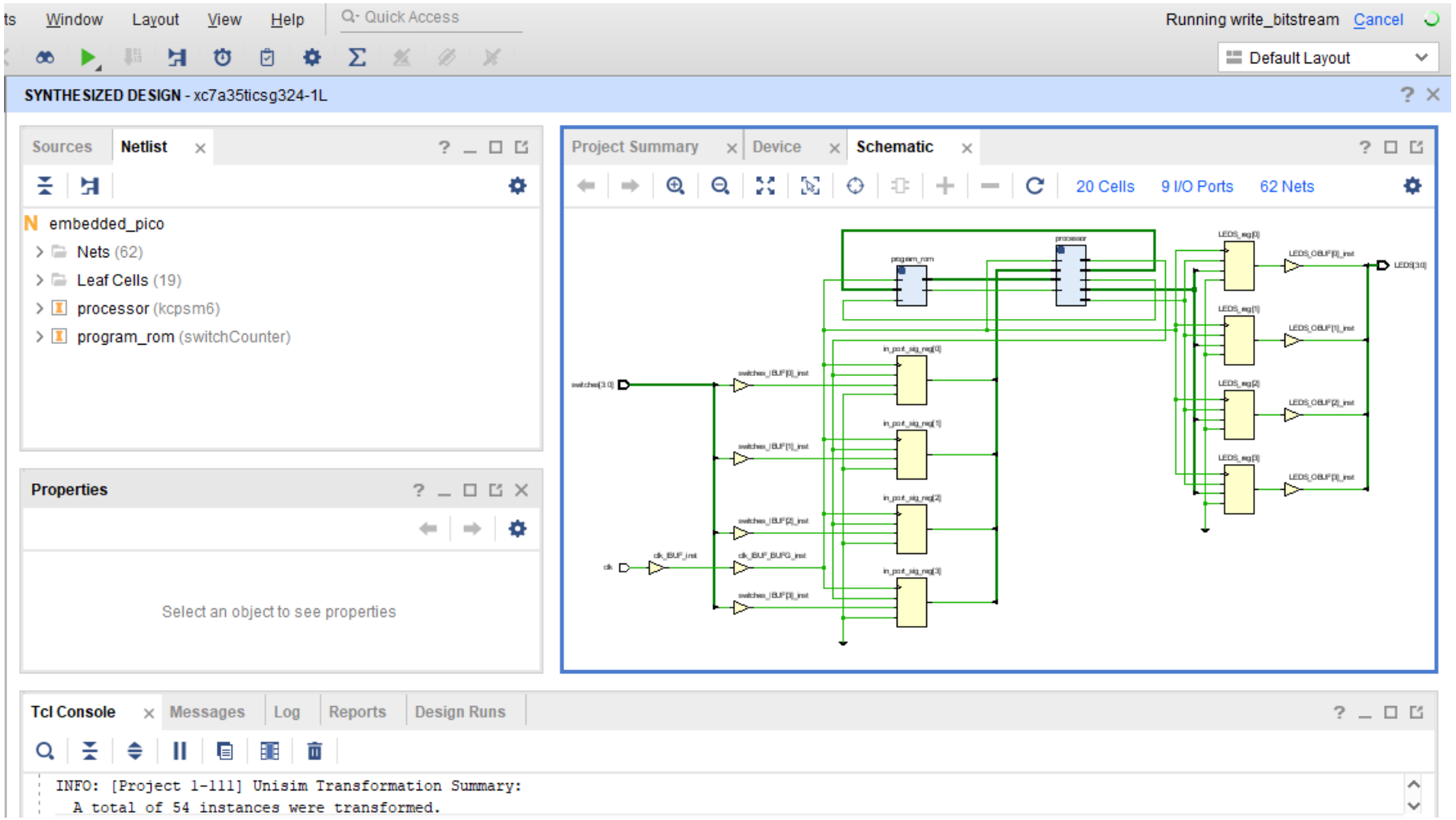


SECTION 3 – SWITCH COUNTER ASSEMBLY CODE AND KCPSM6.EXE OUTPUT

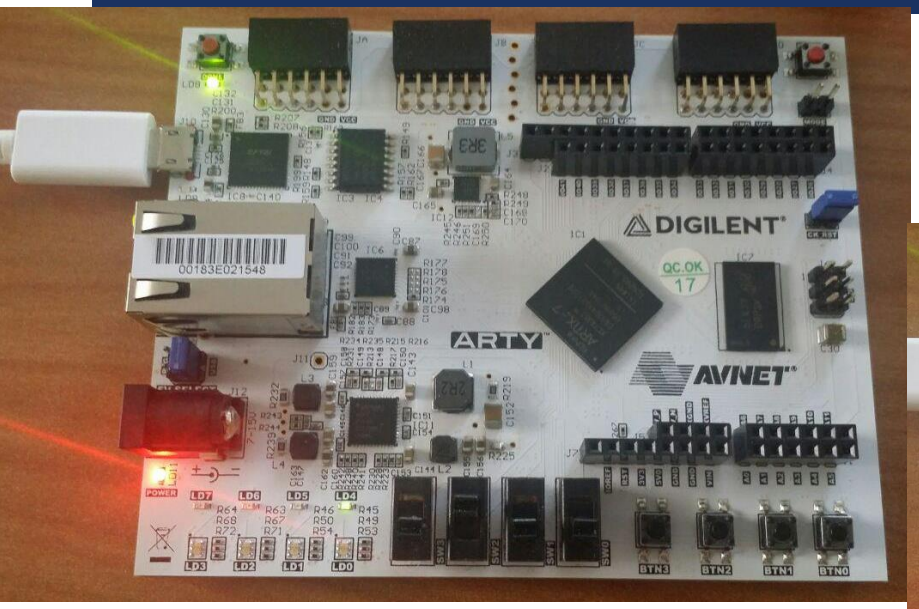
```
38
39  ;==============================
40  ;== MAIN APPLICATION CODE
41  ; Count the number of the switches that are switched on and display the result to the LEDs
42  main:
43      ; Prepare arguments passed through registers
44      INPUT s0, 00 ; Input Register
45      load s1, 00 ; 1 bits number
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 OS 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  ;==============================
```

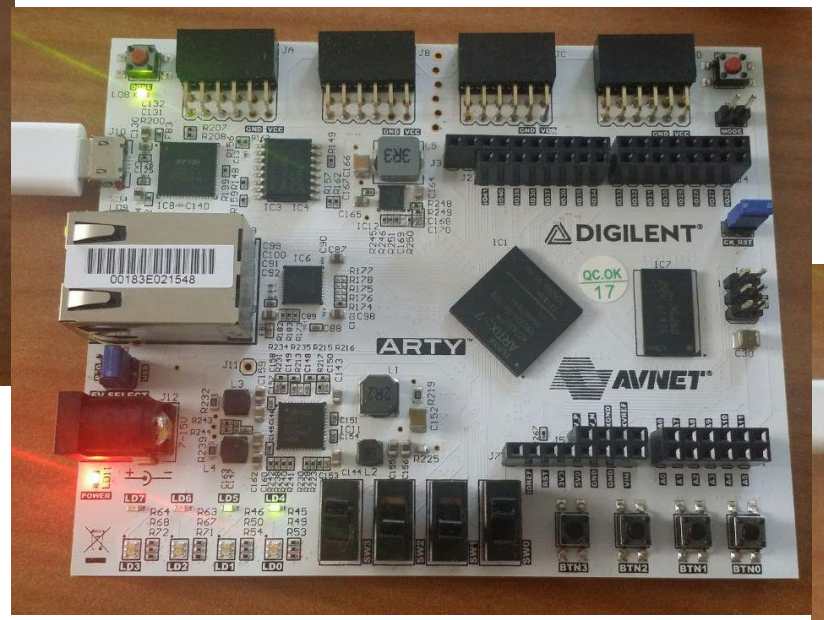
```
kcpsm6.exe
KCPSM6 Assembler v2.70
  Ken Chapman - Xilinx Ltd - 16th May 2014

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
```

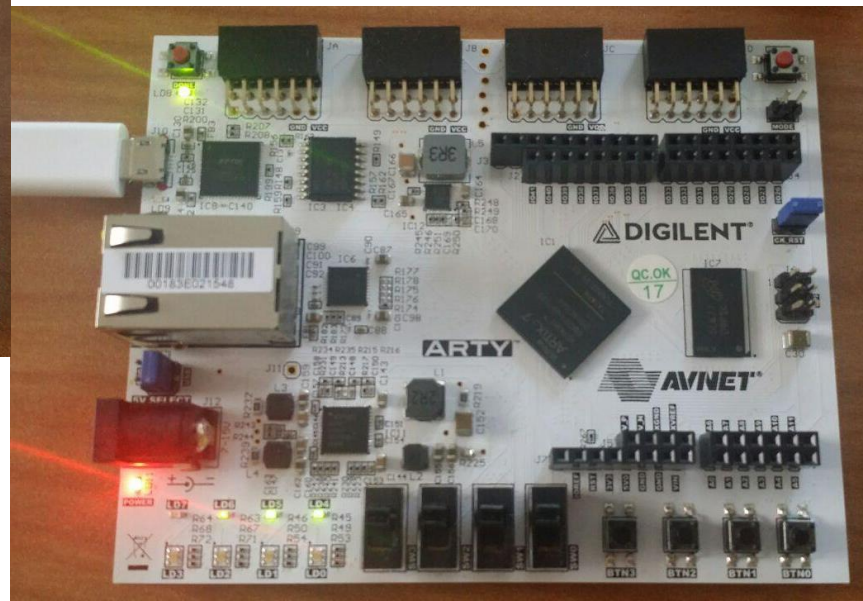
SECTION 3 – SWITCH COUNTER SCHEMATIC



SECTION 3 – SWITCH COUNTER ON ARTY BOARD







SECTION 4

SERIAL COMMUNICATION

SECTION 4 – TO DO

In this last section, I had to:

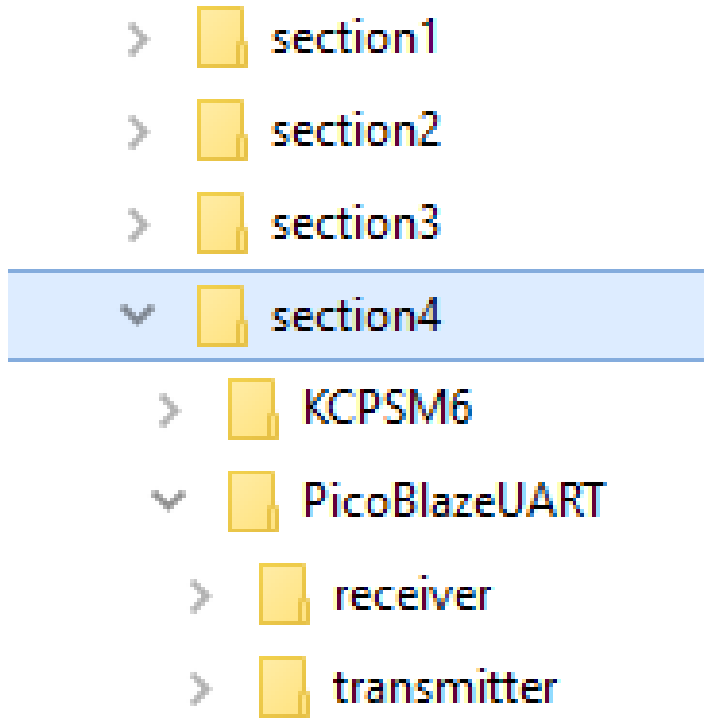
- Given the a test Transmitter:
  - Add proper Constraints (in XDC file)
  - Run it on the Arty Board
- Implement a Receiver and run it on the Arty Board

SECTION 4 – WORKSPACE

I organized this section in two sub-sections:

- Transmitter
- Receiver

KPCSM6 contains downloaded PicoBlaze material.



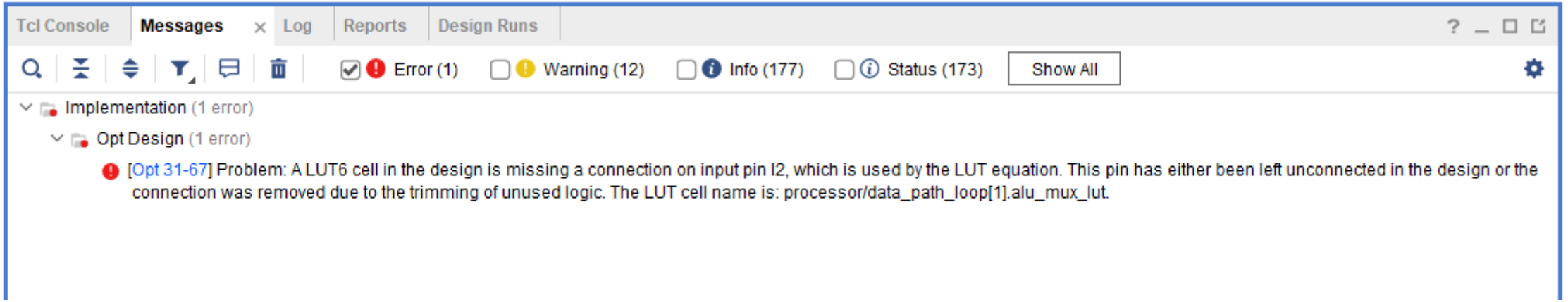
SECTION 4 – TRANSMITTER XDC FILE

```
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   ##ChipKit Single Ended Analog Inputs
```

SECTION 4 – TRANSMITTER VIVADO ERROR

I synthetized the given transmitter correctly, but during implementation some error occured:



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

```
; Software implemented UART Receiver
;
; Receives one character into 'sS' at 115200 baud with 1 start bit, 1 stop bit
; parity. All timing is based on a 50MHz clock where each bit period is equiva
; instructions. A valid character is signified by Z=0 and C=0. A timeout (~51u
; character is invalid then Z=0 and C=1.
;
; Registers used s0, s1, s2 and s5.
;

CONSTANT UART_input_port, 04 ; Receive serial data
CONSTANT UART_output_port, 10 ; Transmit serial data
CONSTANT serial_data, 00000001'b ; bit0 - serial data
LOAD s0, serial_data
OUTPUT s0, UART_output_port ;initialise serial output

UART_RX:
    LOAD s1, 255'd ;Detect beginning of start bit (0) or timeout
rx_timeout:
    INPUT s0, UART_input_port ; 255 x 5 = 1275 instructions or ~51us.
    TEST s0, serial_data ;test serial input for change to '0'
    JUMP Z, start_bit
    SUB s1, 1'd
    JUMP NZ, rx_timeout
    RETURN ;Timeout returns with Z=1 and C=0
;
    RETURN ;Timeout returns with Z=1 and C=0
;
start_bit:
    LOAD s1, 51'd ;Wait until middle of start bit
    mid_start_delay: SUB s1, 1'd ;51 x 2 = 102 instruction delay
    JUMP NZ, mid_start_delay
    INPUT s0, UART_input_port ;test for start bit = '0'
    SR1 s0 ;shift start bit into carry flag and force Z=0
    RETURN C ;Will abort with C=1 and Z=0 if start bit was High
;
LOAD s2, 08 ;8 bits to receive
RX_loop:
    LOAD s1, 105'd ;Loop delay is (105 x 2) + 6 = 216 instructions
rx_bit_delay:
    SUB s1, 1'd
    JUMP NZ, rx_bit_delay
    INPUT s0, UART_input_port ;sample data bit at mid-point
    SR0 s0 ;move data bit into carry flag
    SRA s5 ;Shift data bit into 'sS' LSB first
    SUB s2, 1'd ;count 8 bits
    JUMP NZ, RX_loop
;
; Finally wait one more bit period and sample the stop bit which should be High.
; If it is Low then set carry flag to indicate error. But if it is High the
; character is good and the return must be made with Z=0.
;
stop_bit:
    LOAD s1, 106'd ;Wait until middle of stop bit
stop_bit_delay:
    SUB s1, 1'd ;(106 x 2) + 5 = 217 instructions
    JUMP NZ, stop_bit_delay
    INPUT s0, UART_input_port ;test for stop bit = '1'
    XOR s0, serial_data ;invert bit so that correct value for carry flag
    SR1 s0 ;shift inverted bit into carry flag force Z=0
    RETURN ;For good character return with Z=0 and C=0
```

SECTION 4 – RECEIVER EMBEDDED_PICO.VHD

```
entity embedded_pico is
port (
    clk: in std_logic;
    rxd: in std_logic
);

end embedded_pico;


architecture behavioral of embedded_pico is


  component kcpsm6
    generic(                 hwbuild : std_logic_vector(7 downto 0) := X"00";
                    interrupt_vector : std_logic_vector(11 downto 0) := X"3FF";
             scratch_pad_memory_size : integer := 64);
    port (                   address : out std_logic_vector(11 downto 0);
                         instruction : in std_logic_vector(17 downto 0);
                         bram_enable : out std_logic;
                             in_port : in std_logic_vector(7 downto 0);
                            out_port : out std_logic_vector(7 downto 0);
                             port_id : out std_logic_vector(7 downto 0);
                        write_strobe : out std_logic;
                      k_write_strobe : out std_logic;
                        read_strobe : out std_logic;
                          interrupt : in std_logic;
                      interrupt_ack : out std_logic;
                              sleep : in std_logic;
                              reset : in std_logic;
                                clk : in std_logic);
  end component;


  component receiver
    generic(             C_FAMILY : string := "S6";
                C_RAM_SIZE_KWORDS : integer := 1;
             C_JTAG_LOADER_ENABLE : integer := 0);
    Port (       address : in std_logic_vector(11 downto 0);
             instruction : out std_logic_vector(17 downto 0);
                  enable : in std_logic;
```

```
             instruction : out std_logic_vector(17 downto 0)
                  enable : in std_logic;
                     rdl : out std_logic;
                     clk : in std_logic);
end component;

  component uart_rx6
  Port (
      serial_in : in std_logic;
      en_16_x_baud : in std_logic;
      data_out : out std_logic_vector(7 downto 0);
      buffer_read : in std_logic;
      buffer_data_present : out std_logic;
      buffer_half_full : out std_logic;
      buffer_full : out std_logic;
      buffer_reset : in std_logic;
      clk : in std_logic
  );
  end component;


signal         address : std_logic_vector(11 downto 0);
signal     instruction : std_logic_vector(17 downto 0);
signal     bram_enable : std_logic;
signal  k_write_strobe : std_logic;
signal    kcpsm6_sleep : std_logic;
signal    kcpsm6_reset : std_logic;
signal en_16_x_baud: std_logic;
signal interrupt_ack: std_logic;
signal interrupt: std_logic;
signal pb_in_data: std_logic_vector (7 downto 0);
signal pb_out_data: std_logic_vector (7 downto 0);
signal uart_write: std_logic;
SIGNAL baud_count: integer range 0 to 162 := 0;
signal uart_rx: std_logic;
signal uart_rx_data_out: std_logic_vector (7 downto 0);
signal read_from_uart_rx: std_logic;
signal uart_rx_data_present: std_logic;

--


begin


  processor: kcpsm6
    generic map (                 hwbuild => X"00",
```

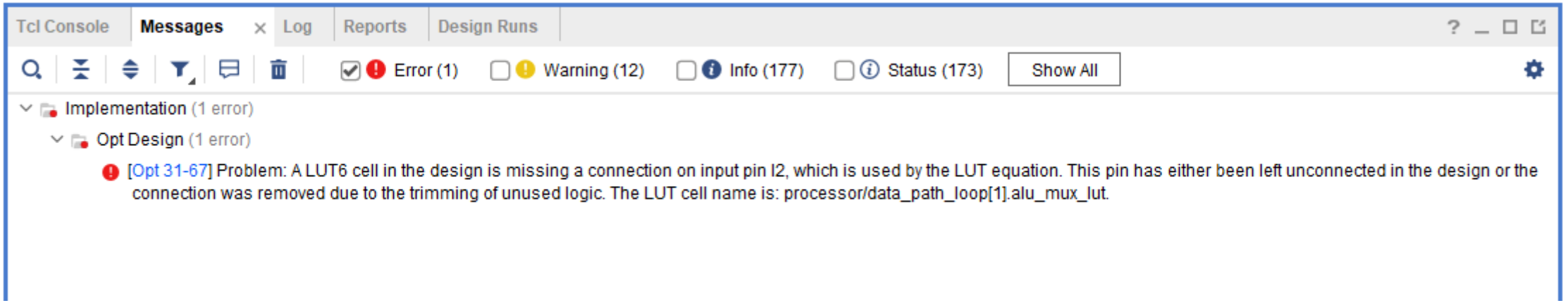
SECTION 4 – RECEIVER EMBEDDED_PICO.VHD

```
86    --
87
88
89    begin
90
91
92      processor: kcpsm6
93        generic map (                 hwbuild => X"00",
94                                    interrupt_vector => X"3FF",
95                             scratch_pad_memory_size => 64)
96        port map(      address => address,
97                   instruction => instruction,
98                   bram_enable => bram_enable,
99                       port_id => open,--
100                 write_strobe => uart_write,
101                k_write_strobe => k_write_strobe,
102                     out_port => pb_out_data,
103                  read_strobe  => read_from_uart_rx,
104                      in_port=> pb_in_data,
105                    interrupt => '0',
106                interrupt_ack => open,
107                        sleep => kcpsm6_sleep,
108                        reset => '0',--
109                          clk => clk);
110
111
112
113     kcpsm6_sleep <= '0';
114
115
116     program_rom: receiver                    --Name to match your PSM file
117       generic map(             C_FAMILY => "7S",   --Family 'S6', 'V6' or '7S'
118                         C_RAM_SIZE_KWORDS => 2,      --Program size '1', '2' or '4'
119                      C_JTAG_LOADER_ENABLE => 1)      --Include JTAG Loader when set to '1'
120       port map(      address => address,
121                  instruction => instruction,
122                       enable => bram_enable,
123                          rdl => kcpsm6_reset,
124                          clk => clk);
```

```
123                          rdl => 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