Fall 2019

California State University, Northridge

Department of Electrical & Computer Engineering



Lab Experiment 1

*Exploring Xilinx Vivado IDE and Zedboard*

Juan Silva & Tumen Khureltulga

(Group 16)

September 17, 2019

ECE 524L

**Table of Contents**

**I. Introduction and Problem Statements…….………………………...…Page 2**

**II. Procedure……..…………………………………………….…………...Page 2**

**III. Testing Strategy………………………………….....………………......Page 3**

**IV. Results…………....…………………………………....…………….......Page 5**

**V. Analysis…………....……………………………....……………….…....Page 6**

**VI. Appendix……………...………………………....………………….….Page 8**

1. **Introduction and Problem Statements**

In this experiment, four 8-bit counters are designed using Vivado software tools and programmed onto an FPGA device located on a development board known as the Zedboard. The purpose of this lab is to design, test, and simulate each counter for peripheral use as well as communicate with the Zedboard. This is done by generating a bitstream and flashing fully synthesizable HDL code onto the development board. Therefore, being familiar with the Zedboard is vital to complete this experiment.  
 The ZedBoard is an evaluation and development board based on the Xilinx ZynqTM-7000 All Programmable SoC (AP SoC). It has combined a dual Corex-A9 Processing System (PS) with 85,000 Series-7 Programmable Logic (PL) cells. The Zynq-7000 AP SoC can be targeted for broad use in many applications. For this application, the LEDs, the user push buttons, and dip switches are used for this lab. Additionally, the on-board oscillator runs at a frequency of 100 MHz. This is enough information to complete the lab objectives.  
 Four 8-bit counters are to be designed and flashed onto the Zedboard: a bi-directional binary counter, a bi-directional ring counter, a Johnson counter, and a Fibonnaci counter. The results of each counter will be displayed via LEDs. Each counter must have the ability to pause, hold, and reset whenever a user pushes the corresponding button. Due to the onboard oscillator running at 100 Mhz, a frequency divider must be designed to allow the user to accurately observe the output results. This is achieved by reducing the clock frequency to 1 Hz for switching of the LEDs can be observed. In order to use the Zedboard peripherals, a constraint file must be included to configure the Zedboard pins.

1. **Procedure**

Preparation for this lab required reading the Zedboard user manual in order to be familiar with the development board. To realize the design of the project, a block diagram, shown on *Figure 1.1*, is designed to identify the necessary hardware needed to implement the project. To address the frequency problem, a frequency divider source was designed and a top-level design source was created to implement the frequency divider by instantiating it. The waveform results from the Vivado software are shown on *Figure 1.2* showing the frequency being divided. Afterwards, the counters are instantiated to the top-level design. Each counter was designed and tested using the Vivado simulator separately. Each counter has a truth table to verify its function. After each counter was verified, a switch case was used to select between each counter. Once completed, a constraint file was added to configure the Zedboard pins. The parameters to modify are the master clock, the LEDs, the push buttons, and the LEDs. Using the Vivado software to configure the pins and connecting a USB cord to the host computer, a bitstream was generated and the HDL code was programmed on the development board. The PROG light turns on and the code should execute.



**Figure 1.1 -** Top Level block diagram of counter design.

1. **Testing Strategy**

The testing strategy for this lab was broken down into two parts: verification using the Vivado software and verification on the Zedboard. First, the plan to test for every counter and the frequency divider was to design each component separately and use a testbench to verify each component functions correctly. The waveforms for every component are shown on the *Results* section. A criteria verifying the components is matching the output waveform to the component’s corresponding truth table on *Table 1.1.*Second, the plan to test the pin configuration on the Zedboard. The LEDs will be used to completely verify each counter. The push buttons are used to test for user inputs such as reset, direction, and pause. Two dip switches are used to switch between each counter. If each peripheral can operate as designed, then the design is complete. The table on *Table 1.2* is a simple functionality checklist to check for each functionality on the development board.

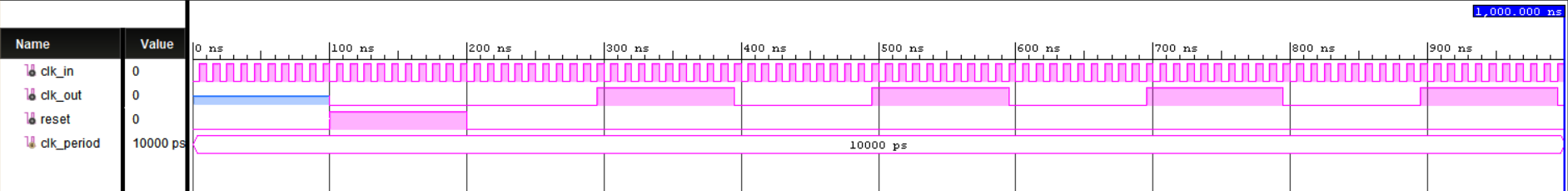
| The frequency divider must receive a clock input and output a slower clock frequency. |
| --- |
| The binary counter must count up or down in sequential fashion. |
| The ring counter must pass a value from the LSB to the MSB in either direction. |
| The Johnson counter must perform similar to a Ring Counter except have an inverted output. |
| The Fibonnaci counter must produce its Fibonacci sequences continuously. |

**Table 1.1**  - Truth table description for each component.

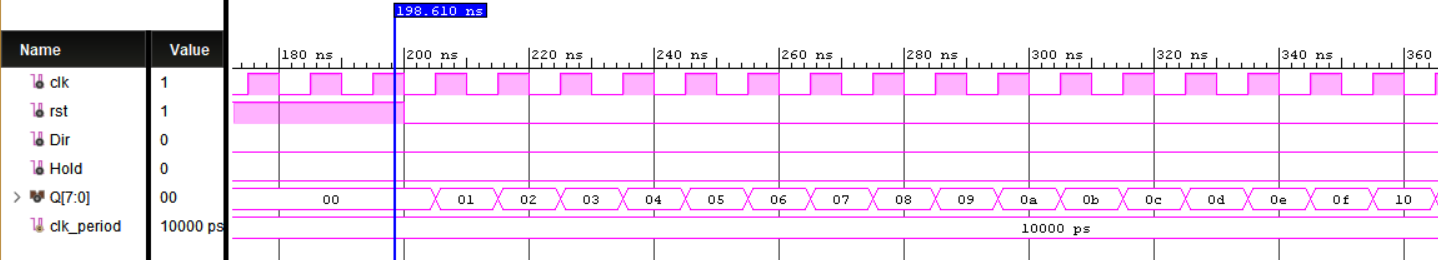
| The LEDs flash the correct counter logic using the clock divider frequency of 1 Hz. | X |
| --- | --- |
| DIP switches are assigned to select a counter. | X |
| A push button is assigned to reset each counter when pressed. | X |
| A push button is assigned to pause each counter when held down. | X |
| A push button is assigned to change the direction for each counter when pressed. | X |

**Table 1.2**  - Verification and Board Testing Checklist

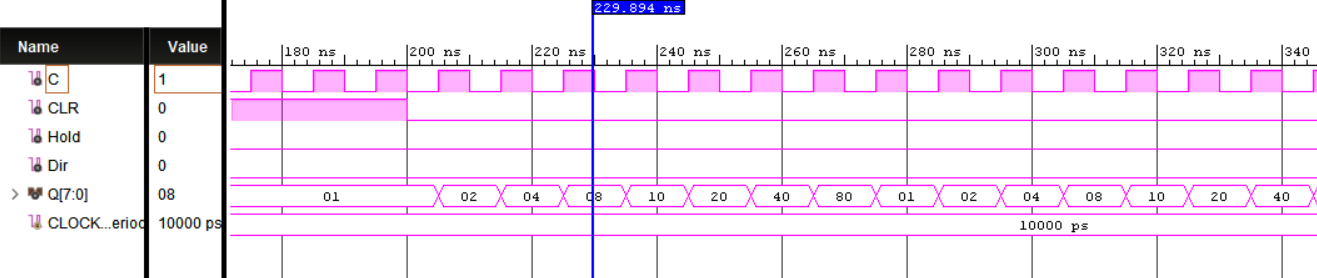
1. **Results & Data**



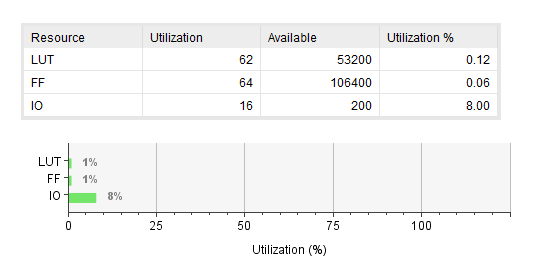
**Figure 1.2** - Frequency Divider Simulated Waveform



**Figure 1.3 -** Binary Counter Simulated Waveform



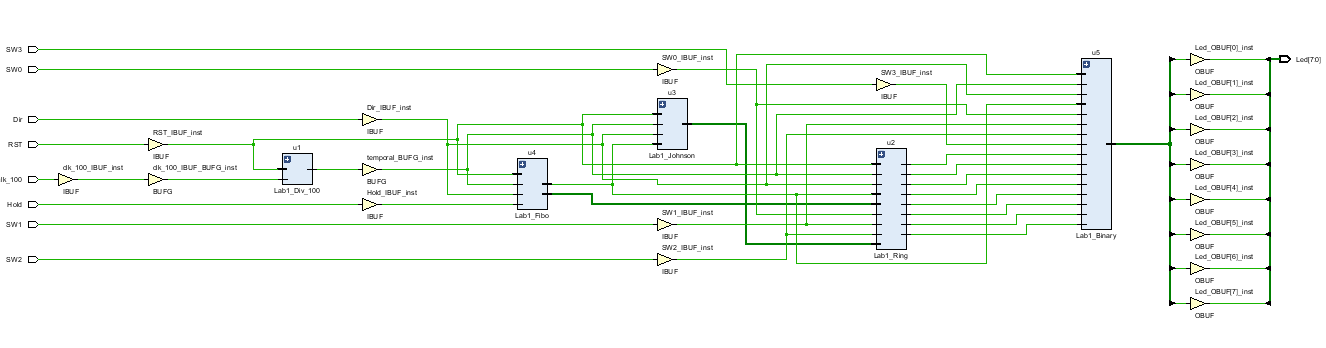
**Figure 1.4** - Ring Counter Simulated Waveform



**Figure 1.6** - Utilization Report

1. **Analysis**

The purpose of this lab was to be familiar with Vivado software tools, design HDL code, and understand how to communicate with the development board. This experiment serves as a tutorial in configuring peripherals as well as overall practice in designing and verifying code. Overall, the procedure and testing strategy went all according to plan. The decision to work on each component separately and verify it via Vivado software tools worked flawlessly. Some setbacks were encountered but we managed to stick to the plan and get the Zedboard to display correct results. Each component worked as intended and using a checklist helped in checking that all requirements were met. One thing to take from this experiment is reading user manuals to understand the hardware being used. Additionally, the use of a testbench is vital in ensuring the code will properly function. Using the Vivado simulator provided information on various possible user input combinations. This was useful when debugging the pin features to see what was working correctly and what wasn’t.  
 Errors and general hardships were encountered throughout the course of the lab. Having each lab member produce separate code for this experiment had its pros and cons. Although this methodology is inefficient, it did prove useful in a situation where one lab partner is late, the other can use their code to work on the lab. This problem is addressed by creating a cloud storage to have the code readily available for the group. Initial design for each counter was designed separately. A switch statement was implemented after each counter was verified. We did experience a few bugs when testing the source files. The Fibonnaci counter was initialized incorrectly and the process for the switch was using the master clock rather than the divided clock. Aside from external issues, the lab equipment did have its own issues. We discovered that the workstation was unable to communicate with the Zedboard. Initially, we believed that it was the hardware that couldn’t connect to the Zedboard. A bitstream was generated successfully but we were unable to program the development board. Switching Zedboards with another group solved the problem.   
 It concluded from the Utilization Report, Fig 1.5, that this lab used a very small part of the FPGA. It used 64 FFs out of possible 106400. It shows that ZynqTM-7000 is such a powerful chip that can handle various applications. FPGA is well suited to handle large amount of data, therefore, it is known to be used in communication and network applications. On the other hand, it is not very suited to control applications because control applications most of time do not need a large amount of data to be processed.  
 Below figure is the schematic generated by Vivado. If the following schematic was implemented using some other technology such as IC chips, it could have been a lot of work. FPGA is easier, quicker, and cost efficient.



**Figure 1.8** - Vivado Generated Schematic

1. **Appendix**

**Prelab Questions:**

1. The part numbers available for the FPGA is the Zynq-700 All Programmable SoC XC7Z020-CLG484-1. The Zynq-7000 represents the available 7 series programmable logic for the Zedboard.
2. The Zedboard has many interfaces such as LEDs, switches, an SD card, USB connections, and push buttons.
3. F21 is the FPGA pin for SW3.   
   R18 is the FPGA pin for the EAST push button.
4. The clock frequency is 100 Mhz. To reduce the clock frequency, implement a frequency divider HDL design source where the master clock is the input and the desired clock is the output.
5. A constraint file is available to configure the Zedboard. This is used to modify peripherals such as the LEDS and push buttons.
6. The jumper settings for programming the FPGA in JTAG mode are JP3 (USB Vbus capacitor setting), JP4 (CFGBVS Select), and JP13 (JTAG PS-RST)
7. .The Zedboard has 512 MB DDR3 and 256 Mb QSPI Flash memory.
8. The user push buttons are debounced. Debounc buttons poll for user input until the switch is closed meaning a user pressed a button.

*Component Source and Test Code*

| **Source Code** | **Testbench Code** |
| --- | --- |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity Lab1\_Div\_100 is  Port (  clk\_in : in STD\_LOGIC;  reset : in STD\_LOGIC;  clk\_out: out STD\_LOGIC  );  end Lab1\_Div\_100;  architecture Behavioral of Lab1\_Div\_100 is  signal temporal: STD\_LOGIC;  signal counter : integer range 0 to 4999999 := 0;  begin  frequency\_divider: process (reset, clk\_in) begin  if (reset = '1') then  temporal <= '0';  counter <= 0;  elsif rising\_edge(clk\_in) then  if (counter = 4999999) then  temporal <= NOT(temporal);  counter <= 0;  else  counter <= counter + 1;  end if;  end if;  end process;  clk\_out <= temporal;  end Behavioral; | library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity tb\_Lab1\_Div\_100 is  -- Port ( );  end tb\_Lab1\_Div\_100;  architecture Behavioral of tb\_Lab1\_Div\_100 is  COMPONENT Lab1\_Div\_100  PORT(  clk\_in : in STD\_LOGIC;  reset : in STD\_LOGIC;  clk\_out: out STD\_LOGIC  );  END COMPONENT;  constant clk\_period : time := 10 ns;  signal clk\_in,clk\_out : std\_logic;  signal reset : std\_logic:='0';    begin  clk\_process :process  begin  clk\_in <= '0';  wait for clk\_period/2;  clk\_in <= '1';  wait for clk\_period/2;  end process;    uut: Lab1\_Div\_100 PORT MAP (  clk\_in => clk\_in,  reset => reset,  clk\_out => clk\_out  );  stim\_proc: process  begin  wait for 100 ns;  reset <= '1';  wait for 100 ns;  reset <= '0';  wait;  end process;  end Behavioral; |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  use IEEE.numeric\_std.ALL;  use IEEE.STD\_LOGIC\_ARITH.ALL;  use IEEE.STD\_LOGIC\_UNSIGNED.ALL;  ENTITY Lab1\_Fibo IS  PORT (C, CLR,Dir,Hold: IN STD\_LOGIC;  Q3: out STD\_LOGIC\_VECTOR(7 downto 0));  END Lab1\_Fibo;  ------------------------------------------------------------  ARCHITECTURE Lab1\_Fibo OF Lab1\_Fibo IS  SIGNAL temp,temp1,temp2: std\_logic\_vector(7 downto 0):="00000000";  BEGIN  PROCESS (C,CLR,Dir,Hold)  BEGIN  IF (CLR='1') THEN  temp1 <= "00000001";  temp2 <= "00000000";  ELSIF (C'EVENT AND C='1') THEN  if(Dir='1') then  temp1<=temp2;  temp<=temp1;  else if (Hold='1') then  temp2<=temp2;  else  temp2 <= temp1;  temp1 <= temp;  end if;  END IF;  end if;  temp <= temp1+temp2;  END PROCESS;  Q3 <= temp2;  END Lab1\_Fibo; | library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity tb\_Lab1\_Fibo is  end tb\_Lab1\_Fibo;  architecture Behavioral of tb\_Lab1\_Fibo is  component HW0Q6  PORT (C, CLR : IN BIT;  Q3: out STD\_LOGIC\_VECTOR(7 downto 0));  end component;  signal C : bit := '1';  signal CLR:bit:='1';  signal Q3: STD\_LOGIC\_VECTOR(7 downto 0  begin  C <= not clk after 10 ns;  CLR<='0' after 40 ns;  uut:tb\_Lab1\_Fibo  PORT MAP (  C => C,  CLR => CLR,  Q3=>Q3  );  end Behavioral; |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity Lab1\_Johnson is  Port ( C : in STD\_LOGIC;  CLR,Dir,Hold : in STD\_LOGIC;  Q2 : out STD\_LOGIC\_VECTOR (7 downto 0));  end Lab1\_Johnson;  architecture Behavioral of Lab1\_Johnson is  signal q\_tmp: std\_logic\_vector(7 downto 0):= "00000000";  begin  process(C, CLR,Dir,Hold)  variable tmp:std\_logic:='0';  begin  if CLR = '1' then  q\_tmp <= "00000000";  elsif (C'event and C='1') then  if(Dir ='1') then  tmp:=not q\_tmp(0);  for i in 0 to 6 loop  q\_tmp(i)<=q\_tmp(i+1);  end loop;  q\_tmp(7)<=tmp;  elsif (Hold='1') then  q\_tmp<=q\_tmp;  else  tmp:=not q\_tmp(7);  for i in 0 to 6 loop  q\_tmp(i+1)<=q\_tmp(i);  end loop;  q\_tmp(0)<=tmp;  end if;  end if;  end process;  Q2 <= q\_tmp;  end Behavioral; | library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity Lab1\_Johnson is  Port ( C : in STD\_LOGIC;  CLR,Dir,Hold : in STD\_LOGIC;  Q2 : out STD\_LOGIC\_VECTOR (7 downto 0));  end Lab1\_Johnson;  architecture Behavioral of Lab1\_Johnson is  signal q\_tmp: std\_logic\_vector(7 downto 0):= "00000000";  begin  process(C, CLR,Dir,Hold)  variable tmp:std\_logic:='0';  begin  if CLR = '1' then  q\_tmp <= "00000000";  elsif (C'event and C='1') then  if(Dir ='1') then  tmp:=not q\_tmp(0);  for i in 0 to 6 loop  q\_tmp(i)<=q\_tmp(i+1);  end loop;  q\_tmp(7)<=tmp;  elsif (Hold='1') then  q\_tmp<=q\_tmp;  else  tmp:=not q\_tmp(7);  for i in 0 to 6 loop  q\_tmp(i+1)<=q\_tmp(i);  end loop;  q\_tmp(0)<=tmp;  end if;  end if;  end process;  Q2 <= q\_tmp;  end Behavioral; |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity Lab1\_Binary is  port(C, CLR,Dir,Hold : in std\_logic;  Q4 : out std\_logic\_vector(7 downto 0));  end Lab1\_Binary;  architecture bhv of Lab1\_Binary is  signal tmp: std\_logic\_vector(7 downto 0);  begin  process (C, CLR,Dir,Hold)  begin  if (CLR='1') then  tmp <= "00000000";  elsif (C'event and C='1') then  if(Dir ='1') then  tmp <= tmp - 1;  elsif (Hold='1') then  tmp<=tmp;  else  tmp <= tmp + 1;  end if;  end if;  end process;  Q4 <= tmp;  end bhv; | LIBRARY ieee;  USE ieee.std\_logic\_1164.ALL;    ENTITY tb\_Lab1\_Ring IS  END tb\_Lab1\_Ring;    ARCHITECTURE behavior OF tb\_Lab1\_Ring IS  COMPONENT Lab1\_Ring  PORT(  C : IN std\_logic;  CLR,Dir,Hold : IN std\_logic;  Q : OUT std\_logic\_vector(7 downto 0)  );  END COMPONENT;  signal C : std\_logic := '0';  signal CLR,Hold,Dir : std\_logic := '0';  signal Q : std\_logic\_vector(7 downto 0);  constant CLOCK\_period : time := 10 ns;  BEGIN  uut: Lab1\_Ring PORT MAP (  C => C,  CLR => CLR,  Hold=>Hold,  Dir=>Dir,  Q => Q  );    CLOCK\_process :process  begin  C <= '0';  wait for CLOCK\_period/2;  C <= '1';  wait for CLOCK\_period/2;  end process;  stim\_proc: process  begin  wait for 100 ns;  CLR <= '1';  wait for 100 ns;  CLR <= '0';  wait;  end process;  END; |

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity top\_module is

Port (Led:out std\_logic\_vector(7 downto 0);

RST,clk\_100,Dir,Hold,SW0,SW1,SW2,SW3:in std\_logic);

end top\_module;

architecture Behavioral of top\_module is

component Lab1\_Ring

port (C,CLR,Dir,Hold: in STD\_LOGIC;

Q1: out STD\_LOGIC\_VECTOR(7 downto 0));

end component;

component Lab1\_Johnson

port (C,CLR,Dir,Hold: in STD\_LOGIC;

Q2: out STD\_LOGIC\_VECTOR(7 downto 0));

end component;

component Lab1\_Fibo

port (C,CLR,Dir,Hold: in STD\_LOGIC;

Q3: out STD\_LOGIC\_VECTOR(7 downto 0));

end component;

component Lab1\_Binary

port (C,CLR,Dir,Hold: in STD\_LOGIC;

Q4: out STD\_LOGIC\_VECTOR(7 downto 0));

end component;

component Lab1\_Div\_100

port(clk\_in : in STD\_LOGIC;

reset : in STD\_LOGIC;

clk\_out: out STD\_LOGIC);

end component;

signal clk\_int:std\_logic;

signal SW:std\_logic\_vector(3 DOWNTO 0);

SIGNAL OUTPUT1,OUTPUT2,OUTPUT3,OUTPUT4:STD\_LOGIC\_VECTOR(7 DOWNTO 0);

begin

SW(0)<=SW0;

SW(1)<=SW1;

SW(2)<=SW2;

SW(3)<=SW3;

u1:Lab1\_Div\_100

port map(clk\_in=>clk\_100,reset=>RST,clk\_out=>clk\_int);

u2:Lab1\_Ring

port map(C=>clk\_int,CLR=>RST,Q1=>OUTPUT1,Dir=>Dir,Hold=>Hold);

u3:Lab1\_Johnson

port map(C=>clk\_int,CLR=>RST,Q2=>OUTPUT2,Dir=>Dir,Hold=>Hold);

u4:Lab1\_Fibo

port map(C=>clk\_int,CLR=>RST,Q3=>OUTPUT3,Dir=>Dir,Hold=>Hold);

u5:Lab1\_Binary

port map(C=>clk\_int,CLR=>RST,Q4=>OUTPUT4,Dir=>Dir,Hold=>Hold);

PROCESS(SW)

BEGIN

CASE SW IS

WHEN "0001" => Led<=OUTPUT1;

WHEN "0010" => Led<=OUTPUT2;

WHEN "0100" => Led<=OUTPUT3;

WHEN "1000" => Led<=OUTPUT4;

WHEN OTHERS => Led<="00000000";

END CASE;

END PROCESS;

end Behavioral;

*Constraint Code*

set\_property PACKAGE\_PIN Y9 [get\_ports {clk\_100}]; # "GCLK"

set\_property PACKAGE\_PIN T22 [get\_ports {Led[0]}]; # "LD0"

set\_property PACKAGE\_PIN T21 [get\_ports {Led[1]}]; # "LD1"

set\_property PACKAGE\_PIN U22 [get\_ports {Led[2]}]; # "LD2"

set\_property PACKAGE\_PIN U21 [get\_ports {Led[3]}]; # "LD3"

set\_property PACKAGE\_PIN V22 [get\_ports {Led[4]}]; # "LD4"

set\_property PACKAGE\_PIN W22 [get\_ports {Led[5]}]; # "LD5"

set\_property PACKAGE\_PIN U19 [get\_ports {Led[6]}]; # "LD6"

set\_property PACKAGE\_PIN U14 [get\_ports {Led[7]}]; # "LD7"

set\_property PACKAGE\_PIN N15 [get\_ports {RST}]; # "BTNL"

set\_property PACKAGE\_PIN R18 [get\_ports {Dir}]; # "BTNR"

set\_property PACKAGE\_PIN T18 [get\_ports {Hold}]; # "BTNU"

set\_property PACKAGE\_PIN N15 [get\_ports {RST}]; # "BTNL"

set\_property PACKAGE\_PIN R18 [get\_ports {Dir}]; # "BTNR"

set\_property PACKAGE\_PIN T18 [get\_ports {Hold}]; # "BTNU"

set\_property IOSTANDARD LVCMOS33 [get\_ports -of\_objects [get\_iobanks 33]];

set\_property IOSTANDARD LVCMOS18 [get\_ports -of\_objects [get\_iobanks 34]];

set\_property IOSTANDARD LVCMOS18 [get\_ports -of\_objects [get\_iobanks 35]];

set\_property IOSTANDARD LVCMOS33 [get\_ports -of\_objects [get\_iobanks 13]];