Lab 6: Microprocessor Based System

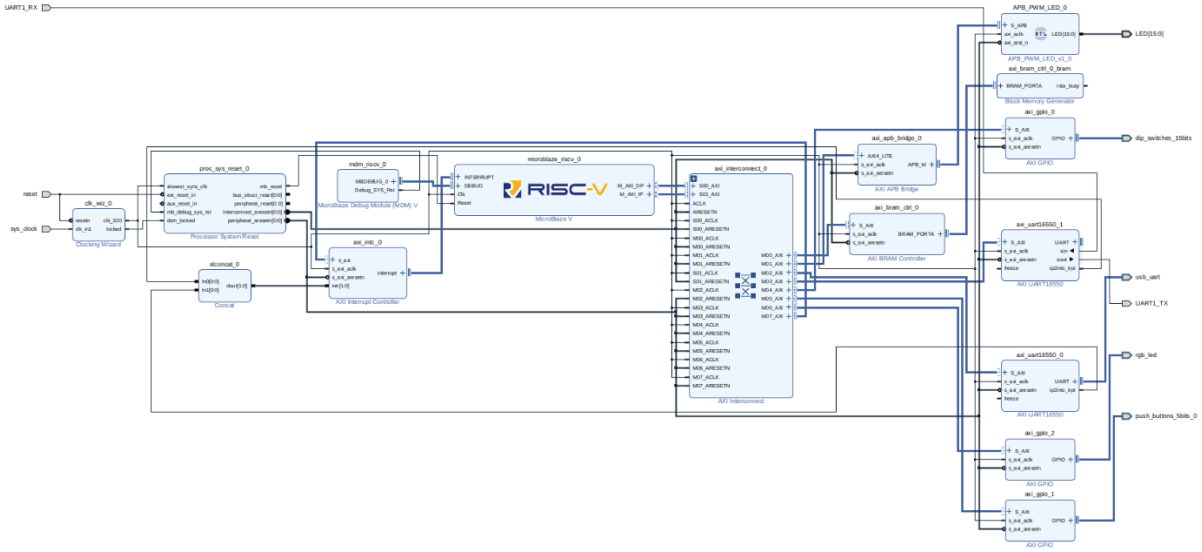
-Overview-

In this lab, I will create a microprocessor-based digital system using Vivado's Block Design tools, integrating a custom PWM LED controller. My task involves designing and connecting all specified components, including my LED controller, which will interface with the system via an APB (Advanced Peripheral Bus) interface. After building the block design, I will test the PWM LED controller in simulation using a provided APB testbench. Once verified, I will generate a bitstream, load it onto the FPGA, and run instructor-provided C/Assembly code to control the hardware.

-Design-

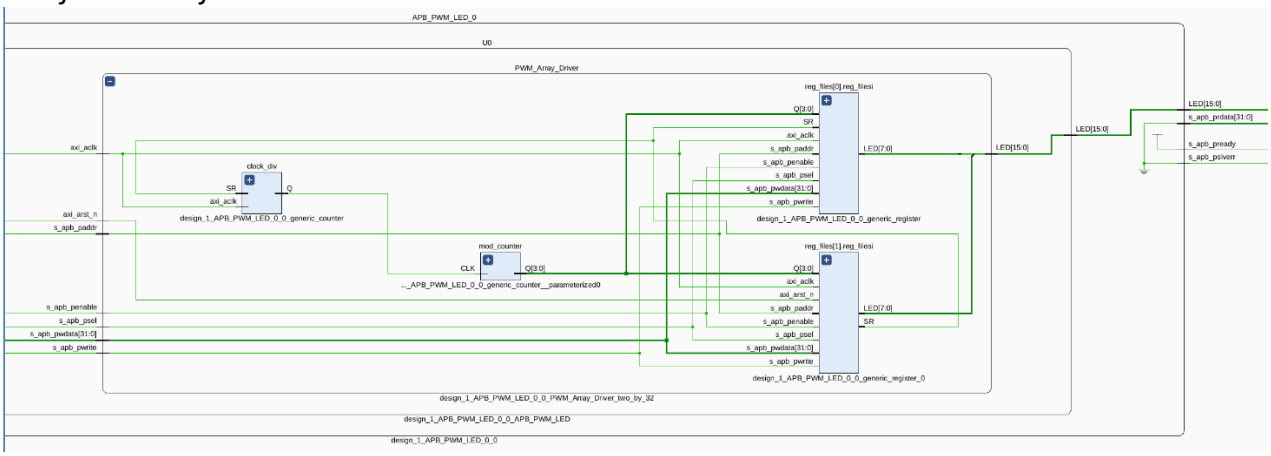
Block Diagram

To drive the APB_PWM_LED entity, the microprocessor communicates through the AXI interconnect, through the AXI APB bridge, then to the APB_PWM_LED.



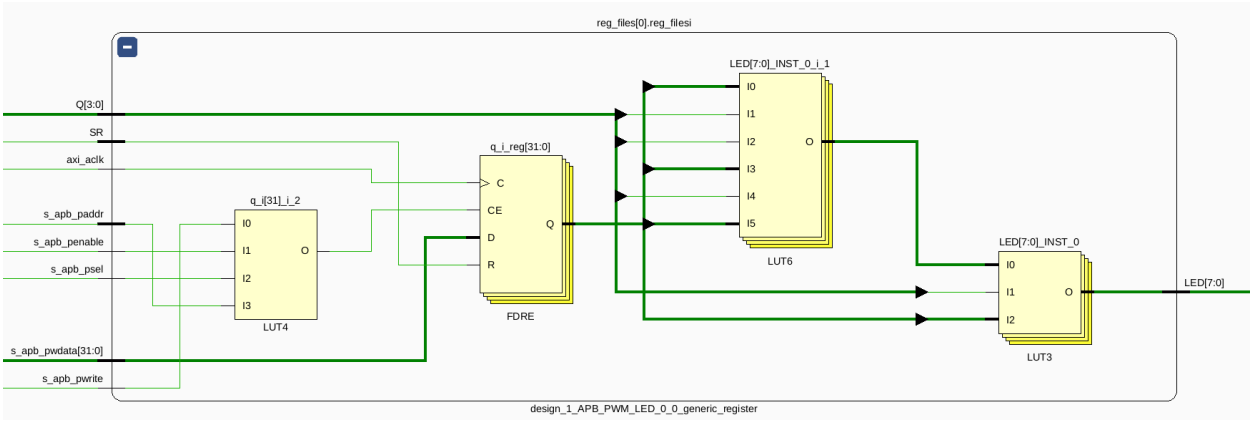
APB_PWM_LED Schematic

To interface the PWM_Array_Driver with the APB bus, the obvious connections were made such as clk to axi_clk, reset was connected to not axi_arst_n (that inverter is in one of the register files), and data in was connected to pwdata. For the write enable of the PWM_Array_Driver, I ANDed the psel, penable, and pwrite signals together because when all those signals are high, that means there is valid write data. For the address, I used the 3rd to left bit of the address as that is the next address after a 32 bit number which is the size of each word of data getting written. Finally, I tied pready to 1 so the entity would always be ready.



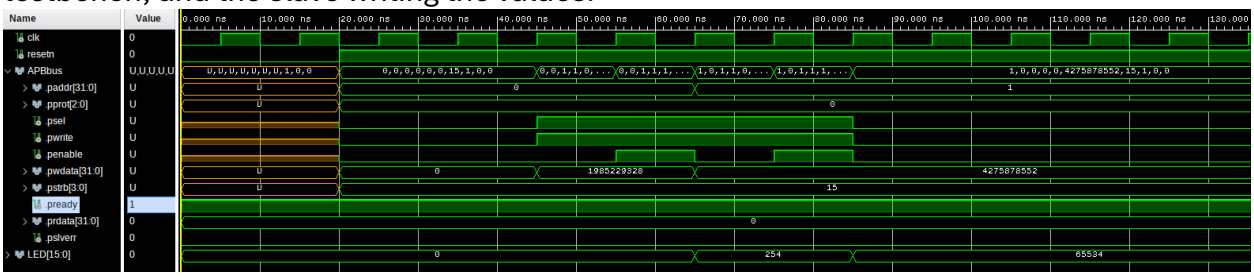
Reg_File Schematic

The first lookup table is the implementation of ANDing psel, penable, and pwrite and decoding the address. The last to 2 lookup tables are the implementation of the comparator of the stored value and the modulo_counter value.

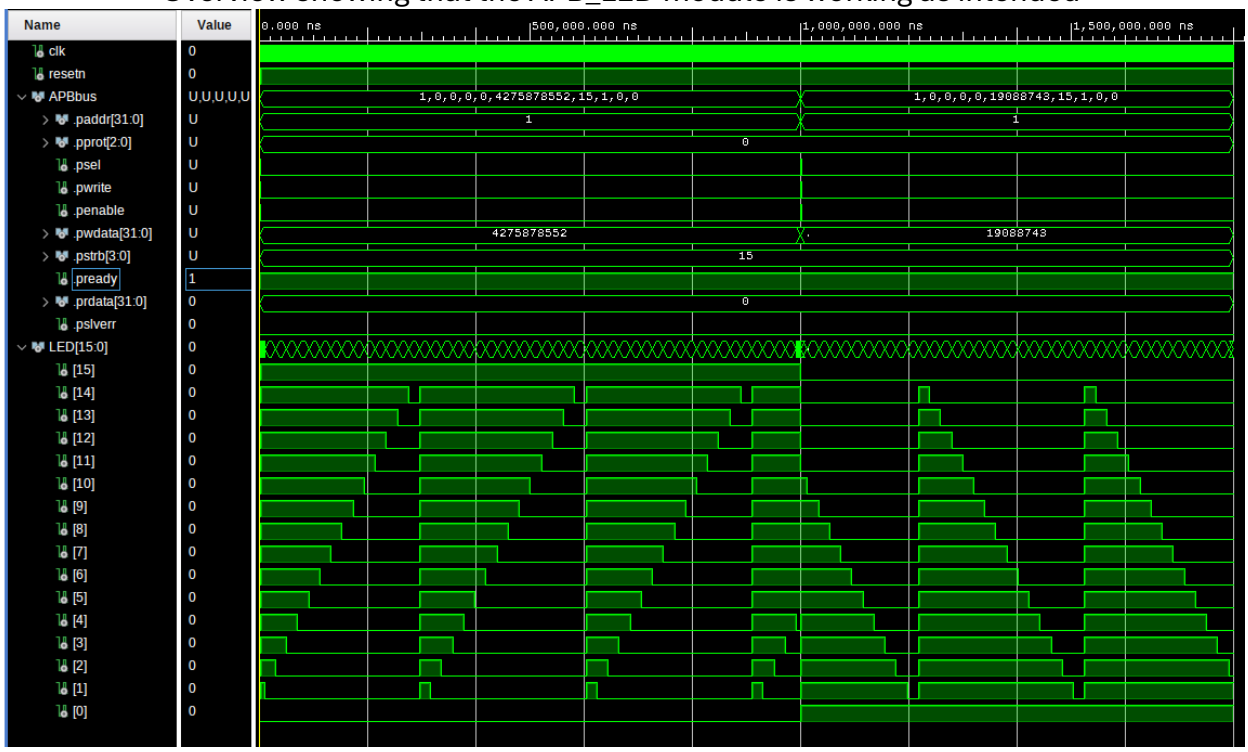


-Simulation –

Showing the initial handshake between the master signals provided by the testbench, and the slave writing the values.

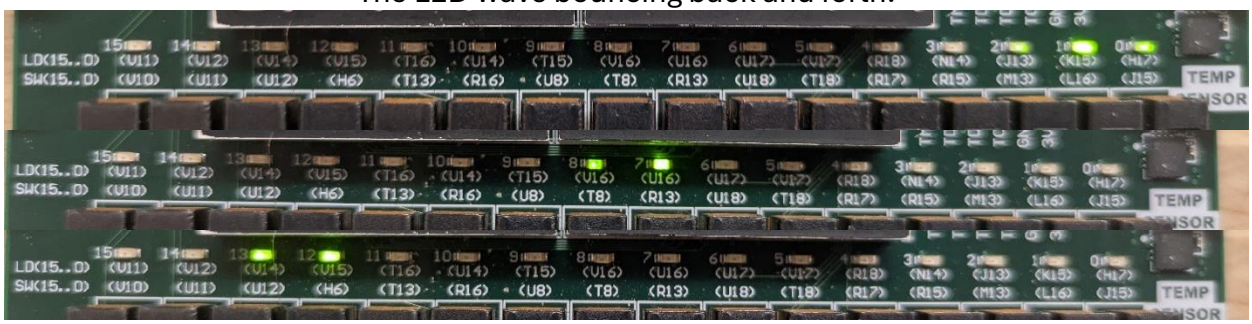


Overview showing that the APB_LED module is working as intended



-Implemented Design Verification-

The LED wave bouncing back and forth.



-Conclusion-

My biggest hiccup in this lab was that I initially didn't have the pwrite signal in consideration when implementing my PWM_Array_Driver to an APB device. When I had the signal unconnected, the top 8 bits of the device were not being written to properly. The processer was setting zeros to the pwdata bus when pwrite was low. My device was writing those values when is shouldn't have, breaking the top 8 bits. Other then that hiccup, the implementation went fairly smoothly, and the cylon eye worked as intended.

-Appendix-

Main VHDL

```
14
15  library IEEE;
16  use IEEE.STD_LOGIC_1164.ALL;
17
18  entity APB_PWM_LED is
19    port (
20      -- Clock and reset
21      axi_aclk        :in  std_logic;
22      axi_arst_n      :in  std_logic;
23      -- The APB port
24      s_apb_paddr     :in  std_logic_vector(31 downto 0);
25      s_apb_psel      :in  std_logic;
26      s_apb_penable   :in  std_logic;
27      s_apb_pwrite    :in  std_logic;
28      s_apb_pwdata    :in  std_logic_vector(31 downto 0);
29      s_apb_pready    :out std_logic := '0';
30      s_apb_prdata    :out std_logic_vector(31 downto 0) := (others => '0');
31      s_apb_pslverr   :out std_logic := '0';
32      -- Port to drive the LEDs
33      LED             :out std_logic_vector(15 downto 0)
34      );
35    ATTRIBUTE X_INTERFACE_INFO : STRING;
36  end APB_PWM_LED;
37
38
39  -- Fill in your architecture here.  This one just sends the data
40  -- register to the LED's.  It is a GPO device (GPIO without the I).
41  -- It gives attributes to the APB ports so that Vivado can create an
42  -- interface for the block design and let you easily connect it with
43  -- the GUI.
44  architecture Behavioral of APB_PWM_LED is
45    ATTRIBUTE X_INTERFACE_INFO of s_apb_paddr :SIGNAL is
46      "xilinx.com:interface:apb:1.0 S_APB PADDR";
47    ATTRIBUTE X_INTERFACE_INFO of s_apb_psel     :SIGNAL is
48      "xilinx.com:interface:apb:1.0 S_APB PSEL";
49    ATTRIBUTE X_INTERFACE_INFO of s_apb_penable  :SIGNAL is
50      "xilinx.com:interface:apb:1.0 S_APB PENABLE";
51    ATTRIBUTE X_INTERFACE_INFO of s_apb_pwrite   :SIGNAL is
52      "xilinx.com:interface:apb:1.0 S_APB PWRITE";
53    ATTRIBUTE X_INTERFACE_INFO of s_apb_pwdata   :SIGNAL is
54      "xilinx.com:interface:apb:1.0 S_APB PWDATA";
55    ATTRIBUTE X_INTERFACE_INFO of s_apb_pready   :SIGNAL is
56      "xilinx.com:interface:apb:1.0 S_APB PREADY";
57    ATTRIBUTE X_INTERFACE_INFO of s_apb_prdata   :SIGNAL is
58      "xilinx.com:interface:apb:1.0 S_APB PRDATA";
59    ATTRIBUTE X_INTERFACE_INFO of s_apb_pslverr  :SIGNAL is
60      "xilinx.com:interface:apb:1.0 S_APB PSLVERR";
61
62    -- Define any signals that your architecture needs.
63      signal axi_arst, enable: std_logic;
64  begin
65
66  enable <= s_apb_psel and s_apb_penable and s_apb_pwrite;
67  axi_arst <= not(axi_arst_n);
68
69  PWM_Array_Driver: entity work.PWM_Array_Driver ( two_by_32 )
70      Port map ( clk => axi_aclk,
71                 wen => enable,
72                 reset => axi_arst,
73                 adr => s_apb_paddr(2 downto 2),
74                 d => s_apb_pwdata,
75                 leds => LED);
76
77  --APB Outputs
78  s_apb_pready <= '1';
79  s_apb_pslverr <= '0';
80  s_apb_prdata <= (others => '0');
81
82  end Behavioral;
83
```

Wrapper VHDL

```
11  library IEEE;
12  use IEEE.STD_LOGIC_1164.ALL;
13  library UNISIM;
14  use UNISIM.VCOMPONENTS.ALL;
15  entity design_1_wrapper is
16    port (
17      LED : out STD_LOGIC_VECTOR ( 15 downto 0 );
18      UART1_RX : in STD_LOGIC;
19      UART1_TX : out STD_LOGIC;
20      dip_switches_16bits_tri_i : in STD_LOGIC_VECTOR ( 15 downto 0 );
21      push_buttons_5bits_0_tri_i : in STD_LOGIC_VECTOR ( 4 downto 0 );
22      reset : in STD_LOGIC;
23      rgb_led_tri_o : out STD_LOGIC_VECTOR ( 5 downto 0 );
24      sys_clock : in STD_LOGIC;
25      usb_uart_ctsn : in STD_LOGIC;
26      usb_uart_rtsn : out STD_LOGIC;
27      usb_uart_rxd : in STD_LOGIC;
28      usb_uart_txd : out STD_LOGIC
29    );
30  end design_1_wrapper;
31
32  architecture STRUCTURE of design_1_wrapper is
33    component design_1 is
34    port (
35      usb_uart_ctsn : in STD_LOGIC;
36      usb_uart_rtsn : out STD_LOGIC;
37      usb_uart_rxd : in STD_LOGIC;
38      usb_uart_txd : out STD_LOGIC;
39      dip_switches_16bits_tri_i : in STD_LOGIC_VECTOR ( 15 downto 0 );
40      rgb_led_tri_o : out STD_LOGIC_VECTOR ( 5 downto 0 );
41      push_buttons_5bits_0_tri_i : in STD_LOGIC_VECTOR ( 4 downto 0 );
42      sys_clock : in STD_LOGIC;
43      reset : in STD_LOGIC;
44      LED : out STD_LOGIC_VECTOR ( 15 downto 0 );
45      UART1_TX : out STD_LOGIC;
46      UART1_RX : in STD_LOGIC
47    );
48    end component design_1;
49  begin
50  design_1_i: component design_1
51       port map (
52        LED(15 downto 0) => LED(15 downto 0),
53        UART1_RX => UART1_RX,
54        UART1_TX => UART1_TX,
55        dip_switches_16bits_tri_i(15 downto 0) => dip_switches_16bits_tri_i(15 downto 0),
56        push_buttons_5bits_0_tri_i(4 downto 0) => push_buttons_5bits_0_tri_i(4 downto 0),
57        reset => reset,
58        rgb_led_tri_o(5 downto 0) => rgb_led_tri_o(5 downto 0),
59        sys_clock => sys_clock,
60        usb_uart_ctsn => usb_uart_ctsn,
61        usb_uart_rtsn => usb_uart_rtsn,
62        usb_uart_rxd => usb_uart_rxd,
63        usb_uart_txd => usb_uart_txd
64      );
65  end STRUCTURE;
66
```

Testbench VHDL

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.APB_test_package.all;

entity APB_PWM_LED_testbench is
end entity;

architecture behavioral of APB_PWM_LED_testbench is
  signal clk,resetn : std_logic := '0';
  signal APBbus : APB_bus_t;
  signal LED : std_logic_vector(15 downto 0);
begin


  clk <= not clk after 5 ns;
  resetn <= '1' after 20 ns;

  uut: entity work.APB_PWM_LED (behavioral)
    port map(
      -- Clock and reset
      axi_aclk        => clk,
      axi_arst_n      => resetn,
      -- APB bus
      s_apb_paddr     => APBbus.paddr,
      s_apb_psel      => APBbus.psel,
      s_apb_penable   => APBbus.penable,
      s_apb_pwrite    => APBbus.pwrite,
      s_apb_pwdata    => APBbus.pwdata,
      s_apb_pready    => APBbus.pready,
      s_apb_prdata    => APBbus.prdata,
      s_apb_pslverr   => APBbus.pslverr,
      -- Port to drive the LEDs
      LED             => LED
      );

  test: process
  begin
    APBbus.pready <= 'Z';
    APBbus.prdata <= (others => 'Z');
    APBbus.pslverr <= 'Z';
    APB_reset_wait(clk,resetn,APBbus);
    wait for 20 ns;

    loop

      APB_write(clk, x"00000000", x"76543210", APBbus);
      APB_write(clk, x"00000001", x"FEDCBA98", APBbus);

      wait for 1 ms;

      APB_write(clk, x"00000000", x"89ABCDEF", APBbus);
      APB_write(clk, x"00000001", x"01234567", APBbus);

      wait for 100 ms;
    end loop;

  end process;

end architecture;
```

Constraint File

```
set_property CFGBVS VCCO [current_design]
#where value1 is either VCCO or GND

set_property CONFIG_VOLTAGE 3.3 [current_design]
#where value2 is the voltage provided to configuration bank 0

##Switches
#set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {switches[0]}]
#set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports {switches[1]}]
#set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports {switches[2]}]
#set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports {switches[3]}]
#set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports {switches[4]}]
#set_property -dict {PACKAGE_PIN T18 IOSTANDARD LVCMOS33} [get_ports {switches[5]}]
#set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {switches[6]}]
#set_property -dict {PACKAGE_PIN R13 IOSTANDARD LVCMOS33} [get_ports {switches[7]}]
#set_property -dict {PACKAGE_PIN T8 IOSTANDARD LVCMOS18} [get_ports {switches[8]}]
#set_property -dict {PACKAGE_PIN U8 IOSTANDARD LVCMOS18} [get_ports {switches[9]}]
#set_property -dict {PACKAGE_PIN R16 IOSTANDARD LVCMOS33} [get_ports {switches[10]}]
#set_property -dict {PACKAGE_PIN T13 IOSTANDARD LVCMOS33} [get_ports {switches[11]}]
#set_property -dict {PACKAGE_PIN H6 IOSTANDARD LVCMOS33} [get_ports {switches[12]}]
#set_property -dict {PACKAGE_PIN U12 IOSTANDARD LVCMOS33} [get_ports {switches[13]}]
#set_property -dict {PACKAGE_PIN U11 IOSTANDARD LVCMOS33} [get_ports {switches[14]}]
#set_property -dict {PACKAGE_PIN V10 IOSTANDARD LVCMOS33} [get_ports {switches[15]}]

## LEDs
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {LED[0]}]
set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports {LED[1]}]
set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports {LED[2]}]
set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports {LED[3]}]
set_property -dict {PACKAGE_PIN R18 IOSTANDARD LVCMOS33} [get_ports {LED[4]}]
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {LED[5]}]
set_property -dict {PACKAGE_PIN U17 IOSTANDARD LVCMOS33} [get_ports {LED[6]}]
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {LED[7]}]
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports {LED[8]}]
set_property -dict {PACKAGE_PIN T15 IOSTANDARD LVCMOS33} [get_ports {LED[9]}]
set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {LED[10]}]
set_property -dict {PACKAGE_PIN T16 IOSTANDARD LVCMOS33} [get_ports {LED[11]}]
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {LED[12]}]
set_property -dict {PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports {LED[13]}]
set_property -dict {PACKAGE_PIN V12 IOSTANDARD LVCMOS33} [get_ports {LED[14]}]
set_property -dict {PACKAGE_PIN V11 IOSTANDARD LVCMOS33} [get_ports {LED[15]}]
```