

San Jose State University
Department of Computer Engineering

CMPE 140 Lab Report

Lab 6 Report

Title Processor Design (2): Processor Hardware Validation

Semester Spring 2019

Date 03/20/19

by



Name Nickolas Schiffer

SID 012279319

Name Salvatore Nicosia

SID 012013599

Lab Checkup Record

Week	Performed By (signature)	Checked By (signature)	Tasks Successfully Completed*	Tasks Partially Completed*	Tasks Failed or Not Performed*
1	 SN		100%		

* Detailed descriptions must be given in the report.

I. INTRODUCTION

The purpose of this lab is to learn processor hardware validation for the ten-instruction 32-bit single-cycle MIPS processor using the Nexys 4 DDR FPGA board.

II. TESTING PROCEDURE

The source code provided (See Appendix) for the initial version of the single-cycle MIPS processor was used to setup the validation environment on the Nexys 4 board as shown in *Figure 1*. A push button generated clock on the board was used to execute each instruction of the test program. Switches 0 to 2 were used to access the content of the relevant registers \$v0, \$v1, \$a0, \$a1, and \$a3. Switches 5 to 7 were used to access the signals applied to the data memory which include the instr, progr, pc_current, wd_dm, and alu_out. These contents were verified through the eight 7-segment LEDs display on board. The results were recorded in a table and were compared to the test log from assignment 2 to corroborate the physical implementation. The validation record table can be found in the testing result section as well as the FPGA results implementation.

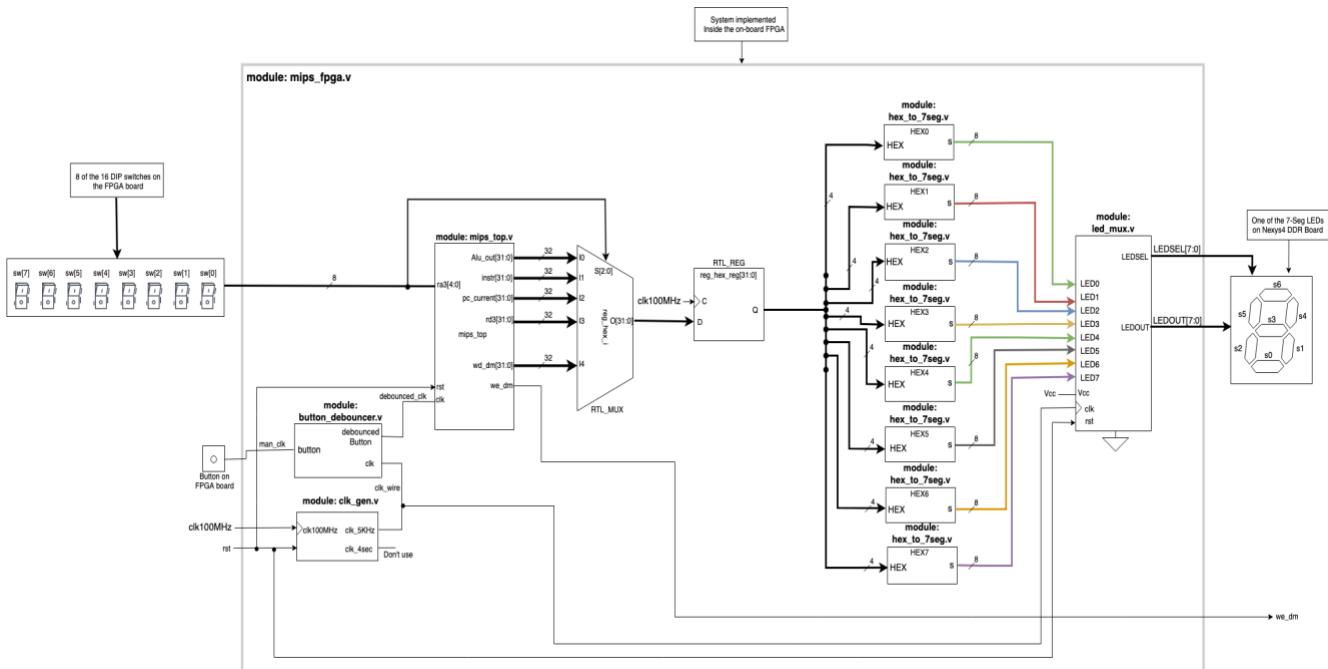


Figure 1: Validation environment setup for MIPS processor

III. TESTING RESULTS

The sample program resides in memory. As the clock is triggered, the instructions are executed accordingly. Using the switch mapping shown in *Legend*, the contents of the registers, current instruction, program counter, and memory were viewed and verified to match the *Table 1*.

Table 1. Validation Record Table MIPS Processor

Adr	Expected Machine Code	Actual Machine Code	PC	Registers				
				\$v0	\$v1	\$a0	\$a1	\$a3
00	20020005	0x20020005	00000	0	0	0	0	0
04	2003000c	0x2003000C	00004	00000005	0	0	0	0
08	2067fff7	0x2067FFF7	00008	00000005	0000000C	0	0	0
0c	00e22025	0x00E22025	0000C	00000005	0000000C	0	0	00000003
10	00642824	0x00642824	00010	00000005	00000005	00000007	0	00000003
14	00a42820	0x00A42820	00014	00000005	0000000C	00000007	00000004	00000003
18	10a7000a	0x10E5000A	00018	00000005	0000000C	00000007	0000000B	00000003
1c	0064202a	0x0064202A	0001C	00000005	0000000C	00000007	0000000B	00000003
20	10800001	0x10040001	00020	00000005	0000000C	00000000	0000000B	00000003
24	20050000	-	-	-	-	-	-	-
28	00e2202a	0x00E2202A	00028	00000005	0000000C	00000000	0000000B	00000003
2c	00853820	0x00853820	0002C	00000005	0000000C	00000001	0000000B	00000003
30	00e23822	0x00E23822	00030	00000005	0000000C	00000001	0000000B	0000000C
34	ac670044	0xAC670044	00034	00000005	0000000C	00000001	0000000B	00000007
38	8c020050	0x8C020050	00038	00000005	0000000C	00000001	0000000B	00000007
3c	08000011	0x08000011	0003C	00000007	0000000C	00000001	0000000B	00000007
40	20020001	-	-	-	-	-	-	-
44	ac020054	0xAC020054	00044	00000007	0000000C	00000001	0000000B	00000007
48	08000000	0x08000000	00048	00000007	0000000C	00000001	0000000B	00000007

Legend:

Switches [0:3] on FPGA	Switches [5:7] on FPGA
010 : \$v0	000 : instr
011 : \$v1	010 : alu_out
100 : \$a0	011 : wd_dm
101 : \$a1	1xx: program_counter
111 : \$a3	

Figures 2 to 8 show the execution results of the instruction 0x00A42820. The results displayed in the pictures show the content of the registers \$v0, \$v1, \$a0, \$a1, \$a3 as well as the program counter and instruction. The results achieved matched the test log from assignment 2.

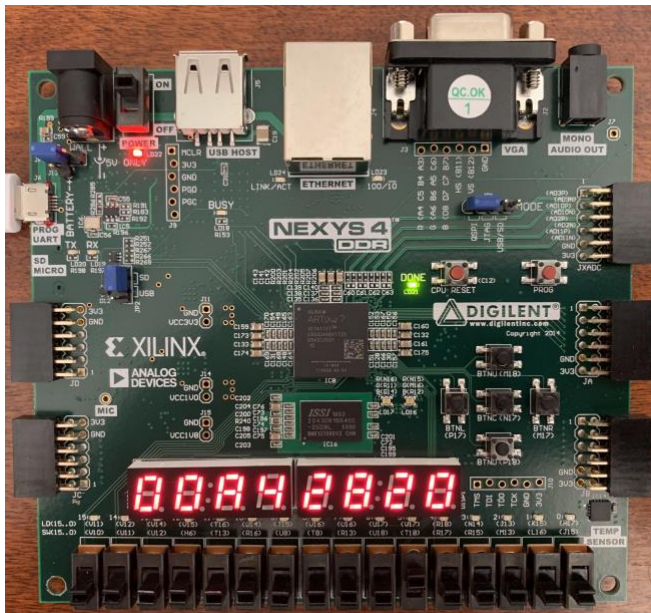


Figure 2. Instruction 0x00A42820

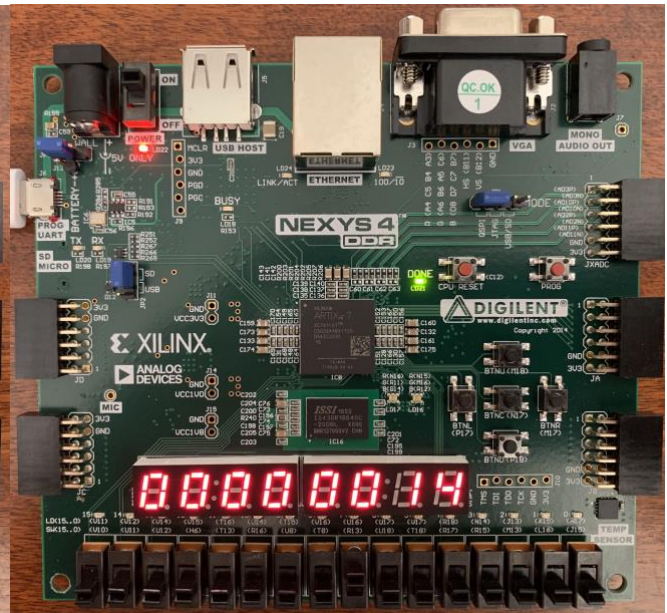


Figure 3. Current program counter = 14

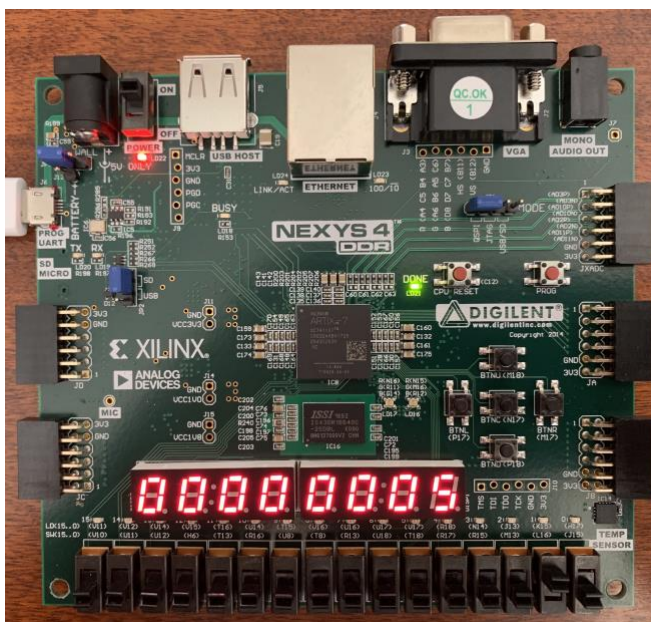


Figure 4. \$v0 register content = 5

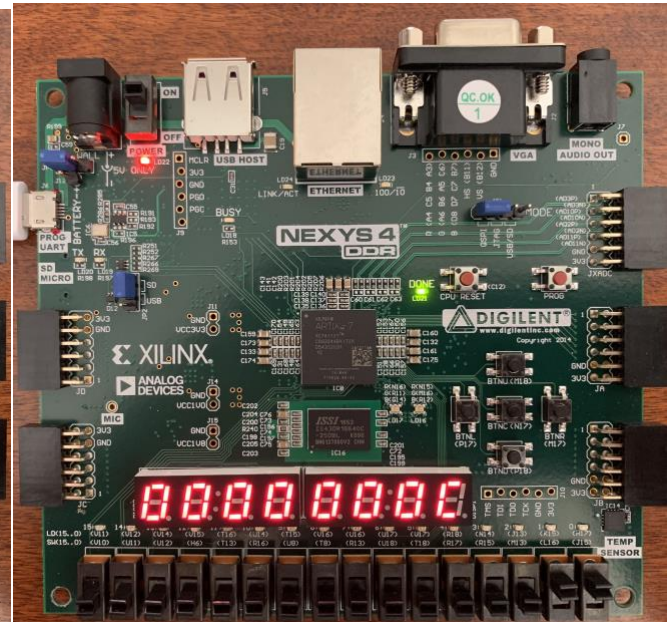


Figure 5. \$v1 register content = C

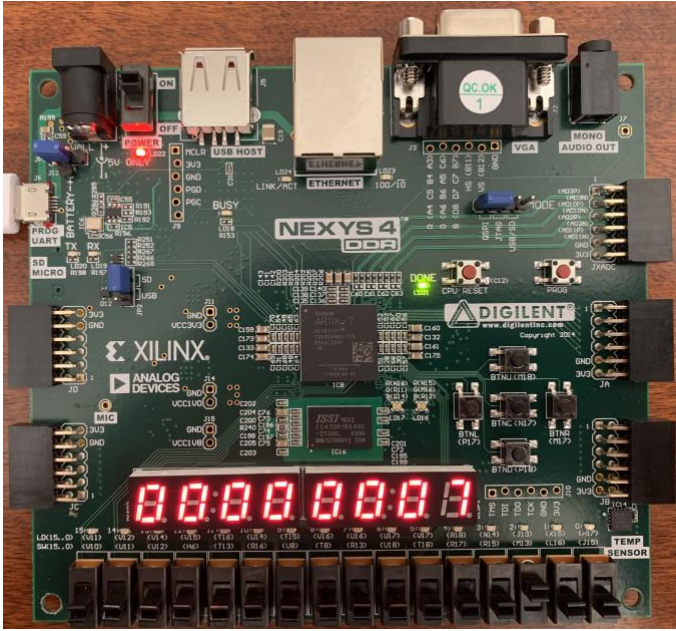


Figure 6. $\$a0$ register content = 7

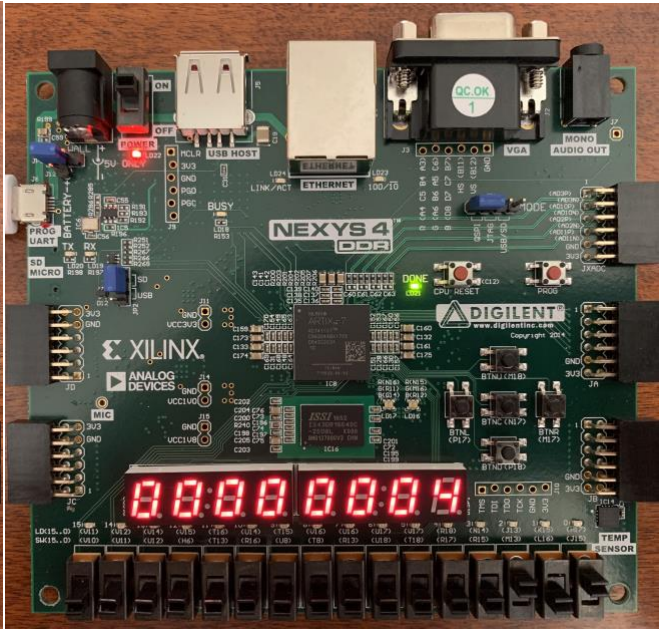


Figure 7. $\$a1$ register content = 4

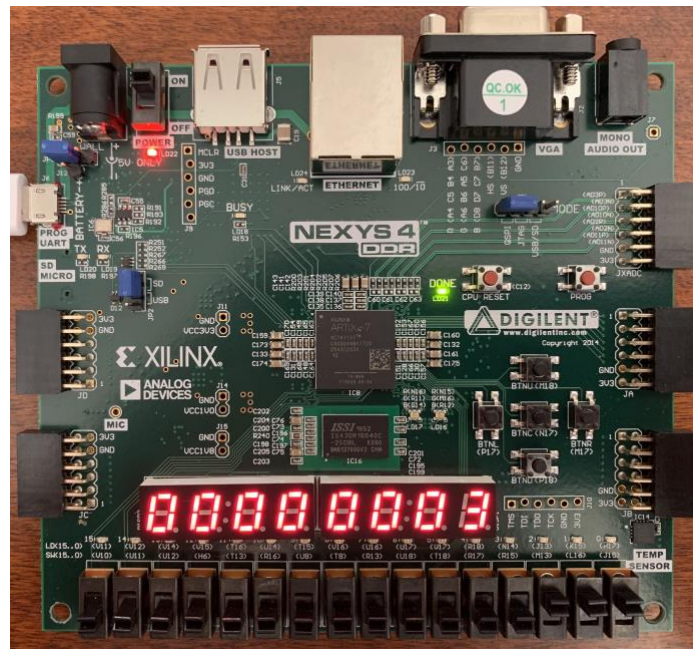


Figure 8. $\$a3$ register content = 3

IV. CONCLUSION

This lab allowed us to verify that the results of the MIPS software testbench match the functionality of the physical implementation on the Nexys 4 DDR development board.

V. SUCCESSFUL TASKS

1. Create a block diagram that illustrates the environment validation setup.
2. Create a validation record table that verifies that the execution of the sample program behaves identically to the execution of the same program software simulation.

VI. APPENDIX

A. SOURCE CODE:

mips_fpga.v

```
module mips_fpga (
    input wire      clk100MHz,
    input wire      rst,
    input wire      button,
    input wire [7:0] switches,
    output wire     we_dm,
    output wire [7:0] LEDSEL,
    output wire [7:0] LEDOUT
);

    reg [31:0] reg_hex;
    wire      clk_sec;
    wire      clk_5KHz;
    wire      clk_pb;

    wire [7:0] digit0;
    wire [7:0] digit1;
    wire [7:0] digit2;
    wire [7:0] digit3;
    wire [7:0] digit4;
    wire [7:0] digit5;
    wire [7:0] digit6;
    wire [7:0] digit7;

    wire [31:0] pc_current;
    wire [31:0] instr;
    wire [31:0] alu_out;
    wire [31:0] wd_dm;
    wire [31:0] rd_dm;
    wire [31:0] dispData;

    clk_gen clk_gen (
        .clk100MHz      (clk100MHz),
        .rst             (rst),
        .clk_4sec        (clk_sec),
        .clk_5KHz        (clk_5KHz)
    );

    button_debouncer bd (
        .clk              (clk_5KHz),
        .button           (button),
        .debounced_button (clk_pb)
    );

    mips_top mips_top (
        .clk              (clk_pb),
        .rst              (rst),
        .ra3              (switches[4:0]),
        .we_dm            (we_dm),
        .pc_current       (pc_current),
        .instr            (instr),
        .alu_out          (alu_out),
        .wd_dm            (wd_dm),
        .rd_dm            (rd_dm)
    );
endmodule
```

```

        .rd_dm      (rd_dm),
        .rd3        (dispData)
    );

    /*
    switches[4:0] are used as the 3rd read address (ra3) of the RF,
    dispData is the register contents from the RF's 3rd read port (rd3).
    */

    hex_to_7seg hex7 (
        .HEX          (reg_hex[31:28]),
        .s             (digit7)
    );

    hex_to_7seg hex6 (
        .HEX          (reg_hex[27:24]),
        .s             (digit6)
    );

    hex_to_7seg hex5 (
        .HEX          (reg_hex[23:20]),
        .s             (digit5)
    );

    hex_to_7seg hex4 (
        .HEX          (reg_hex[19:16]),
        .s             (digit4)
    );

    hex_to_7seg hex3 (
        .HEX          (reg_hex[15:12]),
        .s             (digit3)
    );

    hex_to_7seg hex2 (
        .HEX          (reg_hex[11:8]),
        .s             (digit2)
    );

    hex_to_7seg hex1 (
        .HEX          (reg_hex[7:4]),
        .s             (digit1)
    );

    hex_to_7seg hex0 (
        .HEX          (reg_hex[3:0]),
        .s             (digit0)
    );

    led_mux led_mux (
        .clk           (clk_5KHz),
        .rst           (rst),
        .LED7          (digit7),
        .LED6          (digit6),
        .LED5          (digit5),
        .LED4          (digit4),
        .LED3          (digit3),
        .LED2          (digit2),
        .LED1          (digit1),
        .LED0          (digit0),
        .LEDSEL        (LEDSEL),
        .LEDOUT        (LEDOUT)
    );

    /*
    switches [7:5] = 000: Display word of register selected by switches[4:0]

    switches [7:5] = 001: Display word of instr

    switches [7:5] = 010: Display word of 'alu_out'

```

```

switches [7:5] = 011: Display word of 'wd_dm'

switches [7:5] = 1XX : Display word of 'pc_current'
*/
always @ (posedge clk100MHz) begin
    casez ({switches[7:5]})
        3'b000: reg_hex = dispData[31:0];
        3'b001: reg_hex = instr[31:0];
        3'b010: reg_hex = alu_out[31:0];
        3'b011: reg_hex = wd_dm[31:0];
        3'b1???: reg_hex = pc_current[31:0];
        default: reg_hex = pc_current[31:0];
    endcase
end
endmodule

```

mips_fpga.xdc

```

#Clock
    set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports {clk100MHz}];
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk100MHz}];
#switches
    #n
        set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {switches[0]}}; # Switch 0
        set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports {switches[1]}}; # Switch 1
        set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports {switches[2]}}; # Switch 2
        set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports {switches[3]}}; # Switch 3
        set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports {switches[4]}}; # Switch 4
        set_property -dict {PACKAGE_PIN T18 IOSTANDARD LVCMOS33} [get_ports {switches[5]}}; # Switch 5
        set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {switches[6]}}; # Switch 6
        set_property -dict {PACKAGE_PIN R13 IOSTANDARD LVCMOS33} [get_ports {switches[7]}}; # Switch 7
    #
        set_property -dict {PACKAGE_PIN T8 IOSTANDARD LVCMOS33} [get_ports {switches[8]}}; # Switch 8

#Buttons
    set_property -dict {PACKAGE_PIN N17 IOSTANDARD LVCMOS33} [get_ports {button}]; # Center Button
    set_property -dict {PACKAGE_PIN P17 IOSTANDARD LVCMOS33} [get_ports {rst}]; # Left Button

#LEDs
    set_property -dict {PACKAGE_PIN K13 IOSTANDARD LVCMOS33} [get_ports {LEDOUT[0]}};
    set_property -dict {PACKAGE_PIN K16 IOSTANDARD LVCMOS33} [get_ports {LEDOUT[1]}};
    set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports {LEDOUT[2]}};
    set_property -dict {PACKAGE_PIN L18 IOSTANDARD LVCMOS33} [get_ports {LEDOUT[3]}};
    set_property -dict {PACKAGE_PIN R10 IOSTANDARD LVCMOS33} [get_ports {LEDOUT[4]}};
    set_property -dict {PACKAGE_PIN T11 IOSTANDARD LVCMOS33} [get_ports {LEDOUT[5]}};
    set_property -dict {PACKAGE_PIN T10 IOSTANDARD LVCMOS33} [get_ports {LEDOUT[6]}};
    set_property -dict {PACKAGE_PIN H15 IOSTANDARD LVCMOS33} [get_ports {LEDOUT[7]}};

    set_property -dict {PACKAGE_PIN J17 IOSTANDARD LVCMOS33} [get_ports {LEDSEL[0]}};
    set_property -dict {PACKAGE_PIN J18 IOSTANDARD LVCMOS33} [get_ports {LEDSEL[1]}};
    set_property -dict {PACKAGE_PIN T9 IOSTANDARD LVCMOS33} [get_ports {LEDSEL[2]}};
    set_property -dict {PACKAGE_PIN J14 IOSTANDARD LVCMOS33} [get_ports {LEDSEL[3]}};
    set_property -dict {PACKAGE_PIN P14 IOSTANDARD LVCMOS33} [get_ports {LEDSEL[4]}};
    set_property -dict {PACKAGE_PIN T14 IOSTANDARD LVCMOS33} [get_ports {LEDSEL[5]}};
    set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVCMOS33} [get_ports {LEDSEL[6]}};
    set_property -dict {PACKAGE_PIN U13 IOSTANDARD LVCMOS33} [get_ports {LEDSEL[7]}};

#Inputs out
    #Y
    #
        set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {n_out[0]}};
    #
        set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports {n_out[1]}};
    #
        set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports {n_out[2]}};
    #
        set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports {n_out[3]}};
    #Done/Err
    #
        set_property -dict {PACKAGE_PIN V11 IOSTANDARD LVCMOS33} [get_ports {done}];
    #
        set_property -dict {PACKAGE_PIN V12 IOSTANDARD LVCMOS33} [get_ports {err}];

    set_property -dict {PACKAGE_PIN R11 IOSTANDARD LVCMOS33} [get_ports {we dm}]; # LED 0

```