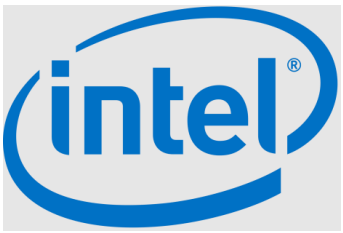


PLATFORM DESIGNER CUSTOM COMPONENT WITH PULSE WIDTH MODULATION (PWM)

Contents

1 Introduction: Test Your Knowledge: Platform Designer Customer Component with PWM

This test your knowledge workshops assumes you have completed the Embedded Nios course.

The resulting embedded system from this lab will be used as the starting point for your lab.

2 Making Platform Designer Components

The basis of this section is the following tutorial:

https://ftp.intel.com/Public/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/Making_Qsys_Components.pdf

You will need to complete a few steps prior to starting the tutorial.

Since copying and pasting text from files to the Quartus editor can introduce hidden characters download these 3 files from the github site:

reg32_avalon_interface.v

PWM_generator.v

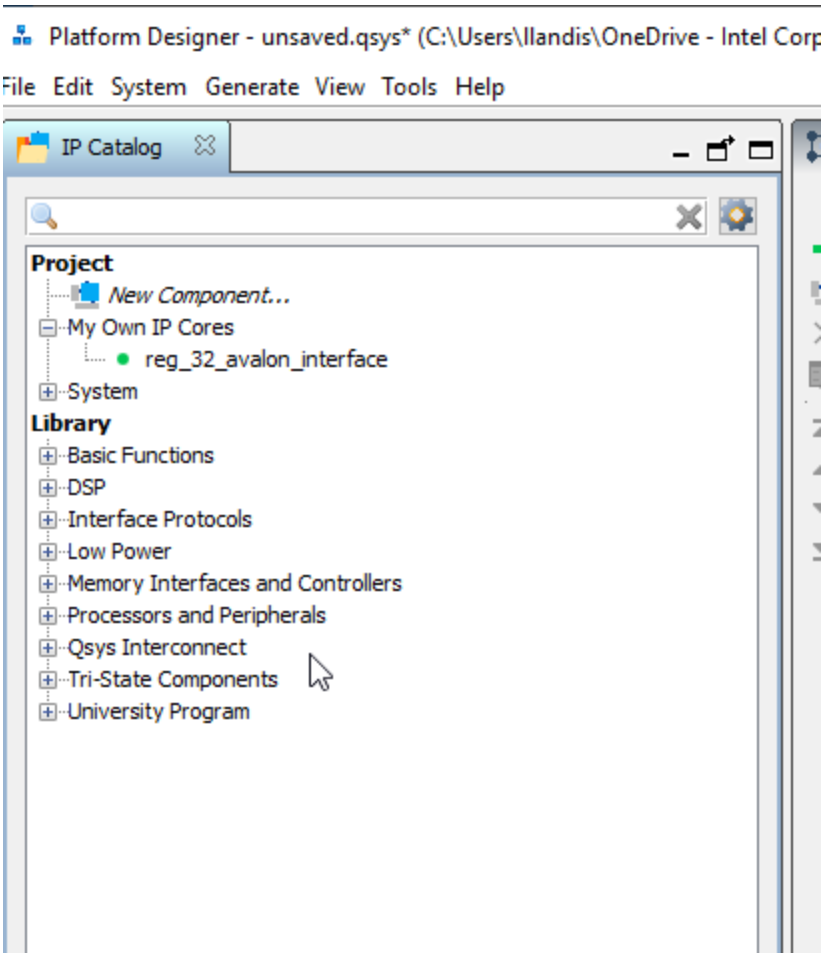
DE_hello_world_PWM.c

Restore your completed project Embedded Nios project in Quartus. This project needs to compile hardware properly and demonstrate that you can download the .sof file to the board, export the .sopcinfo file to Eclipse, write C code, compile it and download the .elf file to the DE10-Lite board. If these steps are not complete, revisit the Embedded Nios course.

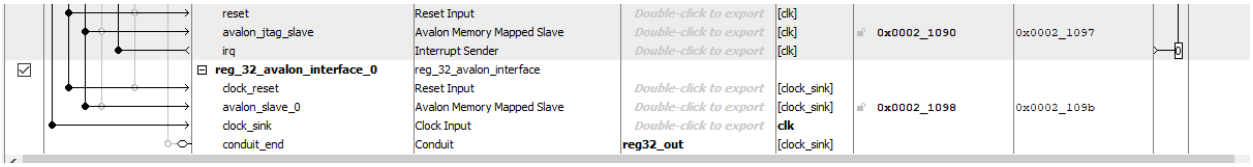
Open the Making Qsys Components pdf file. Follow the steps through page 25.

3 Integrating the reg32_avalon_interface module into your existing Platform Designer project

In Platform Designer, add your new component as new slave device.



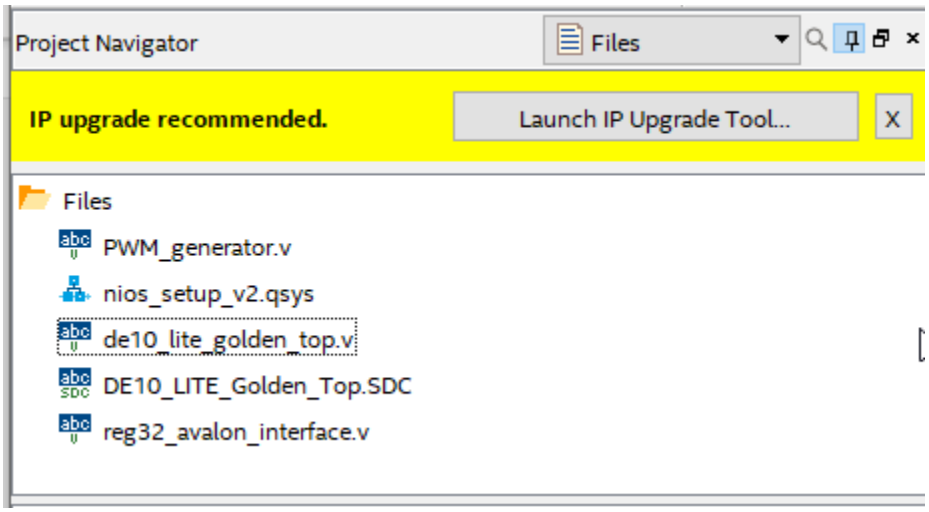
Double click on your new component. Add the component to your design and make connections to clock_reset, clock_sink, and Avalon Memory Mapped Slave. No connection is required to the conduit end. Name components as shown in the figure below.



Use the System → Assign Base addresses, generate your nios_setup_v2 system.

4 Connections from nios_setup_v2 to the top level wrapper

We provided you the PWM_generator.v file. Place that file in the same directory as de10_lite_golden_top.v. If you aren't familiar with PWM (pulse width modulation), this technique is commonly used for control of LED lighting and motors. Click on this link for more details. Add the PWM_generator.v file to your project: Project → Add/Remove Files in Project. Follow the steps as you have done in past projects to add the PWM_generator.v file. If reg32_avalon_interface.v isn't part of your project, add that as well. The list of files should look like this:



The .SDC file is optional.

Edit de10_lite_golden_top.v file. Because you added a new port called reg32_out you need to change the instantiation of that line of code. Note lines 74, 88 and 92 are changed lines from your original version. The line numbers might be slightly different.

```
68  `endif
69
70
71
72  );
73
74  wire [31:0] PWM_ontime;
75
76     nios_setup_v2 u0 (
77        .clk_clk (MAX10_CLK1_50), //clk.clk
78        .led_external_connection_export (ledFromNios[9:0]), //led_external_connection.export //CHANGED TO LEDR
79        .reset_reset_n (1'b1), //reset.reset_n
80        .button_external_connection_export (KEY[1:0]), //button_external_connection.export
81        .switch_external_connection_export (SW[9:0]), //switch_external_connection.export
82        .hex0_external_connection_export (HEX0), //hex0_external_connection.export
83        .hex1_external_connection_export (HEX1), //hex1_external_connection.export
84        .hex2_external_connection_export (HEX2), //hex2_external_connection.export
85        .hex3_external_connection_export (HEX3), //hex3_external_connection.export
86        .hex4_external_connection_export (HEX4), //hex4_external_connection.export
87        .hex5_external_connection_export (HEX5), //hex5_external_connection.export
88        .reg32_out_readdata (PWM_ontime) //                 reg32_out.readdata
89
90     );
91
92     PWM_generator PWM1(.PWM_ontime(PWM_ontime[7:0]), .PWM_out(LEDR[4]) , .clk(MAX10_CLK1_50) , .reset(1'b0));
93
```

Note on line 92, that we have used LEDR[4] as the LED that will get the PWM controlled signal that will control the brightness. Because LEDR[4] is already assigned, you will need to change a bit of code to avoid a conflict.

See line 103 below, LEDR[4] is removed from the list.

```
    //ignore what is below here

    wire [9:0] ledFromNios;
    assign LEDR[1:0] = ledFromNios[1:0];
    assign LEDR[9:5] = ledFromNios[9:5];
    assign HEX0[7] = 1'b1;
    assign HEX1[7] = 1'b1;
    assign HEX2[7] = 1'b1;
    assign HEX3[7] = 1'b1;
    assign HEX4[7] = 1'b1;
    assign HEX5[7] = 1'b1;
endmodule
```

Compile your design and look for errors and fix as necessary.

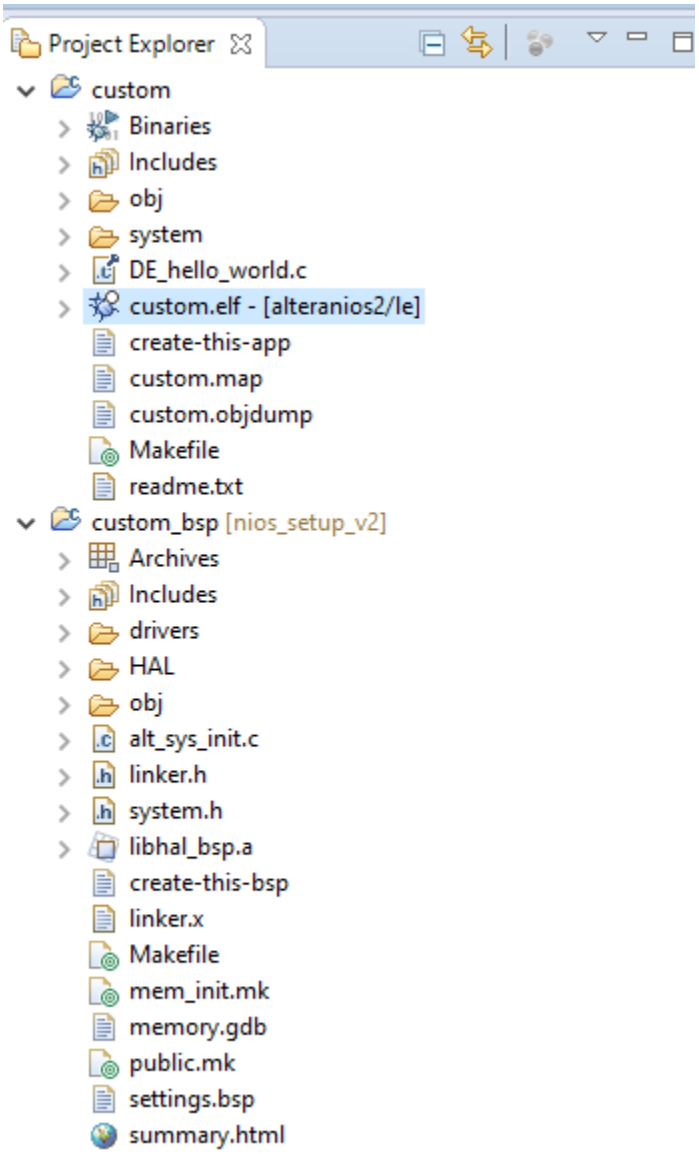
5 Eclipse Project

Launch the Eclipse project that you used for your Nios Embedded project, we will reuse it. Delete the user .c code from the project, and add the code you downloaded called DE_hello_world_PWM.c . Open the code in Eclipse and observe how it works. The main piece of code that controls the PWM function is shown below:

```
24          for(i=1; i<=20;i++){
25              usleep(50000);
26              IOWR_ALTERA_AVALON_PIO_DATA(REG_32_AVALON_INTERFACE_0_BASE, i*12);
27          }
```

The code will cycle through 20 increments of brightness from 12 to 240 (max is 255). Note the use of the usleep command which indicates a delay of 50,000 microseconds. There is a long list of Hardware Adaption Layer (HAL) commands you can use here: Nios® II Software Developer Handbook.

Note that the REG_32_AVALON_INTERFACE_0_BASE code is referenced in system.h. Find the file in the list and open it (its under the BSP).



The system.h has the memory map showing variables assigned to the addresses in the memory map. Make sure the line 26 maps appropriate variable for the reg32. If compile fails, refer to the system.h file. Build your project fix errors and complete.

6 Programming on to the DE10-Lite board

Bring up your Quartus Project and launch the programmer. Select your SOF file and program. If you have problems programming, make sure that Eclipse isn't running a program or the JTAG programmer will conflict.

7 Downloading the .elf file

Download the .elf file to the board and check out the functionality of the PWM running LED[4].

8 Follow on challenges

1. Tune the C code and make the brightness increment up to full brightness and then reverse the PWM to decrease the brightness back to the minimum value.
2. Merge the PWM_generator.v file as a submodule to reg32_avalon_interface.v. recompile and try again. Note the change of bit widths – you now will have a 1 bit PWM as the exported interface.
3. Add additional reg32+PWM modules and write a Knight Rider type sequencer using varying levels of brightness per LED.

Document Revision History

List the revision history for the application note.

Name	Date	Changes
Larry Landis	12/18/2020	Initial Release of lab manual