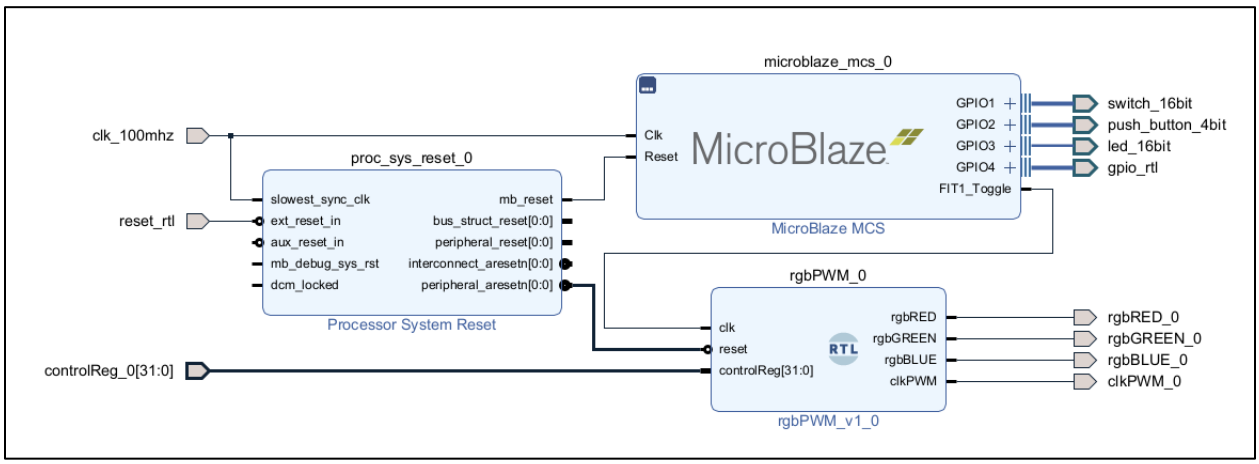
# ECE 544 Getting Started Project Addendum
# RealDigital Boolean Board

*NOTE: I RETARGETED THE NEXYS A7 VERSION OF THE GETTING STARTED WRITE-UP TO THE REALDIGITAL BOOLEAN BOARD. THIS ADDENDUM IS BASED ON MY RECOLLECTION OF THE STEPS I TOOK. THERE MAY BE TYPOS OR SLIGHT DIFFERENCES IN THE MESSAGES, OR MAYBE EVEN A MISSING STEP, BUT IF YOU HAVE A BASIC UNDERSTANDING OF THE PROCESS FOR BUILDING A MICROBLAZE MCS SYSTEM YOU SHOULDN'T HAVE MANY PROBLEMS FOLLOWING THIS ADDENDUM. IF YOU DO, PLEASE POST TO THE GETTING STARTED PROJECT DISCUSSION FORUM.*

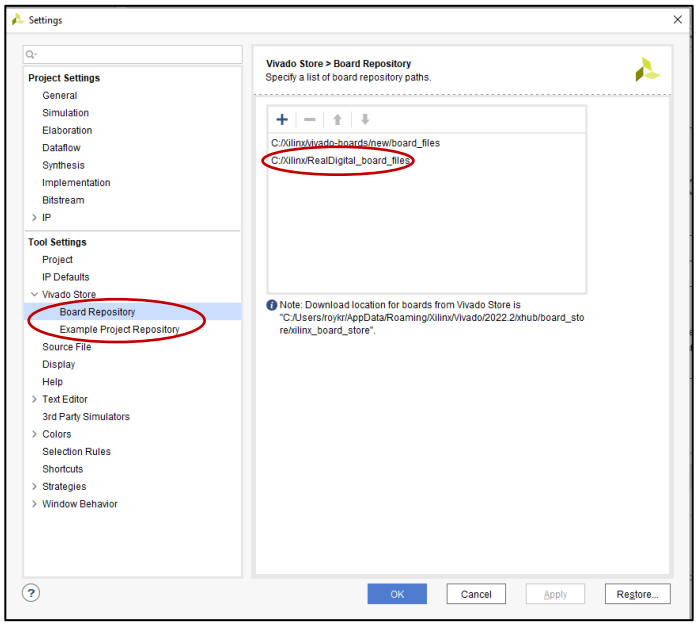
This document is an addendum to the ECE 544 Getting Started Project write-up. It includes guidance on how to target the Getting Started Project write-up which is based on the Nexys4 A7 (Nexys4 DDR) to the Boolean Board. As it turns out the process is straightforward:

## Block diagram

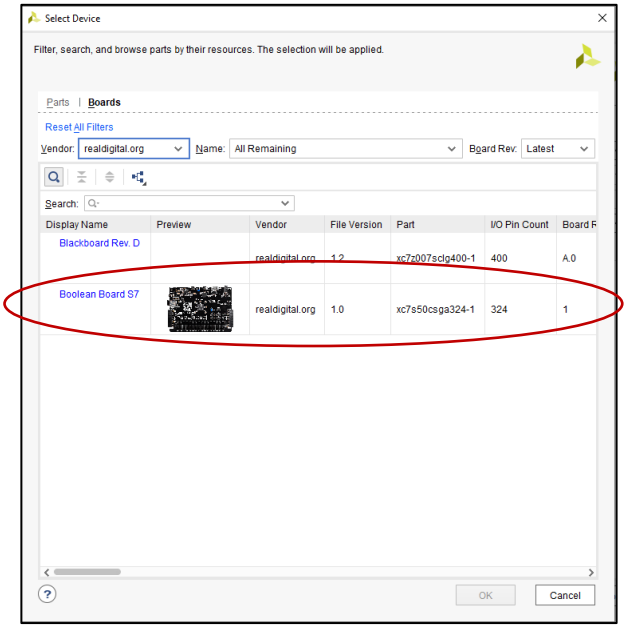


## Hardware

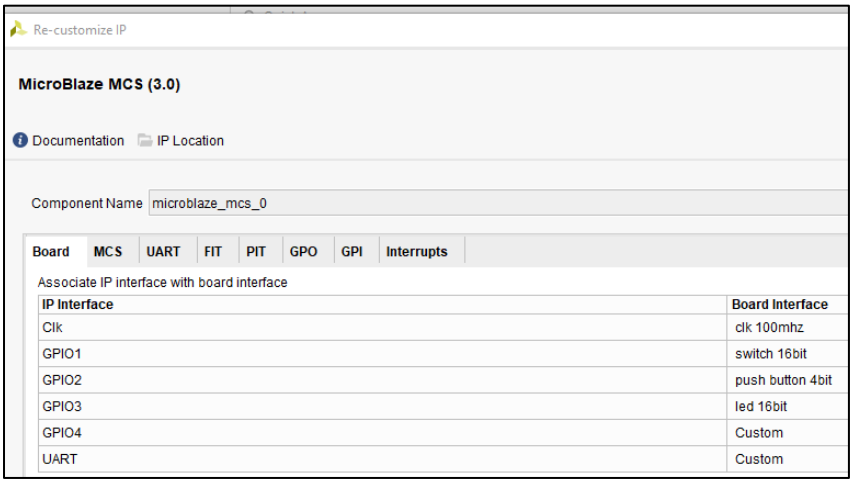
1. Add the RealDigital Boolean Board file repository to Vivado. This is done by adding the repository into the search path for the Vivado Store (may seem odd, but cleaner than the older method)




2. **<Task 1, Step 4>** When you create the project include `hdl/booleanfpga.v` and `constraints/booleanfpga.xdc` instead of the nexysfpga.* files
3. **<Task 1, Step 5>** When you select the `Default Parts/Boards` choose the RealDigital Boolean board instead of Nexys A7 (Nexys4 DDR)



4. **<Task 2, Step 4>** When you Re-customize IP/MicroBlaze MCS/**Board** select the following assignments):

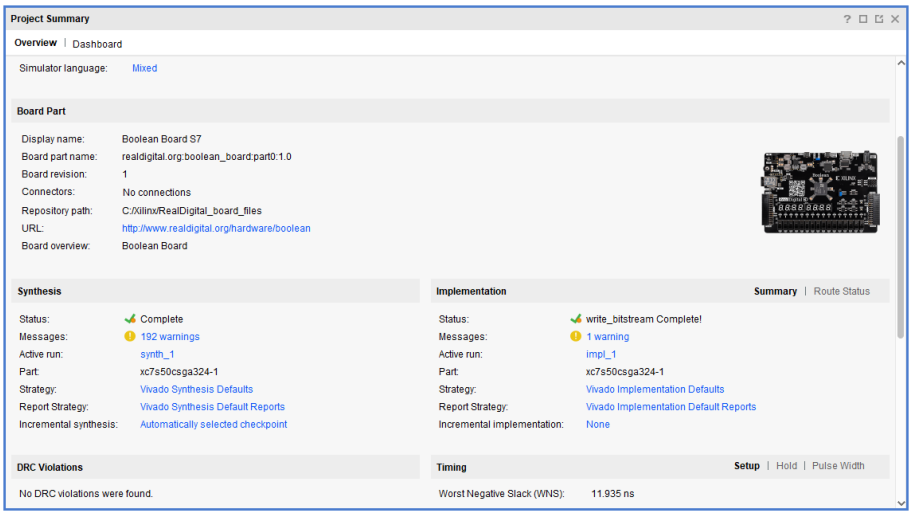



5. **<Task 2, Steps 9 and 10>** When you check the GPO and GPI configurations make sure the GPO and GPI are configured correctly. The Boolean board has 16 switches, 16 LEDs, and 2 RGB LEDs like the Nexys A7 but there are only 4 pushbuttons instead of the 5 pushbuttons on the Nexys A7. The refactored application `gsproj_app.c` for the Boolean board takes that into account.

6. **<Task 3, Step 3>** When you Run `Connection Automation` click your way to the following

   `Clock` – select `clk_100mhz`
   `Reset` – select `reset_rtl`
   `GPIO1` – select `switch_16bit`
   `GPIO2` – select `push_button_4bit`
   `GPIO3` – select `led_16bit`
   `GPIO4` – select Custom (should be named `gpio_rtl`)

7. **<Task 4>** Follow the same process but change file names, etc. for the Boolean Board configuration. Your system should Synthesize and Implement and generate bitstream and export hardware the same way as the Nexys A7 version in the write-up.


Project Summary
Overview | Dashboard
Simulator language: Mixed
Board Part
Display name: Boolean Board S7
Board part name: realdigital.org:boolean_board:part0:1.0
Board revision: 1
Connectors: No connections
Repository path: C:/Xilinx/RealDigital_board_files
URL: http://www.realdigital.org/hardware/boolean
Board overview: Boolean Board
Synthesis
Status: Complete
Messages: 192 warnings
Active run: synth_1
Part: xc7s50csga324-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports
Incremental synthesis: Automatically selected checkpoint
Implementation
Summary | Route Status
Status: write_bitstream Complete!
Messages: 1 warning
Active run: impl_1
Part: xc7s50csga324-1
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental implementation: None
DRC Violations
No DRC violations were found.
Timing
Setup | Hold | Pulse Width
Worst Negative Slack (WNS): 11.935 ns

## Application

1. **<Task 5, Step 3>** When you create your Platform project in Vitis select the `.xsa` file for your Boolean project.
2. **<Task 5, Step 6>** When you import the source code for the application after having created an Empty application, import the files from `application/Boolean`. The Boolean board only has 4 pushbuttons and there are a few other minor differences between the applications.

   Incidentally the Boolean Board does not have a separate CPU Reset button so the hardware in `booleanfpga.v` creates a system reset by: `reset_rtl = ~(btn0 & btn1).`

3. `Build`, `debug_as` and/or `run_as` your application. The way the Boolean Board version of the application controls the RGB LED is different, but both applications have the same functionality:
   - `sw[15]` is used to enable (`sw[15] up,` on) and disable (`sw[15], down off`) the RGB LED PWM outputs.
   - `btn0` – controls the Green segment of the RGB LED
   - `btn1` – controls the Blue segment of the RGB LED
   - `btn2` – controls the Red segment of the RGB LED
   - `btn3` – sets all 3 duty cycles back to 0 (RGB LED will go off)
   - led[15:0] – display the switches whenever any switch changes

*<finis>*