Outline for creating the ECE544 Project #1 Embedded System (Nexys4 DDR)
By Roy Kravitz (roy.kravitz@pdx.edu)
(Last updated: 24-Jan-2025)

Note: The steps for the Boolean Board are pretty much the same, the differences being in the FPGA type, constraints file, and top-level module. This write-up is based on the Nexys A7 to keep it consistent with previous releases.

There are many ways to create the Project #1 embedded system using the Block Design editor and IP Integrator. Some require undoing/redoing and exploring the capabilities, and quirks, of the IP Integrator.

These are the steps I followed (on my third try) that resulted in a working embedded system and a smooth handoff from Vivado to Vitis. I did not use the Digilent NexysA7 board file but rather specified the FPGA directly. I was able to do this because there was no need to include any of the devices supported in the board file, and because I had working nexysa7fpga.v and nexysa7fpga.xdc from the Getting Started project that was easy to modify.

Note: the assumption is that you have added the ece544ip_w25 IP repository as called out in the Project #1 write-up. If you haven't done that, please do so before you try to build embsys.

Step	Screen	Action	Explanation/Comments
1	Add Sources	Create a new RTL project in Vivado. Include: • `hdl/rgbPWM_r2.v` • `hdl/nexysa7fpga.sv` • `Constraints/nexysA7fpga.xdc`	
2	Default Part	Use the dropdowns to select the `xc7a100tcsg324-1` component (Boolean: `xc7s50csga324-1`)	
3	New Project Summary	Check the description to see if it is what you expected. If it is not, press *Back* and make changes. If it is OK, click *Finish*	
4	Flow Navigator	Create a Block Design. Name it `embsys` (or whatever you like)	
5	Block Design/Diagram	Add a MicroBlaze IP block (not the Microblaze MCS). Run *Block Automation* to configure the MicroBlaze as follows: • Local Memory: 64KB • Debug Module: Debug + UART • Interrupt Controller: enabled	Only the non-default options are listed. The default is fine for the others

		• Clock Connection: New Clocking Wizard	
6	Block Design/Diagram	Customize the Clocking Wizard (likely named `clk_wiz_1`) as follows: Clocking Options: • Clock Information/Input Clock/Primary: `Single-ended clock capable` Output Clocks: • `clk_out_2`: Port name: `clkPWM_10MHz`, Requested frequency: `10.000` MHZ. • (scroll down) Reset Type: `Active Low`	The 10MHz clock will be used, in conjunction with the clock divider included in `rgbPWM` to create the PWM clock. The PLL's and MMCM's in the clock generator can create frequencies above and below the input clock but there are limits. 10MHz is within range but KHz frequencies are not.
7	Block Design/Diagram	Run *Connection Automation/All Automation* to create external signals for the system clock and system reset `reset_RTL_0` is a cumbersome name for system reset so let's change it. Click on the signal name and select the *External Port Properties* tab. Change the name to `resetn`.	You can rename any external port this way. Consider changing other "cumbersome" names to names that are more to your liking.
8	Block Design/Diagram	Add a Fixed Interval Timer IP block to the diagram. Customize the FIT: • Number of Clocks: `50_000_000` Customize the `MicroBlaze_0_xlconnect` block • Number of Ports: `1` Connect the `Interrupt` output from the FIT to the `MicroBlaze_0_xlconnect In0[00]` port *Run Connection Automation/All Automation* to connect the clock and reset to the FIT	Why 50000000 for the FIT Number of Clocks? The input clock to the FIT is 100MHz. A count of 50_000_000 results in an interrupt rate of 2 HZ. The FIT interrupt is used to sample

			the switches and pushbuttons in the application – a rate of ½ second is fast enough to be responsive to user input but not so fast that it affects the performance of the app
9	Block Design/Diagram	Add and customize the following AXI IP blocks to the design. Do not Run Connection Automation until all the IP blocks have been added • `AXI Timer` – customize and configure for 32 bits, generate clock • `AXI GPIO` – customize to a single 32-bit output-only GPIO port • `Nexys4IO` – no customization needed • `PWMAnalyzer` x 3 – Three instances of the Digilent PWMAnalyzer, one for the red segment, one for the green segment, and one for the blue segment of RGB2 LED. No customization is needed. Connect the `AXI Timer/generateout0` output to the `Nexys4IO/RGBLED_Clock` input *Run Connection Automation/All Automation*	
10	Block Design/Diagram	*Right-click* on the `Nexys4IO` IP block and select *Make External*. This will make all the `Nexys4IO` ports External which brings them up the hierarchy to the top-level model (`nexysa7fpga.v`) where they are mapped to the correct pins on the FPGA in the top-level port list and specified in the `nexysA7fpga.xdc` pin constraint file. Delete the `RGB1_Red_0`, `RGB1_Green_0`, and `RGB1_Blue_0` signals from `Nexys4IO` by selecting the ports in the block diagram and pressing Delete. The RGB1 LED segments are driven with the `rgbPWM IP` block.	
11	Block	Add the following IP blocks to your embedded system:	

	Design/Diagram	• rgbPWM_r2: *Right-click* in an empty area in the block diagram and select *Add Module*. Select rgbPWM in the dialog box. • Processor System Reset Customize the rgbPWM IP block: • Divide Count: 50 • Use Divider: "1" Connect: • proc_system_reset/peripheral_aresetn[0:0] to rgbPWM/reset • clk_wiz_1/clkPWM_10MHz to rgbPWM/clock • clk_wiz_1/clkPWM_10MHz to proc_system_reset/slowest_sync_clock • resetn to proc_system_reset/ext_reset_in *Right-click* on the rgbPWM IP block and select Make External. This will make the PWM outputs, and the control register input external. The control register input is connected to the GPIO output in the top-level module.	
12	Block Design/Diagram	Connect the following signals: • rgbPWM/rgbRED to PWMAnalyzer_0/pwm • rgbPWM/rgbGREEN to PWMAnalyzer_1/pwm • rgbPWM/rgbBLUE to PWMAnalyzer_2/pwm	
13	Block Design/Diagram	Save your Block Design and validate it. The validation should be successful with no critical warnings Check the Address Map to make sure all the IO devices have unique IO ranges and that the local memory addresses are 0x0000_0000 – 0x0000_FFFF	
14	Flow Navigator	Save (if not saved) the Block design and Generate Block Design	It took about 10 minutes on my laptop to generate the block design, but I have a Quad core laptop w/ lots of memory and an SSD Note: I had 997

			warnings but 0 errors and 0 critical warnings
15	Flow Navigator	Follow the Synthesis/Implementation/Generate Bitstream steps from the Getting Started project: • Match the signals/ports in the embedded system instantiation to the MCU instantiation in `nexysa7fpga.sv` • Synthesize the design and check the warnings • Implement the design and check the warnings • Associate the `bootloop` ELF files (`Tools/Associate ELF Files…`) • Generate the Bitstream • Export the hardware design, including the bitstream	