ELEC 433: Building a Sinusoid Generator

Evan Everett and Michael Wu

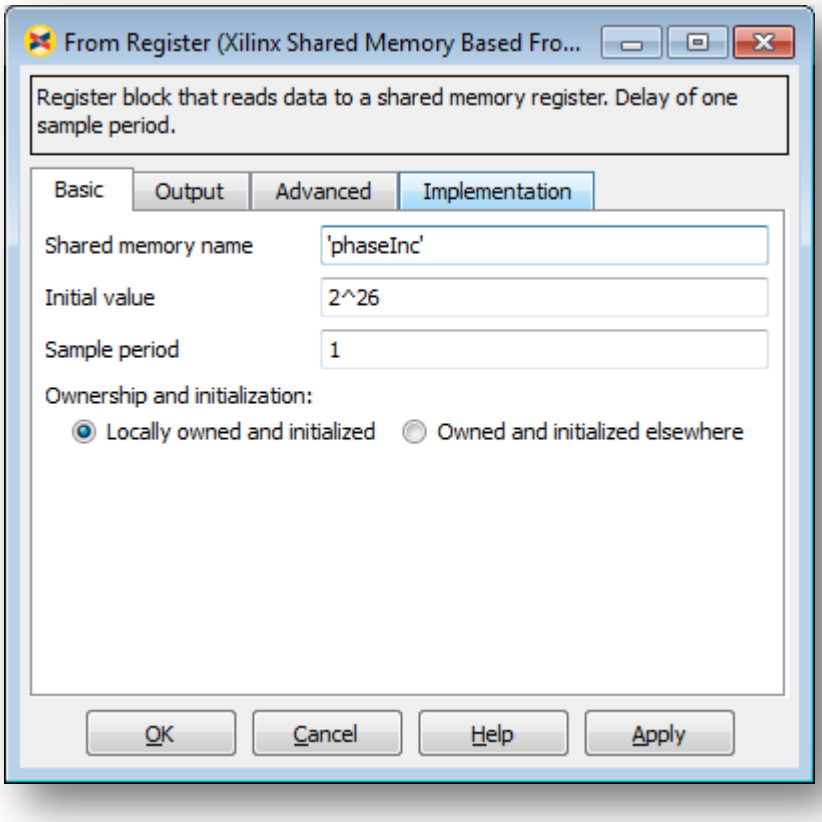
February 2013

Building the Sysgen Model

Download the DDS model from the website. Simulate the model and make sure you understand its behavior.

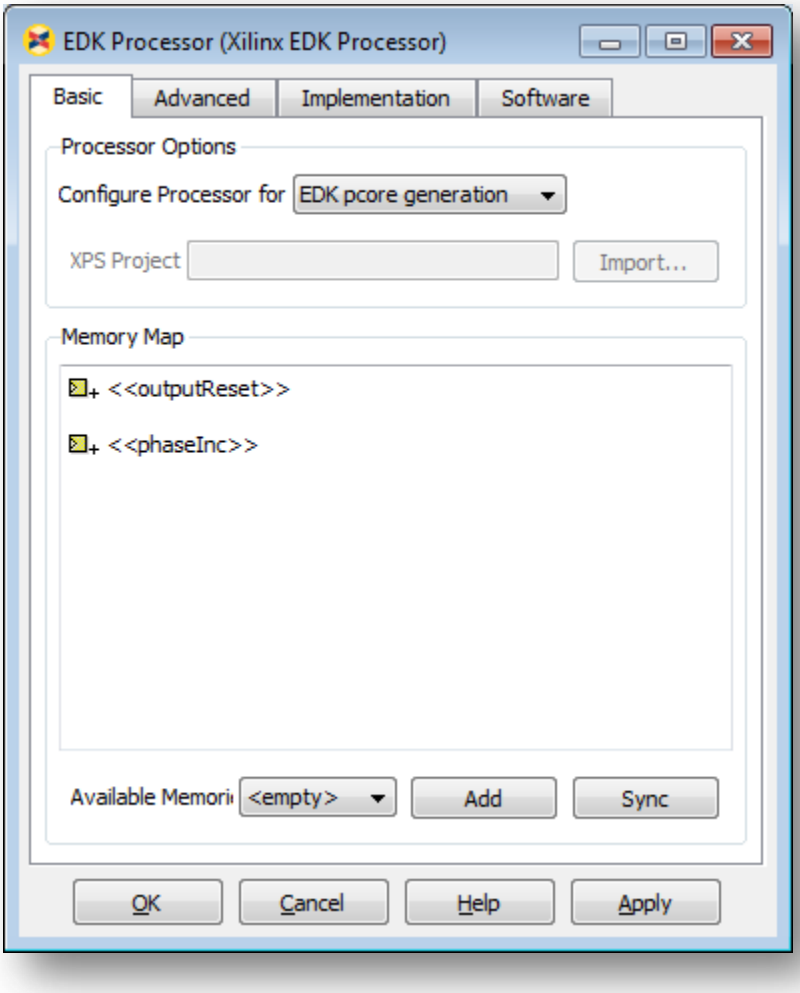
Exporting the Model as a Peripheral Core

Once you have designed and tested your model via simulation, the last step is creating a custom hardware description language (HDL) peripheral core (pcore) that can be inserted into a Xilinx Platform Studio (XPS) project.

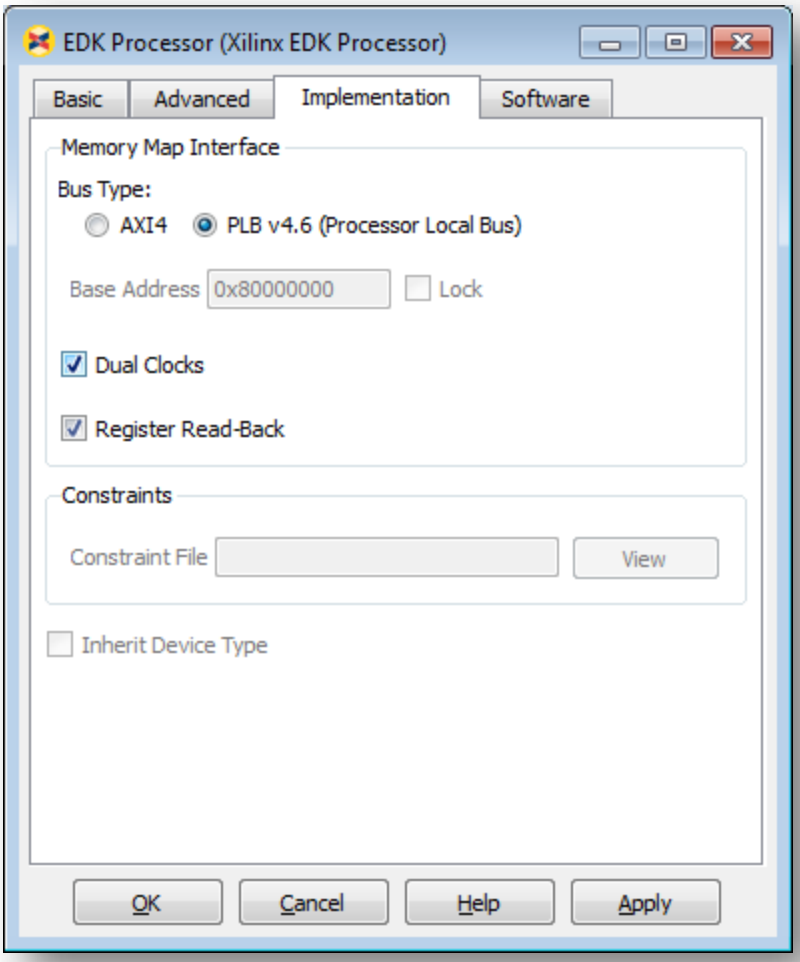


1. Before generating the HDL core, set the initial value of the **phaseInc** register back to 2^26 embedding this value into the hardware design. The From Registers will be memory-mapped in

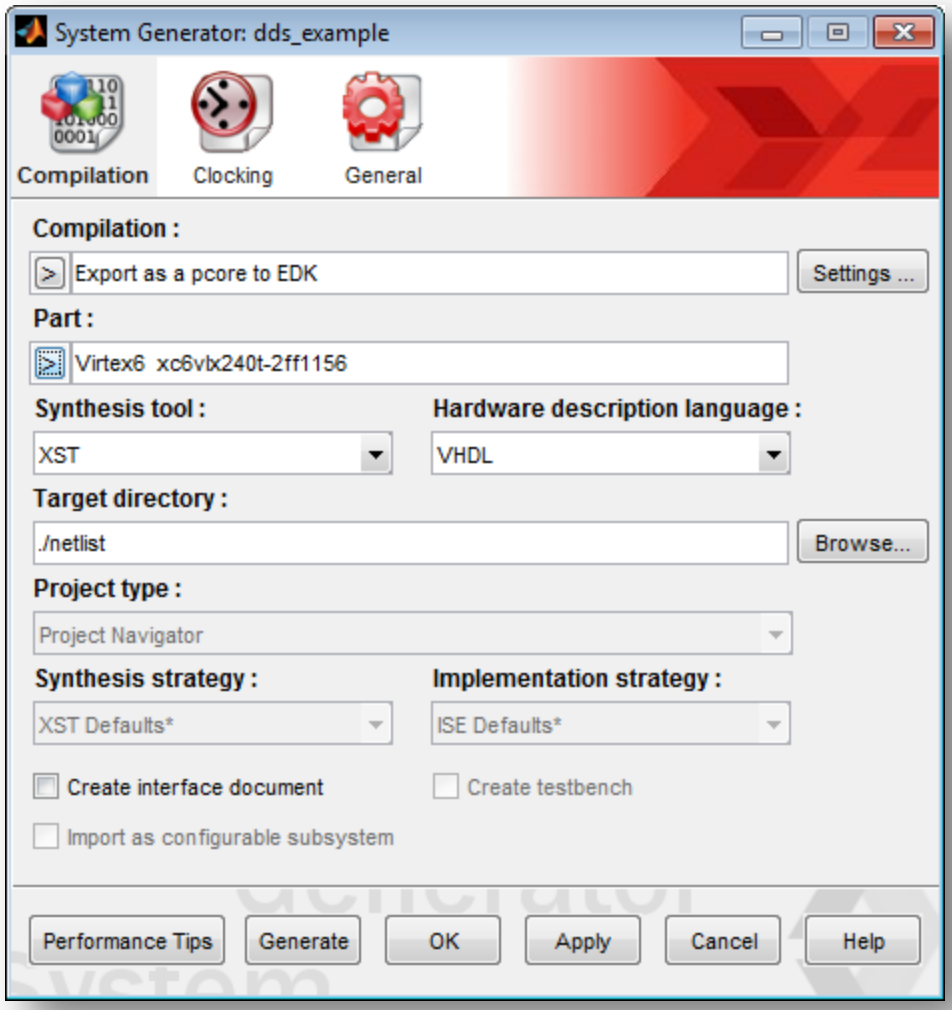
the process of generation and we will be able to change their values by reading and writing certain address locations in c.



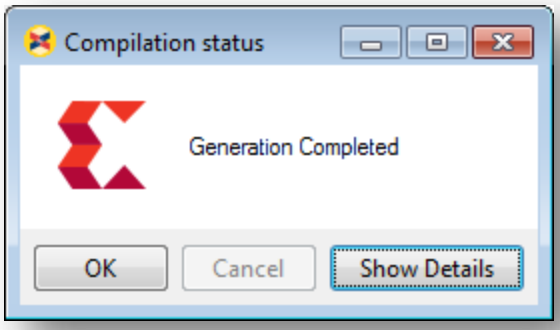
2. In order to create memory map, drag in the **EDK Processor** block from the Xilinx Library. Double-clicking it will reveal the options window. Make sure the processor will be configured for **EDK pcore generation.** Click **Add** to create the memory mapped registers. This will populate the map with the list of to/from registers found in the system. The registers can be expanded to see their address locations. Click **Apply** to save.



3. Move to the **Implementation** tab of the EDK Processor block. For **Bus Type**, select PLB v4.6. AXI4 is a special bus that targets ARM systems. Leave **Dual Clocks** option checked. Also check the unchecked **Register Read-Back** as shown above. **Register Read-Back** will allow us to read the current value of the phaseInc.



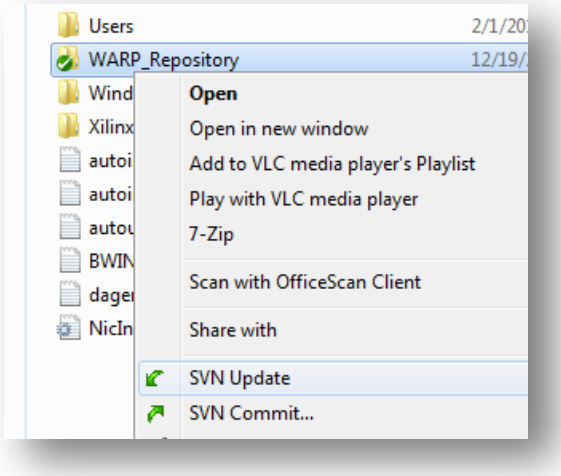
4. Open the **System Generator** block. It will reveal a whole set of options to create the design. As the model is going to be part of a Xilinx Platform Studio project, choose **Export as pcore to EDK** for the Compilation. This will create all the supporting files so it can be recognized by the Embedded Development Kit (EDK) that Xilinx provides. The FPGA we will be targeting is the Virtex6 xc6vlx240t-2ff1156, select it from the dropdown menu by selecting **Virtex6** -> **xc6vlx240t** -> **-2** -> **ff1156**



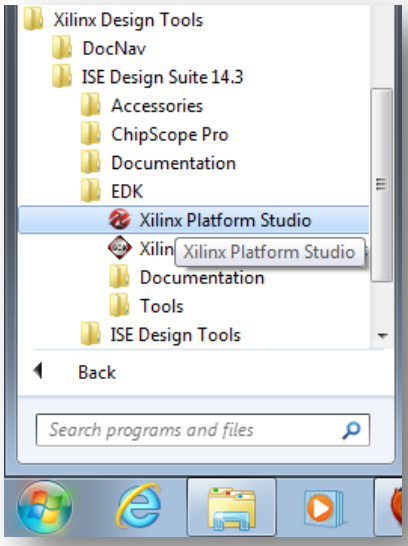
5. Finally, click **Generate** to start the HDL generation. The above window will pop-up once the generation is complete. At this point there is an XPS-compatible peripheral core available in the **./netlist/pcores** folder. This includes the HDL for the core and associated data files that identifies the type of core, the memory-mapped locations etc. If you are familiar with VHDL, you may want to view the VHDL code that has been generated. It will likely be hard to parse, however.

Integrating your Peripheral Core into an XPS Project

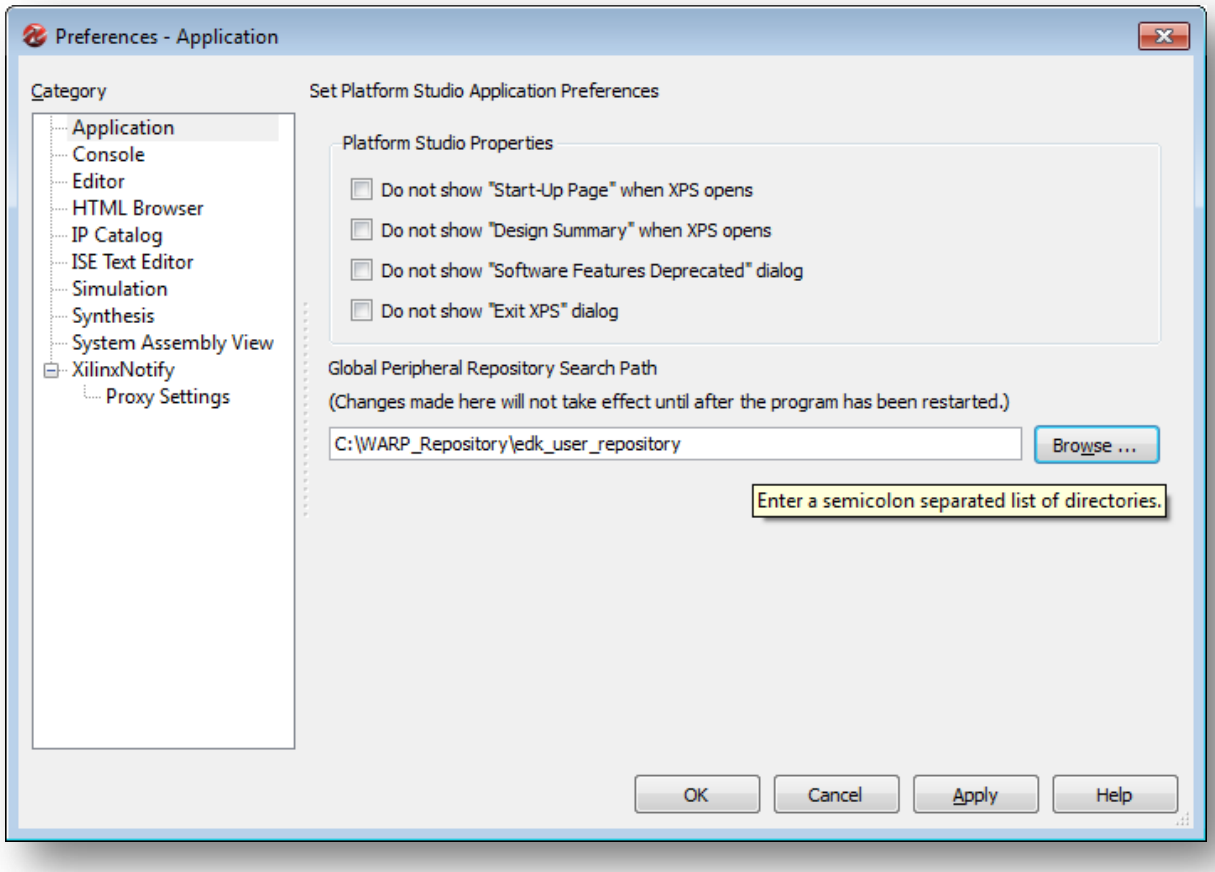
We have provided a pre-built skeleton project in Xilinx Platform Studio for this exercise. Download it form the course website. Unzip this folder in a path on your C: drive **that contains no spaces.** This project contains some relevant pieces of the WARP Platform Support Packages that you will integrate with your DDS pcore.



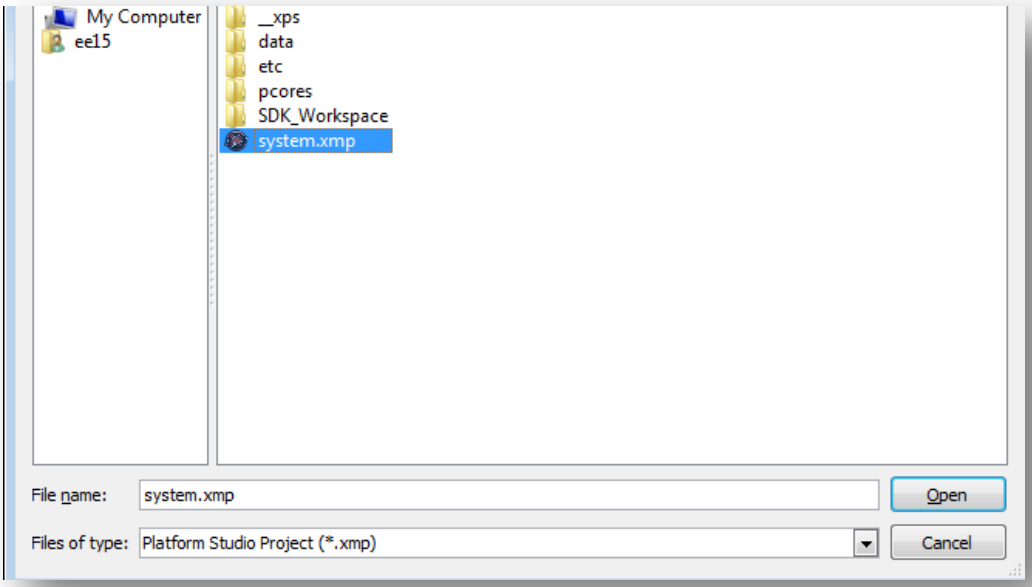
1. First make sure that there is an updated copy of the WARP Repository in your C: drive. Navigate to **WARP_Repository** folder in your the C: drive, right click it, and select **SVN Update**.
2. Copy the `ddsexample_plbw_v1_00_a` pcore from the netlist that was generated from your Sysgen model folder into the \pcores\ folder of the skeleton XPS project.



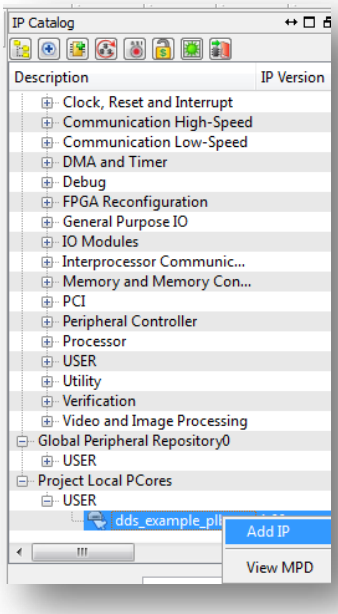
3. Run Xilinx Platform Studio (XPS) from the windows start menu. If it is the first time you have run XPS on the machine, it may take several minutes to initialize.



4. If this is your first time using XPS with WARP on this machine, make sure you tell XPS where to find the WARP support peripherals in the WARP repository. Click **edit->preferences** and select the **Application** category. Browse to the **WARP_Repository** and select the **edk_user_repository** as shown above.

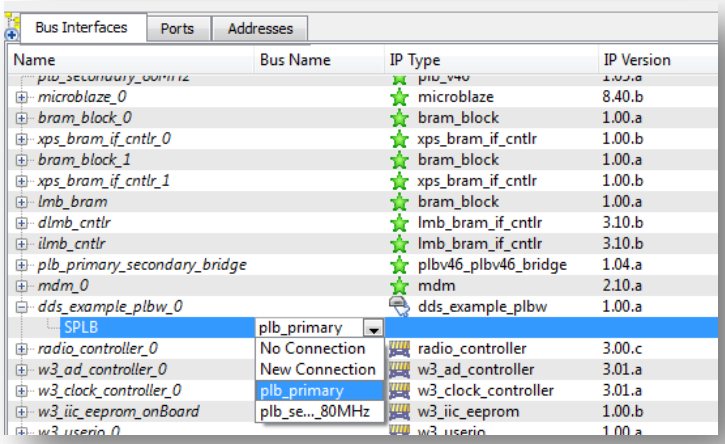


5. Select "Open Project" and navigate to your template project folder. Open the project by selecting "system.xmp".

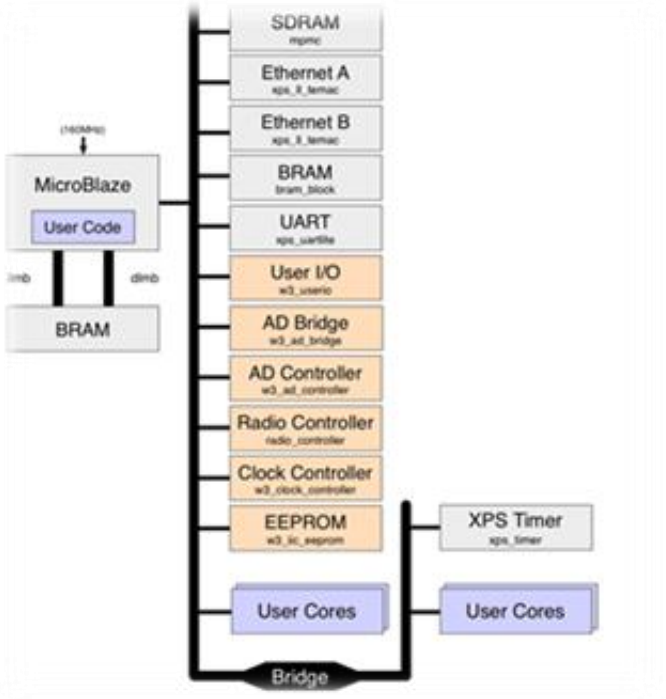


6. Now we need to add the DDS pcore we designed to the project. In the **IP Catalog** tab on the left of the screen, expand the **Project Local PCores/USER** category. There you should see your

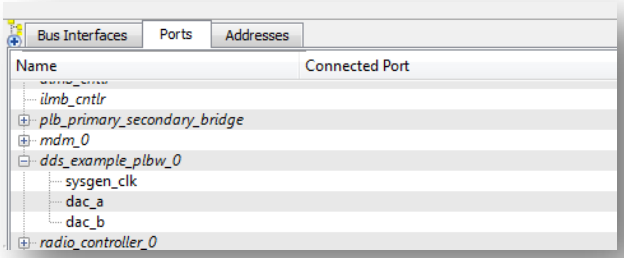
dds_example pcore. Right-click it and select **Add IP**; click yes in the pop-up that asks you if you want to add the pcore to your project. Click OK in the next pop-up and ignore error messages in the console for the moment.



7. Next we need to connect our **dds_example pcore** to a Bus. Click the Bus Interface tab in the middle of the screen and look for the **dds_example_plbw** core in the list of included peripherals. Expand the core's entry assign it to the **plb_primary bus**. The **plb_primary bus** is the processor local bus (PLB) that's used to connect different prepherials. Devices connected to the PLB bus are memory mapped into the MicroBlaze processor's address space, which allows us to controls these cores in software.



8. Now that we've connected our pcore to the PLB bus, the next step is connecting the output ports of our DDS pcore to the input ports of the WARP D/A bridge.



a. To view the ports available for the DDS pcore, select the **Ports** tab and expand the dds_example_plbw entry. You should see the names for three ports. This first port **sysgen_clk** is an "under-the-hood" clock port that needs to be connected to … The other two ports, **dac_a** and **dac_b,** correspond to the Gateway-Outs in your Sysgen model, and are thus the ports we wish to connect the WARP analog board so that we can view the outputs on an oscilloscope. Write down the names of these ports as you will use them in the next step.
b. The file that contains the connections between pcores is the **system.mhs.** We will edit it manually to connect the DDS core to the WARP analog board. Click **File->open->system.mhs,** and the code for the mhs file should appear in an embedded text editor.
c. Scroll to the bottom of the file, and the last two blocks of code should contain sections for the D/A bridge and your DSS pcore. The ports for the D/A bridge are already initialized, but the ports for the DDS are not. You have to define them.
    i. To connect the source the FPGA clock to your DDS pcore type a line of the following line. The net for the system clock is *clk_40MHz*, and you want to assign this net to the **sysgen_clk** port of your DDS by entering the line:

```
PORT sysgen_clk = clk_40MHz
```

    ii. Now we need to connect the outputs of the DDS to the D/A bridge. In a similar way as above, assign the **dac_a** and **dac_b** to ports to a new net (you can name in whatever you wish, but make it meaningful), and then assign the **user_DAC_A** and **user_DAC_A** ports of the D/A to the same net names. Check with an instructor to make sure the code is correct before moving to the next step. In the dds section you make have the lines:

```
PORT dac_a = ddsexample_dac_a
PORT dac_b = ddsexample_dac_b
```

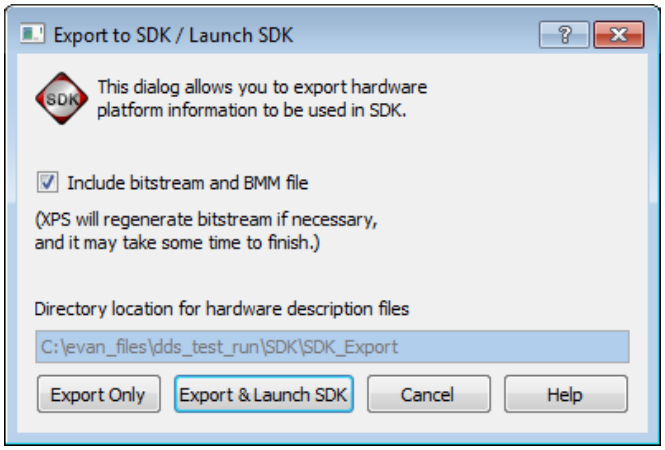
    and in the D/A bridge section you may have the lines:

```
PORT user_DAC_A = ddsexample_dac_a
PORT user_DAC_B = ddsexample_dac_b
```

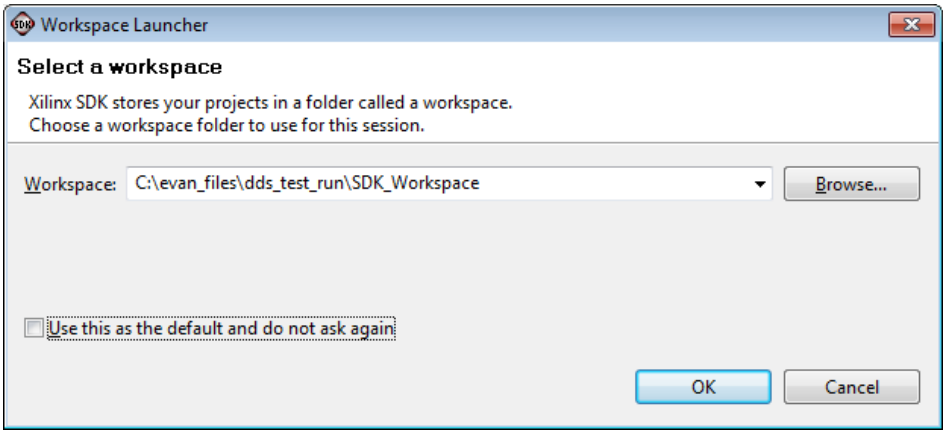
9. Now that the ports of our DDS pcore are connected properly, we next need to generate addresses for the registers in our model that allow us to control the frequency of the DDS as well as reset the DDS. Go back to the "system assembly view" and click the "Addresses" tab. In the upprt right corner, there is a button that when you hover over says " Generate Addresses" Click it.
10. Now you're ready to compile the design. Click hardware->Generate Bitstream, and wait. It may take several minutes.

Creating software to run on your design

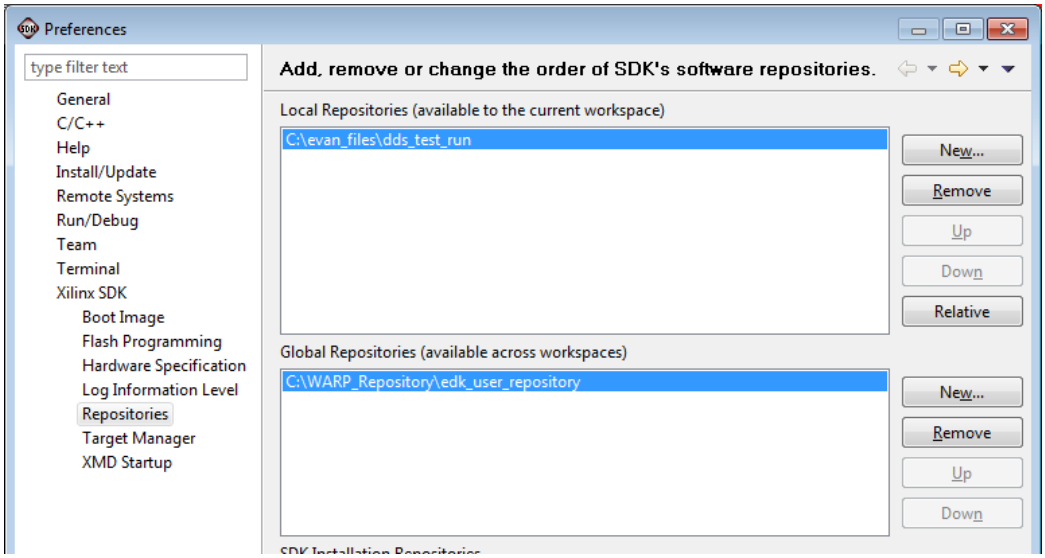
Now we are ready to write our software for interacting with design. To do this, we are going to export our project to a Xilinx Software Development Kit project. And write code that will run on the Microblaze processor within the FPGA.



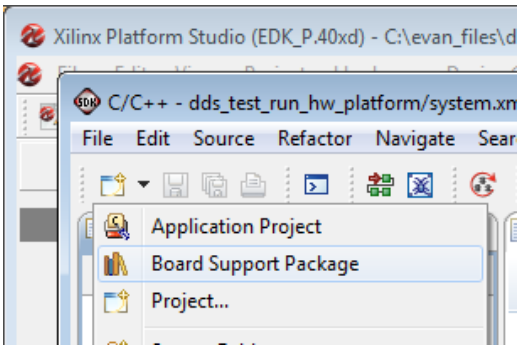
1. First we will export out XPS project to the software development kit. Select Project->Export Hardware Design to SDK. Click Export & Launch



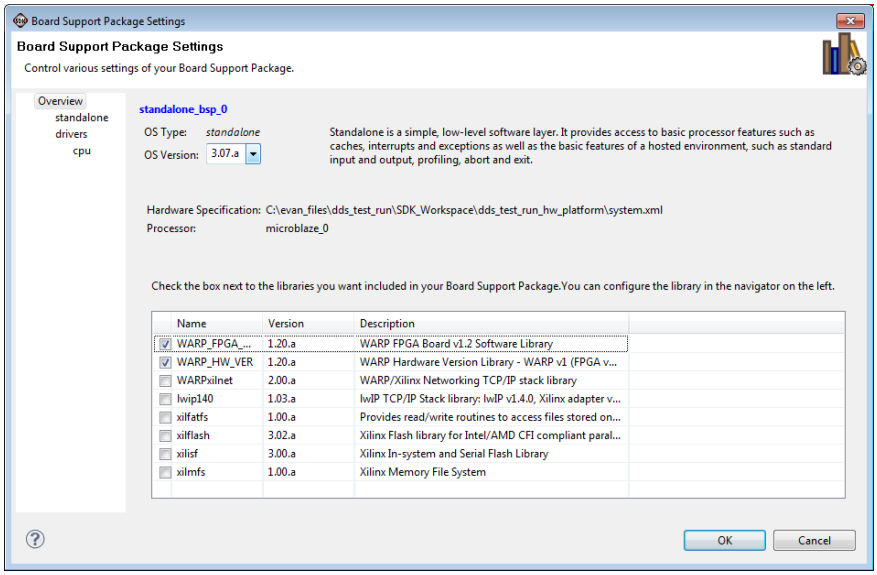
2. Next we will select a software workspace. A workspace is a collection of software projects targeting a single hardware design (in this class we will usually only have one software project per hardware project).
    a. After opening SDK, you will be prompted to select a workspace directory. Make sure the box saying "use this as the default and do not ask again" **unchecked!!** Navigate to your project folder, inside the project polder should be a director called **SDK_Workspace.** Select it, and click OK.
3. Just like we did for XPS, we need to point SDK to the WARP repository (only have to do this first time you set up the workspace).



    i. Go to Xilinx Tools-> repositories. I n the Global Repositories plane select edk_user_repositories from within the WARP_Repository directory.
    b. We also need to point the workspace to the local repository in our XPS project. In the Local Repositories plane, navigate to the top directory of your VPS project. Click OK.

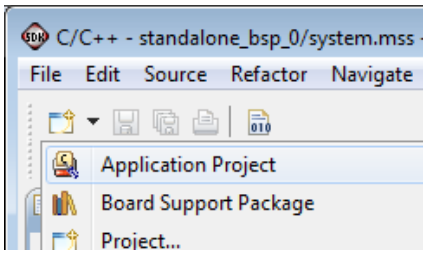


4. Create a new Boars Support Package project by clicking the "new" button the at top left and selecting "Board Support Package".

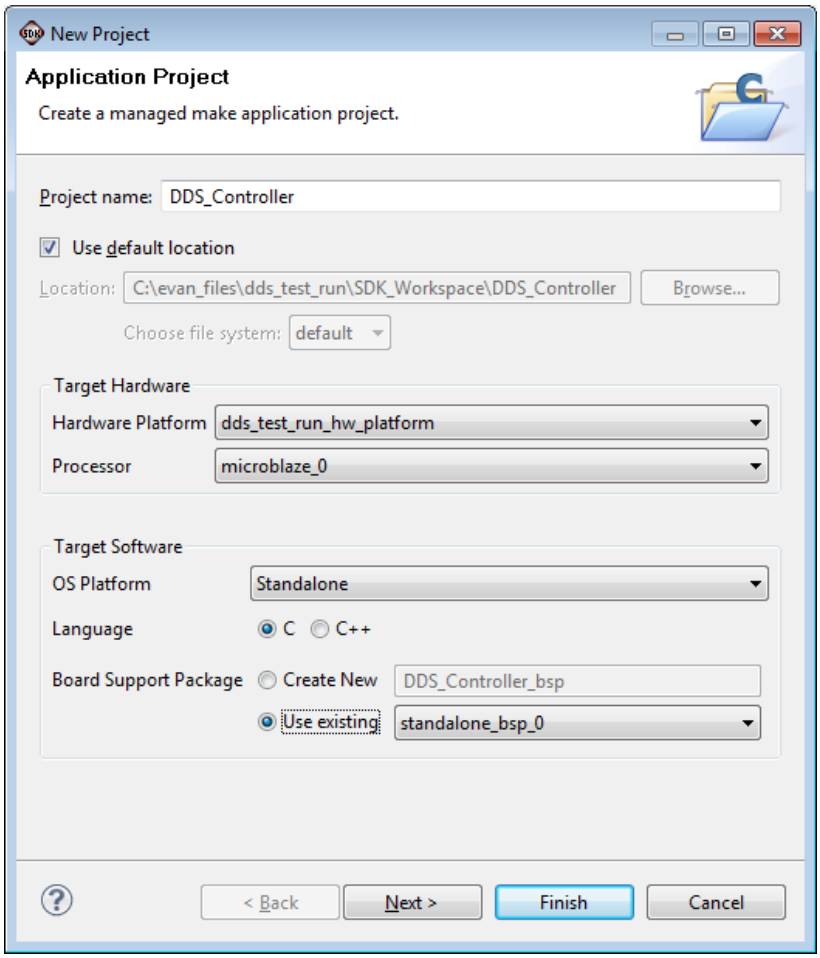


a. Select the optional support packages as shown above.

5. Next, create a new software project.



a. Click the “new” button in the top left and select “**Application Project**”.

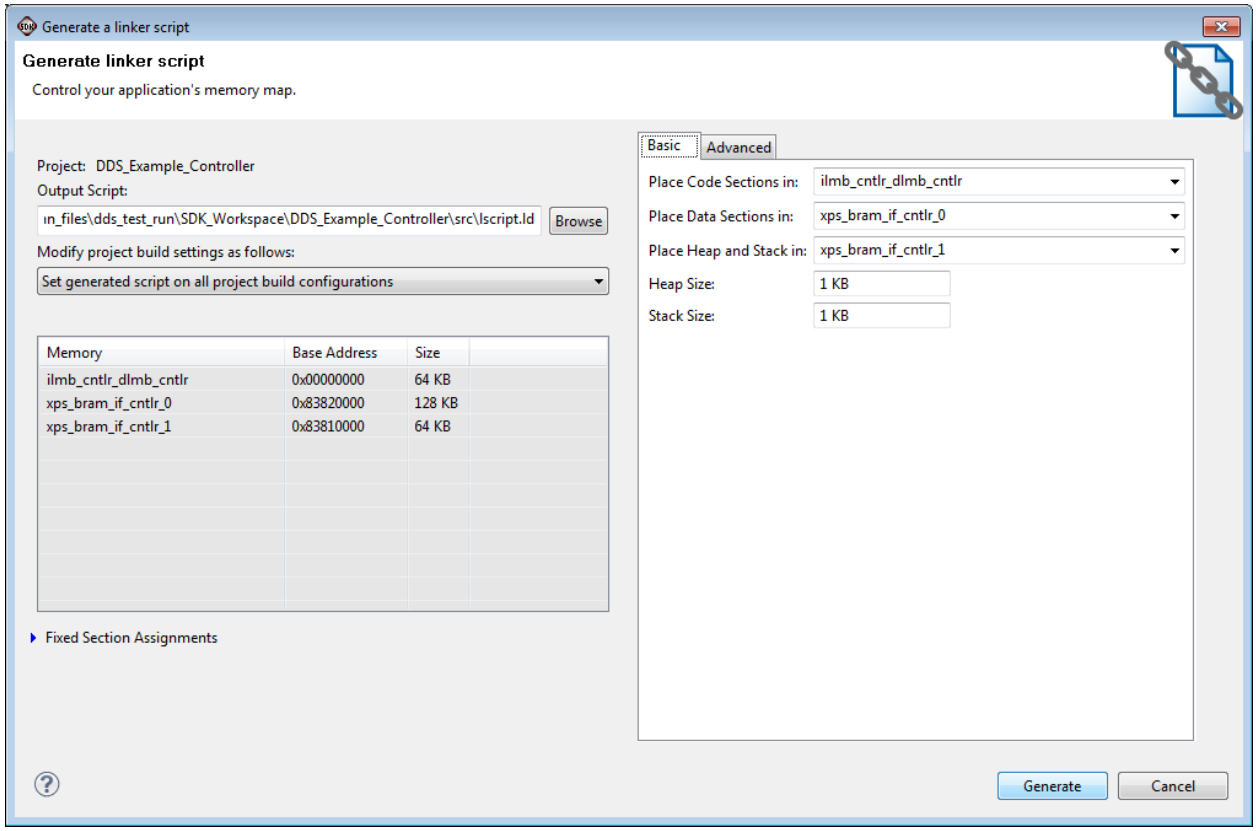


b. Give the project a meaningful name (e.g. "DDS_Controller")
c. For Board Support Package select "Use existing->standalone_bsp_0
d. Click **NEXT (**not "finish"**)** and select the "Empty Application" template.

6. Now it's time to write our code.
   a. Download the example code from the website, unzip the folder and drag the three code files into the "**src**" directory of your new software application project. (Drag from windows explorer directly into the SDK application window).
   b. Read the code thoroughly and make sure you understand its operation.
      i. Look through the C code to understand how the serial port is used. Scroll down in the code and look for the lines which read and write the "phaseInc" register in the model. These lines start with XIo_In32 and XIo_Out32 respectively. Notice the first argument in these function calls: PHASEINC. This is a shorthand for XPAR_DDS_EXAMPLE_PLBWP_0_MEMMAP_PHASEINC, which itself represents the memory address of the register in the model. The tools creates this constant automatically in xparamters.h. The second argument of the write function is the value that gets written to the register
      ii. Look also at xparameters.h, which contains the #defines used in the controller.c and controller.h for writing to the **phaseInc** and register.

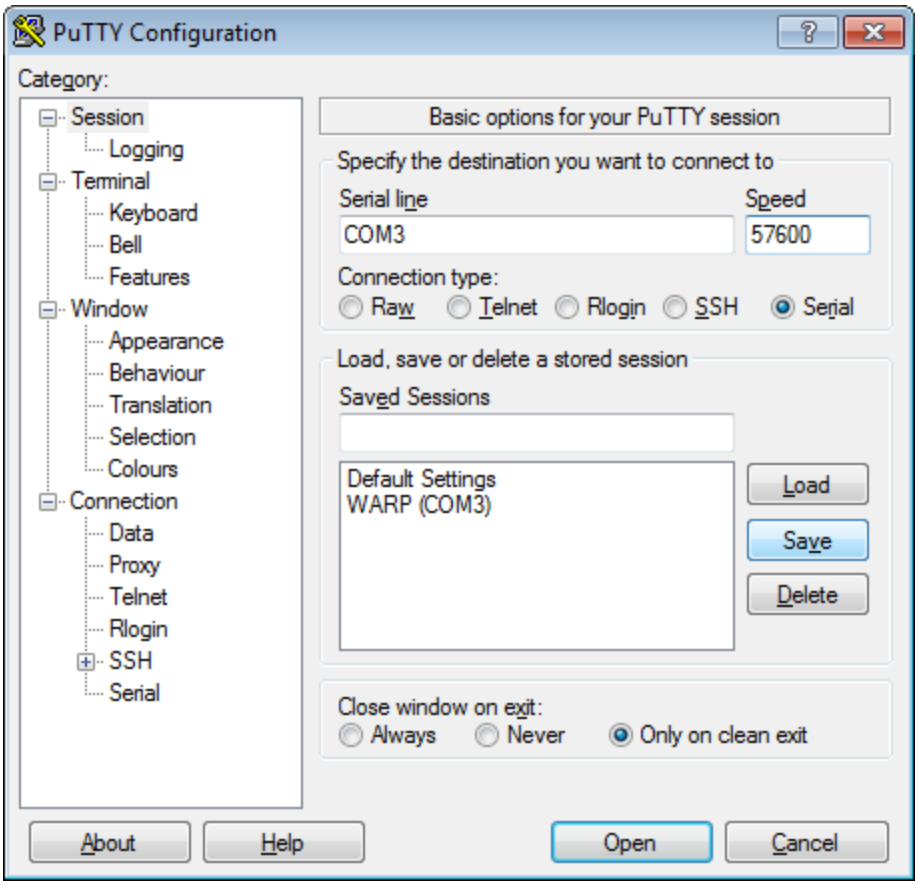
c. Click **save** and the code will compile automatically. Make sure there are no errors.

7. Creating the Linker script.



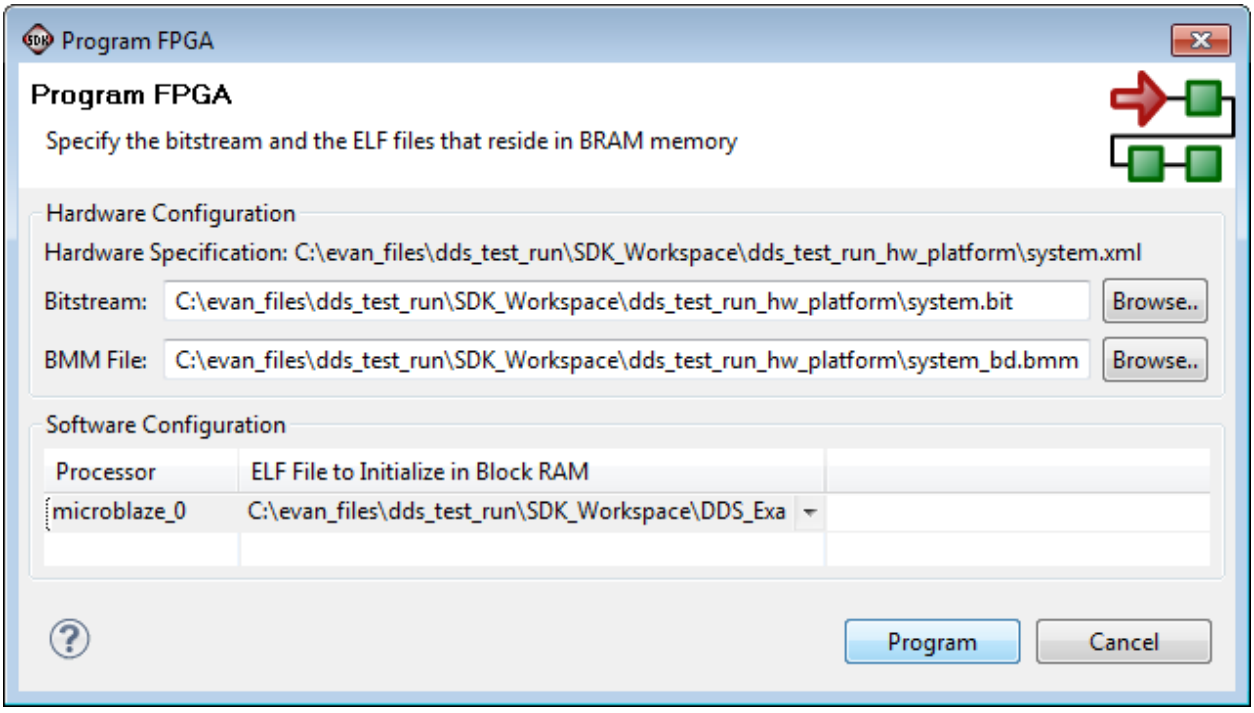
a. Click icon for application project, and click **"Generate linker script"**
b. Stick the Code, Data, and Heap+Stack into the three separate memory maps as shown above.
c. Click **Generate**.

8. Configure PuTTY to talk to the WARP boards.



a. Go to device manager to figure out which COM port is connected to the WARP boards.
b. Open PuTTY.
c. Select the COM port connected to the board.
d. Set the speed to 57600, serial-type connection, and save the configuration for later us.

9. Program the FPGA



a. Select Xilinx Tools -> Program FPGA. In the software configuration tab, change "bootloop" to the ELF file that your application has generated (see above). Click program.
b. You should see text printed to the PuTTY command window, and see the sinusoid on the scope.