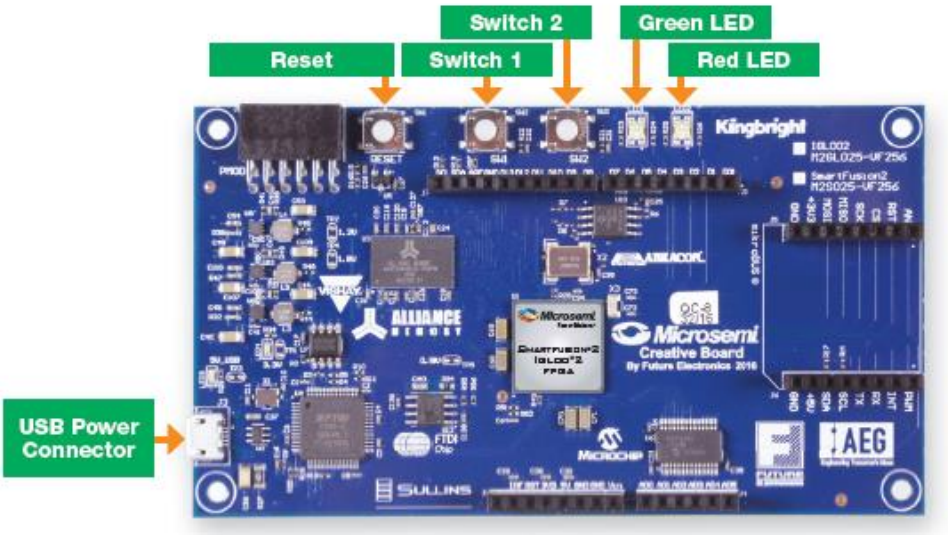
Future Electronics – Microsemi

IGLOO2 Creative Development Board

LAB 3

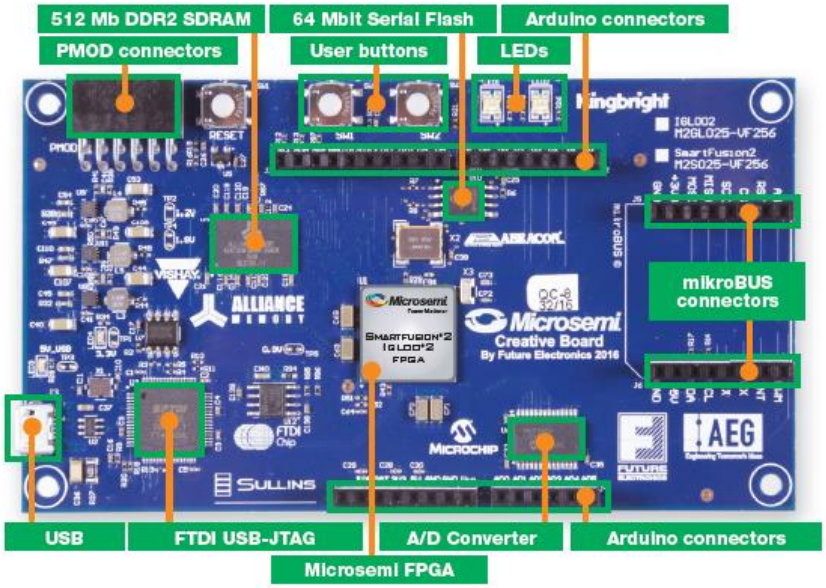
Release - Version A2

The IGLOO2 Board



SW1, SW2, Green LED and Red LED located on top board edge

USB connector
(for power and programming located on left board edge)



Install Libero prior to Lab3

In order to complete this series of Labs you must download the Microsemi SoC Libero Development tools and install them onto your laptop (or desktop).

You will need to obtain a free license from Microsemi and install it on your machine.

This lab was originally written to work with Version 11.7 SP1 of the Libero tool suite.

Refer to the Revision table at the end of this document for updates and improvements to this Lab set.

Install the Labs

If you have not already obtained a copy of the Labs via other sources, please copy the **Future** folder from the USB stick onto the Root of your C: (or D:) drive.

When installed, you will have the following folder paths on your computer:

C:\Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Source

C:\Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Lab2_VHDL

C:\Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Lab2_ver

C:\Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Lab3_part2

Lab 3 Goals

- Create a System Builder design that can access the IGLOO2's eSRAM and eNVM memory
  - Enable access to eSRAM and eNVM in the Device Features section
  - Add User Clients for Memory
  - Add a Peripheral AHBLite Master interface to user logic
  - Use System Builder to configure the internal FPGA clocks
- Use the IP catalog to generate a TPSRAM IP block
- Use Smart Design to import and connect an interface to the Creative Board's Serial Flash Memory (SPI Flash) mounted on the PCB
- Place & Route the design
- Press some buttons and observe eNVM, internal Fabric SRAM, and external SPI Flash activity via Smart Debug

Expected Results for Lab 3:

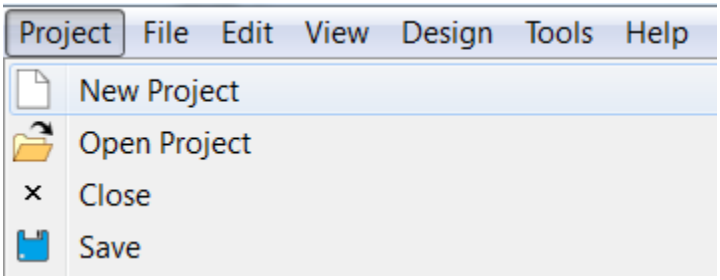
- Become familiar with the Smart Design tool flow
- Successfully step through the GUI and begin to see all the internal features available to you
- Use SW1 to change the values applied to the embedded SRAM or Non Volatile FLASH
- Use SW2 to send SPI traffic between the FPGA and the external SPI Flash on the board
- Monitor the Signals on the SPI interface to see data traffic
- Read the 25th page of the eNVM and confirm that the 0x00 to 0x1F pattern can be written there

CREATING A PROJECT

Start the Libero SoC tool set by double clicking the ICON on your desktop or pathing to the installed folder and double clicking on the file called **Libero SoC v11.7** or via the **Start Menu.**

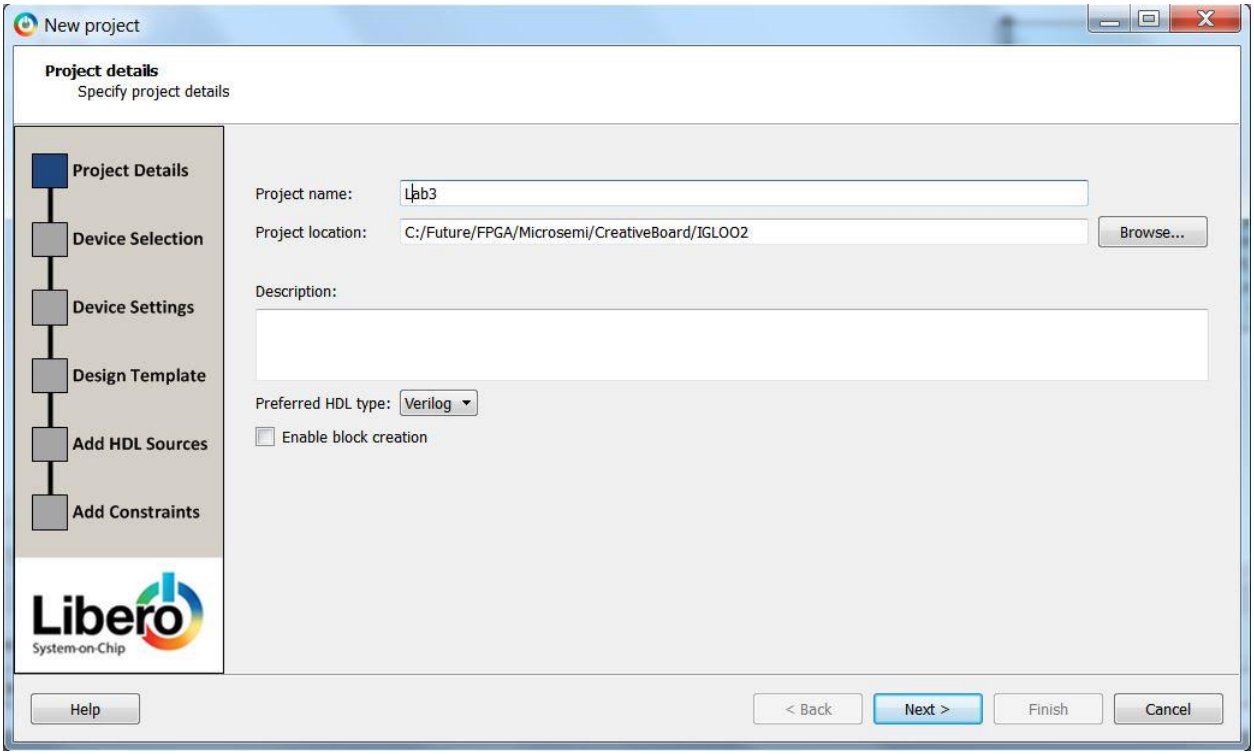
If this is the first time you have started the tools on your machine, it could take up to 45 seconds for the tools to start. If the tools ask you to check for updates, select **NO**.

Like many Windows tools, there are often several ways to accomplish the same results; in this lab we will be showing you how to use the Menu system, or *double-clicking*, or *right-clicking* and picking items to select and run things.



Select **Project**… **New Project**

Project Details



Enter the *Project name:* as: **Lab3** (Note: remember, Verilog is case sensitive.)

Set the *Project location:* to: **C:/Future/FPGA/Microsemi/CreativeBoard/IGLOO2**

(This is the bottom of the folder path that you copied to your hard drive, where you already created Lab1. There will also be solutions folders for Lab2 in case you wish to start from there without completing Lab1. There will also be a folder for Lab3_part2 which we will use later in this Lab. *For now, make sure that you enter the project location to where you actually placed the image if not where we recommend*).

Note: For this lab we will select **Verilog**.

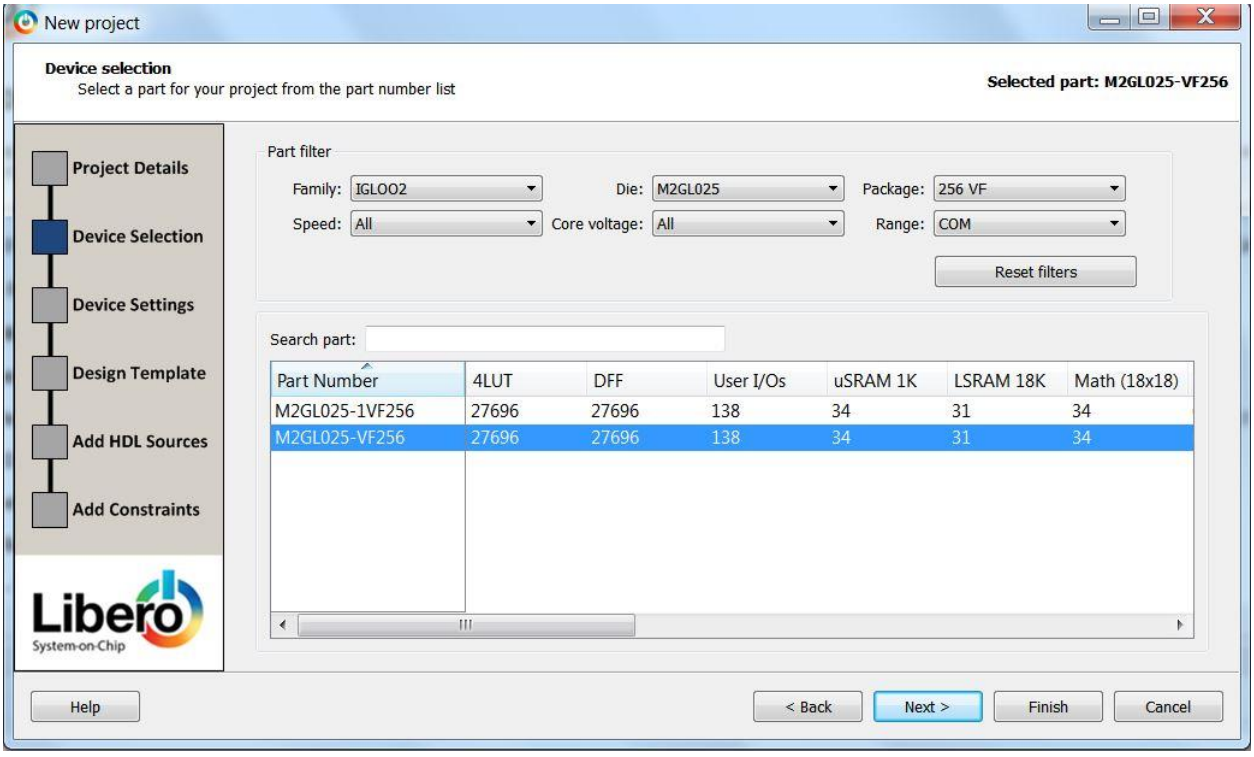
Note: You are free to select either language for your own designs in the future.

Select the *Preferred HDL type:* as: **verilog** using the pull down, if not already set.

Confirm that the check box is **NOT** checked (Enable block creation).

Press the **NEXT** button.

Device Selection



We will be using the following device for this lab: M2GL025-VF256

There are a few ways to set this; for this lab, we will use the **Part Filter** method.

Set Family: IGLOO2 Die: M2GL025 Package: 256 VF

Speed: All Core voltage: All Range: COM

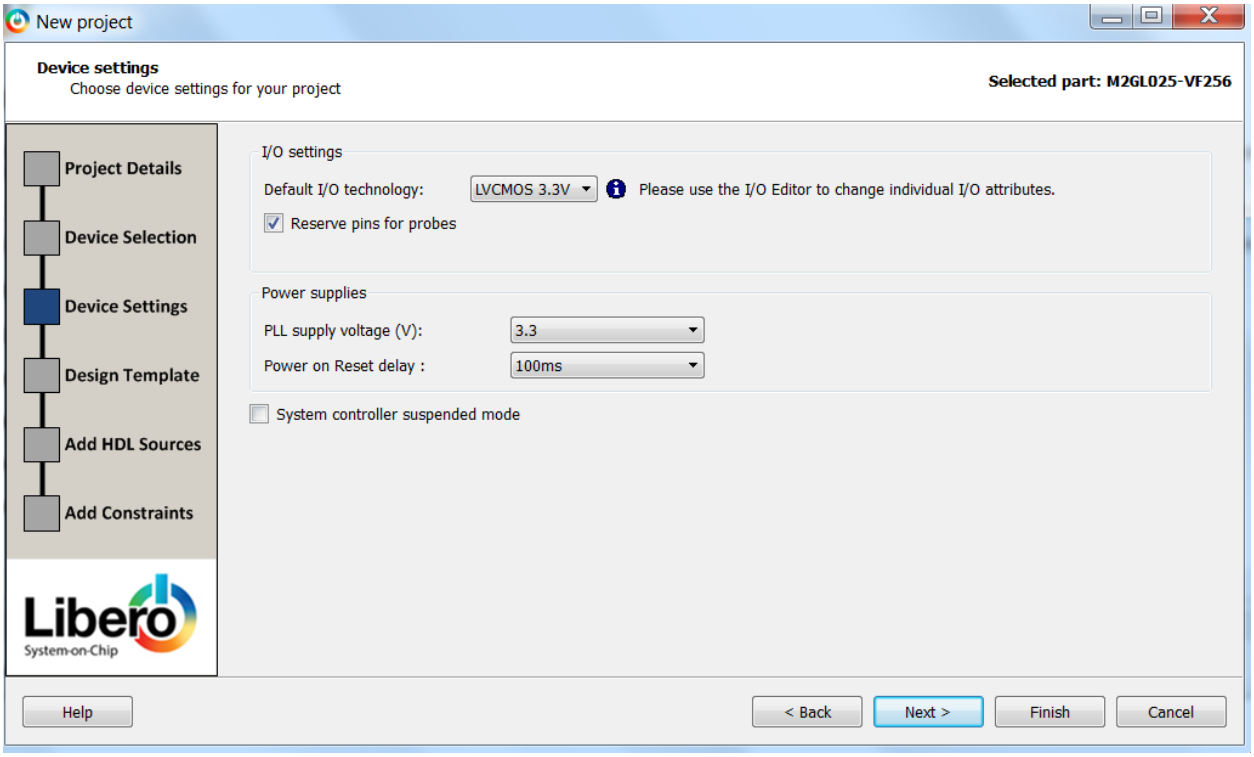
This will reduce your choice to two parts called:

M2GL025-1VF256

M2GL025-VF256 ← select the standard speed one, the one without the "-1" in it.

Press the **NEXT** button.

Device Settings



The Libero tool set will default to **LVCMOS 2.5V** – we will be changing that for this Lab.

Set *I/O Settings -> Default I/O technology:* to **LVCMOS 3.3V**

Confirm the *Reserve pins for probes* **IS** checked.

Leave (or change) the *Power supplies*

PLL supply voltage (V): as **3.3**

Power on Reset delay: as **100ms**

Confirm the *System controller suspended mode* is **NOT** checked.

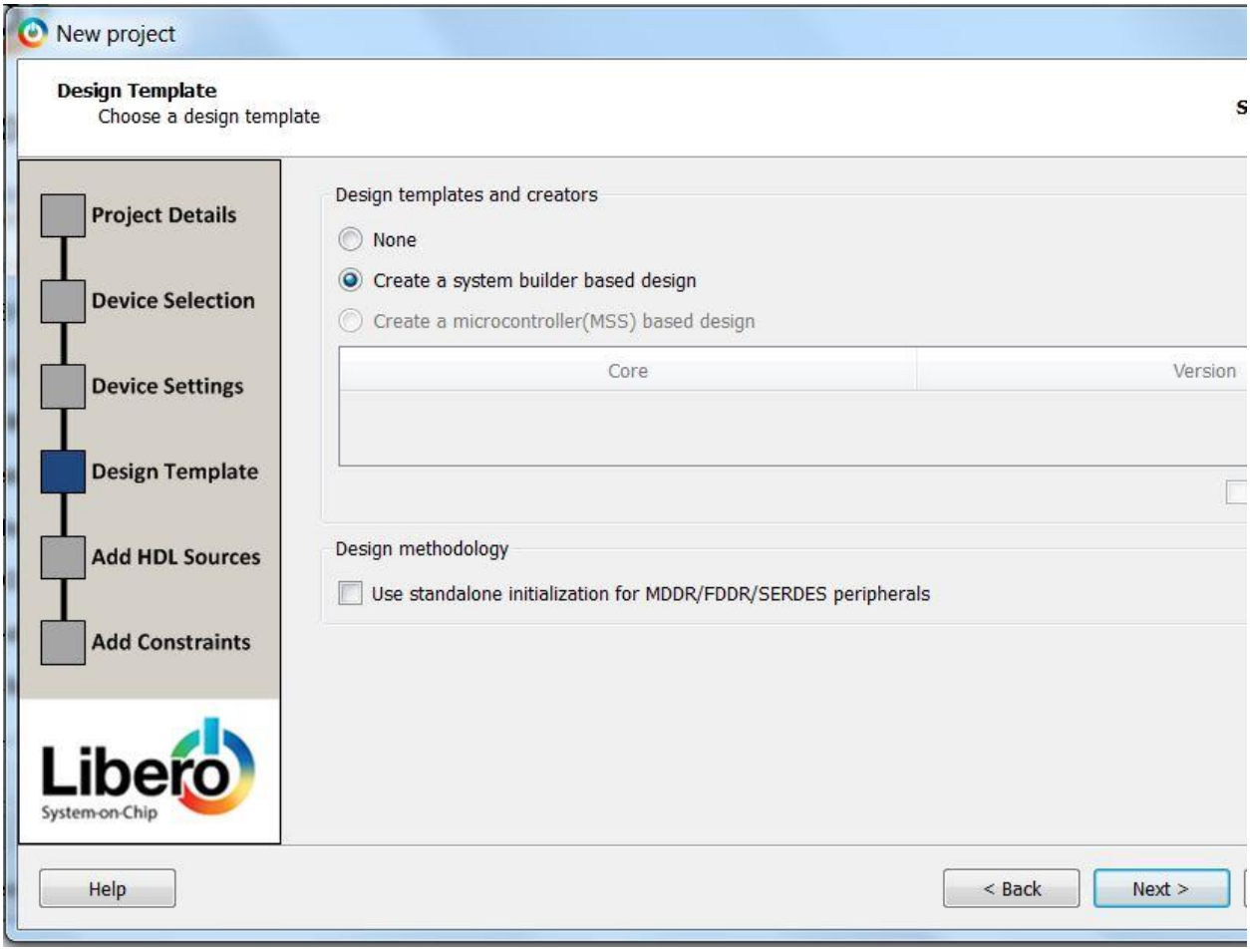
Note: In general, not all banks work at 3.3 volts, so you may wish to leave the default at 2.5 volts. Then you will set the voltages for the different IO pins using the constraints editor or by directly creating the constraints file once you are skilled with the tools.

These decisions are really a matter of how you intend to use the part and what voltages you wish to connect to which banks when planning your overall design.

For this lab it is easier to have you change the default to 3.3 volts like we did in Lab1.

Press the **NEXT** button.

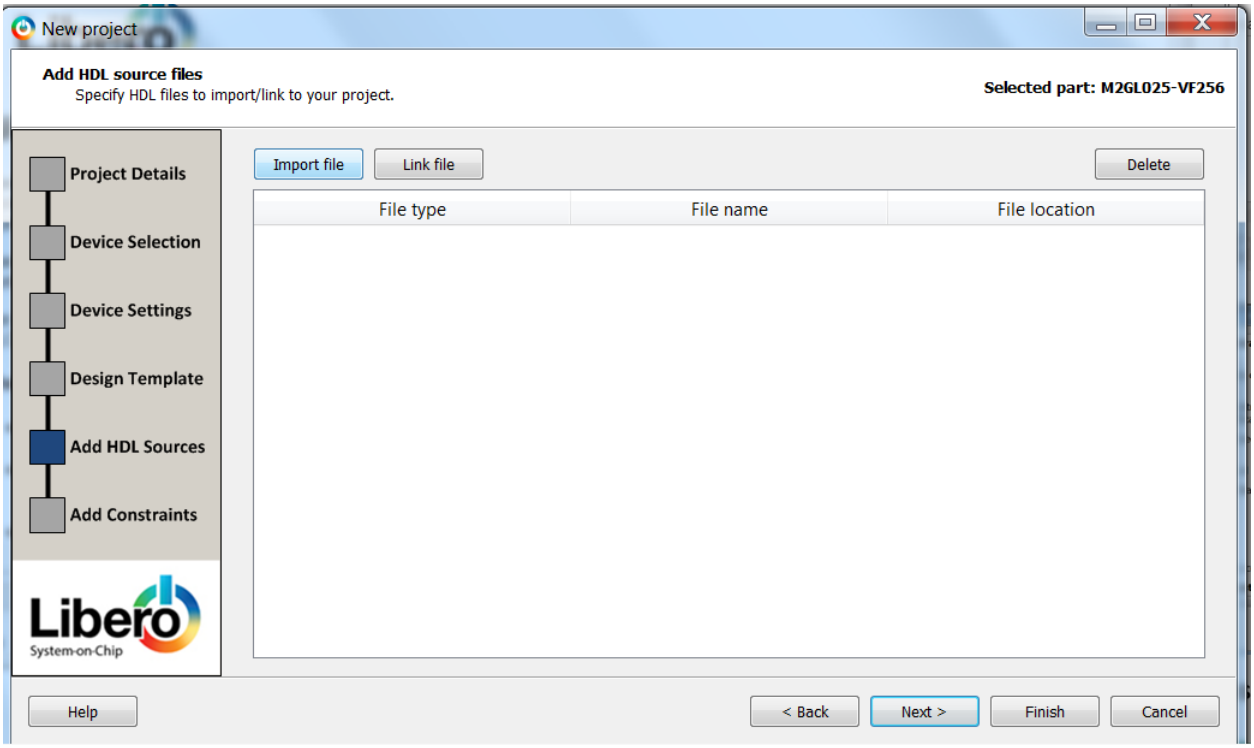
Device Template



Check the radio button called **Create a system builder based design** for this lab.

Press the **NEXT** button.

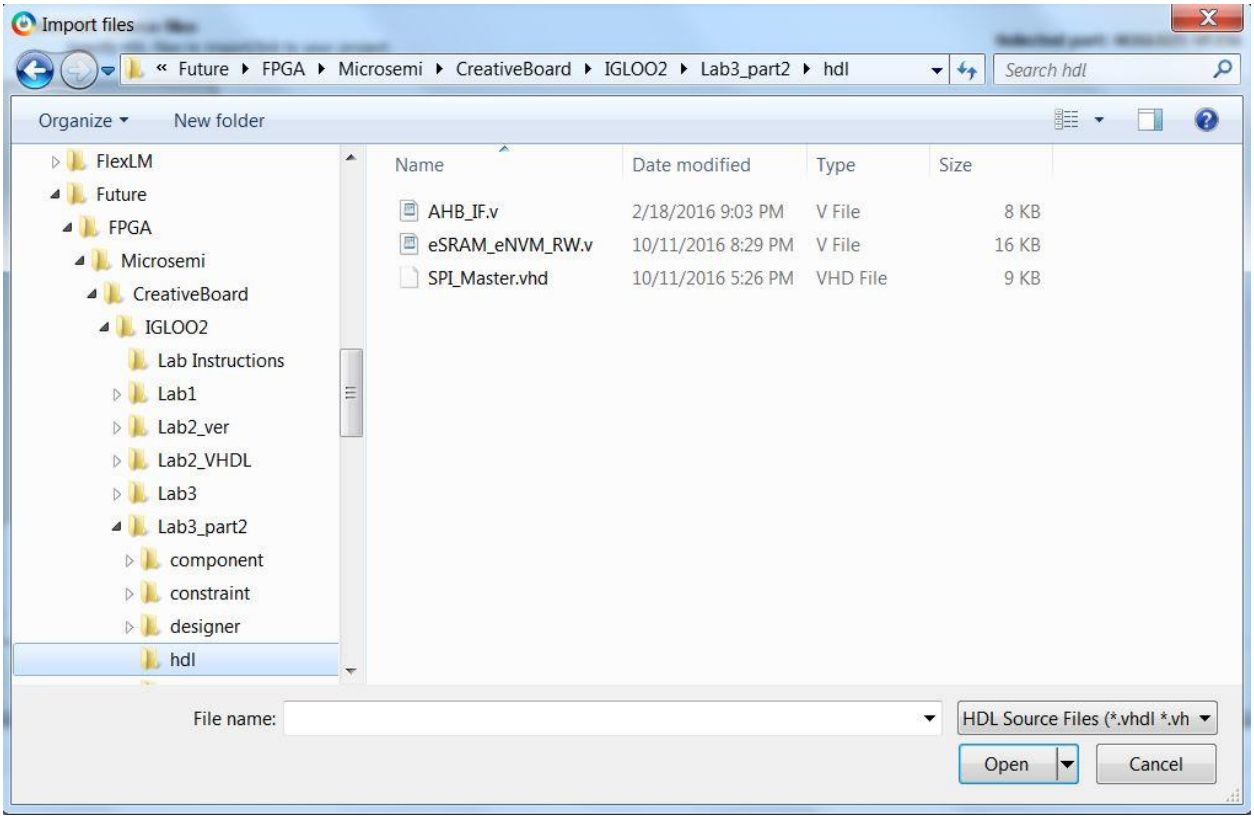
Add HDL Sources



We will now add some HDL source files to the design.

Select the **Import file** button at the top of the screen.

An overlay window will appear.



Path to the following folder:

Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Lab3_part2\hdl

Select and add all three files.

You can use the windows *ctrl-click* function to select multiple files or add them one at a time.

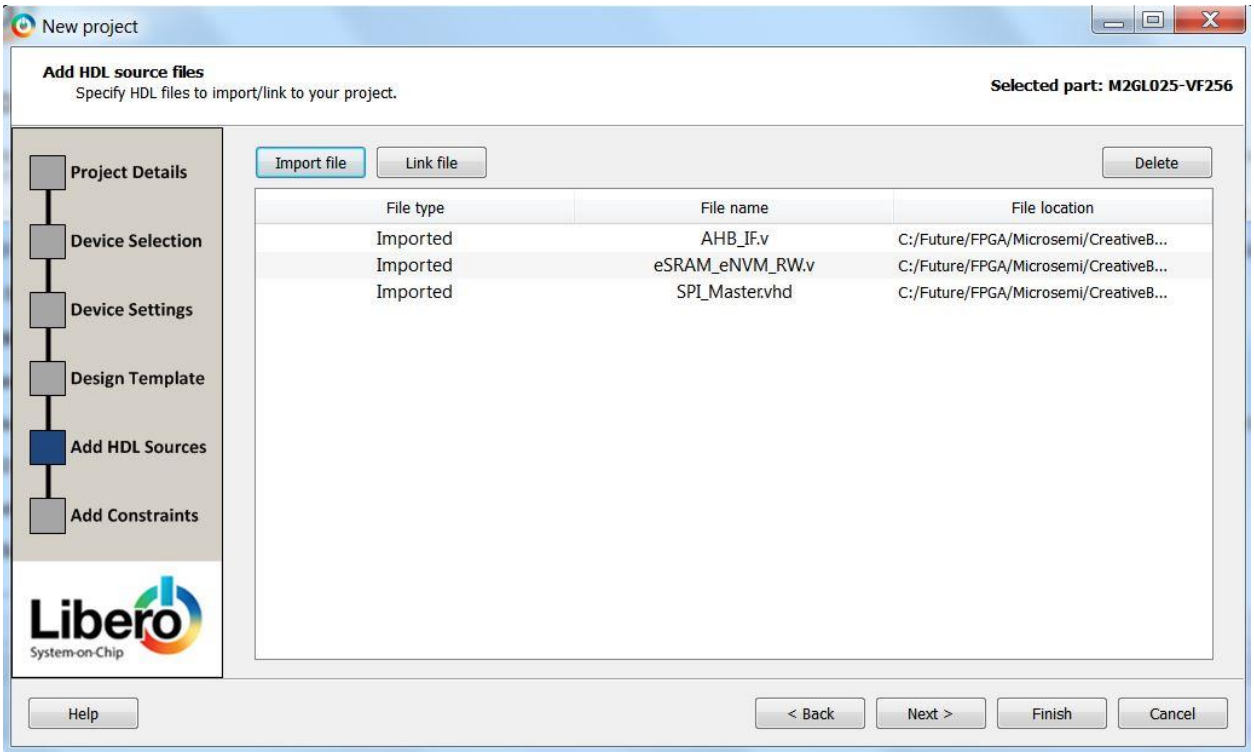
Add AHB_IF.v eSRAM_eNVM_RW.v SPI_Master.vhd

Press the **OPEN** button.

Note: We are adding both VHDL and Verilog files to the design for this lab.

The Libero tools provide dual language support. There are certain things you may need to be very precise about in terms of file and module naming, and capitalization of signals, etc.

For this Lab, all this has been addressed for you. You will also need to be aware that if you have the free included ModelSim ME simulator, it will only support one language at a time for pre-compiled designs (functional simulation). Once your design is compiled and fitted into the part, the post routed design will be simulated in whichever language you have selected.

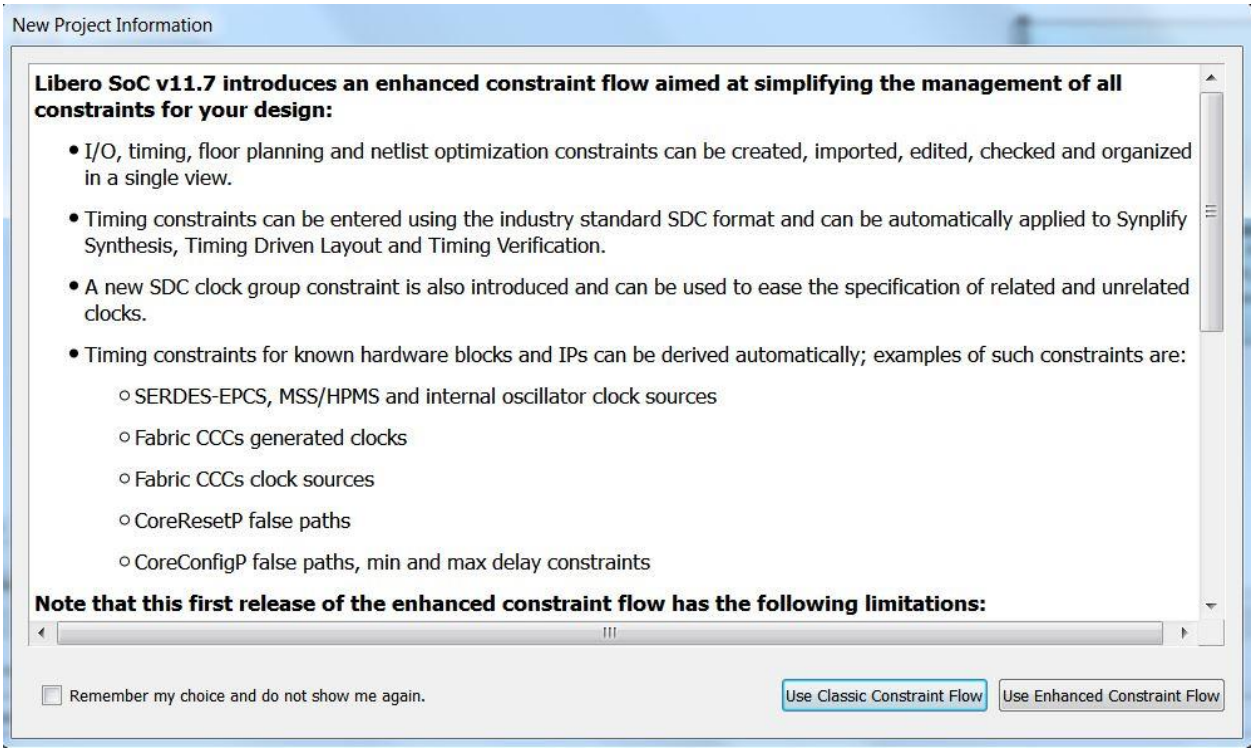


If you have successfully added all 3 files, your screen should look like this.

We will skip adding constraints for now.

Click the **FINISH** button.

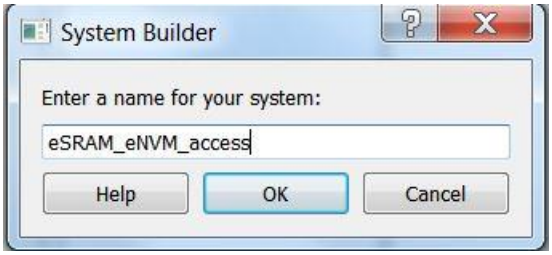
An overlay window will appear.



For this lab we will select the **Use Classic Constraint Flow**.

Click the **Use Classic Constraint Flow** button.

An overlay window will appear for you to name the System you have just described.



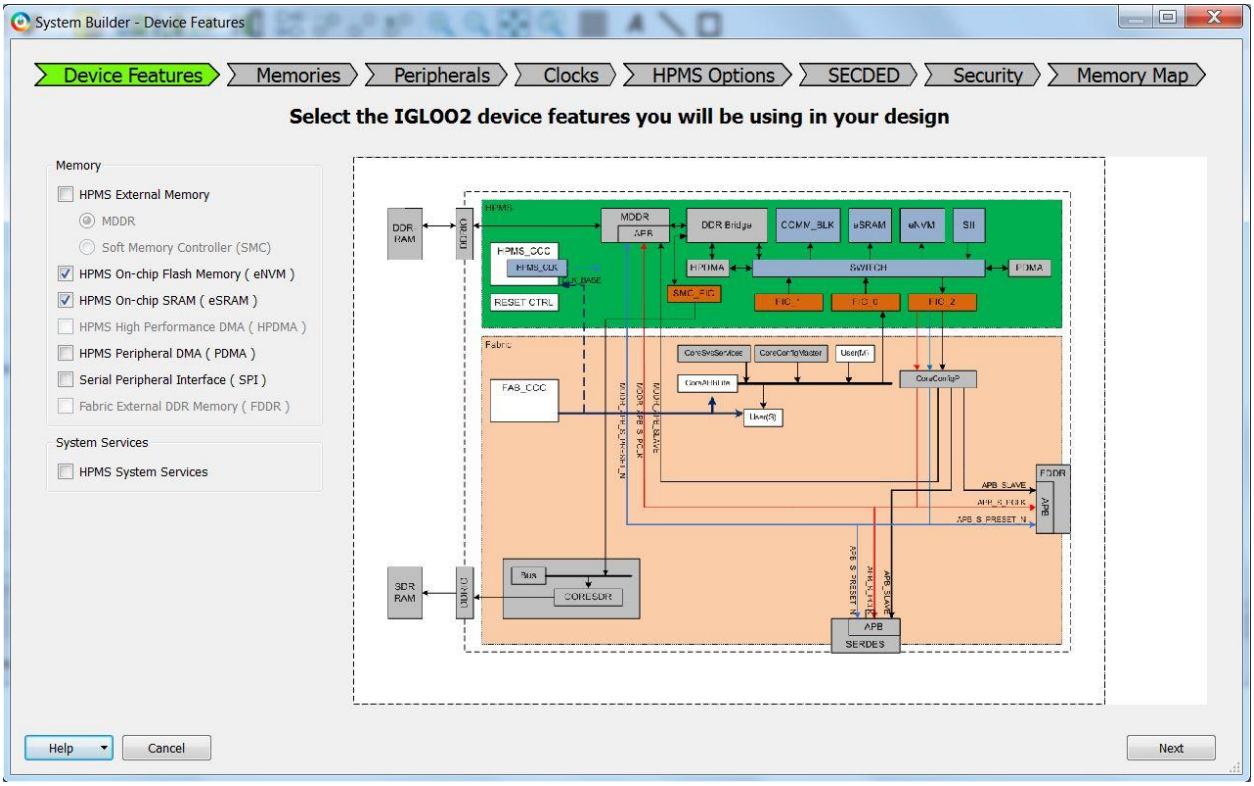
Enter the name **eSRAM_eNVM_access** (Caps and Lower Case matter here)

Everything we have done so far is pretty much just like you did in Lab1. We skipped a few steps and we checked a few different options along the way. Now we are going to utilize a useful tool to help us take advantage of some of the Hard Structures built into the IGLOO2 and SmartFusion2 devices. It will make it much easier to access those elements in your designs.

Click **OK**.

System Builder

After a few seconds, the following screen will appear.



It will look something like this.

From this GUI, you will select the features that you would like to enable in this HPMS (High Performance Memory Subsystem).

You will check or uncheck or enter certain values into each of the screens as you work your way across the different sections as indicated by the Green highlighted section across the top of the GUI.

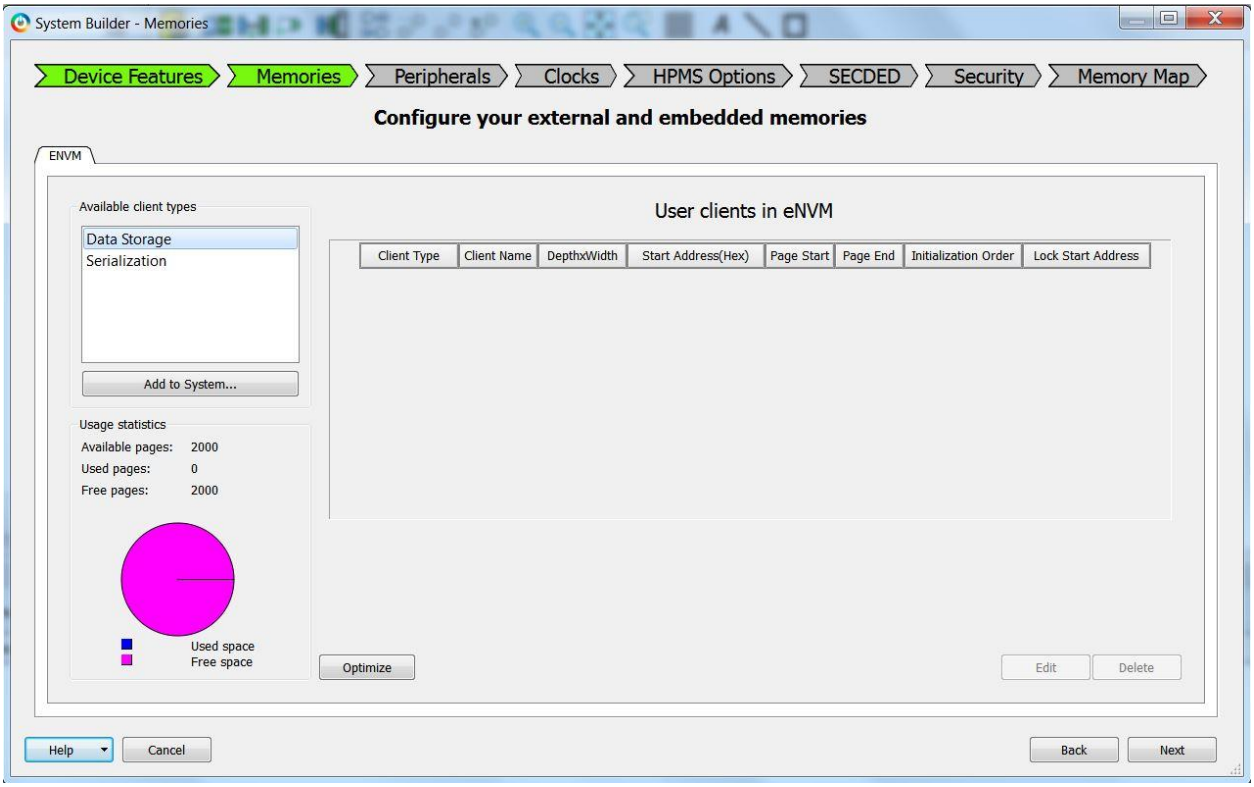
Let's begin with the **Device Features**.

Check the **HPMS On-chip Flash Memory (eNVM)**

Check the **HPMS On-chip SRAM (eSRAM)**

As you checked each one, did you notice that one of the little blocks in the green section to the right changed color slightly?

Click **Next** (in the lower right)



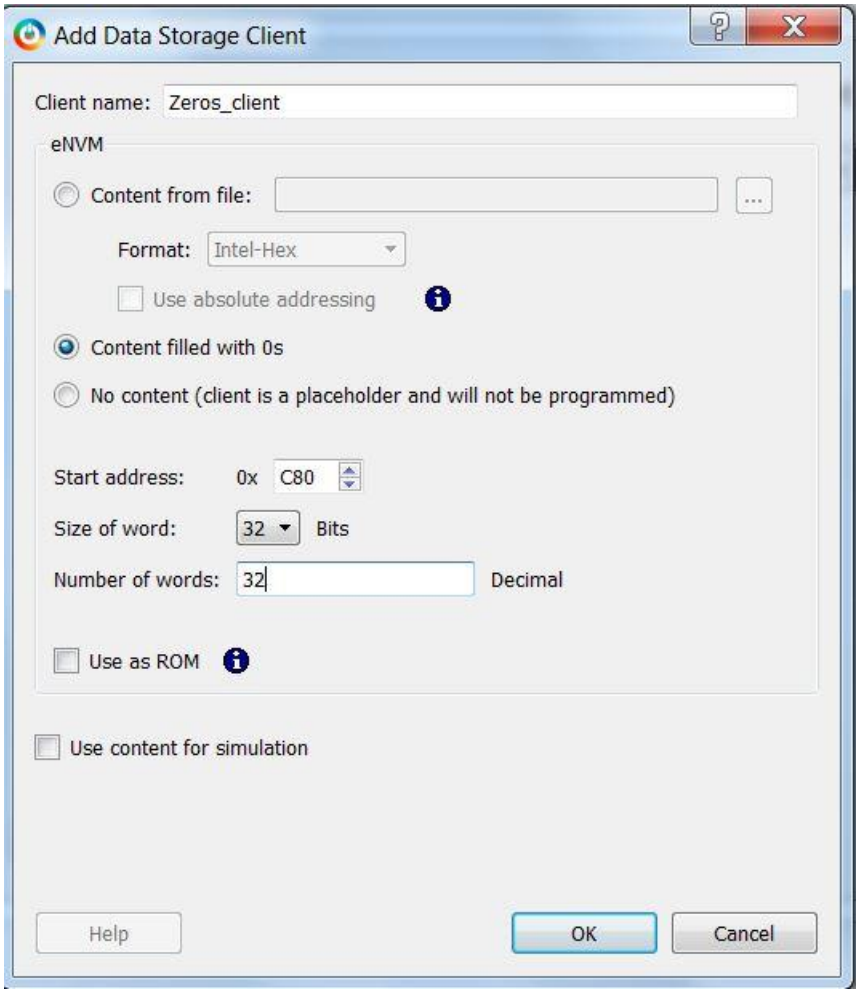
Interesting, we have stepped to the next Green element to configure, and the interface has detected we selected to access some internal memories, so the tool added the **Memories** GUI.

We are going to do something with a data storage client type.

Click to highlight **Data Storage** and notice that the **Add to System…** option becomes available.

Click **Add to System…** and an overlay screen will appear.

Fill in the items indicated below:



Client name: **Zeros_client**

Select: **Content filled with 0s**

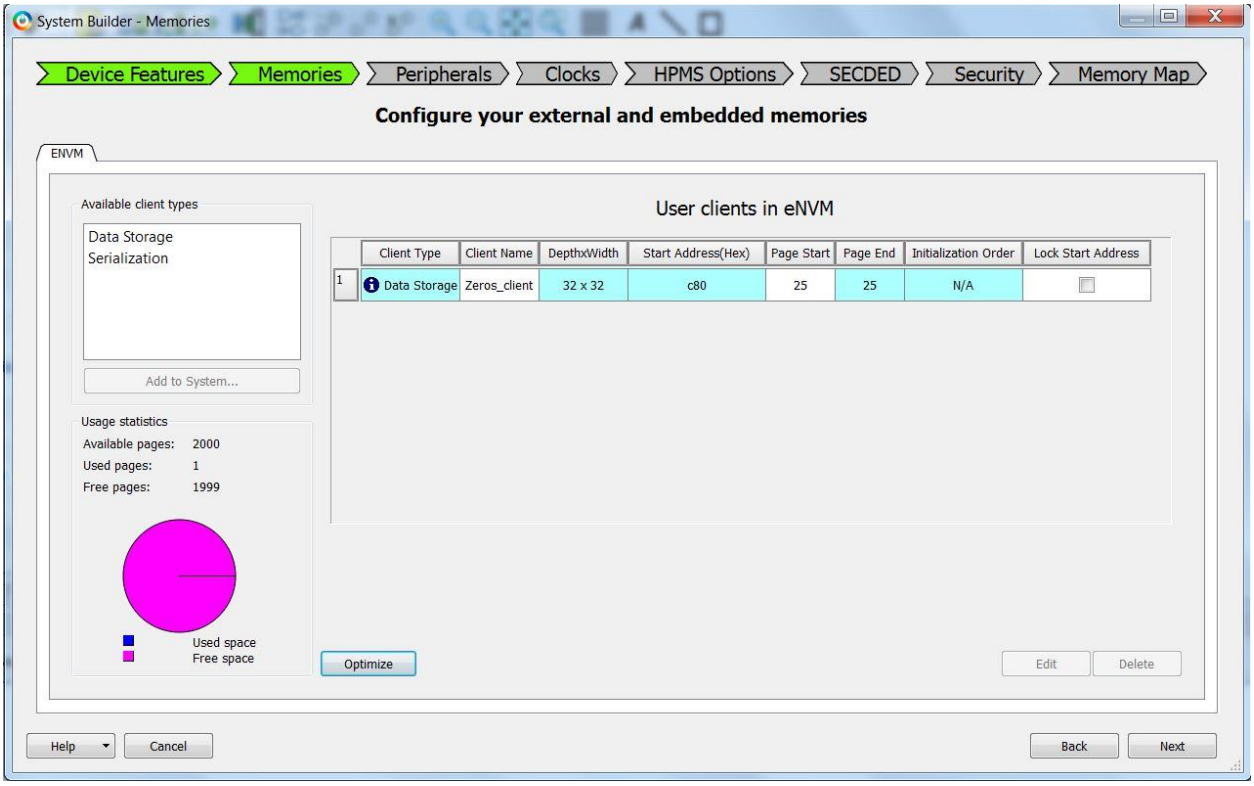
Start address: 0x **C80**

Size of words: **32** bits

Number of words: **32** Decimal

Click **OK**.

Confirm that your screen now looks like the following:

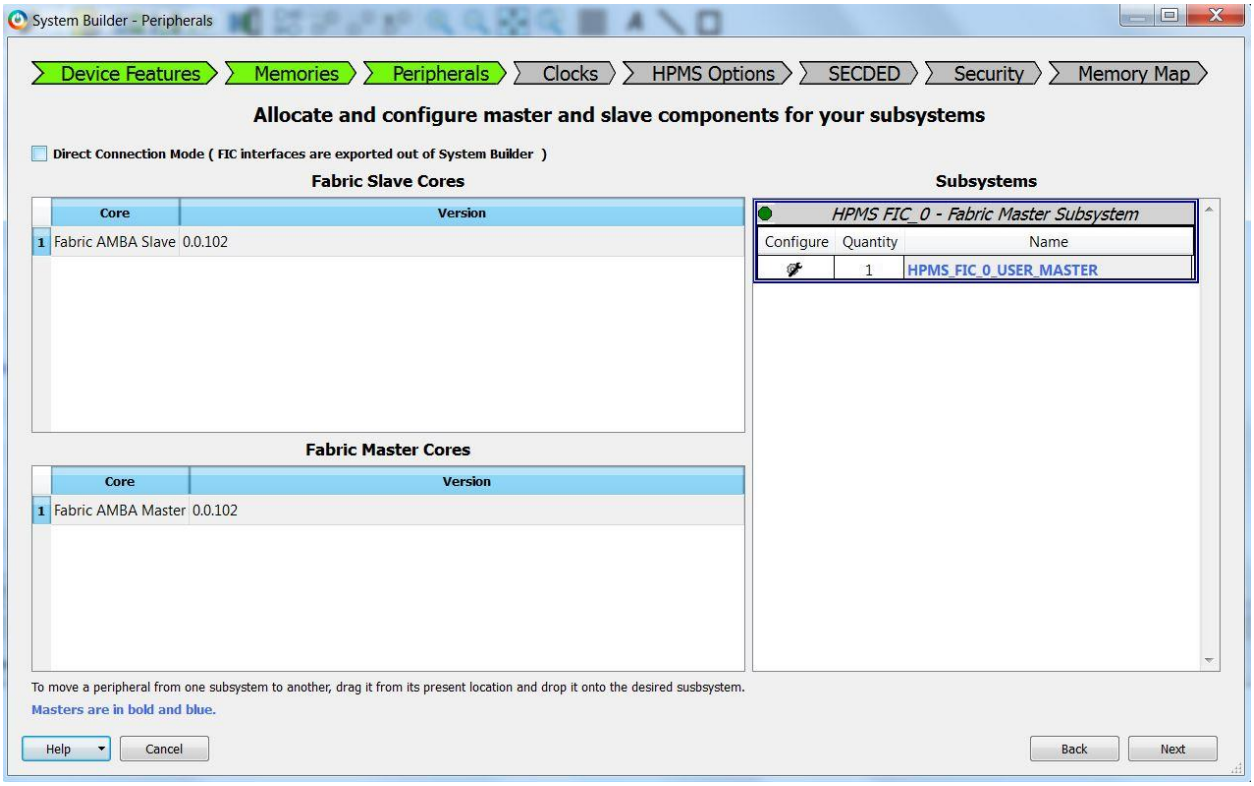


Where we have used the Data Client to help us 'zeroize' the internal eNVM (Non Volatile Memory)

Take note that the starting address of 0x0C80 results in a Page Start of 25 (0, 1,..,25) and we are filling 1 page.

Click **Next**

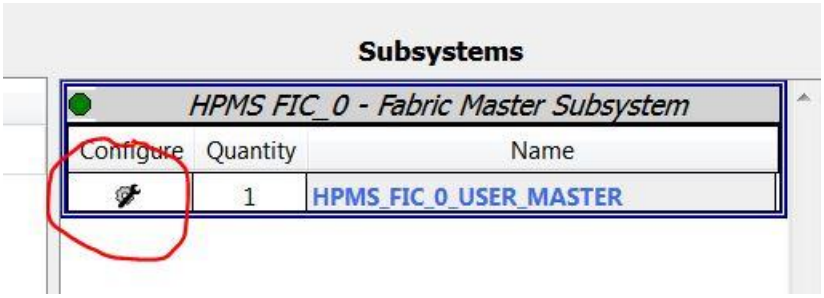
Next is the Peripherals GUI



This GUI is used to help you build the bridge(s) between the FPGA fabric and the HPMS (hard blocks) in the IGLOO2 or SmartFusion2 device. Depending on the size of the part you select, there will be one or more **Fabric Interface Subsystem**s (FIC) in the part. Our selected part has one.

We will take a look at the configuration of this one.

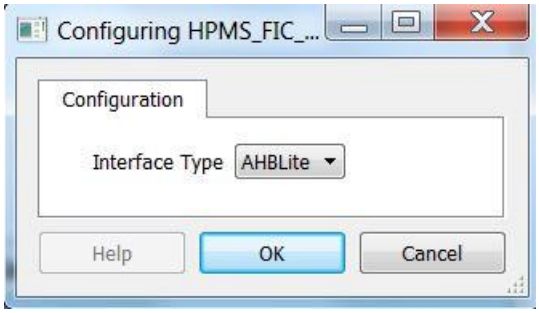
Focusing in on the right hand portion of the above screen:



Click ONCE on the "Gear with Wrench" icon circled above.

Be patient, it takes a looooooong time for the overlay to appear. There are a lot of things going on in the background.

Eventually the following will overlay the screen:



Go ahead and take a look at the different **Interface Type**s the HPMS subsystem provides.

Depending on the complexity of the interface needed, a full bus or one of the more simplified bus interfaces can be selected: AXI AHBLite APB3

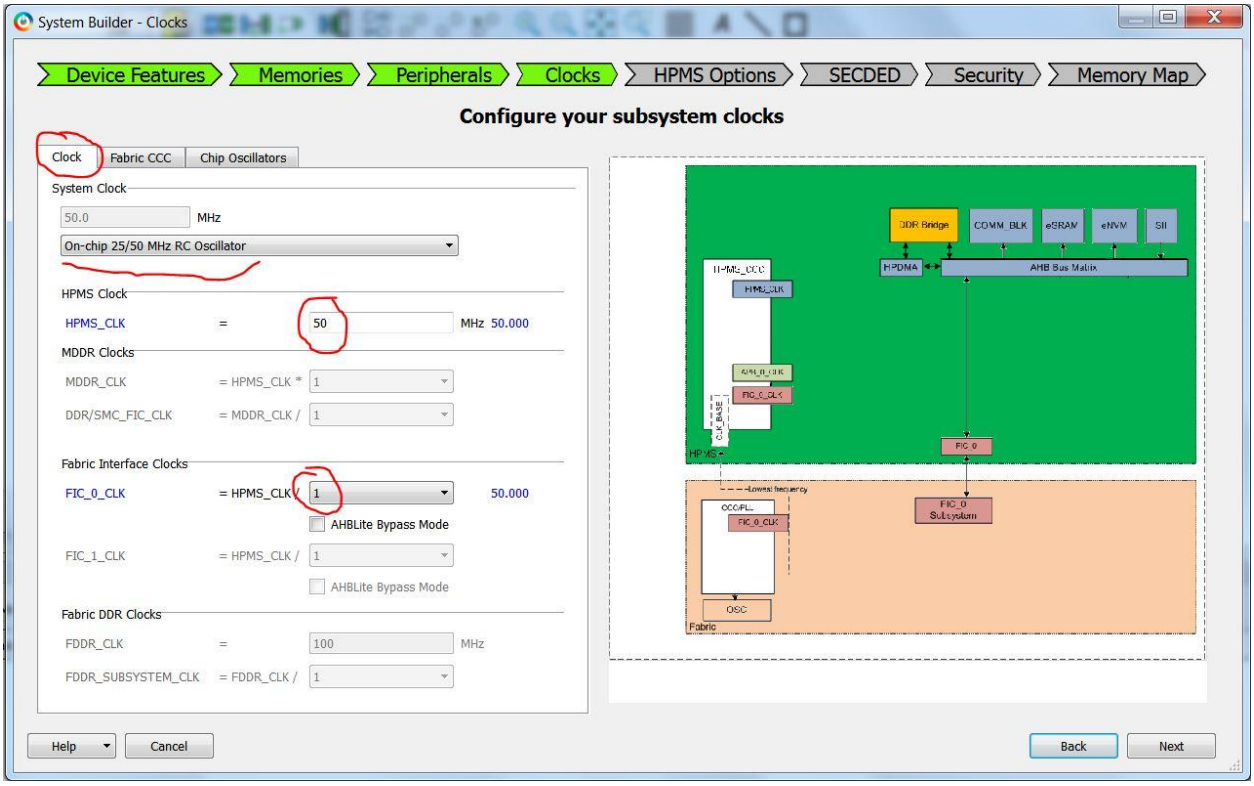
This course will not dig into the details on these. A future course will cover this in greater detail or you can take a Microsemi training course to leverage these features.

When done inspecting, just make sure that it is still selected as **AHBLite**.

Click **Cancel**

Then *Click* **Next**

Next is the **Clocks** GUI



For this exercise we will change from an external clock like we used in Lab1 to an internal Clock.

On the **Clock** sub tab,

Under the section called **System Clock**

Using the *pulldown* – change to the use the **On-chip 25/50 MHz RC Oscillator**

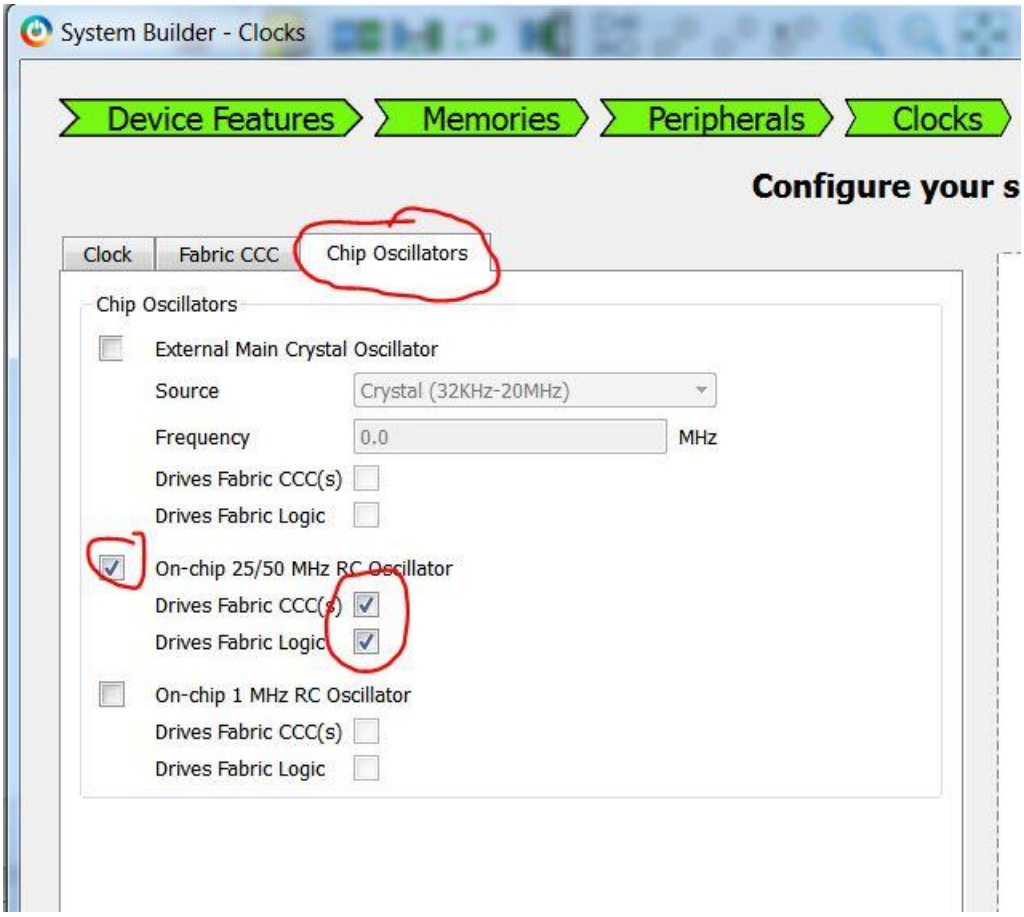
Under the section called **HPMS Clocks**

Set HPMS_CLK to **50** MHz

Under the section called **Fabric Interface Clocks**

Set FIC_0_CLK to **1** this is a divider value so you should see 50.000 reported to the right.

Staying on this screen, change to the Chip Oscillators sub tab.



On the **Chip Oscillators** sub tab, the only section is called **Chip Oscillators.**

Check the second option, and the two sub options below it as in the screen image above.

Checked - On Chip 25/50 MHz rC Oscillator

Drives Fabric CCC(s) - *Checked*

Drives Fabric Logic - *Checked*

Click **Next**

For the HPMS Option – we will take the default values presented. *Click* **Next**

Same for the SECDEC options. *Click* **Next**

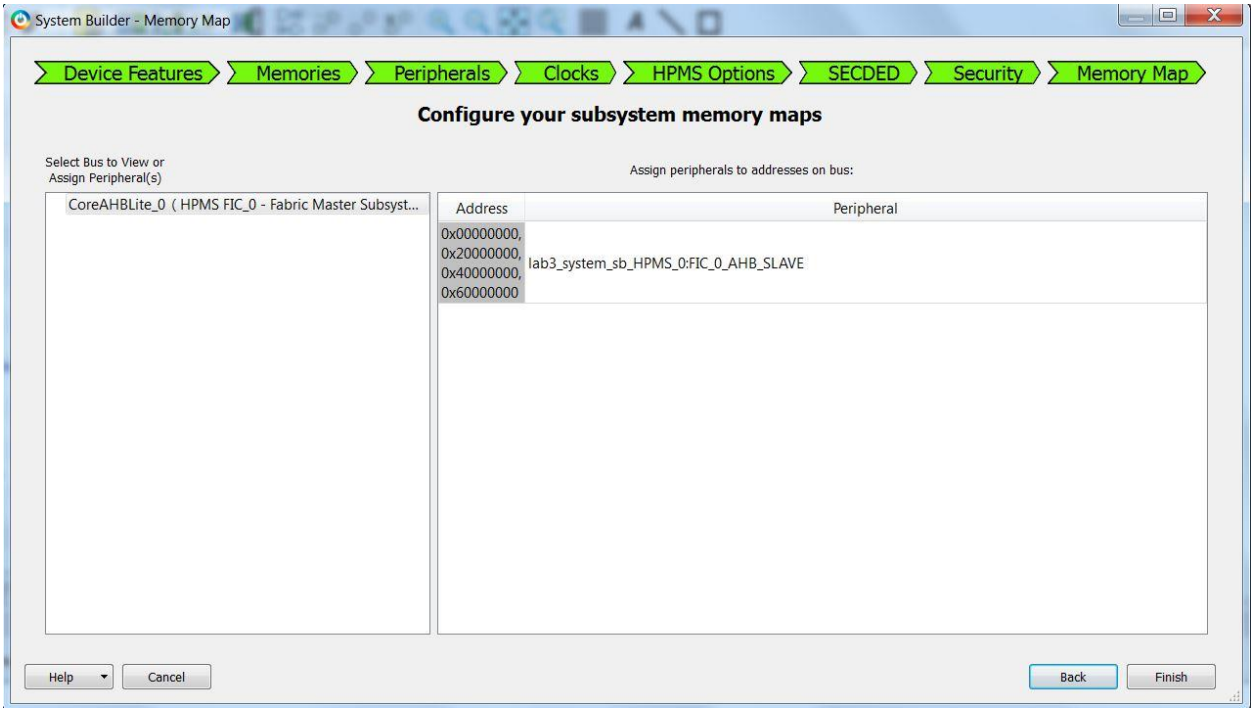
Same for the Security option.

Wow, we just blew by a whole bunch of stuff – all of which is not important for this lab.

Let's just say there are a lot of capabilities and features that you can take advantage of with these parts and we would be more than interested to come and discuss all of them with you in your next design. For now, let's press on.

Click **Next**

The tool will indicate that it is now building the "System Builder" and after a short amount of time present you with the following screen.



The Base Addresses for the **CoreAHBLite_0** (HPMS FIC_0 Fabric Master Subsystem) is displayed.

eNVM virtual base addr = 0

eSRAM_0 base addr = 2000 0000

SPI, IRQ, DMA, other base addrs = 4000 0000

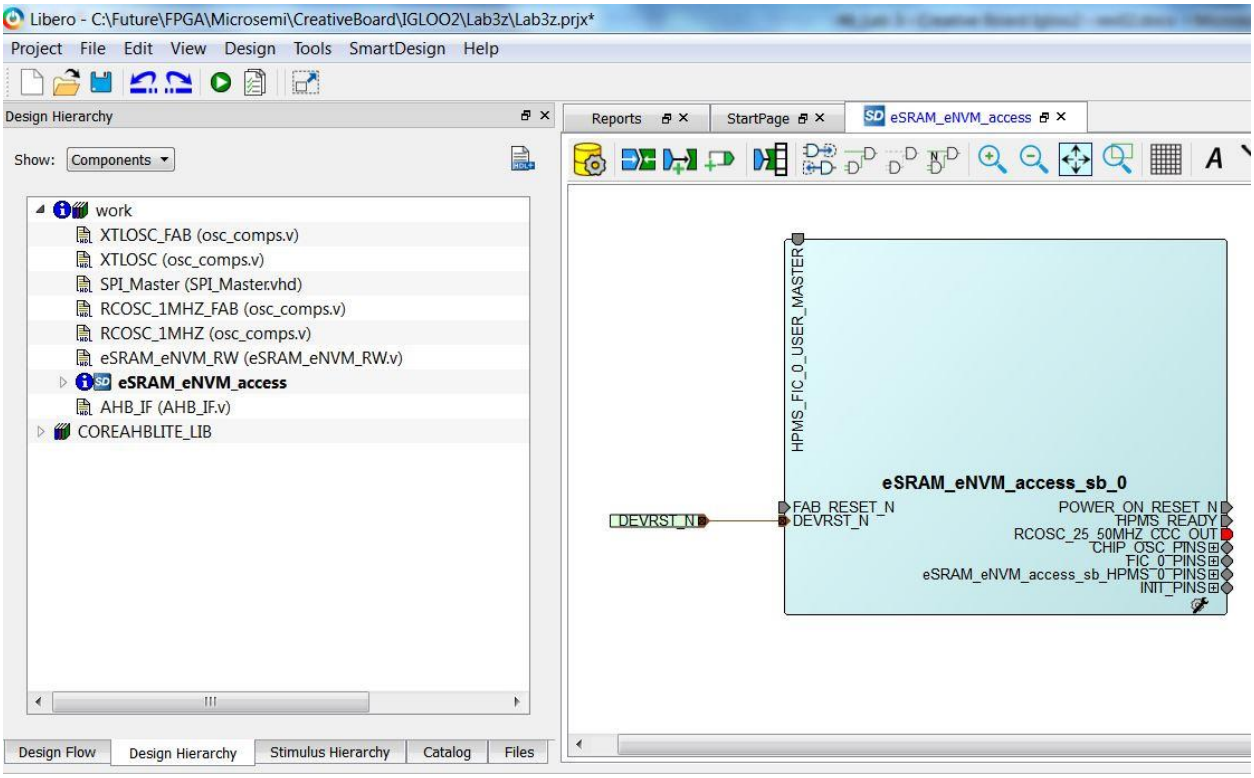
eNVM_0 base addr = 6000 0000

(See the IGLOO2 HPMS User's Guide or the SmartFusion2 MSS User's Guide for details)

If there were other FIC interfaces in the design, they would be listed as well.

Click **Finish.**

After a short amount of time (maybe 15 seconds – watch for the green % done bar across the bottom); the main interface of the Libero Tool set will appear.



A few things to notice here.

We are presented with the **Design Hierarchy** Tab and something in the right window called:

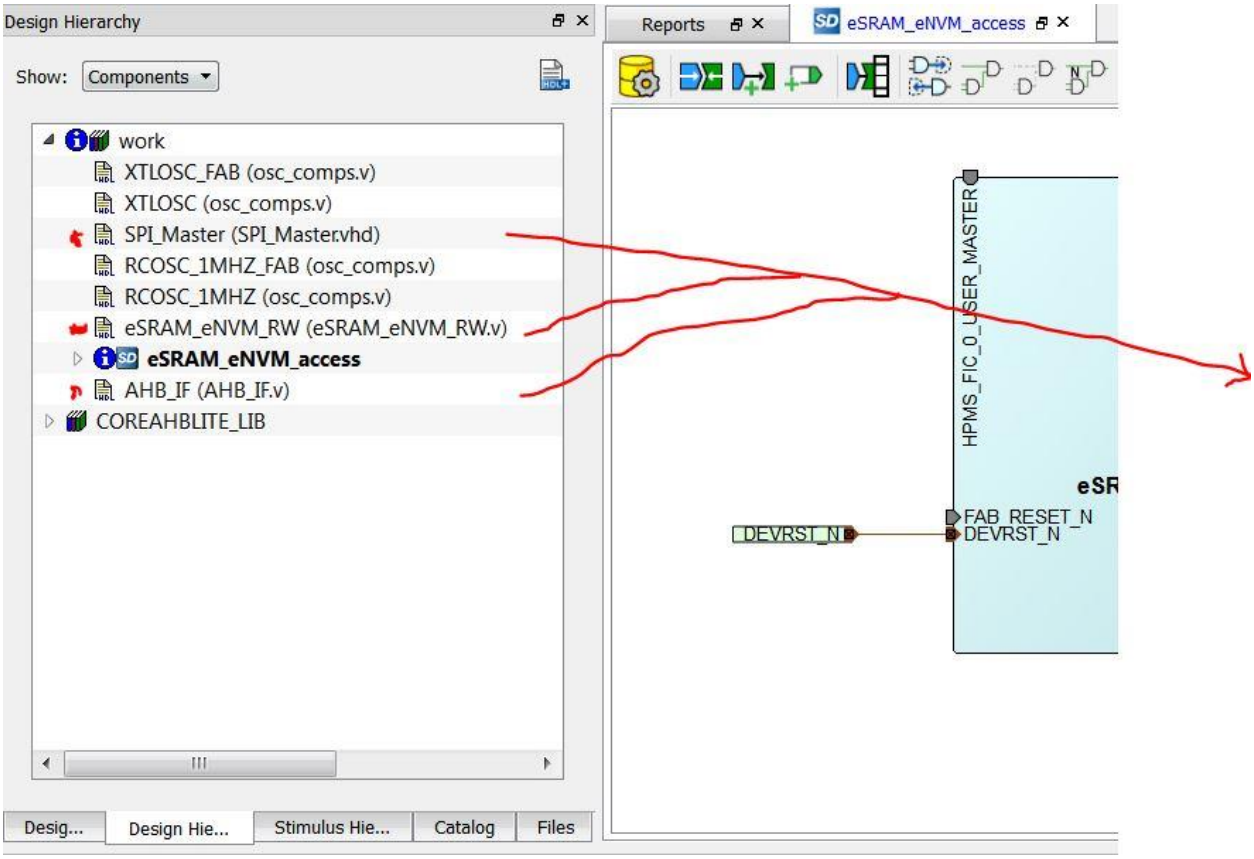
SD eSRAM_eNVM_access

Wow, we have done a lot. We defined a bunch of stuff, added some source files and built an entire sub-system so we would be able to reach into some of the Hard Blocks inside the device.

That is a lot better than instantiating all that in my opinion.

OK, we did skip over a lot, but all we are looking to do here is give you a feel for what can be done with this tool.

Let’s add a few more elements to the design window to the right and hook it all up.



Select and **drag** the above three HDL code modules:

SPI_Master.vhd

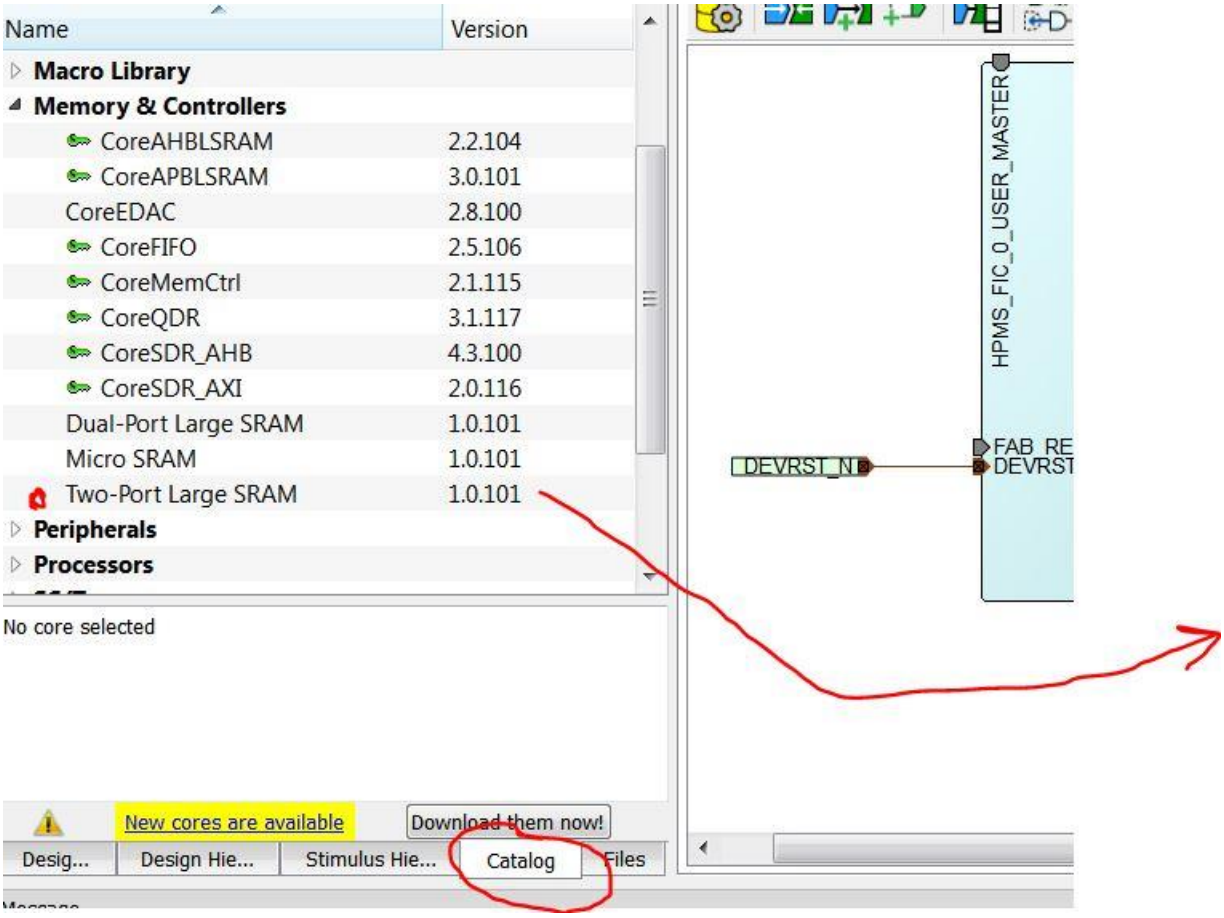
eSRAM_eNVM_RW.v

AHBLite_IF.v

from the Design Hierarchy tab to the SmartDesign canvas.

You can place them anywhere. We will reposition them in two more steps.

Now *select* the **Catalog** Tab so we can get some Free IP.



On the **Catalog** tab, expand the **Memory & Controllers** section.

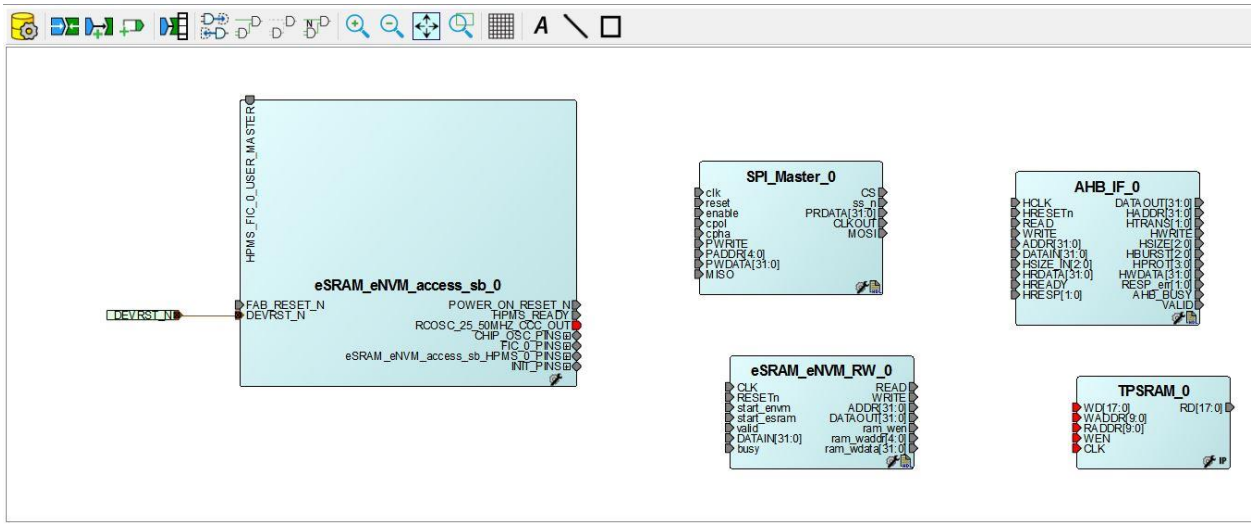
You may need to scroll about in the Catalog window to find it.

You could also use the search feature in the catalog. Type LSRAM to find the core.

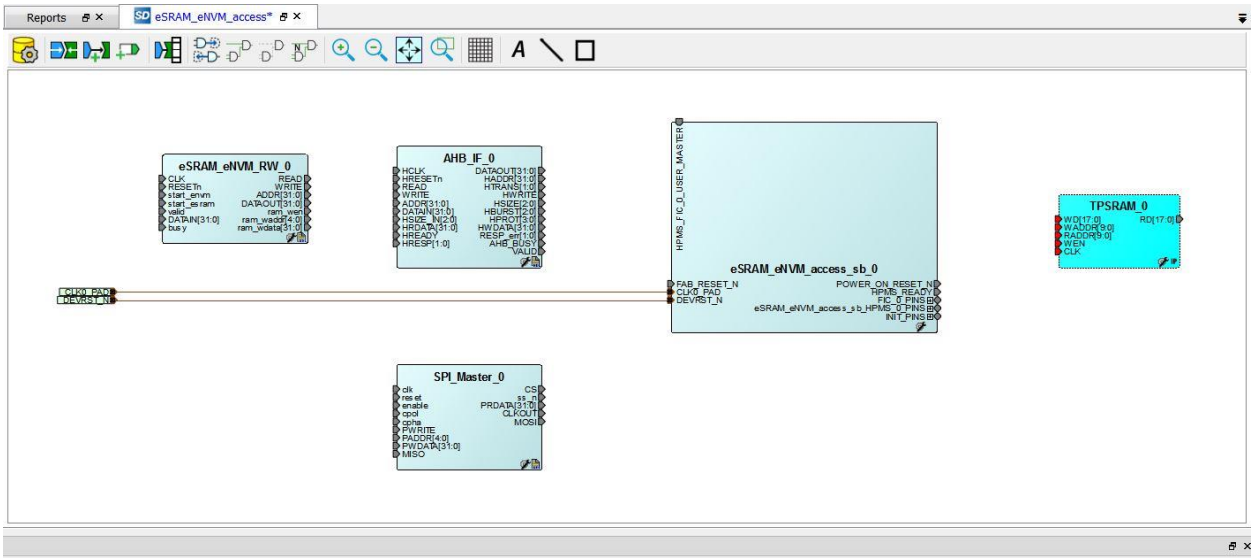
Select and **drag** the **Two-Port Large SRAM** IP into the Smart Design window anywhere.

We will reposition the items in the next step, and then configure the TPSRAM IP.

Your design space should look something like this:



Reposition the elements so it looks like this. (Yeah, there is another Input Port in this picture. It did not magically appear, but we will get to that soon.)

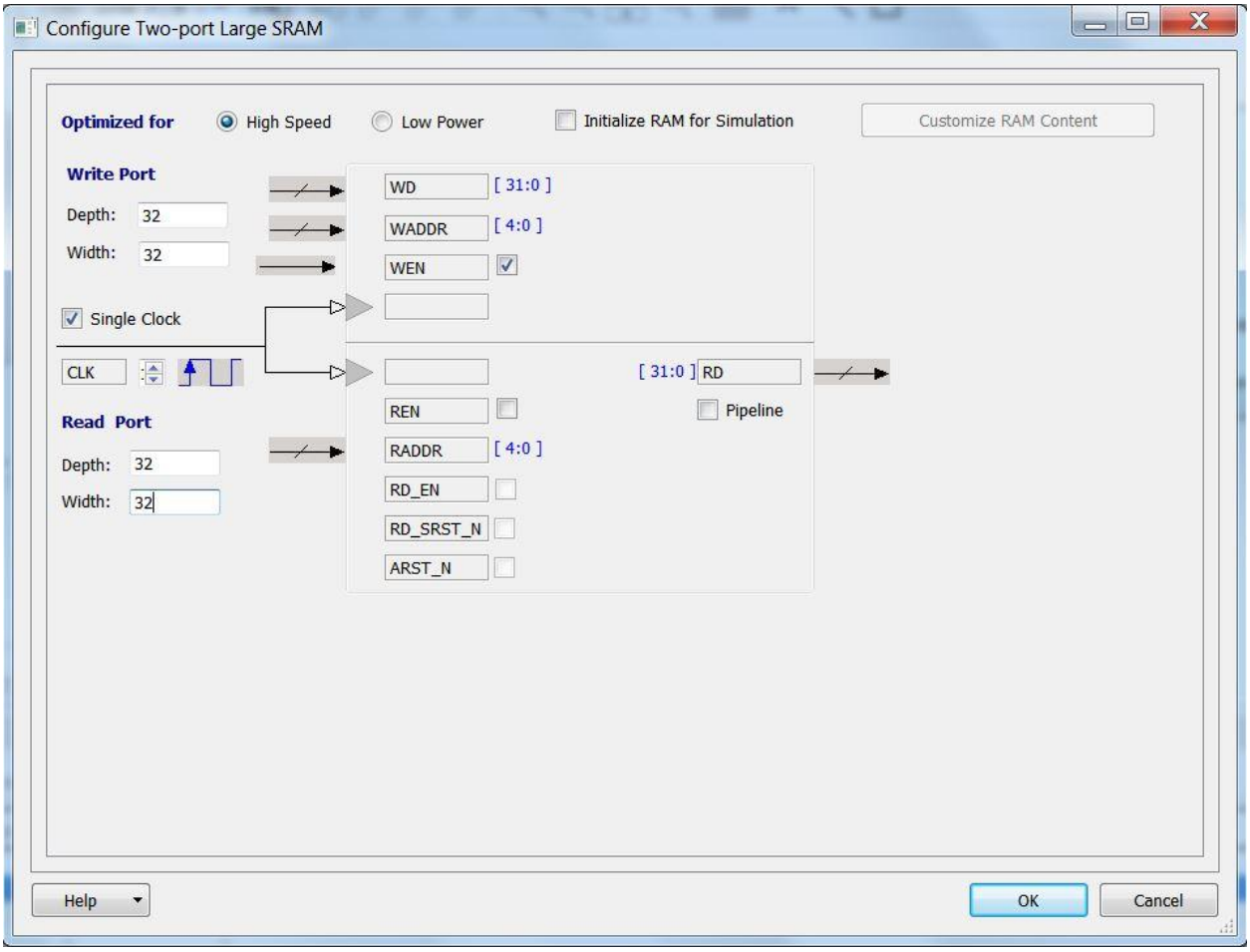


Do you see that blue square button with the 4 arrows pointing N-E-W-S?

If you click it, then the window will re-size to fit all the elements nicely.

Double-click on **TPSRAM_0** so we can configure the IP. (It may take 30 seconds for the wizard to open.)

We are now going to configure the Two-Port SRAM IP for our design.



Change the memory configuration to have **32** in all fours locations as indicated above.

And make sure that you leave the **Single Clock** checked.

(Notice that if you enter a value in to the GUI that results in Libero not being able to build the IP, you will get a warning indicator. It will come and go as you enter or change values.)

When done looking about, *click* **OK**.

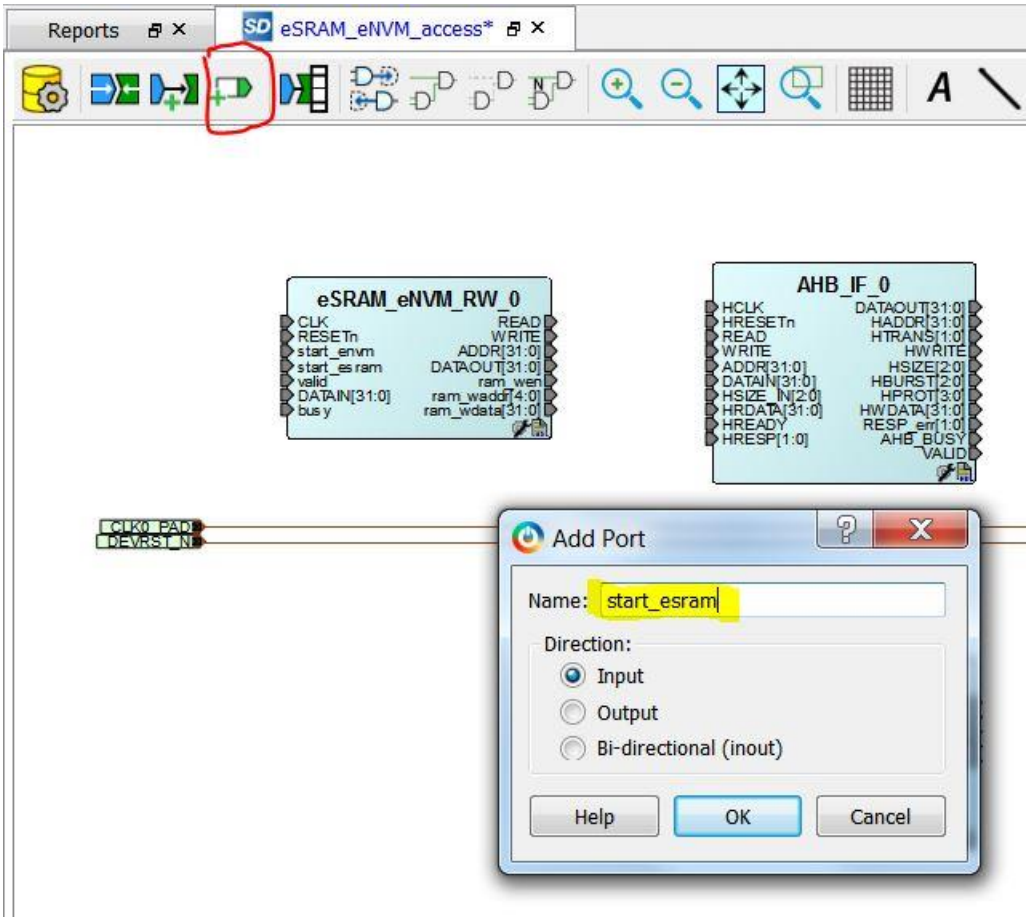
OK, are you ready? Now we are going to add all the IO ports and hook up all the signals!

Zoom in a little bit so we are looking at the area to the left of, and below, the eSRAM_eNVM_RW_0 as shown in the window below.

Some of the buttons in the area I have indicated are *toggle* type buttons.

Press them down to do the function desired, un-press them to stop doing that function.

Press the **Add Port** Button as shown. An overlay window will appear.



Type in the Name: **start_esram**

Make sure that the *direction* is **Input**

Click **OK**

If it is not already just to the left of the **eSRAM_eNVM_RW_0** block, position it there.

It does not need to be exactly in line as shown.

And it will not yet be connected as shown either.

Press the **Connection Mode** Button as shown below.

The cursor will change shape.

Position the cursor over the **dark** end of the **Input port**.

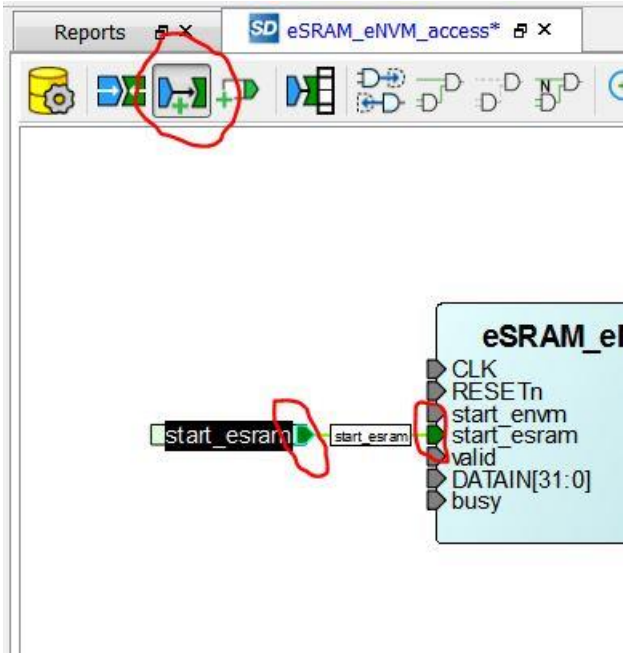
Press down on the left mouse button **AND HOLD IT DOWN**.

Drag the mouse over to the **eSRAM_eNVM_RW_0** block and hover over the **start_esram** input.

The cursor should change to a green "+ with a circle" to indicate it is OK to make the connection.

Release the mouse button. BOOM – you have just connected the two components.

It should now look like the picture below.

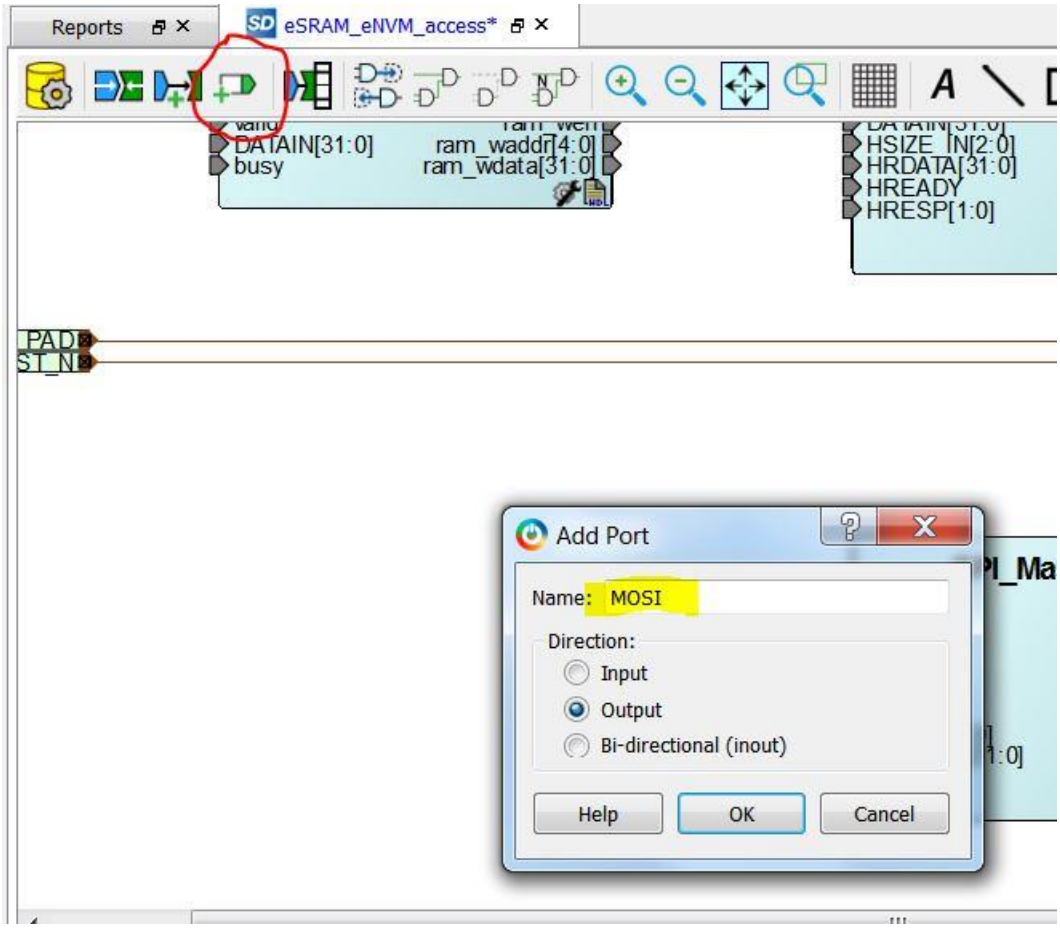


Unselect **Connection Mode** by clicking on it so that it is now **NOT** 'depressed'.

This is easy, only 20 more to go.

Let's do one more, than we will discuss this a little.

Press the **Add Port** Button as shown. An overlay window will appear like before.



Enter the *Name:* **MOSI** – this is an output signal from the SPI interface.

Set the *Direction*

to: **Output**

Click **OK** Where did it go??? Scroll about the window to find it – look way over to the right.

Drag it over to the right of the SPI_MASTER_0 block. Alignment is not critical.

As before,

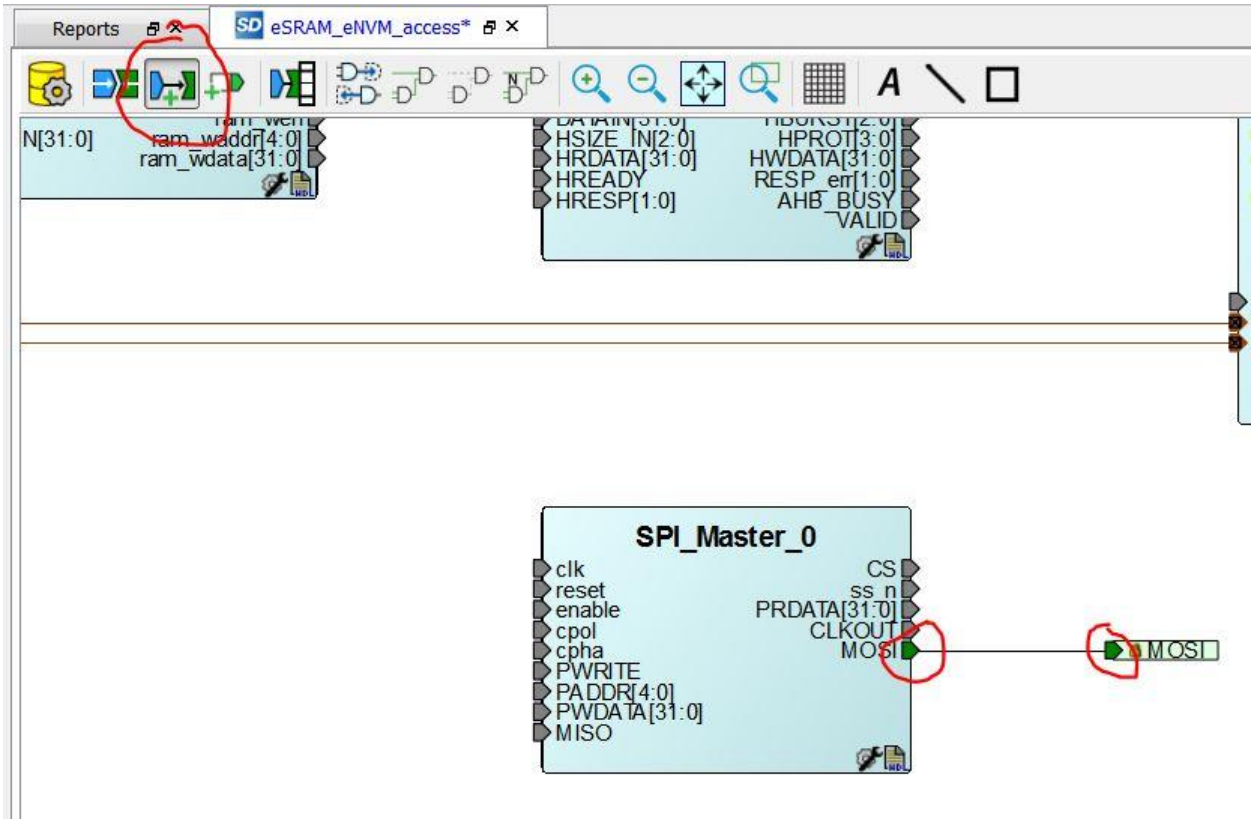
Press the **Connection Mode** Button as shown. The cursor will change shape.

Position the cursor over the **dark** end of the **Output port**.

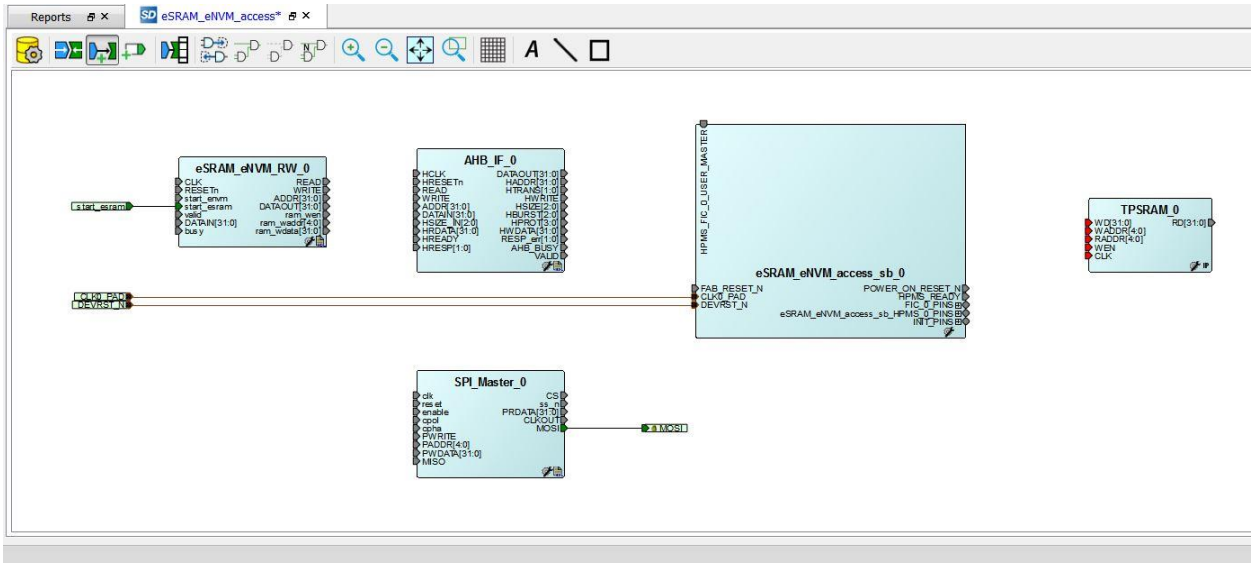
Press down on the left mouse button **AND HOLD IT DOWN**.

Drag the mouse over to the **SPI_MASTER_0** block and hover over the **MOSI** output.

Release the mouse button. It should now look like the picture below.



Your screen should look like this. Maybe hit that **Zoom the design to fit the Screen** button?



Unselect **Connection Mode** by clicking on it so that it is now **NOT** depressed.

Now let's go add all the other IO, then hook up all the ins and outs of each of the blocks.

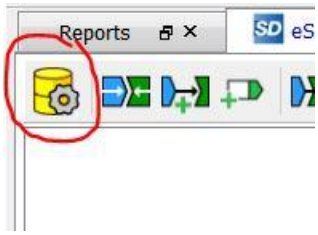
Just kidding.

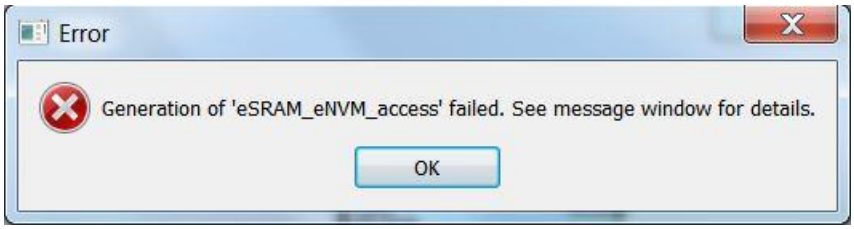
We could connect all the remaining IO in a similar manner.

And we could connect the signal between the different blocks as well.

There is a faster way to make these interconnections, but the purpose of this lab is not to teach you how to draw lines. We have other stuff to do.

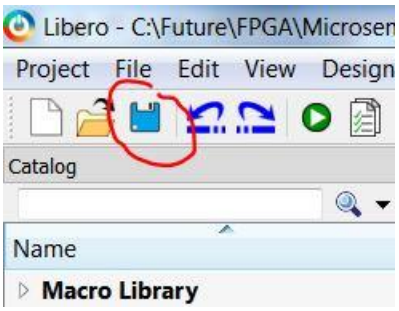
We cannot yet press the Generate Component button because we do not have all the Module's IOs connected. If you do press it, the tool will issue an error message similar to the one below.





If you did the above, *click* **OK** to close the error message.

Click on **Save** in the upper left of the screen to save the project.



Close the **Project -> Close.** Libero might prompt you again if you want to save it, *click* **Yes**

End of **Lab3_part1**

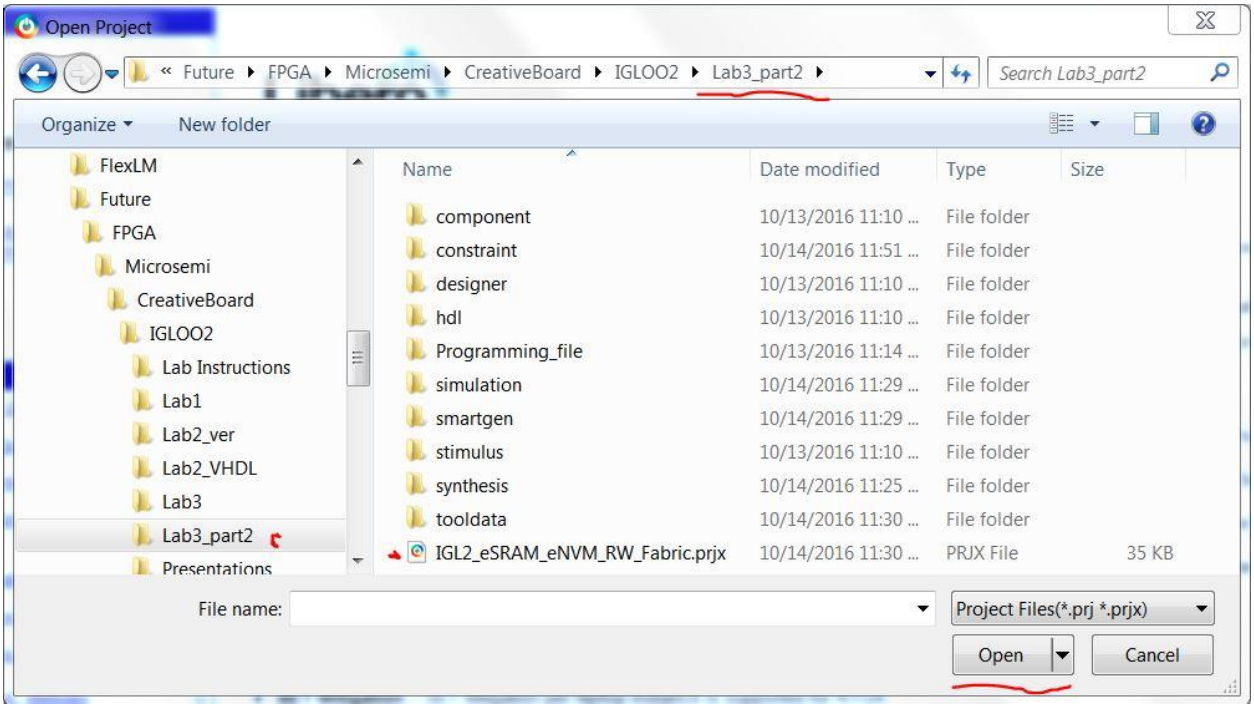
Lab3_part2

Start Libero, if it is not already running.

Open up the completed Lab3_part2 project.

Projects -> Open Project…

Path to the Lab3_part2 folder as shown below:



Double-Click the Project file called **IGL2_eSRAM_eNVM_RW_Fabric.prjx** or *select* it and *Click* **Open**

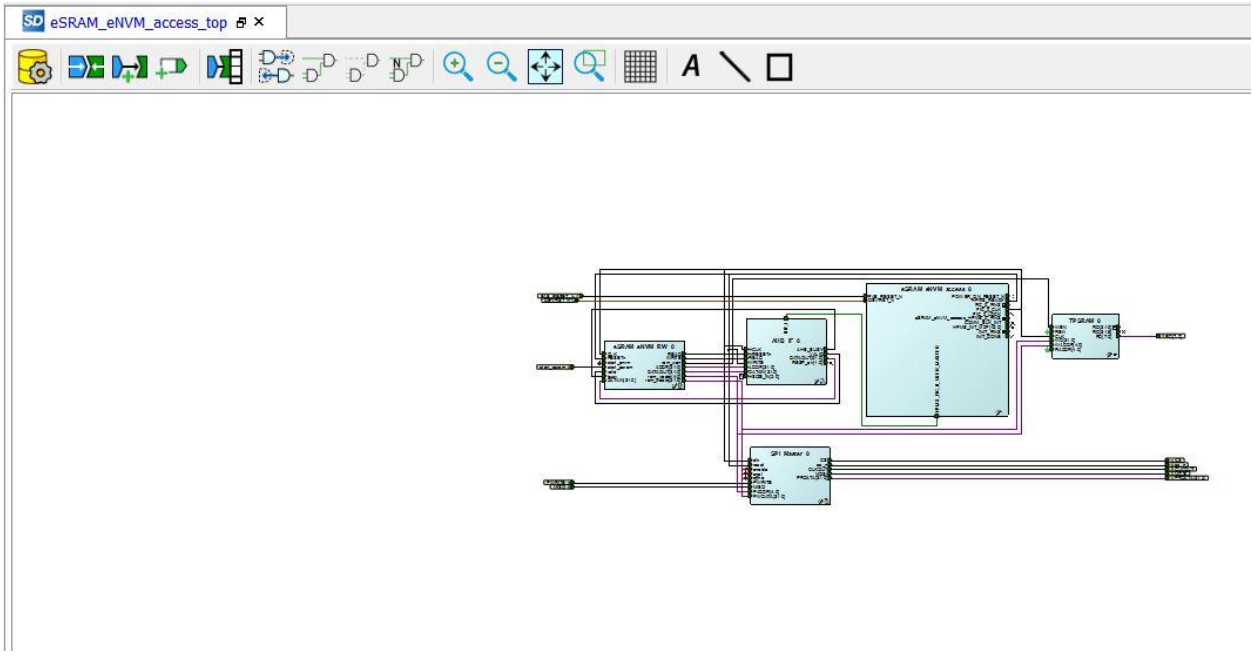
Libero will open the project and a few windows on the right side.

We really only need the **SD eSRAM_eNVM_access_top c**anvas open; you can close all the others.

Close the **Start Window** if it opens.

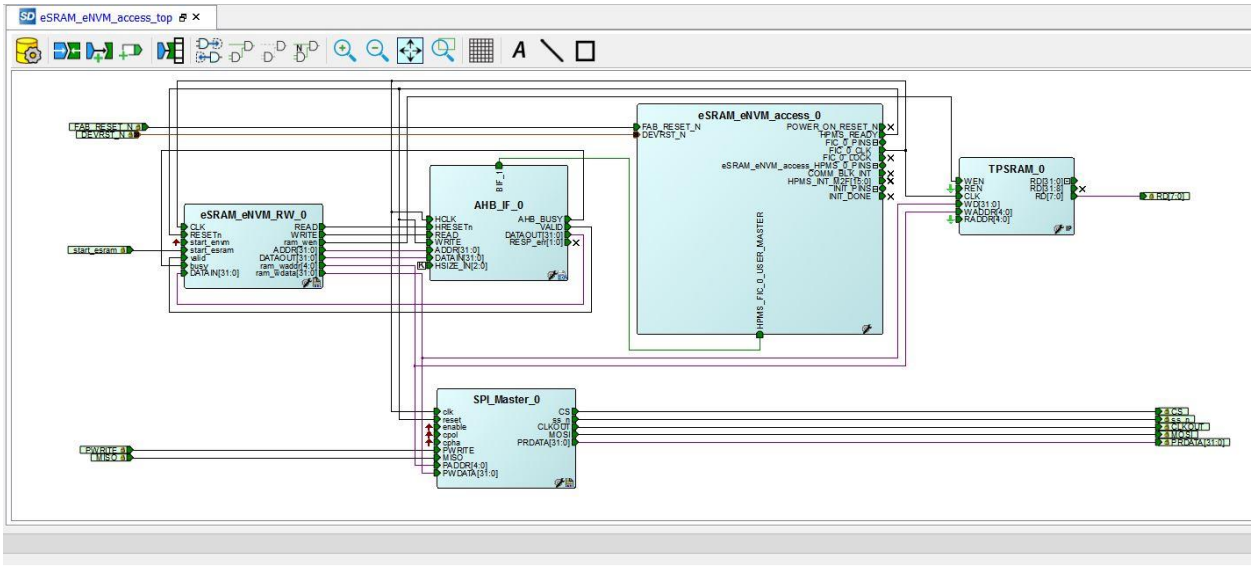
(The info is useful when first learning, but gets in the way once you are familiar with the tools.)

Libero will open in either the **Design Flow** or **Design Hierarchy** Tab on the left, and you should see the **SmartDesign** window on the right. It should look like below: Pretty small.

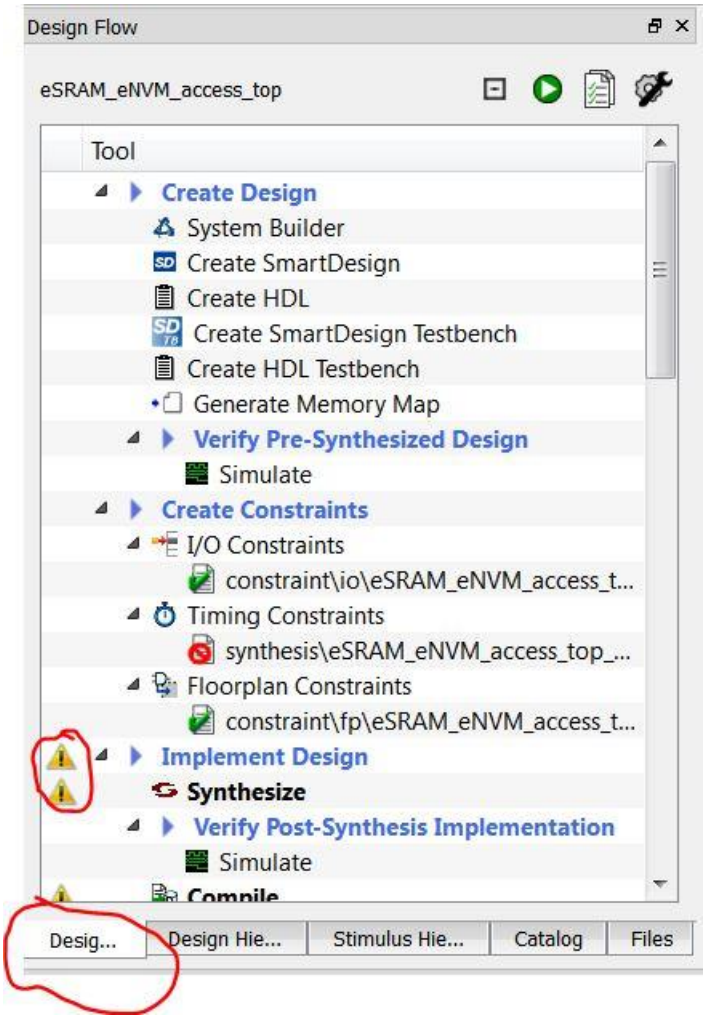


Click on the **Zoom the design to fit in the current window** button.

The window should look like below. If some of the lines look odd, *Click* the **Reroutes all nets** button.



If not already positioned in the **Design Flow** tab, select it as shown below.



If you see **Caution Triangles** then you will need to **ReRun** the design flow.

(If your Laptop went to sleep, unplug and re-plug the USB cord to wake the USB port.)

Scroll down until you can see **Run Program Action**.

Note: If you have Green checks all the way down thru **Generate Bitstream** you can skip this next step.

Right-Click and then *Click* **Clean and Run All** (you've got about 4 minutes – go get a snack.)

If you have all Green Check marks, all you need to do is *Right-Click* and *Click* **Run** to begin Programming.

You will see the overlay advising you that the part is getting programmed (about 55 seconds).

Once programed, *Double-Click* on **SmartDebug Design** (or *Right-Click* and *Click* **Open Interactively**).

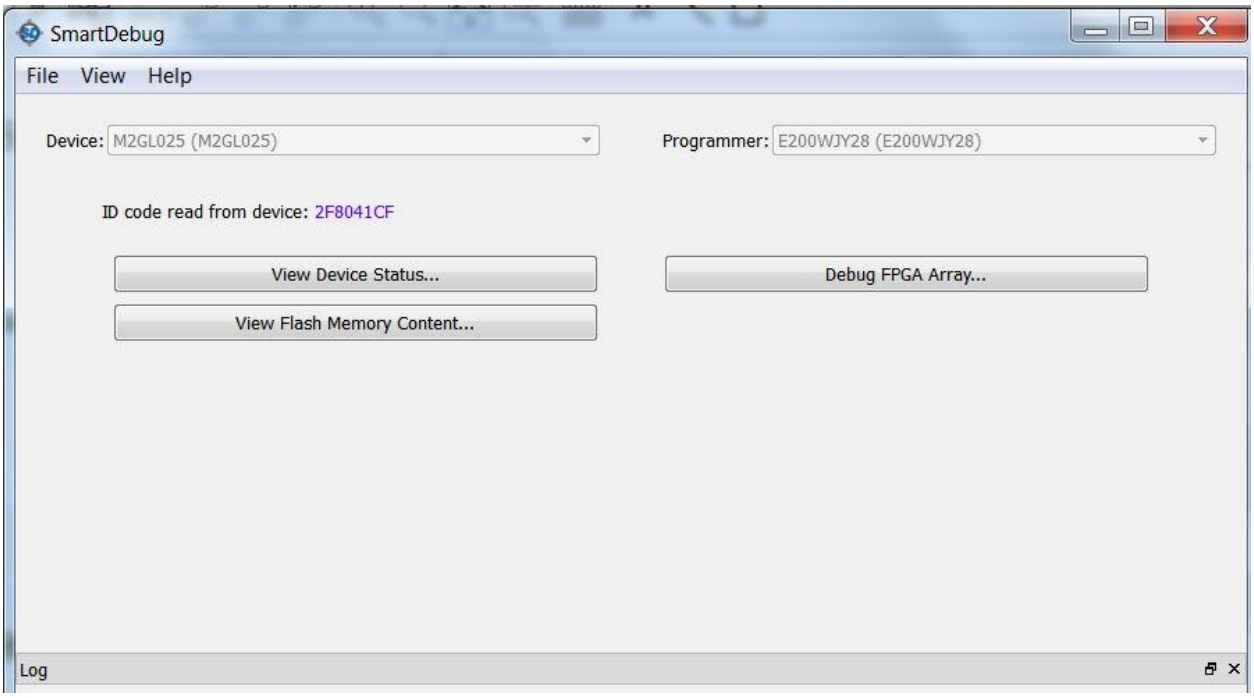
SmartDebug will launch, and after a while the **Splash Microsemi Logo** will appear and go away.

You will know that **SmartDebug** is ready when you see **SD** on the **Task Bar.**

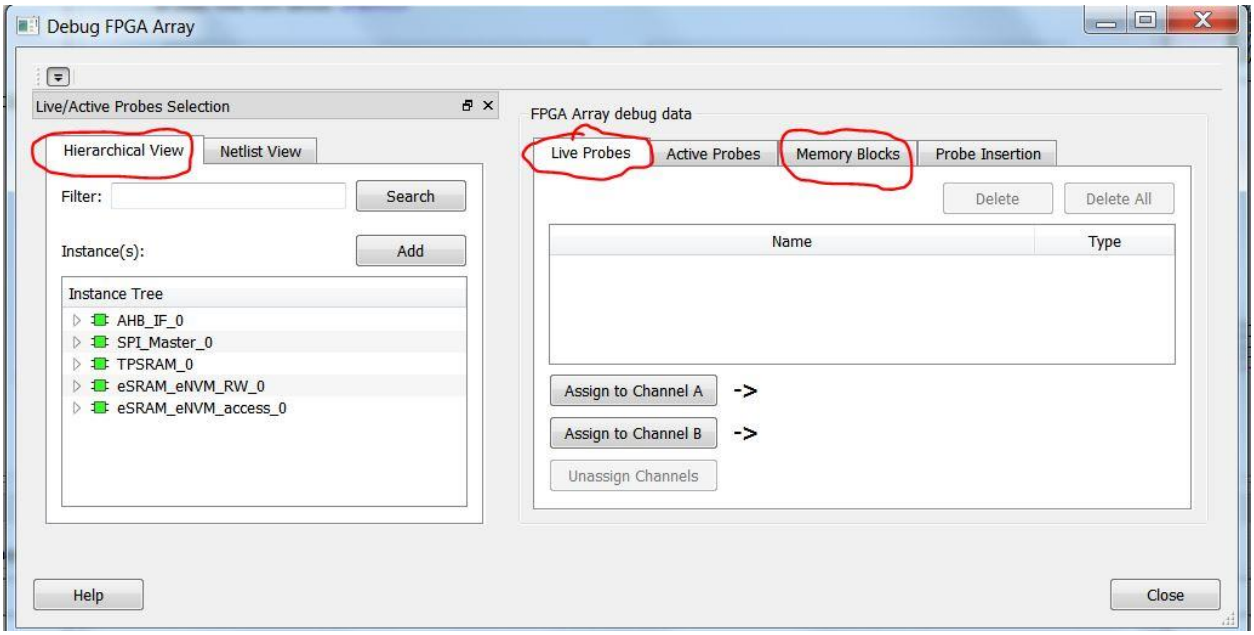
If the **SmartDebug** GUI is not open, *Click* on the **ICON** on the task bar to open it.

SmartDebug

If you completed Lab2 this should look familiar.



Click **Debug FPGA Array…** Shift from the **Live Probes** tab to the **Memory Blocks** tab.

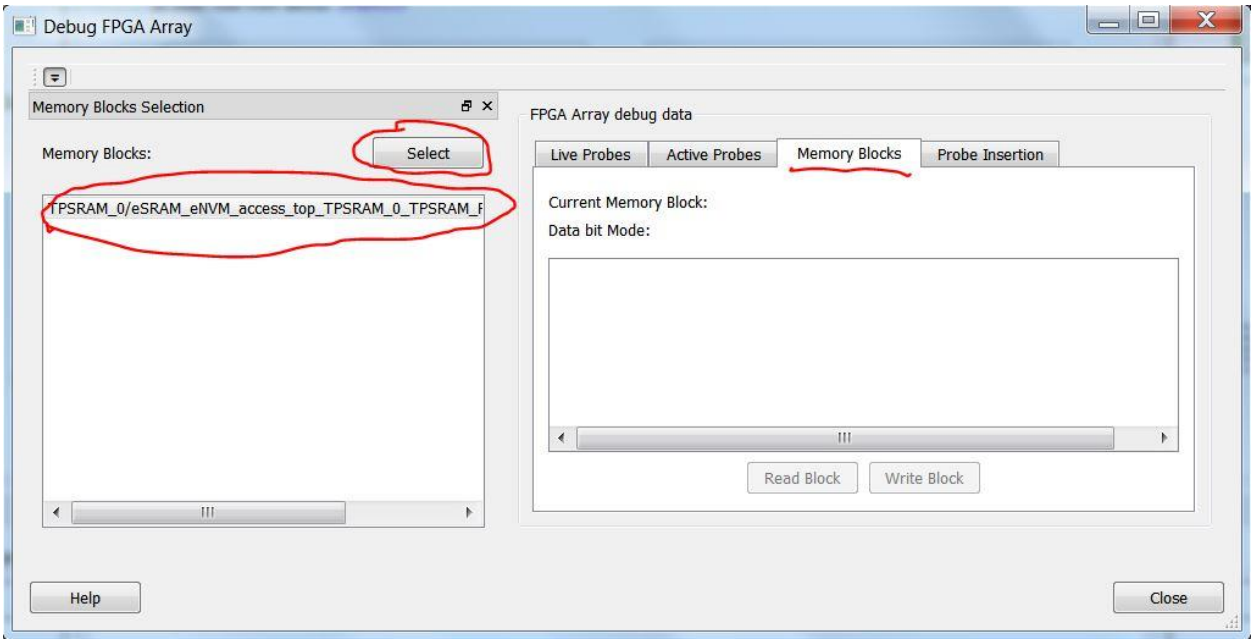


On the Memory Blocks Tab,

Click on the only Memory Block visible in the left window,

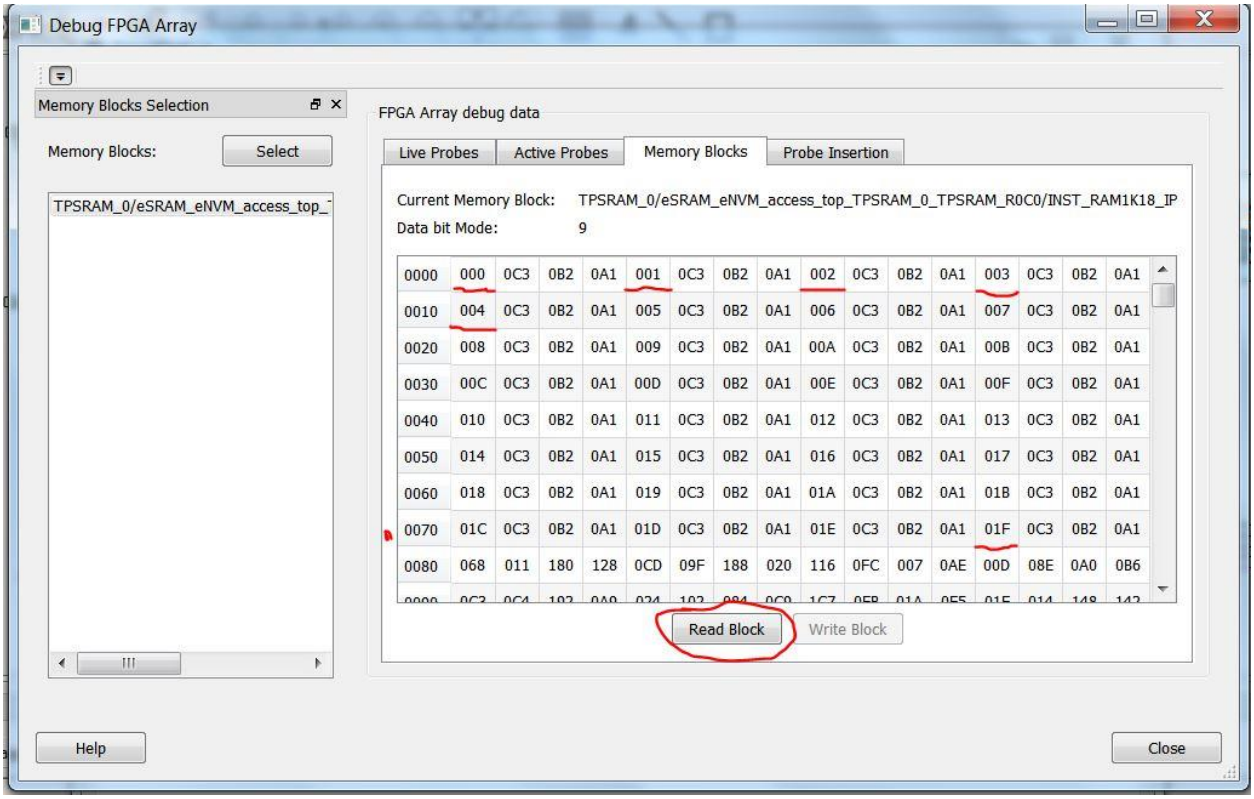
TPSRAM_0/eSRAM_eNVM_access_top_TPSRAM_0_TPSRAM_ etc.

Then *Click* the **Select** button to add it to the **Current Memory Block:** window on the right.



Click the **Read Block** button when it becomes available.

Resize the window so that you can see address row **0080.**



When the design first started, it ran some code and filled an *ascending pattern* into the memory.

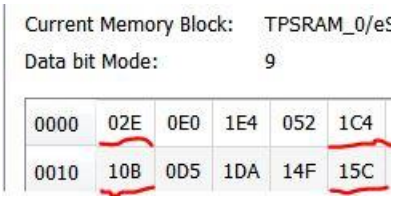
The left most column is the address.

Can you see the pattern? 000 001 002 003 … thru 01F

Press and *Release* the **RESET** button on the board.

This will reset the internal FPGA and the TPSRAM contents will get scrambled.

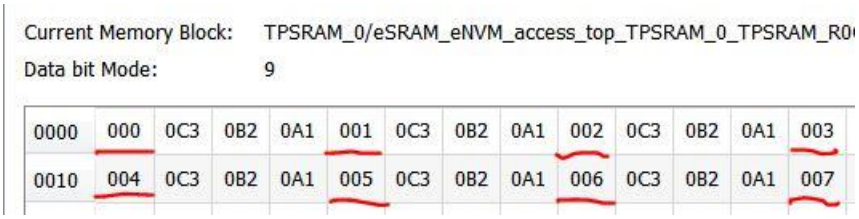
Click the **Read Block** button. This will read the memory again.



(your data may be different, but the 000 001 is now gone)

Press and *Release* the **SW1** button on the board.

The design will fill the TPSRAM with the ascending pattern again.



This can be a useful feature when trying to debug designs.

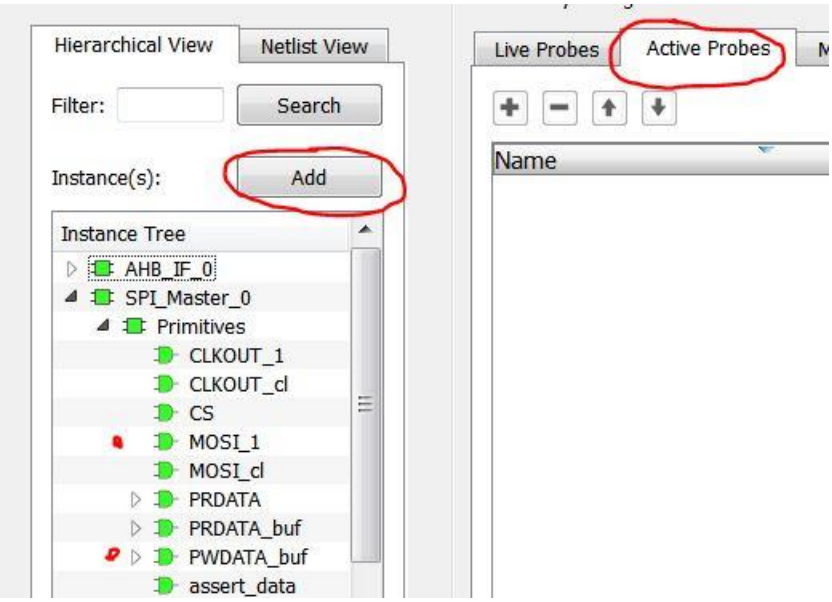
Active Probes

Change from the **Memory Blocks** tab to the **Active Probes** tab.

Expand the **SPI_Master_0** instance until you can see all the Primitive elements.

Click on **MOSI_1** and *Click* on the **Add** button.

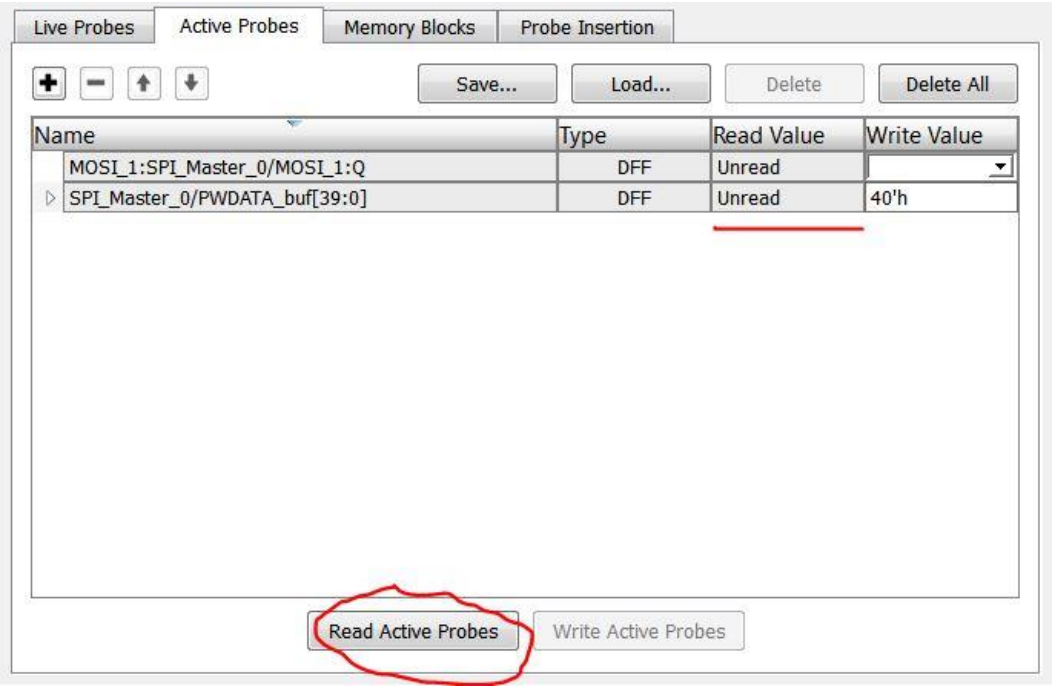
Click on **PWDATA_buf** and *Click* on the **Add** button.



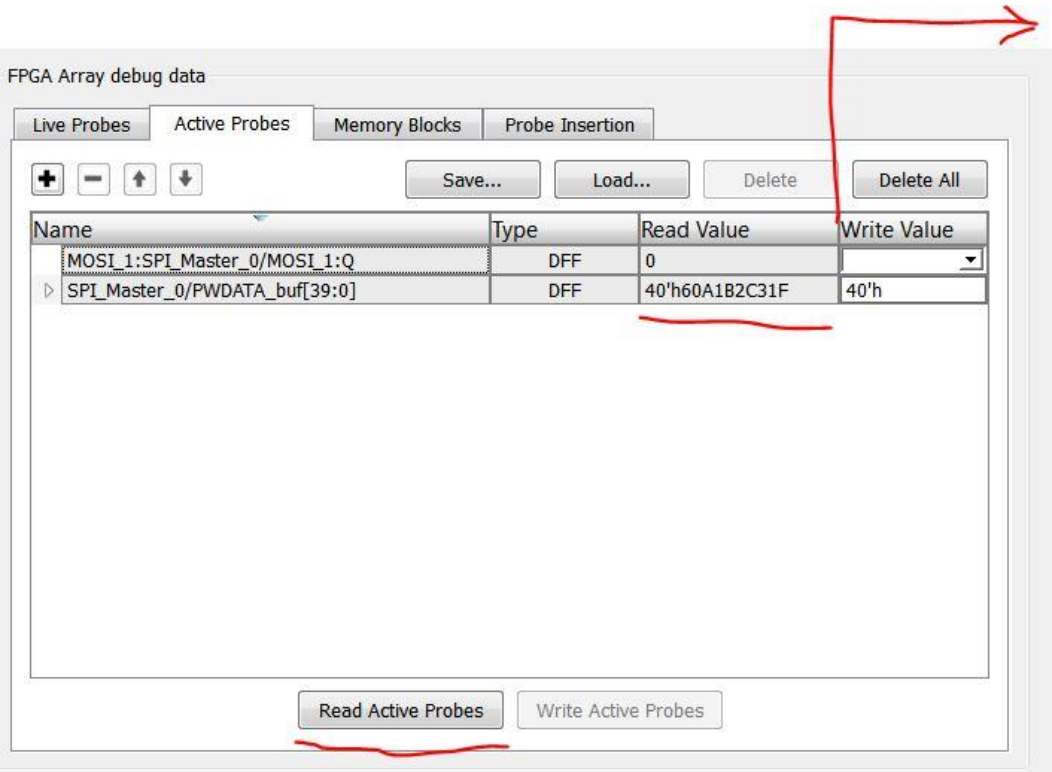
You have just added these two Primitives to the right side window.

The right side window should now look like this.

Notice that the **Read Value** column indicates that you have not read anything yet (Unread).

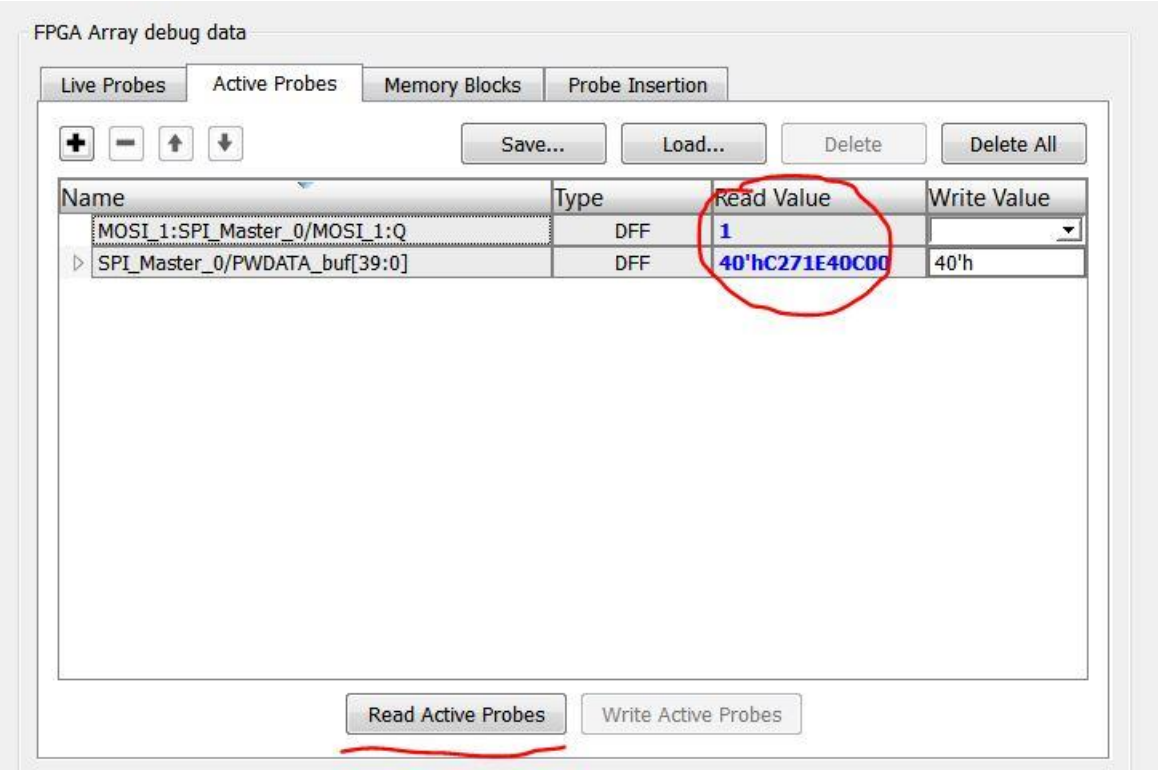


Click on **Read Active Probes**. Adjust the size of **Read Value** so you can see all.



Click on **Read Active Probes** a few times. Did anything change? _____

Hmmm ???



PRESS and **HOLD DOWN SW2** And

Click on **Read Active Probes** a few times. Did anything change now? _____

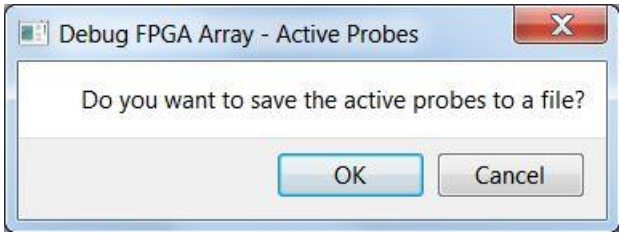
Both the MOSI and PWDATA values have now changed a lot.

Click on **Read Active Probes** repeatedly very fast.

You should be able to see the **MOSI** signal toggling from **1** to **0** to **1**.

When done, close the **Debug FPGA Array** window (lower right button).

If it asks if you want to **save the active probes**; you can if you wish.



The next section is optional if you have time or you can do it later.

Remember in Lab2 we took a look at the internal eNVM (internal Flash Memory)?

Well, this design also does something to that memory and we can look at it with SmartDebug as well.

Go Back into the **SmartDebug** GUI.

Click on **View Flash Memory Content…**

Set **Start Page** to **25**

Set **End Page** to **25**

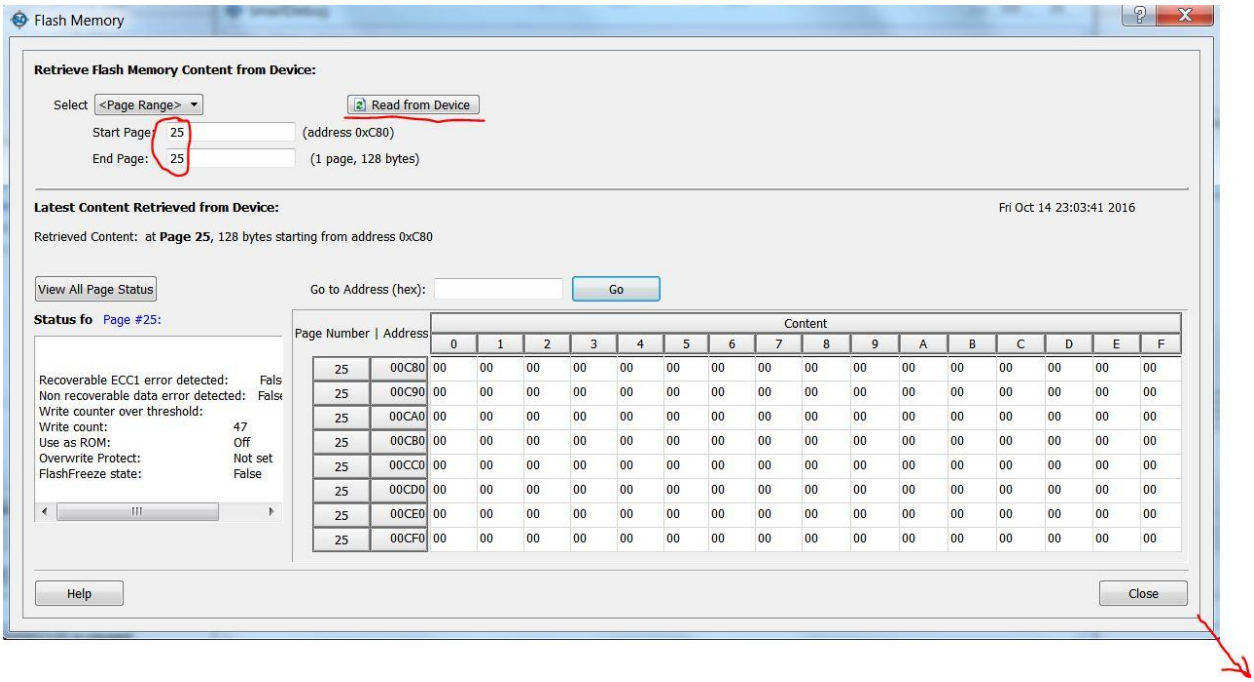
When we added the eNVM to the design, we selected an option to fill it with all **00**s.

Do you remember doing that step?

Click on **Read from Device**

Resize the window *downward* so that you see the entire **Content** Screen.

It should look like below:



Close the **Flash Memory** window and *exit* the **SmartDebug** window.

We can modify the design to load that ascending value into the eNVM instead of the TPSRAM.

You will need to go into the **SmartDesign** GUI and change a few things.

Here is what you need to do…

(And I am **NOT** going to show you a *step by step* on how to do it)

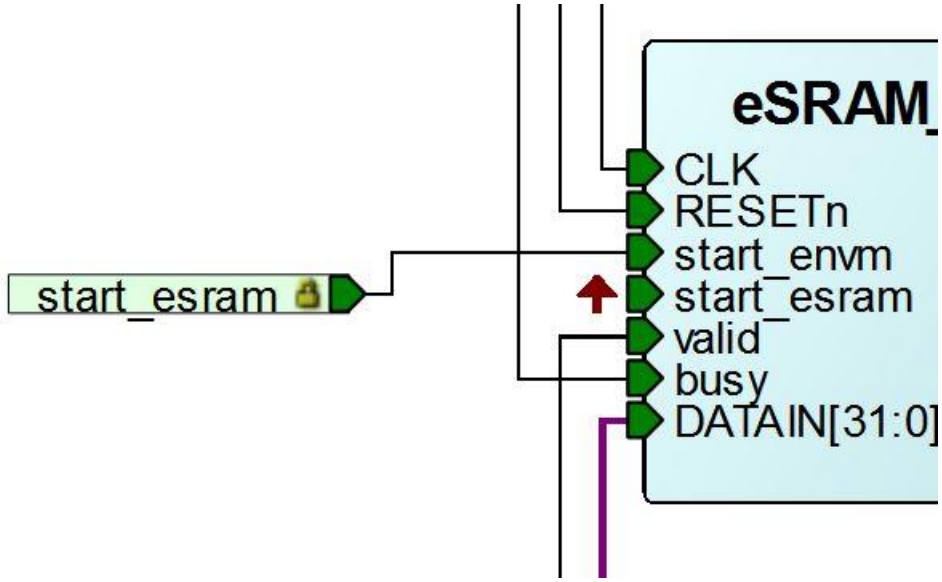
You need to:

- Delete the connection from the s**tart_esram** input module to the block to its right.

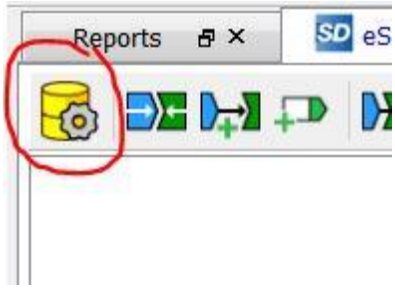
- Delete the Pull-up arrow from the **start_envm** input port on the block (hint: *Right-click*).

- Set the **start_esram** input port on the block to have a Pull-up.

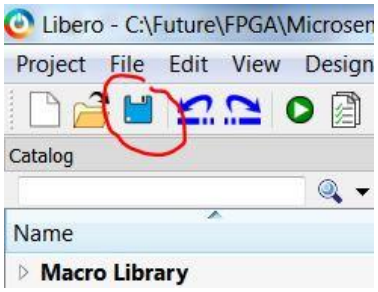
- Connect the net from the s**tart_esram** input module to the block to its right, the **start_envm** port.



When your design looks like above, *Click* the **Generate Component** icon to rebuild the SmartDesign.



then **SAVE** the Project



Re-**Synthesize**, **Compile**, **Place and Route**, **Generate Bitstream**, and **Run Program Action** all in one step.

Right-Click on **Run Program Action** and *Click* on **Clean and Run All**.

(4 minutes run time – start filling out the Survey Form – tell us a little bit about a new project!)

When done, restart **SmartDebug**.

Click on **View Flash Memory Content…**

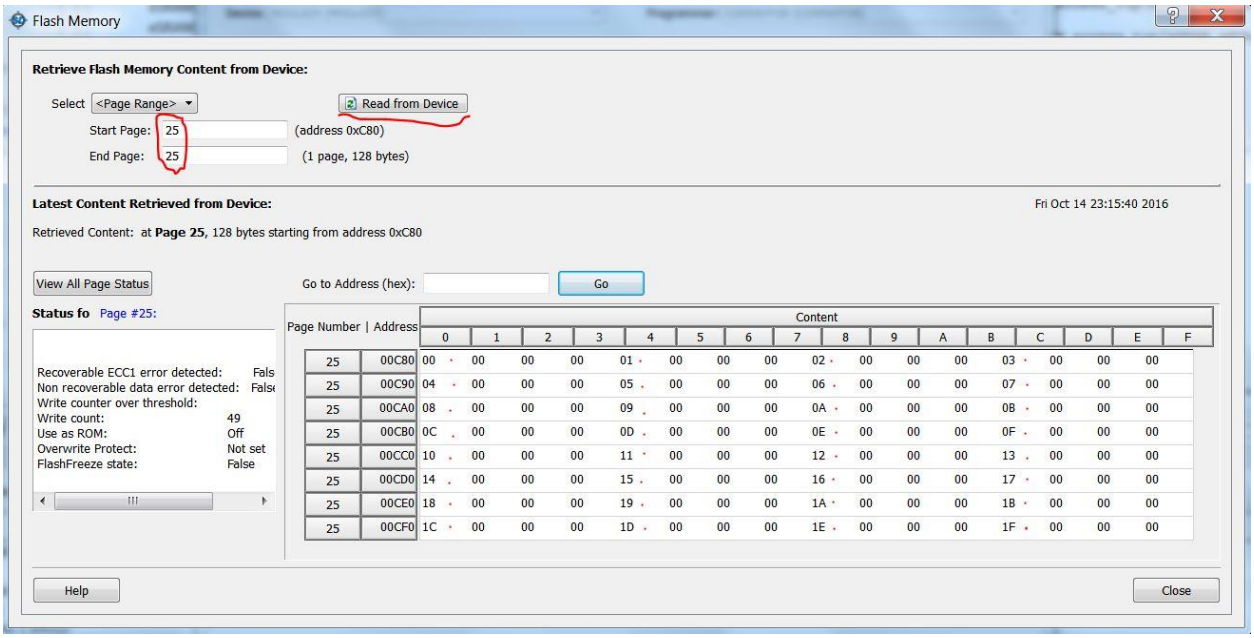
Set **Start Page** to **25**

Set **End Page** to **25**

Click on **Read from Device**

Resize the window *downward* so that you see the entire **Content** Screen. It should look like below.

Now we can see the ascending pattern in columns **0 - 4 - 8 - C**



Here is a closer look.

Page Number	Address	Content															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
25	00C80	00	00	00	00	01	00	00	00	02	00	00	00	03	00	00	00
25	00C90	04	00	00	00	05	00	00	00	06	00	00	00	07	00	00	00
25	00CA0	08	00	00	00	09	00	00	00	0A	00	00	00	0B	00	00	00
25	00CB0	0C	00	00	00	0D	00	00	00	0E	00	00	00	0F	00	00	00
25	00CC0	10	00	00	00	11	00	00	00	12	00	00	00	13	00	00	00
25	00CD0	14	00	00	00	15	00	00	00	16	00	00	00	17	00	00	00
25	00CE0	18	00	00	00	19	00	00	00	1A	00	00	00	1B	00	00	00
25	00CF0	1C	00	00	00	1D	00	00	00	1E	00	00	00	1F	00	00	00

If you downloaded the design into the FPGA, it would get erased back to Zeroes.

Congratulations!! **You are done with Lab3_part2.** Close everything, complete the survey, and go home.

Revision History:

First Release	A	16-Oct-16	Daryl Lee Specter
Corrected typos and added clarity (skipped A1)			
Pages 21, 30, 37, 42	A2	13-Nov-16	Daryl Lee Specter