Exercise Manual

for

The Intel® Quartus® Prime Software: Foundation

Software Requirements to complete all exercises

Intel® Quartus® Prime Pro Edition software version 17.1 with Intel® Arria® 10 device family installed

Link to the Intel Quartus Prime Handbook:

https://www.altera.com/products/design-software/fpga-design/quartus-prime/support.html

Use the link below to download the design files for the exercises:

http://www.altera.com/customertraining/ILT/Quartus_Prime_Foundation_17_1_v1.zip

For Intel Quartus Prime Standard Edition version of these exercises:
http://www.altera.com/customertraining/ILT/P-CSTN-QP-F-15-1-v1.zip

Exercise 1

Exercise 1

***Objectives*:**

- *Create a project using the New Project Wizard*
- *Name the project*
- *Pick a device*

Note: In these exercises, you'll create a brand new project and complete an existing design. You'll have the choice of creating the design using three different types of design entry: Verilog, VHDL, or as a Intel® Quartus® Prime Design Software schematic. Where noted, be sure to only follow the instructions appropriate for your choice of design entry method. By the end of the class, you'll have a final, optimized design, ready for programming into an Intel® Arria® 10 FPGA device.

Be sure to completely read the instructions for each step and sub-step in this lab manual. Each step first summarizes what you'll be doing in that step before providing complete instructions. Use the lines next to each step (____) to keep track of your progress or to check off completed steps in the exercises.

If you have any questions or problems, please ask the instructor for assistance.

Step 1: Create new project for use in the lab exercises

____ 1. Before starting the Intel® Quartus® Prime software, unzip the lab project files:

a. If you are using an Intel® training laptop (either locally or remotely through the Virtual Classroom), go to the **C:\fpga_trn\Quartus_Prime_Software_Foundation** folder in a Windows Explorer window.

b. **Important:** ***delete*** any old lab file folders that may already exist there labeled **QIIF*** or **QPF***.

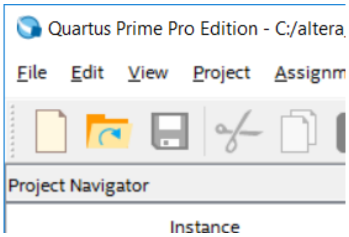
c. Double-click the executable file (**Quartus_Prime_Software_Foundation_17_1_v1.exe**) found in that location. If you cannot find this file, please ask your instructor for assistance.

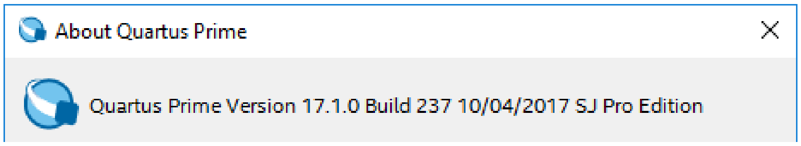
d. In the WinZip dialog box, simply click **Unzip** to automatically extract the files in place to a new folder named **QPF17_1** in the directory mentioned above.

e. Clock **OK** then **Close** to close WinZip.

____ 2. Start the Intel Quartus Prime Pro Edition software, version 17.1, from the Start menu (All Programs → Intel FPGA 17.1 Pro Edition → Quartus (Quartus Prime Pro) 17.1 or from a shortcut on the desktop.

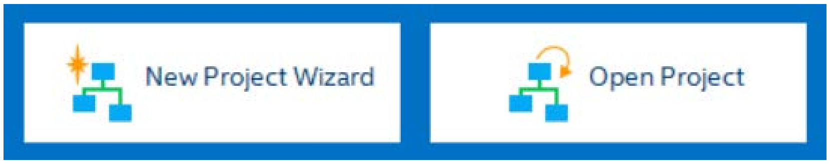
IMPORTANT: *Make sure you are using the* ***Pro Edition*** *and* ***not*** *the* ***Standard Edition****! You can check in the title bar or go to the* ***Help*** *menu and select* ***About Quartus Prime****.*





Quartus Prime Version 17.1.0 Build 237 10/04/2017 SJ Pro Edition

____ 3. Start the New Project Wizard. You can find the New Project Wizard button on the **Home** tab (**Help** menu → **Start Screen**) that appears when you start the Intel Quartus Prime software. If you've closed this tab or it does not appear, you can select **New Project Wizard…** from the **File** menu. Be sure to select **New Project Wizard** and not **New** from the **File** menu, which is used to create a new file, not a project.



The New Project Wizard appears. If the ***Introduction*** *screen appears, click* ***Next****.*

____ 4. Select one of the working directories shown below depending on the type of design entry you want to perform. Once you've made the choice of design entry method, it should be used as your entry method for **all** of the exercises. (The ***<lab_install_directory>*** is **C:\fpga_trn\Quartus_Prime_Software_Foundation**.)

<lab_install_directory>\QPF17_1\VHDL

<lab_install_directory>\QPF17_1\Verilog

<lab_install_directory>\QPF17_1\Schematic

____ 5. Set the **Name** of the project to **pipemult** and leave the top-level entity name as **pipemult.**

____ 6. Click **Next**.

____ 7. On the **Project Type** page, leave the type set to the default: **Empty project**. Click **Next**.

*If you'd like, you can click the **Design Store** link to see project templates available for current devices and new and recent versions of the Intel Quartus software (requires an Internet connection).*

____ 8. On the **Add Files** page, add the top-level design file to the project.

a. Click the browse button .... Navigate to the project directory selected earlier as the **Select File** dialog box may not automatically point there.

b. Select the top-level file **pipemult** (**.v**, **.vhd**, or **.bdf**, depending on the design entry method you chose in #4).

c. Click **Open.**

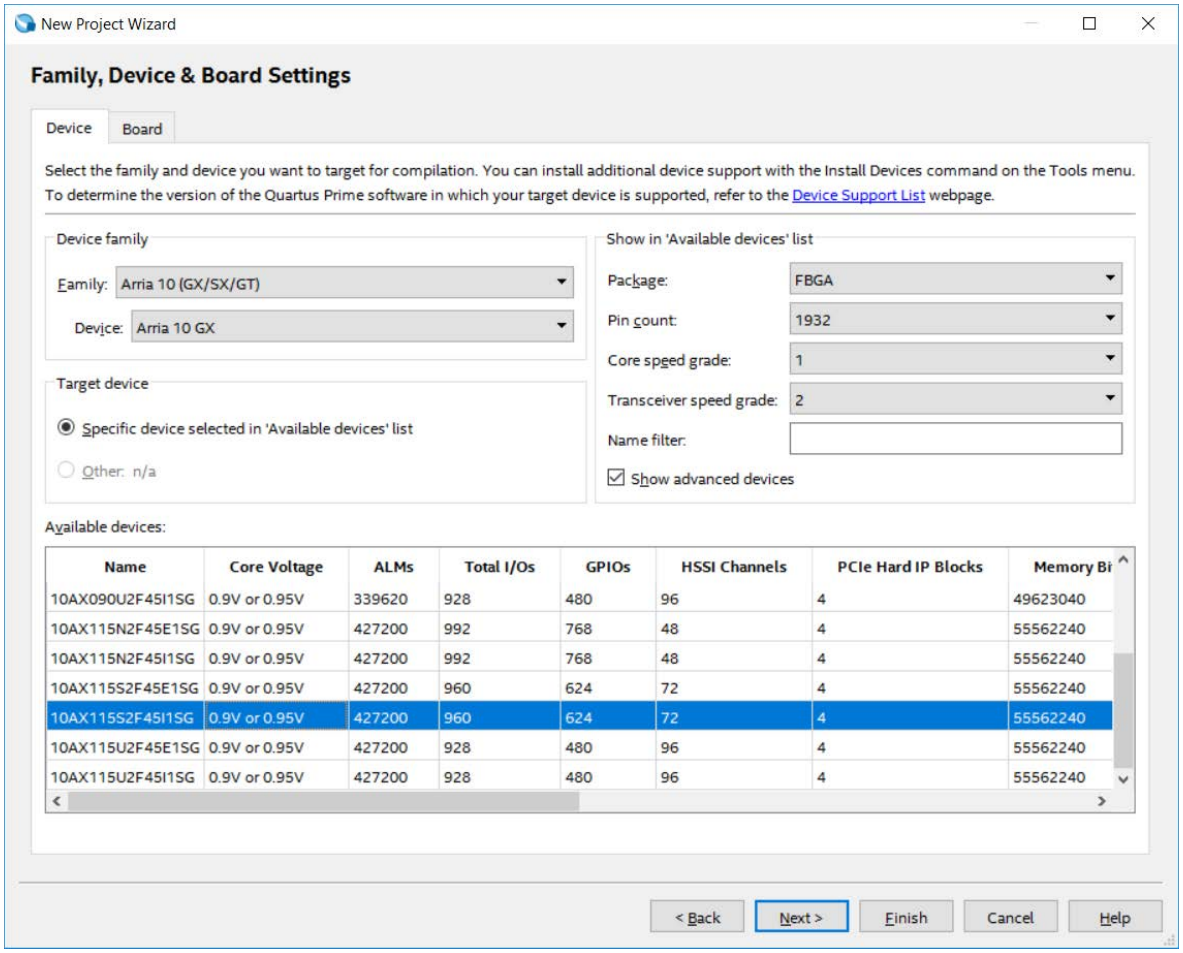
d. Click **Add** to actually add the file to the project *(easy to miss!)*.

e. Click **Next**.

Note that this step isn't really necessary since the design file is already located in the project working directory. The new project would automatically include the design file as part of the project. Files or file directories (libraries) only need to be added in the New Project Wizard if they are not located in the project directory. Adding the file to the project only removes a warning message that indicates the file has not been added.

____ 9. On the **Family & Device Settings** page, **Intel Arria 10 (GX/SX)** is selected as the **Family**. In the **Devices** drop-down, choose the **Intel Arria 10 GX** variation.

____ 10. In the **Show in 'Available devices' list** section, set **Package** to **FBGA**, **Pin count** to **1932**, **Core Speed grade** to **1**, and **Transceiver speed grade** to **2**. This filters the list of available devices. Select the **10AX115S2F45I1SG** device from the **Available devices:** list. Double-check to make sure you select the correct device.



New Project Wizard

Family, Device & Board Settings

Select the family and device you want to target for compilation. You can install additional device support with the Install Devices command on the Tools menu.
To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family

Family: Arria 10 (GX/SX/GT)

Device: Arria 10 GX

Target device

● Specific device selected in 'Available devices' list

○ Other: n/a

Show in 'Available devices' list

Package: FBGA

Pin count: 1932

Core speed grade: 1

Transceiver speed grade: 2

Name filter:

☑ Show advanced devices

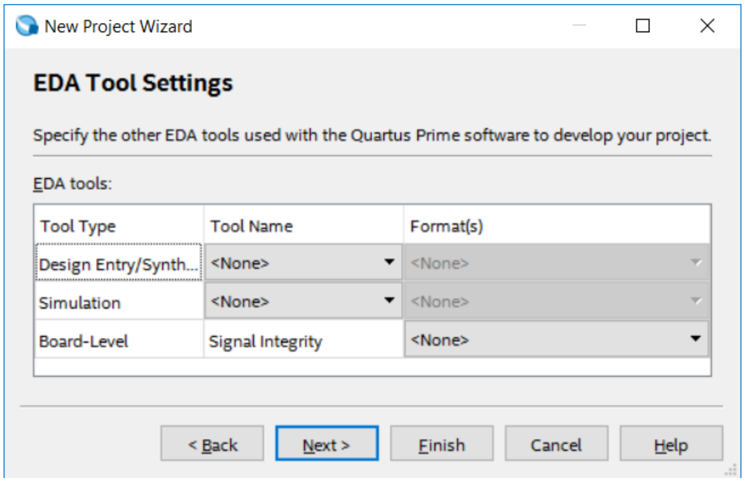
Available devices:

Name	Core Voltage	ALMs	Total I/Os	GPIOs	HSSI Channels	PCIe Hard IP Blocks	Memory Bi
10AX090U2F45I1SG	0.9V or 0.95V	339620	928	480	96	4	49623040
10AX115N2F45E1SG	0.9V or 0.95V	427200	992	768	48	4	55562240
10AX115N2F45I1SG	0.9V or 0.95V	427200	992	768	48	4	55562240
10AX115S2F45E1SG	0.9V or 0.95V	427200	960	624	72	4	55562240
10AX115S2F45I1SG	0.9V or 0.95V	427200	960	624	72	4	55562240
10AX115U2F45E1SG	0.9V or 0.95V	427200	928	480	96	4	55562240
10AX115U2F45I1SG	0.9V or 0.95V	427200	928	480	96	4	55562240

The device you've selected is the one used on Intel's Arria 10 GX FPGA Development Kit (https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-a10-gx-fpga.html).

____ 11. Click **Next**.

____ 12. On the **EDA Tool Settings** page, you can specify third-party EDA tools you may want to use along with the Intel Quartus Prime software for different functions such as synthesis, or simulation. Since these exercises will be done entirely within the Intel Quartus Prime software without any other tools, set all the selections in the **Tool Name** column to **<None>** to save some compile time later. Click **Next**.

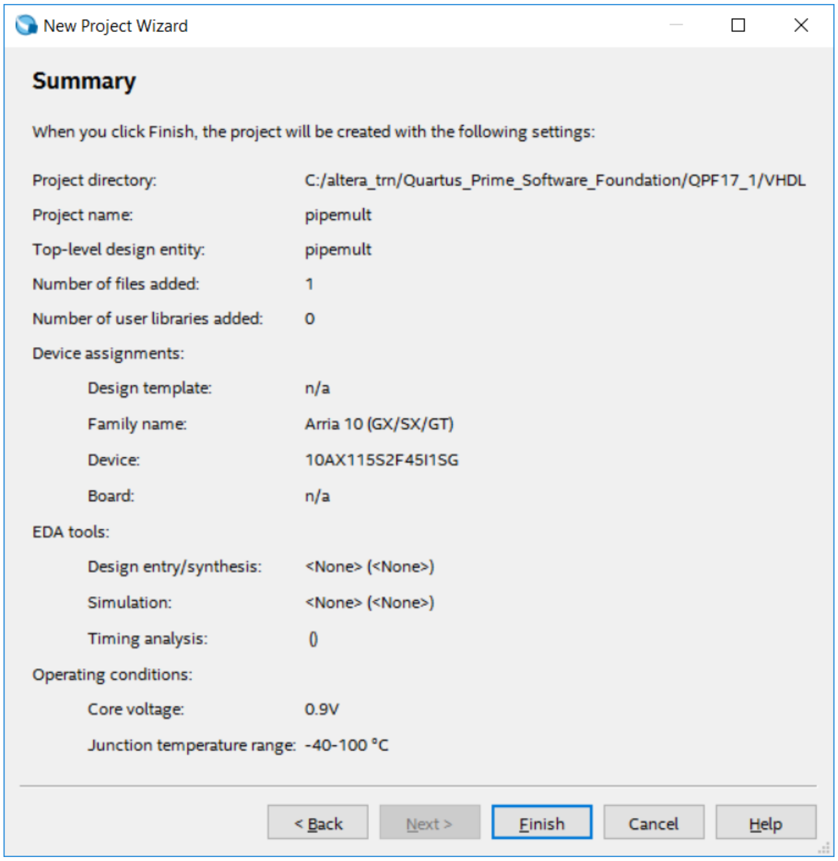


Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)
Design Entry/Synth...	<None>	<None>
Simulation	<None>	<None>
Board-Level	Signal Integrity	<None>

The summary screen appears.



When you click Finish, the project will be created with the following settings:

Project directory:	C:/altera_trn/Quartus_Prime_Software_Foundation/QPF17_1/VHDL
Project name:	pipemult
Top-level design entity:	pipemult
Number of files added:	1
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Arria 10 (GX/SX/GT)
Device:	10AX115S2F45I1SG
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	<None> (<None>)
Timing analysis:	0
Operating conditions:	
Core voltage:	0.9V
Junction temperature range:	-40-100 °C

____ 13. Click **Finish**.

The project is now created.

Keep the project open as you continue through the rest of the exercises. There is no need to close the project. If you do close the project for some reason, be sure to select

> ***Open Project*** *instead of just* ***Open*** *from the* ***File*** *menu. The* ***Open*** *command is used to simply open a single file instead of a project, preventing the ability to perform many project-based operations, such as compilation.*

Exercise Summary

- Created a project using the New Project Wizard
  - *Named the project*
  - *Picked a device*

END OF EXERCISE 1

Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Exercise 2

Exercise 2

<u>Objectives</u>:

- *Create a multiplier and RAM block using IP from the IP Catalog to complete the design*
- *Create a **.hex** file to initialize the RAM block using the Memory Editor*
- *Analyze and elaborate the design to check for errors*

Pipelined Multiplier Design

Figure 1 shows a schematic representation of the top-level design file you will be using today. It consists of a multiplier and a RAM block. Data is fed to the multiplier from an external source and stored in the RAM block, which is also controlled externally. The data is then read out of the RAM block by a separate address control.

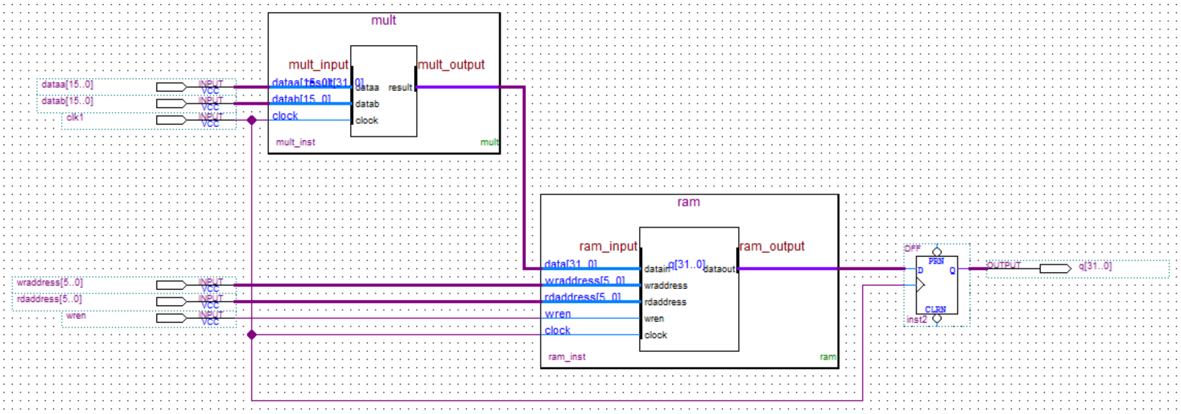


Figure 1

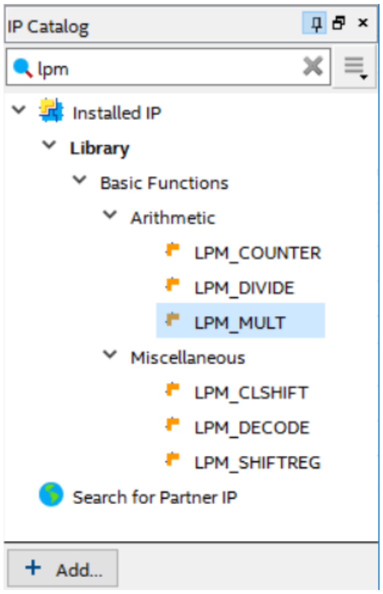
IMPORTANT NOTE: *For exercises 2-5, you should continue working from the files of the previous exercise. The **Solutions** directory contains a Word document with the answers to questions asked in the exercises as well as an archive of the final project as it would be set up at the end of exercise 5.*

Step 1: Build an 8x8 multiplier using the IP Catalog

____ 1. If the **IP Catalog** is not visible, select it from the **Tools** menu, the toolbar , or from the **View** menu → **IP catalog**).

____ 2. Type **lpm** into the search field of the **IP Catalog**.

This filters the list of installed IP to only those that are part of the Library of Parameterized Modules (LPM) discussed in the presentation. These are basic logic functions that can be used in any design on any device.



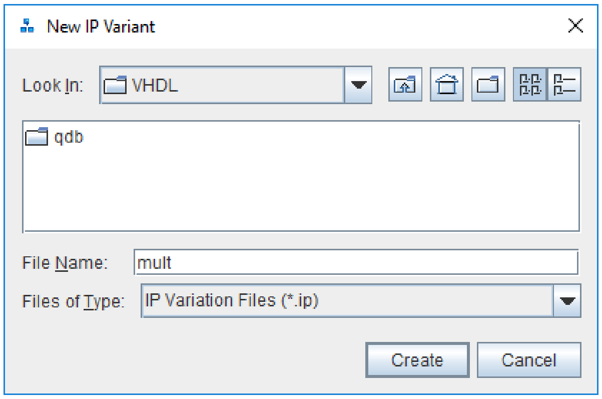
____ 3. Right-click the IP named **LPM_MULT** and hover over the **Details** submenu.

Here you can get some basic information about the selected IP, as well as where the files for the IP are located. Sometimes, datasheets or web links for more information can be found here. In this case, the IP is part of the Intel Quartus Prime Pro Edition installation, so its files are located in the Intel Quartus Prime installation directory.

____ 4. Create an instance of the **LPM_MULT** IP by either double-clicking it in the IP Catalog or clicking it once to select it and then click **Add**. You can also do this by right-clicking the IP and selecting **Add version 17.1**.

The IP Parameter Editor opens.

____ 5. In the **New IP Variant** dialog that appears, set the **File Name** to **mult**. The **Save in folder** field should match the project directory.



New IP Variant

Look In: VHDL

qdb

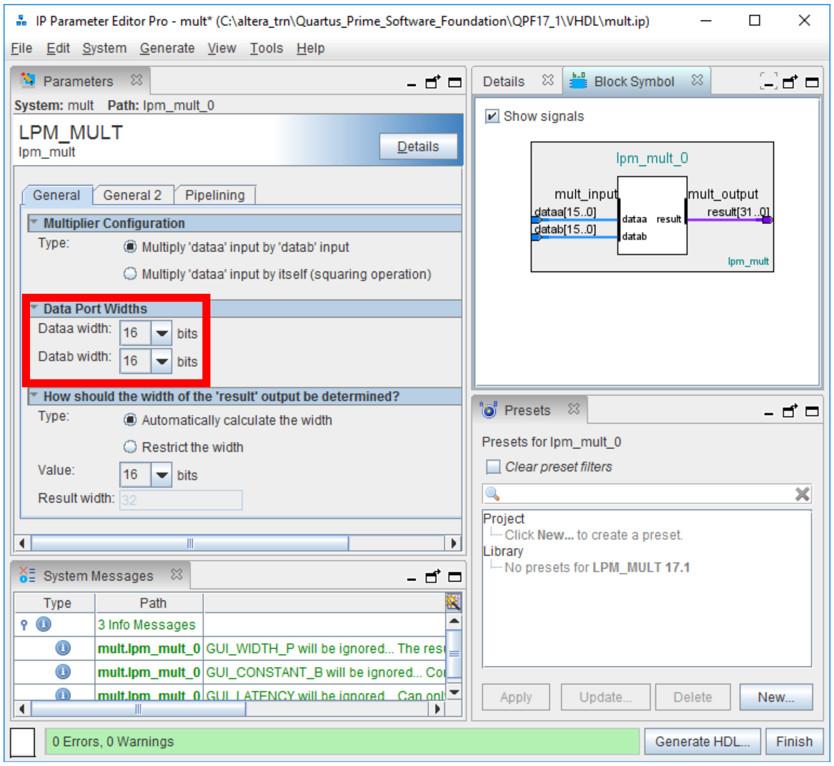
File Name: mult

Files of Type: IP Variation Files (*.ip)

All IP is parameterized in the Pro Edition using the Platform Designer-based IP Parameter Editor.

____ 6. On the **General** tab of the IP Parameter Editor, set the **Dataa width** and **Datab width** to **16** bits if they are not already set. For the remaining settings on this tab, use the default settings.

As you make changes to the parameter settings, notice the changes in the ***Block Symbol*** *tab of the parameter editor.*



IP Parameter Editor Pro - mult* (C:\altera_trn\Quartus_Prime_Software_Foundation\QPF17_1\VHDL\mult.ip)

System: mult Path: lpm_mult_0

LPM_MULT
lpm_mult

Multiplier Configuration

Type: ● Multiply 'dataa' input by 'datab' input
○ Multiply 'dataa' input by itself (squaring operation)

Data Port Widths

Dataa width: 16 bits

Datab width: 16 bits

How should the width of the 'result' output be determined?

Type: ● Automatically calculate the width
○ Restrict the width

Value: 16 bits

Result width: 32

☑ Show signals

Presets for lpm_mult_0

☐ Clear preset filters

Project
└ Click New... to create a preset.
Library
└ No presets for LPM_MULT 17.1

Type	Path	
ⓘ	3 Info Messages	
ⓘ	mult.lpm_mult_0	GUI_WIDTH_P will be ignored... The resu
ⓘ	mult.lpm_mult_0	GUI_CONSTANT_B will be ignored... Co
ⓘ	mult.lpm_mult_0	GUI_LATENCY will be ignored... Can only

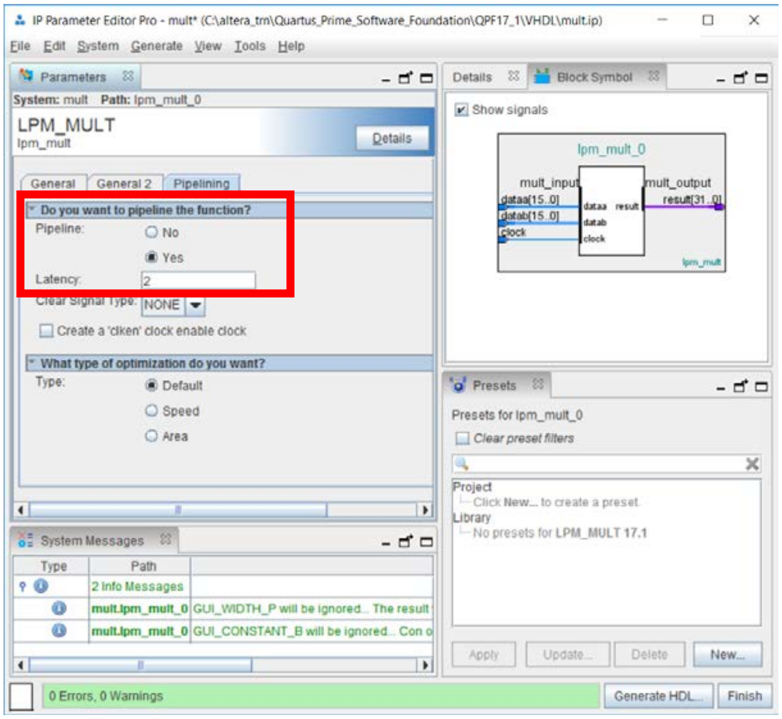
0 Errors, 0 Warnings

____ 7. Switch to the **General 2** tab.

____ 8. On this tab, keep all of the default settings (i.e. **datab** input does <u>NOT</u> have a constant value, use unsigned multiplication, and select the default multiplier implementation).

____ 9. Switch to the **Pipelining** tab.

____ 10. On this tab, set **Pipeline** to **Yes** and the **Latency** to **2**. Leave all other settings at their defaults.



IP Parameter Editor Pro - mult* (C:\altera_trn\Quartus_Prime_Software_Foundation\QPF17_1\VHDL\mult.ip)

System: mult Path: lpm_mult_0

LPM_MULT
lpm_mult

General | General 2 | Pipelining

Do you want to pipeline the function?

Pipeline: ○ No ● Yes

Latency: 2

Clear Signal Type: NONE

□ Create a 'clken' clock enable clock

What type of optimization do you want?

Type: ● Default ○ Speed ○ Area

☑ Show signals

Presets for lpm_mult_0

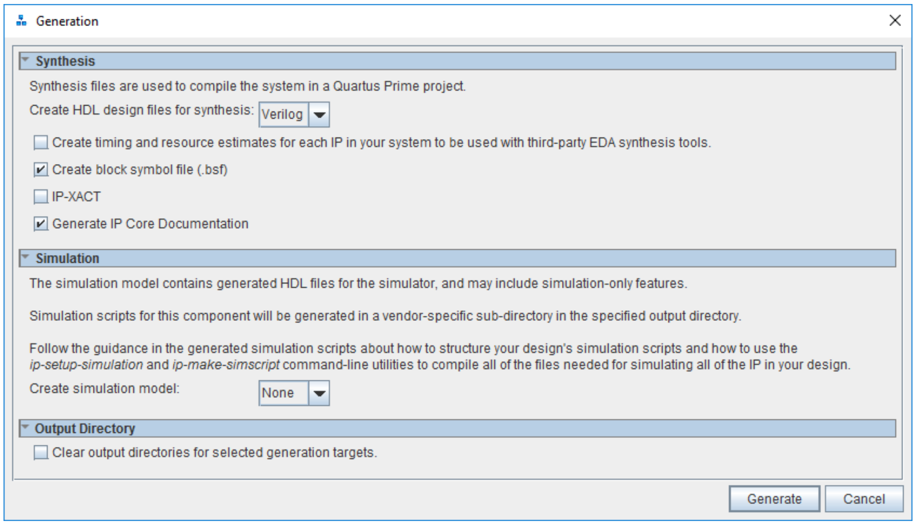
□ Clear preset filters

Project
└ Click New... to create a preset.
Library
└ No presets for LPM_MULT 17.1

Type	Path	
	2 Info Messages	
	mult.lpm_mult_0	GUI_WIDTH_P will be ignored... The result
	mult.lpm_mult_0	GUI_CONSTANT_B will be ignored... Con o

0 Errors, 0 Warnings

____ 11. Click **Generate HDL** in the lower-right corner of the parameter editor, or select **Generate HDL** from the **Generate** menu.



Generation

Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: Verilog

□ Create timing and resource estimates for each IP in your system to be used with third-party EDA synthesis tools.

☑ Create block symbol file (.bsf)

□ IP-XACT

☑ Generate IP Core Documentation

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

Output Directory

□ Clear output directories for selected generation targets.

*The **Generation** dialog box appears. In this dialog, you select options for what files should be generated by the IP Parameter Editor. An **.ip** file, the main file type used by the Platform Designer system design tool, is always created and stored in the **Output Directory Path**. Other files for synthesis and/or simulation of the IP are generated based on the other options selected.*

____ 12. In the **Synthesis** section, select the HDL file type based on your preferred HDL language selection from Exercise 1. If you chose to create a schematic design, select either **VHDL** or **Verilog**.

____ 13. If you chose to create a schematic design, turn on the option **Create block symbol file (.bsf)** if it's not already enabled. If you chose to create an HDL-based design, you can just ignore this option.

____ 14. Make sure the **Create simulation model:** option in the **Simulation** section is set to **None.**

While creating simulation model files for the multiplier would not cause any issues, we will not be simulating the design and we want to simplify the project directory, so we're not going to generate simulation files.

____ 15. Click **Generate**. Click **Yes** if asked to save the settings.

____ 16. Click **Close** when generation completes. If there were any errors, go back to the IP Parameter Editor and fix the parameters.

____ 17. Click **Finish** to close the IP Parameter Editor.

*An information dialog appears explaining how to edit the IP's parameters by opening its **.ip** file using Platform Designer.*

____ 18. Click **Close**.

*The multiplier IP has been created. If you look in the project directory, you'll see the generated file, **.ip**, and directory (**mult** containing **.qip**, **.bsf**, and others) discussed in the presentation.*

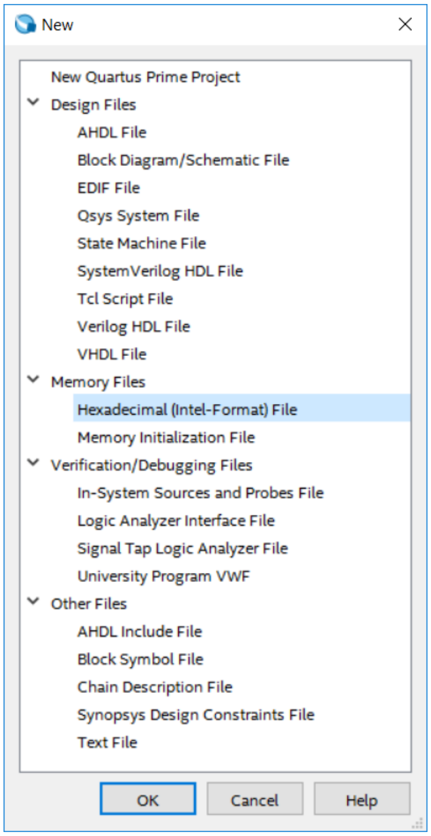
*If for some reason you later find that the parameter settings for the IP core you created are incorrect or you did not generate the correct output files, select **IP Components** in the pop-up list in the Project Navigator, and double-click the IP core you would like to edit. Check the parameters on all the tabs of the IP Parameter Editor to fix any errors and regenerate the IP.*

Step 2: Create .hex file using the Memory Editor

____ 1. From the **Tasks** window under **Project Files**, click **New**. You can also go to the **File** menu and select **New** or click 🗋 in the toolbar.

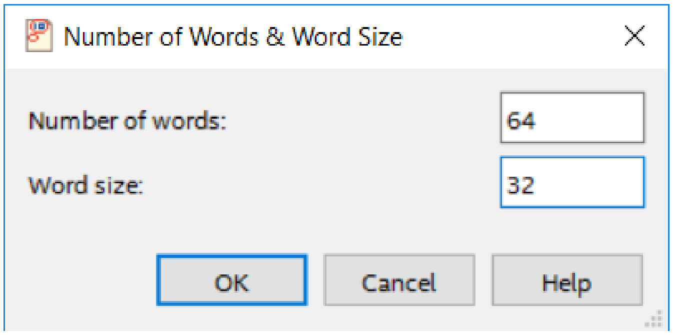
____ 2. In the **New** dialog box, expand the **Memory Files** category and select **Hexadecimal (Intel-Format) File**.

This hex file will be used to initialize a dual-port RAM you will create in the next step.


New
New Quartus Prime Project
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
Qsys System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
Signal Tap Logic Analyzer File
University Program VWF
Other Files
AHDL Include File
Block Symbol File
Chain Description File
Synopsys Design Constraints File
Text File
OK
Cancel
Help

____ 3. Click **OK**.

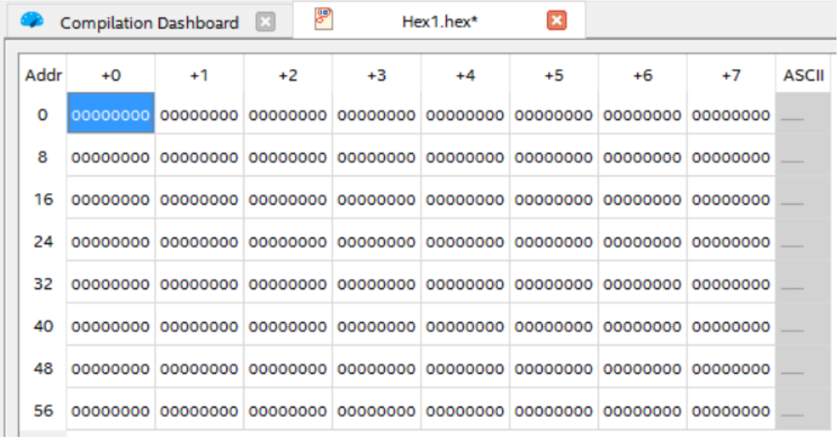
____ 4. In the memory size dialog box, enter **64** as the **Number of words** and **32** as the **Word size**.



Number of words: 64

Word size: 32

____ 5. Click **OK**.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

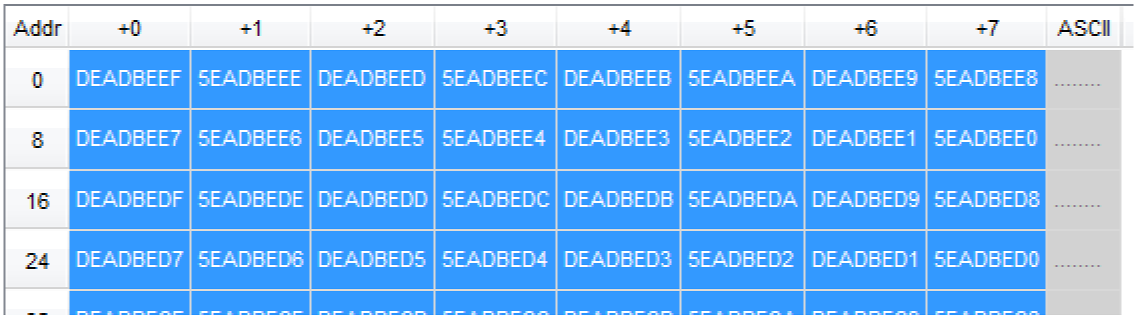
*The **Memory Editor** now displays the memory space. If it is not displayed exactly as shown above, you can change the number of cells per row (**View** menu) to **8**, the memory radix (**View** menu) to **Hexadecimal**, and the address radix to **Decimal**.*

____ 6. Select all of the memory locations. Right-click and select **Custom Fill Cells**.

____ 7. Use the **Custom Fill Cells** dialog box to enter your own values to initialize the memory. You can enter any values you want. Select one of the following:

a. Repeating sequence: Enter a series of numbers separated by commas or spaces to be repeated in memory.

b. Incrementing/decrementing: Enter a start value and another value by which to increment or decrement the start value.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	DEADBEEF	5EADBEEE	DEADBEED	5EADBEEC	DEADBEEB	5EADBEEA	DEADBEE9	5EADBEE8	....
8	DEADBEE7	5EADBEE6	DEADBEE5	5EADBEE4	DEADBEE3	5EADBEE2	DEADBEE1	5EADBEE0	....
16	DEADBEDF	5EADBEDE	DEADBEDD	5EADBEDC	DEADBEDB	5EADBEDA	DEADBED9	5EADBED8	....
24	DEADBED7	5EADBED6	DEADBED5	5EADBED4	DEADBED3	5EADBED2	DEADBED1	5EADBED0	....

____ 8. Click **OK** when complete to close the dialog box.

____ 9. Save the file as **ram.hex** in the project directory (you may have to navigate to the project directory again before you save). Make sure the option to **Add file to current project** is enabled.

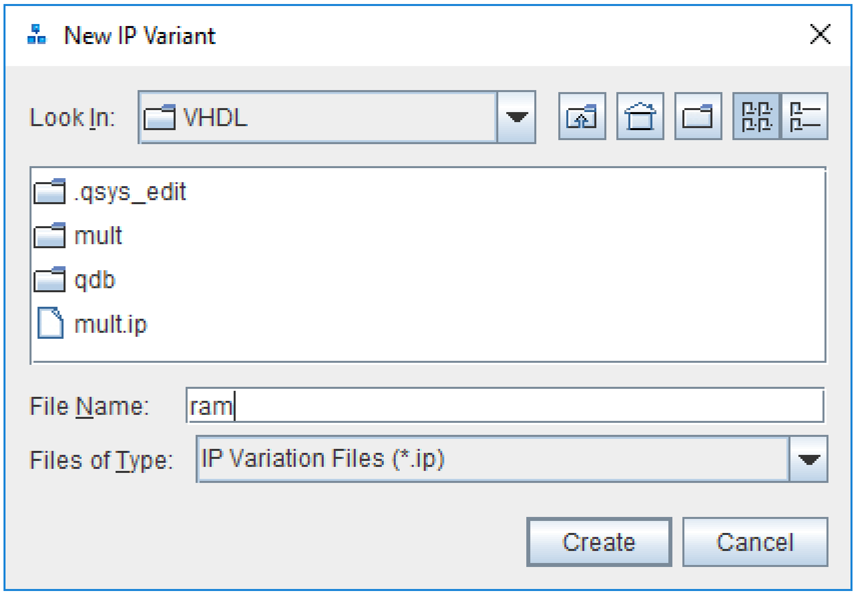
____ 10. Close **ram.hex**.

Step 3: Create a 64x32 RAM from the IP Catalog

____ 1. In the IP Catalog search field, enter **ram:** *(be sure to include the colon)*.

____ 2. Add the IP named **RAM: 2-PORT** by double-clicking it or selecting it and clicking **Add**.

____ 3. For the **File Name**, enter **ram**.


New IP Variant

Look In: VHDL

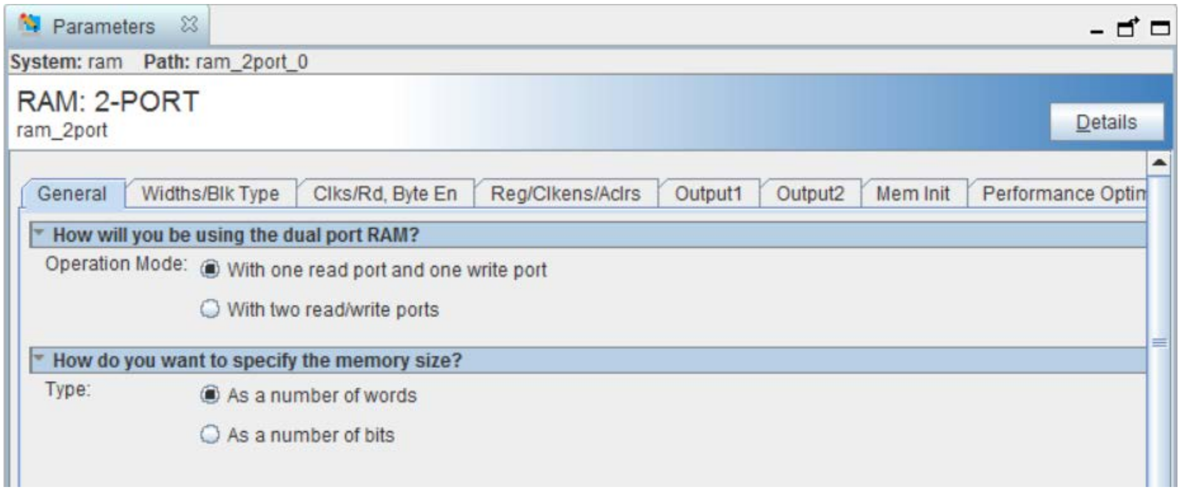
.qsys_edit
mult
qdb
mult.ip

File Name: ram

Files of Type: IP Variation Files (*.ip)

____ 4. Click **Create**.

____ 5. On the **General** tab, make sure **With one read port and one write port** is selected for the **Operation Mode**. Make sure the **Type** is set to **As a number of words**.



System: ram **Path:** ram_2port_0

RAM: 2-PORT
ram_2port

How will you be using the dual port RAM?

Operation Mode: ● With one read port and one write port
○ With two read/write ports

How do you want to specify the memory size?

Type: ● As a number of words
○ As a number of bits

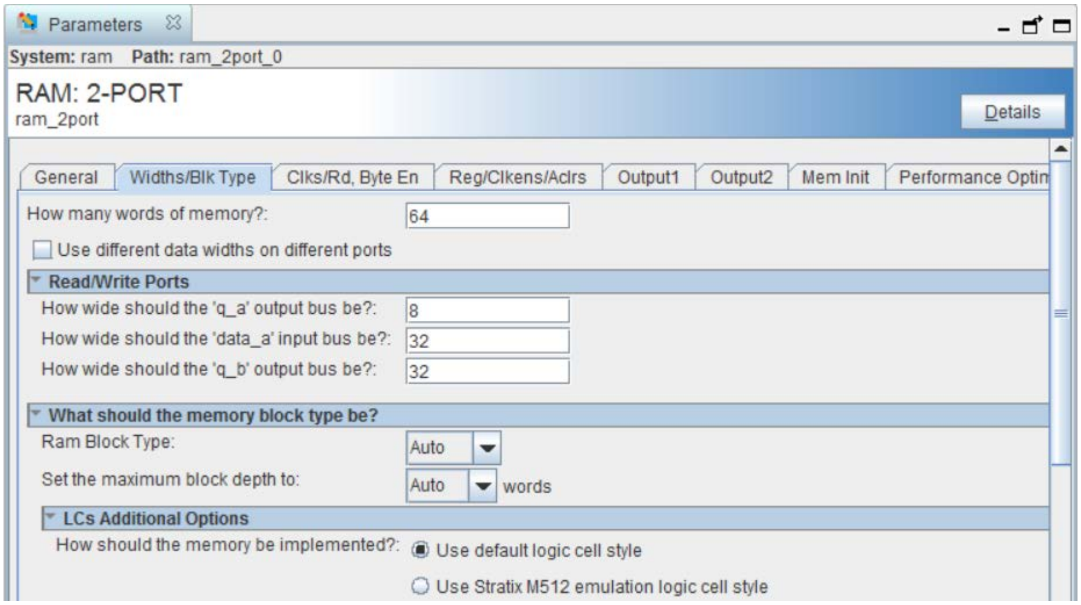
____ 6. Switch to the **Widths/Blk Type** tab.

____ 7. On this tab, set the number of **words of memory** to **64** and the width of the **data_a input bus** and **q_b output bus** to **32**.

The ***q_a*** *output bus is only used for true dual port operation, so it can be left at its default.*

Don't worry if you see red error messages appear as you are adjusting these parameters. They will go away as soon as you finish setting the values.

____ 8. Set the **RAM Block Type** to **Auto** and the **maximum block depth** to **Auto**. Leave all other settings at their defaults.



System: ram **Path:** ram_2port_0

RAM: 2-PORT
ram_2port

How many words of memory?: 64

□ Use different data widths on different ports

Read/Write Ports

How wide should the 'q_a' output bus be?: 8

How wide should the 'data_a' input bus be?: 32

How wide should the 'q_b' output bus be?: 32

What should the memory block type be?

Ram Block Type: Auto

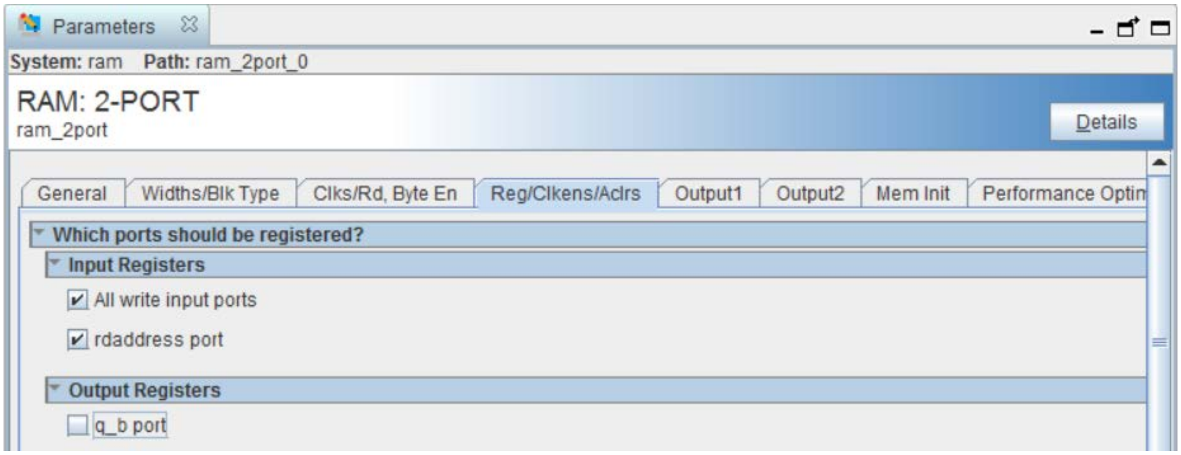
Set the maximum block depth to: Auto words

LCs Additional Options

How should the memory be implemented?: ● Use default logic cell style
○ Use Stratix M512 emulation logic cell style

____ 9. Switch to the **Reg/Clkens/Aclrs** tab.

____ 10. On this tab, disable the option for an **Output Register** on the **q_b port**. Leave all other settings at their defaults.



Parameters

System: ram Path: ram_2port_0

RAM: 2-PORT
ram_2port

Details

Which ports should be registered?

Input Registers

☑ All write input ports

☑ rdaddress port

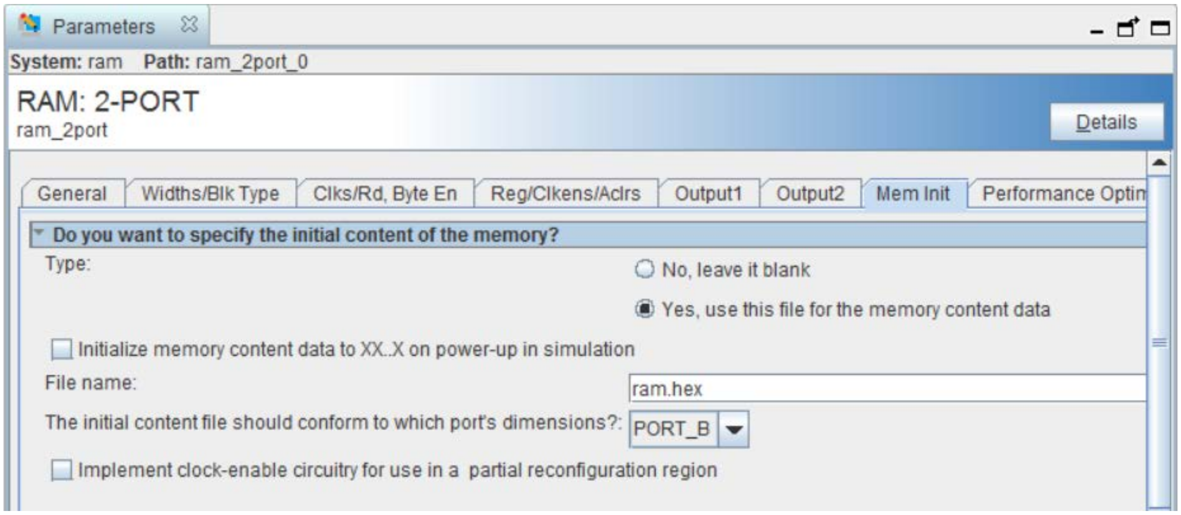
Output Registers

□ q_b port

____ 11. Switch to the **Mem Init** tab.

____ 12. For the **Type** of initialization, select **Yes, use this file for the memory content data**.

____ 13. Type in **ram.hex** for the **File name**. This is the file you created in the previous step.



Parameters

System: ram Path: ram_2port_0

RAM: 2-PORT
ram_2port

Details

Do you want to specify the initial content of the memory?

Type: ○ No, leave it blank

● Yes, use this file for the memory content data

□ Initialize memory content data to XX..X on power-up in simulation

File name: ram.hex

The initial content file should conform to which port's dimensions?: PORT_B

□ Implement clock-enable circuitry for use in a partial reconfiguration region

*If an error appears in the **System Messages** tab at the bottom of the IP Parameter Editor, you may not have set the word size and memory depth correctly in either the **ram.hex** file or in the **Widths/Blk Type** tab of the parameter editor. If the issue is with the **ram.hex** file, you can use the Memory Size Wizard, found in the Intel Quartus **Edit** menu after reopening **ram.hex**, to adjust the file to the correct dimensions.*

____ 14. Click **Generate HDL** or select **Generate HDL** from the **Generate** menu.

____ 15. Choose the same synthesis generation options you selected earlier for the **mult** IP core and click **Generate**. If requested, choose to save changes.

____ 16. Click **Close** once generation is complete. Click **Finish** to close the IP Parameter Editor.

Once again, you can check the lab installation directory to see all the files generated by the IP Parameter Editor. You have now created the two components needed for this design.

Step 4: Instantiate and connect design blocks according to design entry method

*Choose **ONE** of the following procedures based on your design entry method (VHDL, Verilog, or Schematic). Follow **only** the directions for your selected design entry method and then proceed to **Step 5** on page 24.*

VHDL

Perform these instructions only if you are using VHDL.

____ 1. Open **pipemult.vhd**. You can use the **Open** command from the **File** menu, click the 📂 toolbar button, or double-click the top-level entity in the Project Navigator. You can also click **Open** in the **Project Files** section of the **Tasks** window.

*This is the top-level file for the design. Normally, you would have to instantiate both **ram** and **mult** and connect them together. In the interest of time, the file has been almost completed for you by including the **ram** component declaration and instantiation. However, it is missing the instantiation of the multiplier.*

____ 2. Open the file **mult.cmp**, found in the **mult** folder in the project directory.

____ 3. Copy the **component** declaration from **mult.cmp** and paste it into the architecture declaration section of **pipemult.vhd** where indicated.

____ 4. Close **mult.cmp**.

____ 5. Open the file **mult_inst.vhd**, also in the **mult** directory.

____ 6. Copy the contents of **mult_inst.vhd** (the component instantiation part starting with **u0**) and paste into the architecture body of **pipemult.vhd** where indicated.

____ 7. Change the copied instance name from **u0** to **mult_inst**.

____ 8. Change the following signal names in the instantiation: (Note: While typing, the auto-complete pop up window may appear and show you a list of the available signal names which have been previously defined in the design file. You can use the **cursor keys**, *not the mouse*, to select a name from the list and press **Enter** to accept it.)

CONNECTED_TO_dataa	to	**dataa**
CONNECTED_TO_datab	to	**datab**
CONNECTED_TO_clock	to	**clk1**
CONNECTED_TO_result	to	**mult_to_ram**

____ 9. While you have this file open, try double-clicking any signal or port name to select it.

Notice how all instances of the selected signal or port name are highlighted in the file. This makes it very easy when editing HDL code to see where nodes are used or how they are connected together.

____ 10. Save **pipemult.vhd**.

____ 11. Close **mult_inst.vhd**.

____ 12. Continue to **Step 5: Check the design**.

Verilog

Perform these instructions only if you are using Verilog entry.

____ 1. Open **pipemult.v**. You can use the **Open** command from the **File** menu, click the 📂 toolbar button, or double-click the top-level entity in the Project Navigator. You can also click **Open** in the **Project Files** section of the **Tasks** window.

*This is the top-level file for the design. Normally, you would have to instantiate both **ram** and **mult** and connect them together. In the interest of time, the file has been almost completed for you by including the **ram** instantiation. However, it is missing the instantiation of the multiplier.*

____ 2. Open the file **mult_inst.v**, found in the **mult** folder in the project directory.

*This is the instantiation template generated by the IP Parameter Editor. If you do not see the folder or file, open the **mult.ip** file found in the project directory to reopen the IP Parameter Editor, and click **Generate HDL** to generate the files for the IP core.*

____ 3. Copy the contents of **mult_inst.v** (the component instantiation) and paste into the body of **pipemult.v** where indicated.

____ 4. Change the copied instance name from **u0** to **mult_inst**.

____ 5. Change the following signal names in the instantiation. (Note: While typing, the auto-complete pop up window may appear and show you a list of the available signal names which have been previously defined in the design file. You can use the **cursor keys**, *not the mouse*, to select a name from the list and press **Enter** to accept it.)

_connected_to_clock_	to	**clk1**
_connected_to_dataa_	to	**dataa**
_connected_to_datab_	to	**datab**
_connected_to_result_	to	**mult_to_ram**

____ 6. While you have this file open, try double-clicking any signal or port name to select it.

Notice how all instances of the selected signal or port name are highlighted in the file. This makes it very easy when editing HDL code to see where nodes are used or how they are connected together.

____ 7. Save **pipemult.v**.

____ 8. Close **mult_inst.v**.

____ 9. Continue to **Step 5: Check the design**.

Schematic

Perform these instructions only if you are using schematic entry.

____ 1. Open **pipemult.bdf**. You can use the **Open** command from the **File** menu, click the 📂 toolbar button, or double-click the entity in the Project Navigator. You can also click **Open** in the **Project Files** section of the **Tasks** window.

*This is the top-level schematic file for the design. Normally, you would have to instantiate both **ram** and **mult** sub-designs as well as all I/O pins and other connections. In the interest of time, the schematic file has been almost completed for you with some of the I/O and other connections already created. However, the design is missing the **ram** and **mult** blocks themselves and the output pins **q[31..0]**.*

____ 2. In the schematic file, double-click any empty space to open the **Symbol** window.

a. Click the browse button for the **Name:** field.

b. Browse to the **mult** directory in the project directory, and **Open** the **mult.bsf** symbol file.

c. Click **OK** in the **Symbol** window.

d. Click the left mouse button to place the symbol inside the schematic file where indicated.

*<u>Note</u>: The three ports on the left side of the multiplier should line up exactly with the wires coming from the input pins (no X's). If they do not, you may not have specified the multiplier parameters correctly when configuring the IP. If this is the case, hit the **Esc** key to cancel the symbol placement, and regenerate the incorrectly created IP from the IP Catalog using the steps listed earlier, choosing to overwrite the incorrect files when prompted.*

____ 3. Right-click on the **mult** symbol and choose **Properties**.

____ 4. In the Symbol Properties dialog box, change the **Instance name:** from **inst** to **mult_inst**. Click **OK**.

____ 5. Double-click an open area again to reopen the **Symbol** window. This time, go into the **ram** folder to open the **ram.bsf** symbol file and place it in the schematic file where indicated.

*<u>Note</u>: With **ram**, the lower 4 ports on the left side of the symbol should line up with the wires coming from the input pins. The **data[31..0]** port should be unconnected.*

____ 6. As you did with mult, use the **Symbol Properties** dialog box to change the name of the **ram** instantiation from **inst** to **ram_inst**.

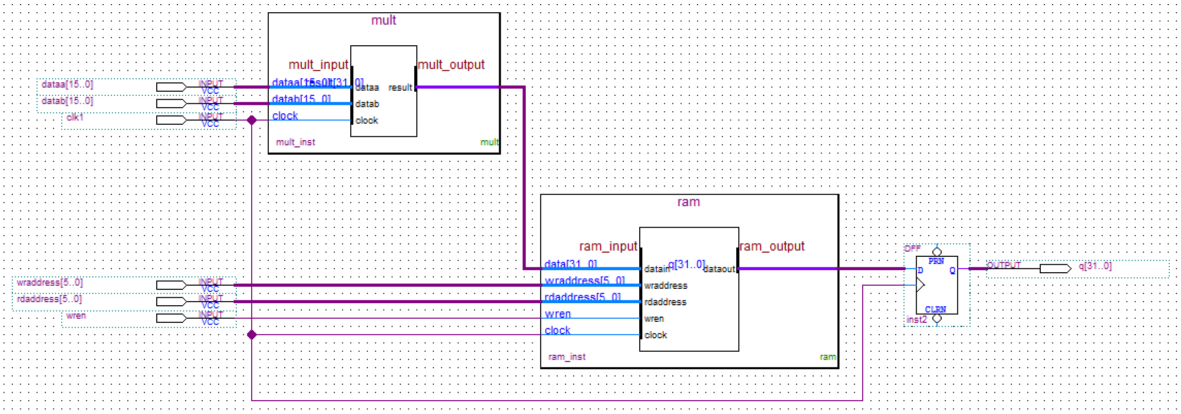
____ 7. Open the **Symbol** window again and this time, as a shortcut, type **output** in the **Name:** field.

*The **Symbol** window found the output symbol automatically by name.*

____ 8. Click **OK** and place the output pin near the output port of the **inst2** register symbol.

____ 9. Double-click the **pin_name1** text and change it to **q[31..0]**.

____ 10. Click the bus drawing tool ⌝ found in the schematic editor toolbar, and click and drag to draw the bus connection between the **mult_output** output of **mult** to the **data[31..0]** input of **ram**.



The resulting schematic should look like the above figure.

____ 11. Save **pipemult.bdf**.

Step 5: Check the design

____ 1. From the **Assignments** menu, select **Settings**.

____ 2. Go to the **Compiler Settings** category on the left and click **Advanced Settings (Fitter)**.

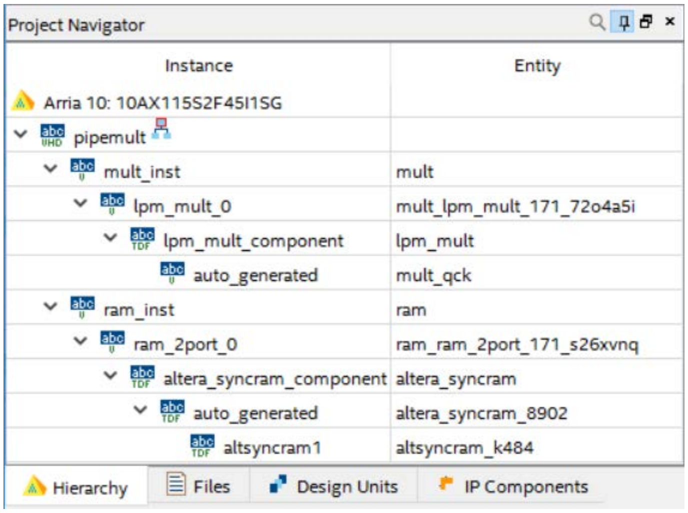
____ 3. From the **Processing** menu, go to **Start** and select **Start Analysis & Elaboration**. You can also click in the toolbar.

Analysis and elaboration checks that all the design files are present and connections have been made correctly. It also establishes the project hierarchy. You can safely ignore any warnings that appear in the ***Messages*** *window.*

____ 4. If a dialog appears at the end of the analysis and elaboration process, click **OK**.

____ 5. If there are any errors detailed in the **Messages** window, check your connections or, if necessary, return to the IP Parameter Editor (by opening the appropriate **.ip** file or through the **IP Components** category of the Project Navigator) for either IP core to fix the problem. You'll need to regenerate the files for an IP core if you make a change.

Feel free to explore the Project Navigator that now contains the complete project hierarchy.



Instance	Entity
Arria 10: 10AX115S2F45I1SG	
pipemult	
mult_inst	mult
lpm_mult_0	mult_lpm_mult_171_72o4a5i
lpm_mult_component	lpm_mult
auto_generated	mult_qck
ram_inst	ram
ram_2port_0	ram_ram_2port_171_s26xvnq
altera_syncram_component	altera_syncram
auto_generated	altera_syncram_8902
altsyncram1	altsyncram_k484

Exercise Summary

- Generated the HDL files for multiplier and RAM IP cores using the IP Catalog and incorporated them into a design
- Created a **.hex** file for RAM initialization using the Memory Editor
- Checked the design files using Analysis and Elaboration

END OF EXERCISE 2

Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Exercise 3

Exercise 3

Objectives:

- *Perform a full compilation*
- *Locate information in the Compilation Report*
- *Explore cross-probing capabilities by viewing logic in various windows*

Device Name	**10AX115S2F45I1SG**
Total Design	
Logic utilization (in ALMs)	
Total registers	
Total pins	
Total block memory bits	
Total DSP Blocks	
mult sub-design	
ALMs needed (mult)	
Combinational ALUTs (mult)	
Dedicated Logic Registers (mult)	
DSP Blocks (mult)	
ram sub-design	
ALMs needed (ram)	
Combinational ALUTs (ram)	
Dedicated Logic Registers (ram)	
Block memory bits (ram)	
M20Ks (ram)	
Control Signals	
Name	**Fanout**

Step 1: Compile the design

____ 1. If it’s not already open, select **Compilation Dashboard** from the **Processing** menu or the **Tasks** window.

The Compilation Dashboard provides quick access for performing all or only individual stages of compilation.

____ 2. Select **Start Compilation** from the **Processing** menu or click ▶ located in the toolbar to perform a full compilation of the design. You can also click ▶ next to **Compile Design** in the Compilation Dashboard.

Compilation may take up to 5 or 10 minutes. You can safely ignore any warnings that appear, but as the compilation proceeds try experimenting with the filtering options in the ***Messages*** *window to examine the different messages. Right-click a message to get more information about that message from the built-in help or the Intel web site (if an Internet connection is available).*

Even though we won't be using the individual Fitter compilation stages in these exercises, you can see how you can run each stage individually and jump to that stage's compilation report, which we'll examine next.

Step 2: Gather information from the Compilation Report

The Compilation Report provides all information on design processing. You use it to understand how the compiler interpreted your design and to verify results. It is organized by compiler executables, with each one generating its own folder. Throughout the following steps, feel free to explore the many reports generated by the tool.

____ 1. The **Compilation Report** should have opened in the Intel Quartus Prime GUI automatically when you started the compilation. If it didn't, click in the toolbar or next to any compilation stage in the Compilation Dashboard. You can also choose **Compilation Report** from the **Processing** menu.

____ 2. From the **Flow Summary** section at the top of the **Compilation Report**'s Table of Contents, record the **Logic utilization (in ALMs)**, **Total registers**, **Total pins**, **Total block memory bits**, and **Total DSP Blocks** in the table at the beginning of this exercise.

An adaptive logic module (ALM) is an FPGA logic structure made up of 2 adaptive look-up tables (ALUTs). These two ALUTs can be used together to implement a single logic equation of up to 7 inputs or independently to implement two logic functions of up to 6 inputs. ALMs are arranged in groups of ten. A grouping of 10 ALMs is a logic array block, or LAB. This relatively simple design does not use any ALMs.

An ALM can also be configured to implement a 32x2, 64-bit memory. When configured in this way, the LAB is referred to as a memory LAB, or MLAB, of 640 bits.

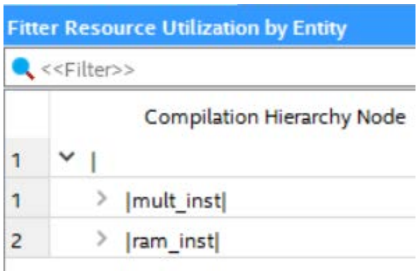
Larger memories in Intel® Arria® 10 devices are implemented using M20K blocks. These memory blocks can each store 20 Kb. M20K blocks can be combined together as needed.

A DSP block contains two multipliers and an adder unit to support various multiply-add/accumulate configurations. See the device handbook or user guide for more details.

____ 3. Expand the **Fitter** folder and examine Fitter reports in the Compilation Report.

a. Expand the **Place Stage** folder.

b. From the **Resource Utilization by Entity** report, record in the table the resource counts for the **mult_inst** and **ram_inst** sub-designs. Collapse the sub-designs in the **Compilation Hierarchy Node** column to make it easier to view the report, as shown below, and scroll to the right to find the appropriate columns of information.



Fitter Resource Utilization by Entity

<<Filter>>

	Compilation Hierarchy Node
1	⌄ |
1	› |mult_inst|
2	› |ram_inst|

c. Back in the Table of Contents, expand the **Plan Stage** folder.

d. Select the **Control Signals** report, and record the control signals found and their fan-out.

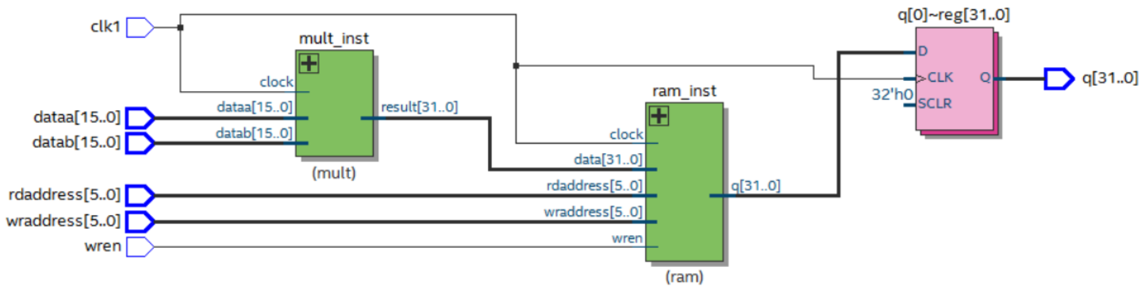
From these results, you can see that this small design is using very few device resources. No ALMs are used for the design. Memory and DSP blocks are used to implement ***ram*** *and* ***mult****, respectively. No other logic is being used. Feel free to explore some of the additional reports listed here and compare these resource reports with the reports found in the Analysis & Synthesis folder.*

In the next few steps, you will take a look at some additional ways to analyze the results of compilation and determine where the input and output registers are located (since they don't appear as used resources in the ***Resource Utilization by Entity*** *report).*

Step 3: Explore the design logically using the RTL Viewer

The ***RTL Viewer*** *allows you to view a logical representation of an analyzed design graphically. It is a very helpful tool for debugging HDL synthesis results.*

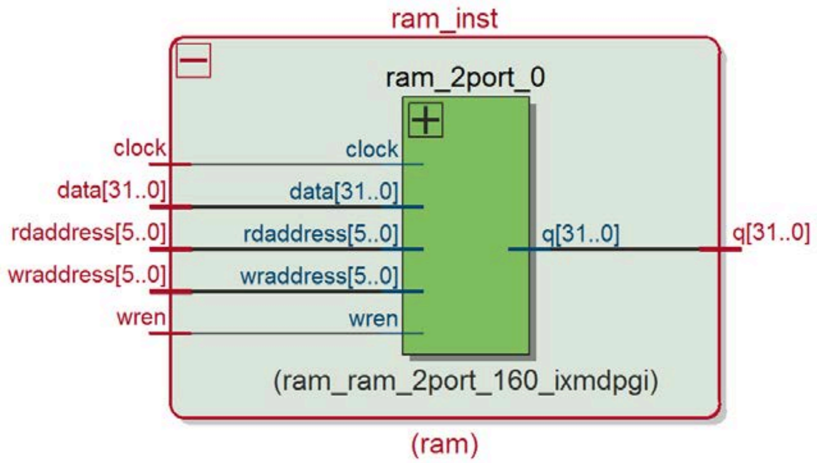
____ 1. From the **Tools** menu, open the **RTL Viewer** (under **Netlist Viewers**).



*You should see a schematic similar to the one shown here. The output register block is named **inst2[31..0]** in the schematic version of the project, but the rest of the diagram is the same for all three versions. This is a graphical view displaying the logical representation of the design. It shows the I/O, the instantiation of the **mult** and **ram** IP cores, and an additional set of output registers. Notice the registers are external to the memory block per the original design.*

____ 2. Select the **ram** sub-design to highlight it. The outline of the block turns bright red when the block is selected. Notice **ram_inst** is highlighted in the **Netlist Navigator** tab on the left side of the RTL Viewer.

____ 3. Double-click the **ram** sub-design block.



*You have now descended the hierarchy into the **ram** sub-design. As a result, the view changes to the above image. This shows that the **ram** sub-design is made up of a single instance named **ram_2port**.*

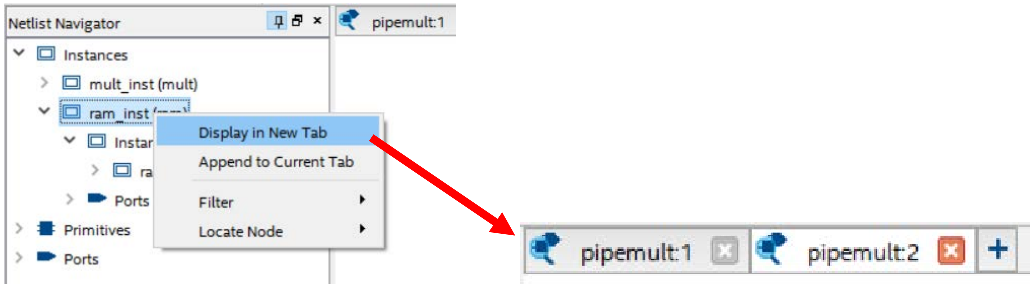
____ 4. Double-click **ram_2port_0** to descend one more level.

*You can now see the name of the IP core itself: **altera_syncram** (with **_component** added on at this level). You could continue double-clicking blocks to descend the hierarchy to its lowest level: single-bit RAM functions. But let's view this lowest level of the hierarchy in a different way.*

____ 5. With the outer **ram_2port_0** block still highlighted in red, right-click on any empty area and select **Expand to Upper Hierarchy**. Do this a second time after selecting the **ram_inst** hierarchy ring to highlight it.

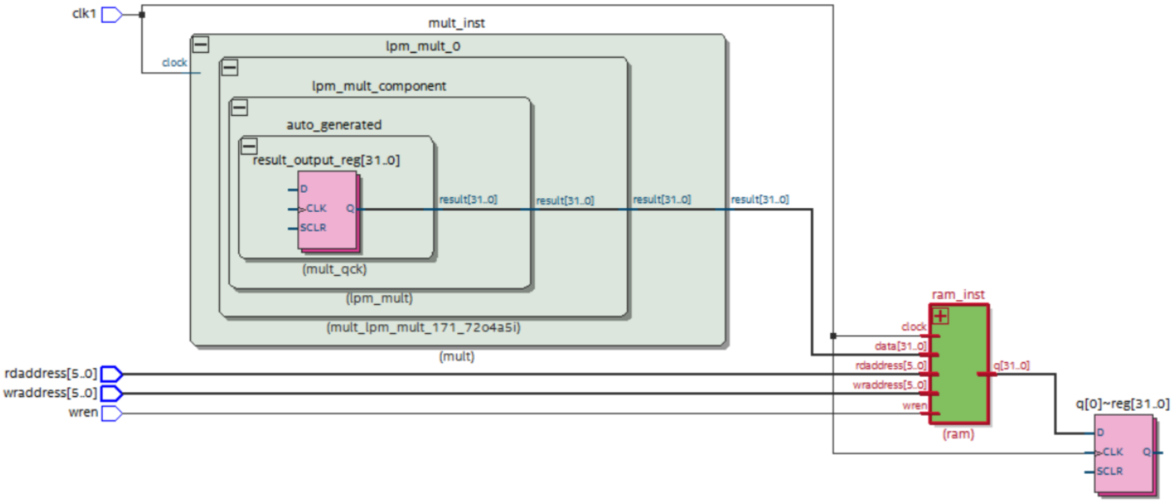
*This returns the Viewer to the top hierarchical level. If you've descended further into the hierarchy, you may need to do this a few times to return to the top. To ascend the hierarchy this way, the outer block needs to be highlighted. You can get back to the top quickly by selecting **Goto Top Level** from the right-click menu instead of **Expand to Upper Hierarchy**. You can also just double-click on the top-level module **pipemult** in the **Netlist Navigator**.*

____ 6. Right-click the instance **ram_inst (ram)** in the Netlist Navigator and select **Display in New Tab**.



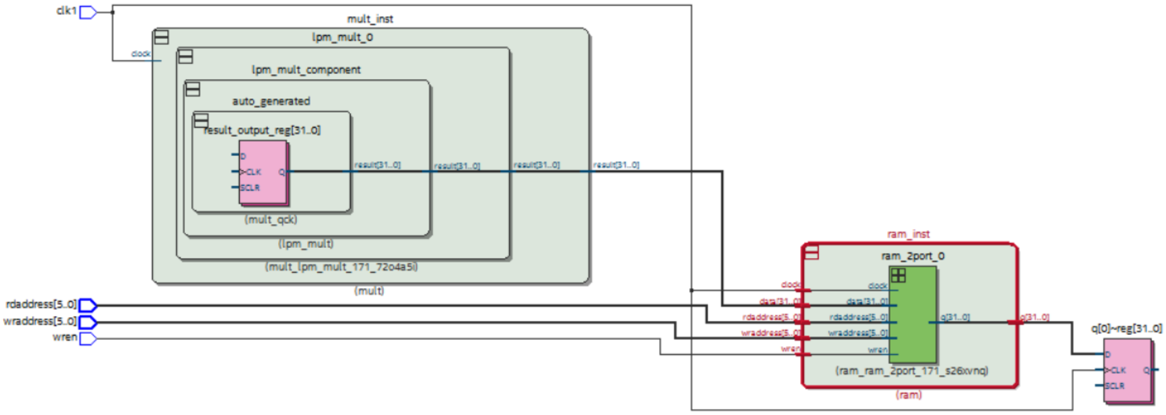
*A second tab named **pipemult:2** opens in the **RTL Viewer**. Displaying information in a new tab allows you to have multiple views open so you can compare and correlate what you see.*

____ 7. Right-click the displayed **ram_inst** block and select **Filter** → **Sources and Destinations**.



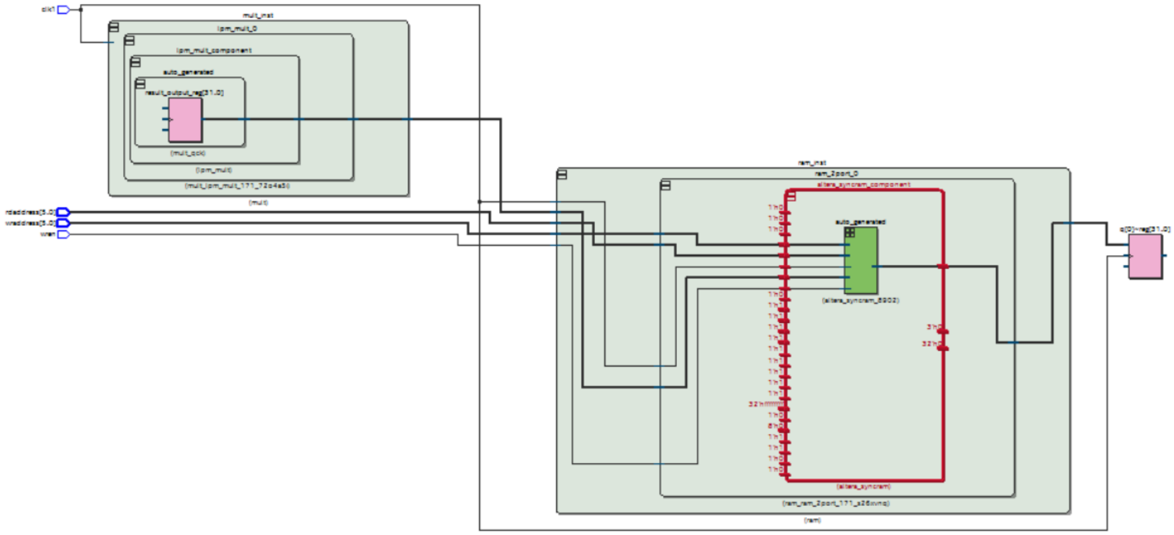
*The display is now restricted to the logic driving and driven by the **ram** block. This can help greatly in understanding the connections in a design, even ones you may not know are there. The shaded area around a block indicates a hierarchical boundary.*

____ 8. Click the "unwrap" ⊞ symbol on **ram_inst**. (Use the **View** menu → **Fit in Window** option or the ⛶ toolbar button to see the entire diagram.)



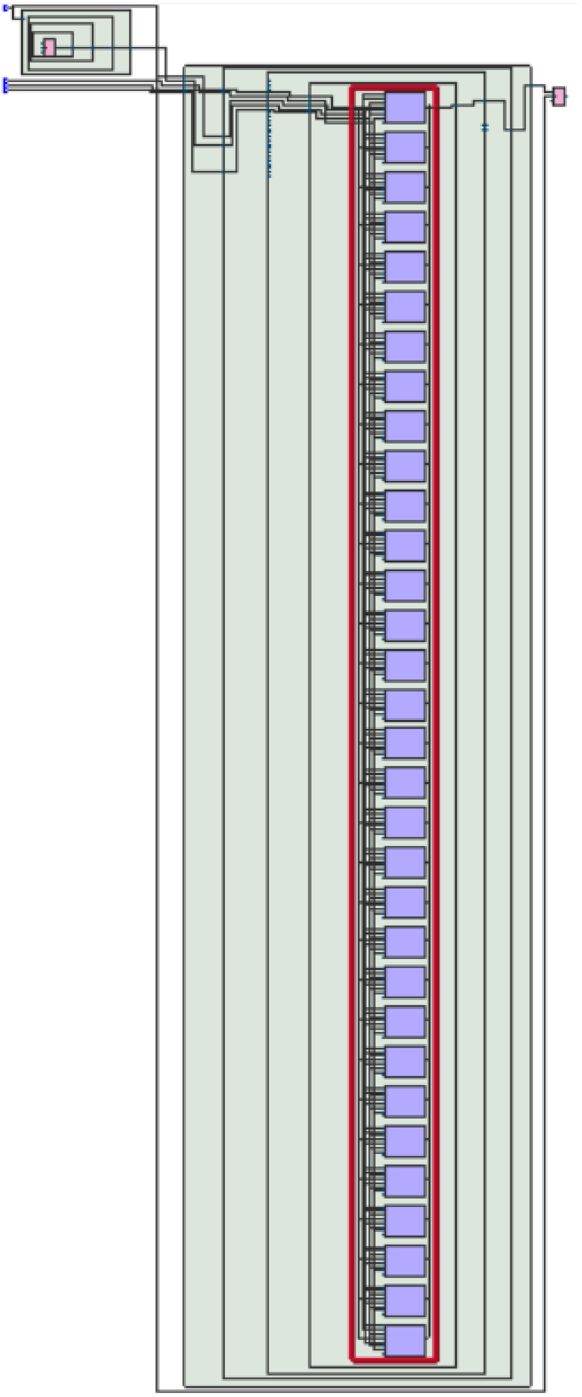
*Instead of displaying the lower hierarchical level alone (like when you double-clicked on the **ram_inst** block earlier), the **unwrap** feature displays the lower level within the current hierarchical level (may appear slightly different than the above depending on design entry method). The **ram_2port_0** module is now visible, and this could be unwrapped again to see **altera_syncram_component**.*

____ 9. Unwrap **ram_2port_0** and then **altera_syncram_component**.



*Descending the hierarchy again, the internal logic of the **altera_syncram_component** module is displayed including any module I/O that are not connected.*

____ 10. Continue unwrapping the **altera_syncram** module until you can’t unwrap it any more. Zoom out with the magnifying glass tool (right-click) or use the **Fit in Window** zoom to see more of the schematic, if necessary.



Moving down into the lower levels of the ***ram_inst*** *module, the viewer indicates that this module is represented by 32 smaller RAM functions, one function for each input/output data bit.*

Step 4: Explore the ram and mult sub-designs physically using the Technology Map Viewer

The Technology Map Viewer allows you to see how a design is actually implemented using FPGA/CPLD resources. Use this tool as an aid during constraining and debugging to see changes in resource usage as settings and options are adjusted.

____ 1. Cross-probe the **ram** block from the RTL Viewer into the Technology Viewer by doing the following:

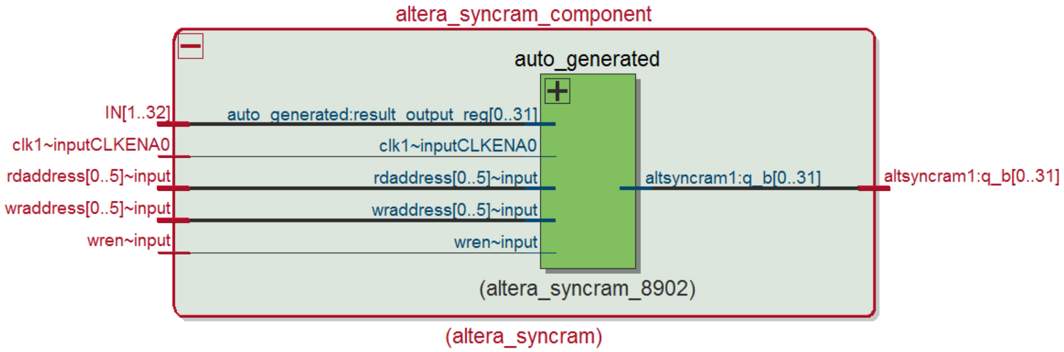
a. In the RTL Viewer, switch back to the standard arrow cursor if you switched to the magnifying glass to zoom in or out.

b. **Wrap** the innermost hierarchical rings for the ram function by clicking the ⊟ buttons until you can see the **altera_syncram_component** hierarchical level.

c. Click **altera_syncram_component** to select it.

d. Right-click the selected hierarchy, go to the **Locate Node** submenu and select **Locate in Technology Map Viewer**.

e. Unwrap the **altera_syncram_component** block.



*The Technology Map Viewer opens and displays the **auto_generated** module inside the **altera_syncram_component**. You may see an additional tab at the top of the viewer that contains the top-level design. You can now explore the rest of the design in a similar fashion to the RTL Viewer.*

____ 2. Unwrap the **auto_generated** module and then the **altsyncram1** module.

Now you are looking at the 32 memory bits as they are implemented in the single M20K memory block. You can verify this through the resource's properties.

____ 3. Right-click any of the 32 **ram_block**s and select **Properties**.

*In the Properties tab, you can look at a schematic view of the block resource and how it was configured for this design, including signal connections (**Fan-in** and **Fan-out** tabs) and parameter settings (**Parameters** tab).*

____ 4. Wrap up the **altsyncram1** module.

____ 5. Right-click in any empty space and select **Goto Top Level** to view the entire design in the Technology Map Viewer.

Compare this view to the RTL Viewer. All parts of the design are now represented by actual hardware resources in the device instead of the functional schematic view seen in the RTL Viewer.

____ 6. Unwrap the **mult_inst** block until it cannot be unwrapped anymore.

The multiplier function has been placed in a single DSP resource.

____ 7. Select the DSP resource, **mult0~mac**, and examine the properties of the DSP block resource. If you closed the Properties tab, right-click **mult0~mac** and select **Properties** to reopen it.

The Properties tab displays a schematic representation of the resource along with IP parameter settings and input and output port information, just like the memory block resource. If you'd like, try going back to the RTL Viewer and following the same steps. This comparison should help you to better understand the ways the different viewers present information about the design at each stage of compilation.

Step 5: Use the Chip Planner to view connections to the ram sub-design

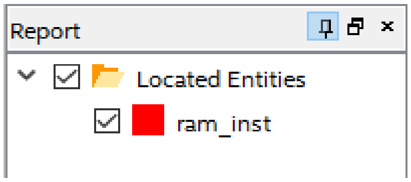
*The **Chip Planner** will give you a sense of where logic has been placed in the design. This can be very helpful when trying to understand design performance, as proximity is the key to performance in most newer FPGAs and CPLDs. Though the **Chip Planner** can be used for manually locating and moving logic, we only want to use it to evaluate results.*

____ 1. In the **Technology Map Viewer**, go back to the top level and select the RAM block **ram_inst** to highlight it, if it is not already highlighted. Right-click on the block.

____ 2. From the **Locate Node** submenu, select **Locate in Chip Planner**.

*You should now see the **ram** sub-design zoomed in in the Chip Planner and highlighted in color. If it's not zoomed in, double-click **Located 32 nodes** in the **Locate History** at the bottom of the window. Cross-probing to the Chip Planner creates a report, a colored overlay of the cross-probed object. You can use the zoom cursor in a similar manner to the other netlist viewers to zoom out (right-click) to see where in the Intel Arria 10 device it was placed by the Fitter. Feel free to explore the view of the device.*

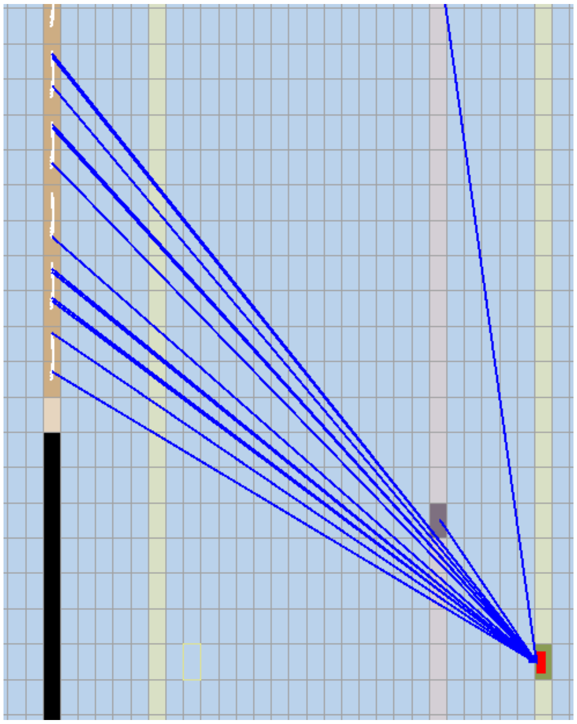
____ 3. In the **Report** window, turn off the report overlay by clicking the checkbox next to **Located Entities** or **ram_inst**.



After turning off the report, if you click elsewhere in the Chip View and deselect the memory block, you can find and highlight it again by going to the ***Locate History*** *tab at the bottom of the Chip Planner. Click or double-click any of the* ***ram_block2a**** *items. Clicking one of these objects highlights the RAM block in blue, while double-clicking highlights the block and zooms into it.*

If you further explore around the Chip Planner Chip View, you should see that the red shaded device resource in the Chip Planner is the DSP block that implements ***mult_inst****.*

____ 4. With the memory block highlighted in the Chip Planner, click the **Generate Fan-In Connections** toolbar button. As mentioned above, if you clicked on anything else so that the memory block is no longer highlighted, you can re-highlight the memory block using the **Locate History** tab.



The Chip Planner now displays the signal connections that are feeding into the ***ram*** *sub-design (coming from device I/O pins in an I/O column and the multiplier). You can use a similar method to see the fan-out.*

____ 5. Turn off the fan-in/fan-out by first clicking on any non-highlighted part of the device. Then click the **Clear Unselected Connections** toolbar button.

____ 6. *(Optional; time permitting)* Experiment with the **Layers Settings** and the **Tasks & Reports** windows. Try to understand the differences between enabling/disabling a layers setting and creating a report overlay.

____ 7. When you are done exploring, close the RTL Viewer, Technology Map Viewer, and Chip Planner.

Exercise Summary

- Performed a full compilation
- Gathered information from the compilation report
- Cross-probed between tools to analyze design processing results in different ways using the RTL Viewer, Technology Map Viewer, & Chip Planner

END OF EXERCISE 3

Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

Exercise 4

Exercise 4

Objectives:

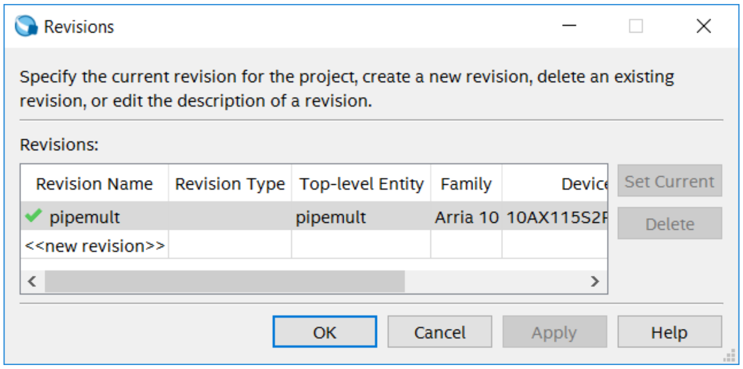
- *Create a new revision to store new constraint settings*
- *Make design constraints using the Assistment Editor*

Device Name	**10AX115S2F45I1SG**
Total Design	
Logic utilization (in ALMs)	
Total registers	
Total pins	
Total block memory bits	
Total DSP Blocks	
mult sub-design	
ALMs needed (mult)	
Combinational ALUTs (mult)	
Dedicated Logic Registers (mult)	
DSP Blocks (mult)	
ram sub-design	
ALMs needed (ram)	
Combinational ALUTs (ram)	
Dedicated Logic Registers (ram)	
Block memory bits (ram)	
M20Ks (ram)	
Control Signals	
Name	**Fanout**

Step 1: Create a new revision to store constraint changes

*In order to experiment with different settings or assignments and to see how these changes affect your results, the Intel Quartus Prime software has support for creating revisions, with each revision building a new Intel Quartus Settings File (**.qsf**). You can also compare the compilation results of your various revisions.*

____ 1. From the **Project** menu, select **Revisions**.



Revisions

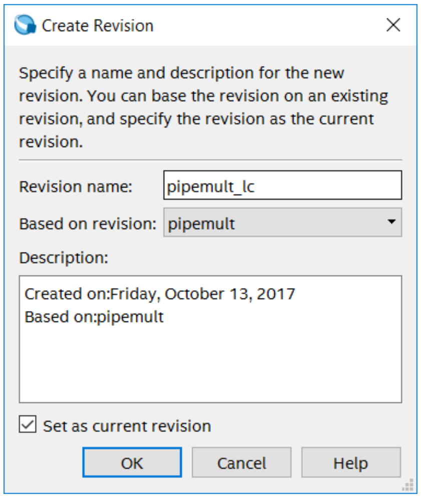
Specify the current revision for the project, create a new revision, delete an existing revision, or edit the description of a revision.

Revisions:

Revision Name	Revision Type	Top-level Entity	Family	Devic
✓ pipemult		pipemult	Arria 10	10AX115S2F
<<new revision>>				

____ 2. In the **Revisions** dialog box, double-click **<<new revision>>**.

____ 3. In the Create Revision dialog box, set the **Revision name** to **pipemult_lc**.



Create Revision

Specify a name and description for the new revision. You can base the revision on an existing revision, and specify the revision as the current revision.

Revision name: pipemult_lc

Based on revision: pipemult

Description:

Created on:Friday, October 13, 2017
Based on:pipemult

☑ Set as current revision

*Notice the revision is automatically based on the original revision **pipemult**. You could use the drop-down to base it on a blank revision (like starting a new project).*

____ 4. Leave all other settings at their defaults and click **OK.**

____ 5. Click **OK** to close the **Revisions** dialog box if it remains open.

Step 2: Implement the multiplier in logic elements instead of embedded multipliers

Intel® Arria® 10 DSP blocks are a valuable resource for implementing multiply operations. They provide a better usage of resources for multiplication compared to using ALMs. However, DSP blocks are a limited resource compared to the number of ALMs available. DSP blocks also have dedicated locations whereas ALMs are available all throughout the device. If your design has many multipliers, it may be advantageous to implement smaller or non-speed critical multipliers in ALMs (or even memory blocks) instead. This can be done using an option in the IP Parameter Editor or on a multiplier-by-multiplier basis using a logic option in the Assignment Editor.

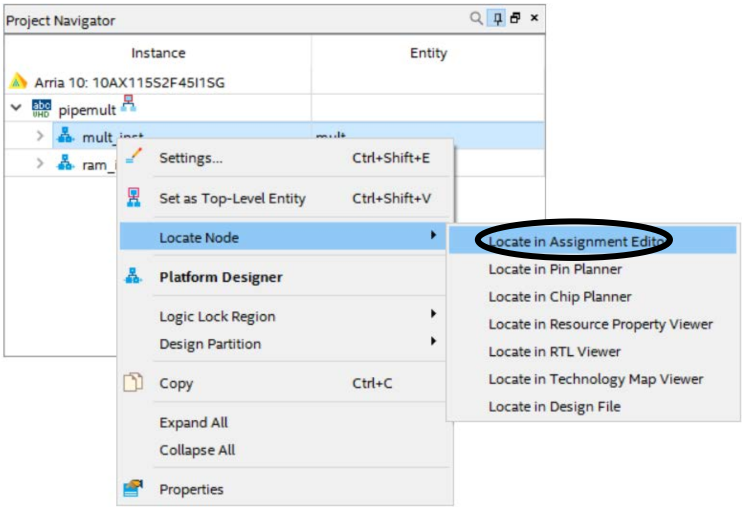
____ 1. In the toolbar, click the button to run **Analysis & Elaboration**.

New project revisions don't retain the original revision's compilation database, so the project hierarchy must be re-established through Analysis & Elaboration.

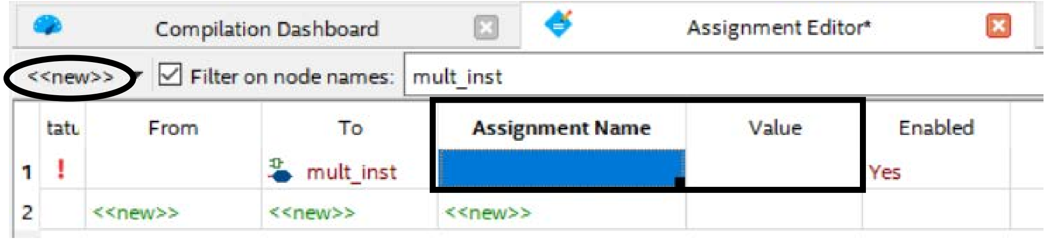
____ 2. In the Project Navigator Hierarchy, expand the **pipemult** hierarchy.

____ 3. Right-click the **mult_inst** entity in the hierarchy.

____ 4. From the **Locate Node** submenu, select **Locate in Assignment Editor**.



____ 5. Click the **<<new>>** button in the top left hand corner of the Assignment Editor.



	tatu	From	To	Assignment Name	Value	Enabled
1	!		mult_inst			Yes
2		<<new>>	<<new>>	<<new>>		

*The name **mult_inst** gets automatically entered into the **To** field.*

____ 6. Double-click the cell in the **Assignment Name** column and select **DSP Block Balancing** from the list.

Tip: By quickly typing "dsp", you may find the option faster than scrolling.

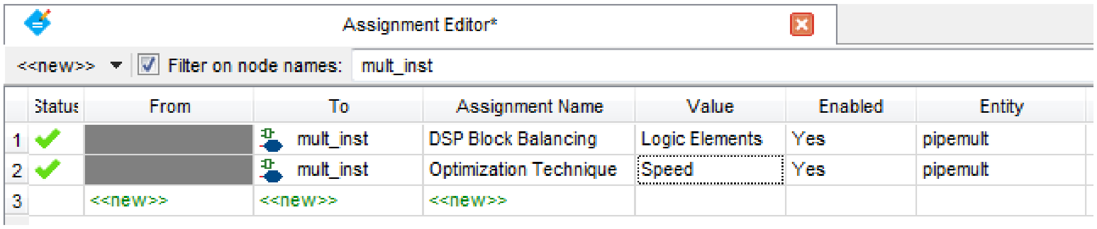
____ 7. Hit the **Enter** key or click elsewhere to accept the assignment name.

____ 8. Double-click the cell in the **Value** column for the **DSP Block Balancing** assignment, and select **Logic Elements** from the drop-down menu.

____ 9. Hit the **Enter** key or click elsewhere to accept the assignment value.

____ 10. Click the **<<new>>** button in the upper left again, and set the **Assignment Name** to **Optimization Technique**.

____ 11. Double-click the cell in the **Value** column for the **Optimization Technique** assignment and select **Speed**. The Assignment Editor should look like this:



Assignment Editor*

<<new>> ▼ ☑ Filter on node names: mult_inst

	Status	From	To	Assignment Name	Value	Enabled	Entity
1	✓		mult_inst	DSP Block Balancing	Logic Elements	Yes	pipemult
2	✓		mult_inst	Optimization Technique	Speed	Yes	pipemult
3		<<new>>	<<new>>	<<new>>			

____ 12. From the **File** menu, **Save** the Assignment Editor.

____ 13. Click ▶ in the toolbar or next to **Compile Design** in the Compilation Dashboard to compile the design.

Again, the compilation may take 5-10 minutes.

Step 3: Gather resource information from the Compilation Report

____ 1. Open the **Compilation Report** from the **Processing** menu or Compilation Dashboard if it's not already open.

____ 2. From the **Flow Summary** section at the top of the **Compilation Report**'s Table of Contents, record the **Logic utilization (in ALMs)**, **Total registers**, **Total pins**, **Total block memory bits**, and **Total DSP Blocks** in the table at the beginning of this exercise.

____ 3. Again expand the **Fitter** folder and extract information from Fitter reports.

a. Expand the **Place Stage** folder.

b. From the **Resource Utilization by Entity** report, record in the table the resource counts for the **mult_inst** and **ram_inst** sub-designs, collapsing the sub-designs in the **Compilation Hierarchy Node** column to make it easier to view the report.

c. Back in the Table of Contents, expand the **Plan Stage** folder.

d. Select the **Control Signals** report, and record the control signals found and their fan-out.

In the Verilog and VHDL versions of the project, you may notice that the ***ALMs needed*** *is a fractional value. How can half (0.5) of an ALM be used? ALMs can be split in two to create separate logic functions. The Compilation Report is indicating that some logic in the design only requires half of the logic available in an ALM.*

From the results, you can see that in this revision, the multiplier implementation was moved from the embedded multipliers into the logic array blocks. This move also required more fanout on the clock network to clock the added logic needed to implement the multiplier.

Feel free to open up the tools used in the prior exercise (RTL Viewer, Technology Map Viewer, Chip Planner) to examine the results of this compilation. The unwrapped ***mult*** *block should look* ***significantly*** *different in the Technology Map Viewer.*

Exercise Summary

- Created a new revision to evaluate different constraints
- Controlled logic options (constraints) using the Assignment Editor

END OF EXERCISE 4

----

*Other names and brands may be claimed as the property of others.

Exercise 5

Exercise 5

Objectives:

- *Assign I/O pins and perform I/O Assignment Analysis*
- *Back-annotate pin assignments to lock placement*
- *Optionally, use the Interface Planner tool to create I/O assignments for a design*

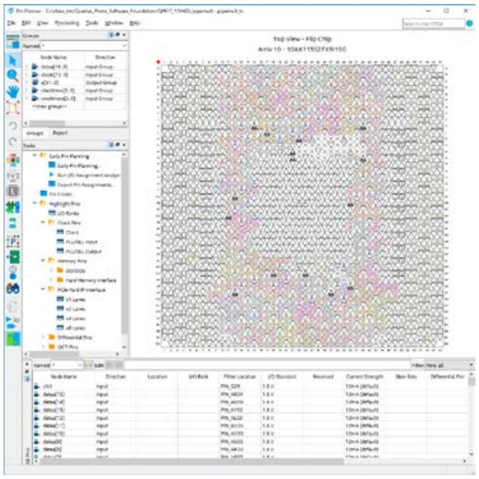
For this lab exercise, you have a choice: you can continue working on the design you've been using all day to explore the Intel Quartus Pin Planner, the main tool for creating I/O-related assignments for a design, or you can switch to a different design to learn about the Interface Planner, a Intel Quartus Prime Pro edition-only feature that makes it easy to create a complete, validated I/O floorplan. If there is sufficient time at the end of class, you can go through both parts of the lab.

*To work with the Pin Planner, simply continue below. If you want to work with Interface Planner, skip to page **59**.*

Using the Pin Planner

Step 1: Use Pin Planner to assign I/O pins & set I/O standards

____ 1. From the **Assignments** menu or the **Assignments** section of the **Tasks** window, open the **Pin Planner**.



Remember, if not already detached, you can detach the Pin Planner from the main Intel Quartus Prime application window to make it easier to work with. You can also undock the Groups and All Pins lists.

____ 2. Make sure the **Top View** of the device is displayed, indicated by the red dot in the upper left hand corner of the package. If not, select **Package Top** from the **View** menu.

____ 3. Open up the Pin Legend and experiment with different views using the options in the **View** menu. Also try generating Pin Planner reports using the **Highlight Pins** tasks in the **Tasks** window.

Try to understand the differences between how information is presented using the views vs. the Pin Planner reports.

____ 4. Turn off any reports you may have created in the previous step by disabling all the reports in the **Report** window, or right-clicking in the **Report** window and selecting **Delete All**.

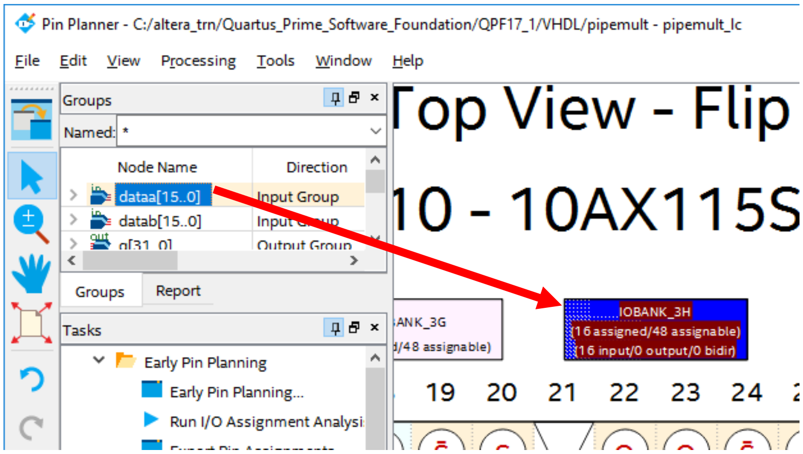
____ 5. In the **Pin Planner** toolbar, make sure that the **Show I/O Banks** toolbar button is selected. This is the original default view in the Pin Planner. If you are unsure of which button this is, you can also find this option in the **View** menu.

____ 6. In the **Groups** list window of the **Pin Planner** *(select **View**, **Groups List** if it is not visible)*, locate the **dataa** input bus, and click it once to select it.

*Tip: You may notice, if you left either the Technology Map Viewer or the Chip Planner open from the previous exercises, that the **dataa** bus is automatically selected and highlighted in those other tools when you select it here in the Pin Planner. The Chip Planner even does a highlighting animation to show you exactly where the selected I/O pins have been placed by the Fitter. This is even faster than using cross-probing!*

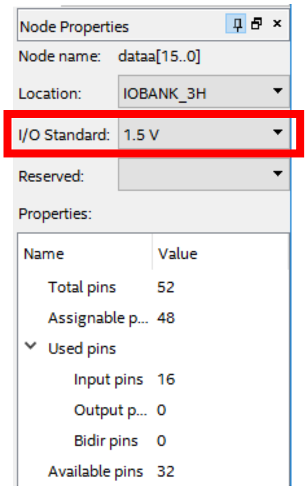
____ 7. Still in the Groups list, right-click the **dataa** bus and select **Node Properties**.

____ 8. In the Node Properties dialog box (located to the right of the Package View by default), from the **Location** menu, select **IOBANK_3H**.



The display for IOBANK_3H will now change to indicate that the bank has 16 assigned pins out of 48 total assignable pins. Assigning signals to an I/O bank like this (instead *of to individual pins) gives the Fitter the freedom to place the signals on any pins within the specified I/O bank.*

____ 9. Back in the **Node Properties** dialog box, set the **I/O standard** for **dataa** to **1.5 V**.



Node Properties

Node name: dataa[15..0]

Location: IOBANK_3H

I/O Standard: 1.5 V

Reserved:

Properties:

Name	Value
Total pins	52
Assignable p...	48
Used pins	
Input pins	16
Output p...	0
Bidir pins	0
Available pins	32

*You could have assigned the location and changed the I/O standard using the All Pins list instead of the Groups list. However, be aware that **individual** assignments made in the All Pins list (or on individual signals within groups in the Groups list) take precedence over **group** assignments made in the Groups list. The best way to ensure that all the signals in an entire bus or signal group inherit an assignment like this is to make the assignment to the whole group in the Groups list.*

____ 10. Click in any empty space in the Package View to de-select **dataa**.

____ 11. Using the Groups list with the Node Properties window or with drag and drop, also assign the **datab** bus to **IOBANK_3H**. *Do not change the I/O standard for this bus yet!*

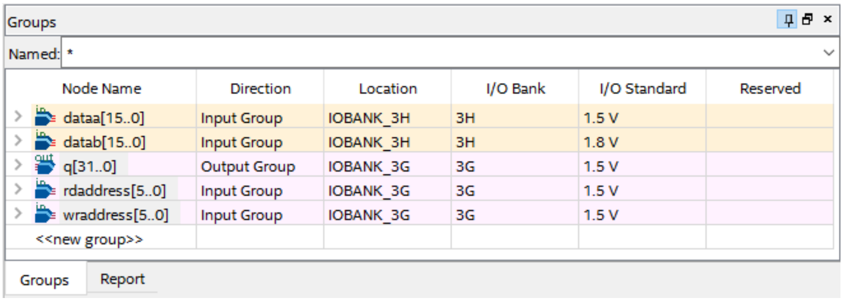
____ 12. Click in any empty space in the Package View to de-select **datab**.

Using drag and drop for location assignments to I/O banks selects the I/O bank the group is dragged to. This can affect subsequent assignments made in the Node Properties window. De-selecting pins assigned to I/O banks prevents this from happening.

____ 13. Assign the **q** output bus to **IOBANK_3G** (located just to the left of **IOBANK_3H**) and set the bus's **I/O standard** to **1.5 V**. You can do this in either the Node Properties or by scrolling right in the Groups list and double-clicking the cell in the **I/O Standard** column. De-select the **q** bus in the Chip View when finished.

____ 14. Assign both the **rdaddress** and **wraddress** buses to **IOBANK_3G** and set their I/O standard to **1.5 V**, de-selecting each bus after making assignments for each.

____ 15. Verify the assignments you've made in the Groups list.

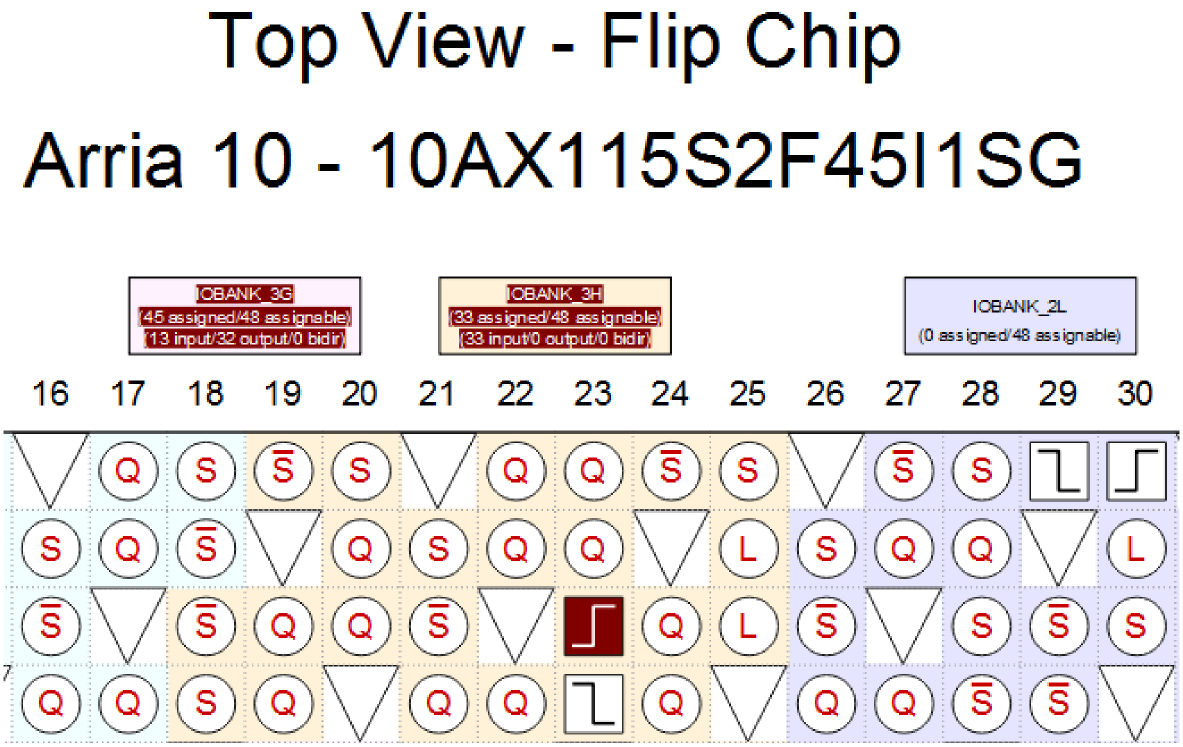


Node Name	Direction	Location	I/O Bank	I/O Standard	Reserved
dataa[15..0]	Input Group	IOBANK_3H	3H	1.5 V	
datab[15..0]	Input Group	IOBANK_3H	3H	1.8 V	
q[31..0]	Output Group	IOBANK_3G	3G	1.5 V	
rdaddress[5..0]	Input Group	IOBANK_3G	3G	1.5 V	
wraddress[5..0]	Input Group	IOBANK_3G	3G	1.5 V	
<<new group>>					

____ 16. Using the **All Pins** list, the Node Properties window, or drag and drop, assign **clk1** to pin **C23** (pin has a clock edge ↑ symbol) and set its I/O standard to **1.8 V** if it isn't already.

____ 17. Using any method, assign **wren** to **IOBANK_3G** and set its **I/O standard** to **1.5 V**.

The Package View of the Pin Planner should look like the following screenshot:


Top View - Flip Chip
Arria 10 - 10AX115S2F45I1SG
IOBANK_3G
(45 assigned/48 assignable)
(13 input/32 output/0 bidir)
IOBANK_3H
(33 assigned/48 assignable)
(33 input/0 output/0 bidir)
IOBANK_2L
(0 assigned/48 assignable)

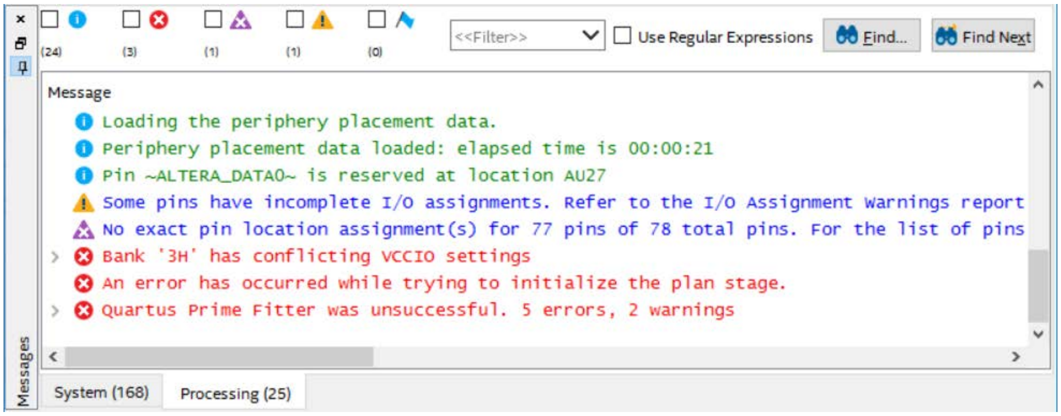
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
▽	Q	S	S̄	S	▽	Q	Q	S̄	S	▽	S̄	S	↓	↑
S	Q	S̄	▽	Q	S	Q	Q	▽	L	S	Q	Q	▽	L
S̄	▽	S̄	Q	Q	S̄	▽	↑	Q	L	S̄	▽	S	S̄	S
Q	Q	S	Q	▽	Q	Q	↓	Q	▽	Q	Q	S̄	S̄	▽

Step 2: Analyze I/O assignments

Now that you have made general I/O assignments, you can check the validity of those assignments without running a full compilation using I/O Assignment Analysis. This way you can quickly and easily find I/O placement and assignment issues and correct them.

____ 1. From the Pin Planner toolbar, click or select **Start I/O Assignment Analysis** from **Processing** menu.

____ 2. Switch to the main Intel Quartus Prime window and look at the **Processing** tab in the **Messages** window.



```
Message
  Loading the periphery placement data.
  Periphery placement data loaded: elapsed time is 00:00:21
  Pin ~ALTERA_DATA0~ is reserved at location AU27
  Some pins have incomplete I/O assignments. Refer to the I/O Assignment Warnings report
  No exact pin location assignment(s) for 77 pins of 78 total pins. For the list of pins
> Bank '3H' has conflicting VCCIO settings
  An error has occurred while trying to initialize the plan stage.
> Quartus Prime Fitter was unsuccessful. 5 errors, 2 warnings
```

Was the analysis successful? Check the messages. I/O Assignment Analysis has found errors with the I/O assignments.

____ 3. Review the messages and determine the cause of the error. Expand the error messages to get more detail as to why there is an issue.

Determining the cause of the I/O placement failure requires reading the error messages carefully and having a little understanding of ***Intel® Arria® 10*** *devices or general FPGA I/O blocks. See if you can understand and correct the cause of the errors on your own. If you do not have a lot of* ***Intel Arria 10*** *device or general FPGA experience, the next steps will show you how to correct the errors.*

____ 4. Bring the **Pin Planner** to the foreground.

____ 5. Scroll to the right in the Groups list to see the **I/O Standard** column. If the **I/O Standard** column does not appear, right-click in the Groups list and select **Customize Columns** to make the column visible.

*Notice that you have assigned the **dataa** and **datab** input buses to **I/O Bank 3H** but set different **VCCIO (1.8 & 1.5)** voltage levels for them. Intel Arria 10 FPGAs, like all Intel FPGA devices, allow for only one V_{CCIO} per I/O bank.*

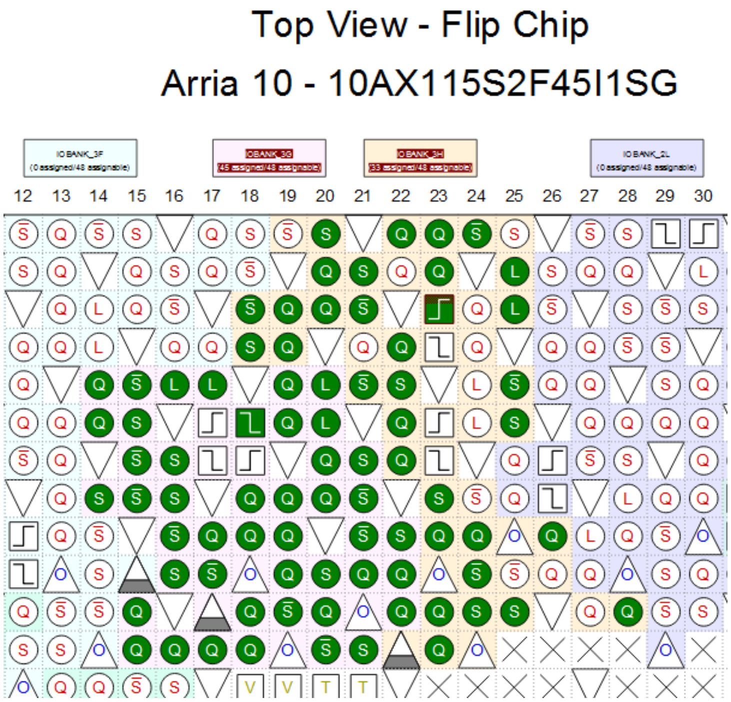
____ 6. Using the Groups list, change the I/O standard assignment for the **dataa** group to **1.8 V** (or **1.8 V (default)**). Click any empty space to accept the assignment.

____ 7. Re-run I/O Assignment Analysis.

See how quickly and easily you can check your I/O placement assignments without running a full compilation!

Now let's take a look at the I/O location assignments selected by the Fitter.

____ 8. In the Pin Planner, click the button in the toolbar or, from the **View** menu, go to the **Show** submenu and select **Show Fitter Placements**.

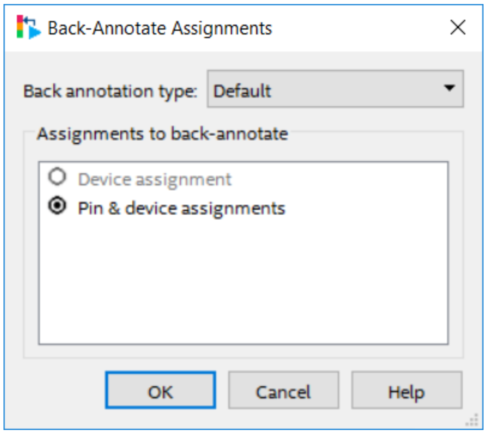

Top View - Flip Chip
Arria 10 - 10AX115S2F45I1SG
IOBANK_3F
(0 assigned/48 assignable)
IOBANK_3G
(45 assigned/48 assignable)
IOBANK_3H
(33 assigned/48 assignable)
IOBANK_2L
(0 assigned/48 assignable)
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

You can see that Fitter-selected pin locations appear in green in the Pin Planner. You can also see these assignments in the All Pins list in the ***Fitter Location*** *column. Performing I/O Assignment Analysis partially ran the Fitter, and by adding I/O bank location assignments, you constrained the signals to be placed in the selected I/O banks. Without these assignments, the Fitter could have placed the signals anywhere on the chip. Notice that* ***clk1****, the only signal you manually assigned to a specific pin (C23), appears green with a brown hatched pattern (you may have to zoom in a bit to see this). Remember that this indicates that your user assignment overlaps the location assignment selected by the Fitter.*

Step 3: Back-annotate pin assignments to lock placement

At this point in the I/O design flow, you may decide that you like the I/O location assignments made by the Fitter, and you've verified the pin-out to begin board design. Now you need to make sure that the pin locations are not moved during successive compilations.

____ 1. From the **Assignments** menu in the main Intel Quartus Prime window, select **Back-Annotate Assignments** to open the **Back-Annotate Assignments** dialog box. The dialog box may appear slightly different from the screenshot below.


Back-Annotate Assignments
Back annotation type: Default
Assignments to back-annotate
○ Device assignment
● Pin & device assignments
OK
Cancel
Help

____ 2. In **Assignments to back-annotate**, enable **Pin & device assignments** (default setting) as shown above. Click **OK**.

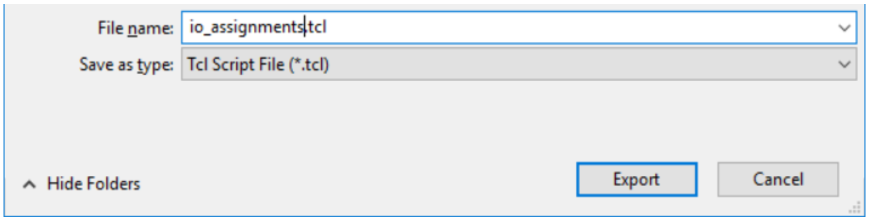
Notice how all the I/O pins that were green earlier have changed to solid brown. If you look in the ***Location*** *column in either the All Pins list or the Groups list, you should also now see specific assignments for all I/O. These changes indicate that the locations that were Fitter-assigned I/O are now user-assigned I/O and have been written into the* ***.qsf*** *file as constraints.*

Step 4: Transfer pin assignments to original revision

Now you should carry these pin assignments from the current design revision to the original revision (in case you decide to choose that revision later). Remember that all assignments, including I/O-related assignments, are unique to a particular project revision, so the assignments must be copied over if you want them to match between the two revisions. To do this, you will export the assignments as a Tcl script and import them into the original pipemult revision by running the script.

____ 1. With the **Pin Planner** active, go to the **File** menu and select **Export**.

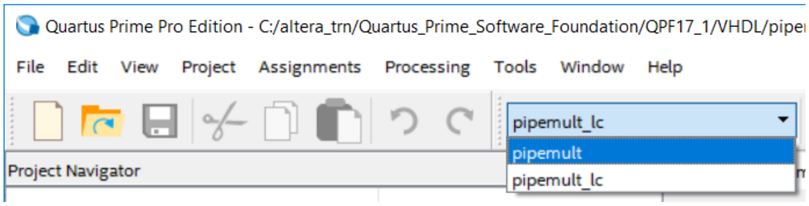
____ 2. In the **Export** dialog box, choose Save as type **Tcl Script File**. Set the **File name** to **io_assignments.tcl** and click **Export**.



File name: io_assignments.tcl

Save as type: Tcl Script File (*.tcl)

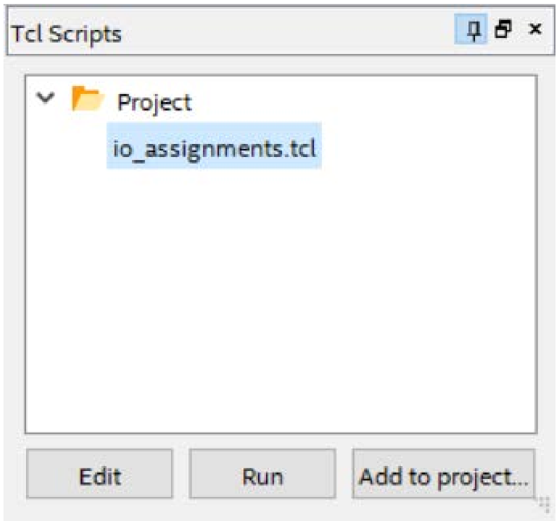
____ 3. Go back to the main Intel Quartus Prime window. From the drop-down menu at the top of the window, change the revision back to **pipemult**.



Quartus Prime Pro Edition - C:/altera_trn/Quartus_Prime_Software_Foundation/QPF17_1/VHDL/pipe

____ 4. From the **Tools** menu, select **Tcl Scripts…**

____ 5. Highlight the **io_assignments.tcl** file in the Project folder. Click **Run**.



____ 6. Click **OK** when complete and then click **Close**.

____ 7. Reopen the **Pin Planner** to see that the assignments were imported correctly. If **Show Fitter Placements** is turned off, turn it back on.

*Remember that the green pins are the Fitter-placed locations from the previous compilation of the revision **pipemult**. The reddish brown pins are the pins imported from the **pipemult_lc** revision, and the green and red hatched pins (if any; depends on how the Fitter placed the I/O in the earlier compilation) are intersecting assignments (think of the intersecting area of a Venn diagram) between the two revisions of the project. The green locations would all be removed if this revision were compiled again since the user-assigned I/O pins (in reddish brown) have precedence.*

____ 8. Turn off the viewing of Fitter placements using the toolbar button or by turning off the option in the **Show** submenu of the **View** menu.

*You can now easily see that the locations match the back-annotated locations from the **pipemult_lc** revision.*

You would now perform a full compilation of the completely constrained design. Once the compilation completes, you would be able to perform static timing analysis to verify all timing requirements are met and then program the FPGA device using a bitstream file generated by the Intel Quartus Prime Assembler.

Exercise Summary

- Assigned pin locations using the Pin Planner
- Back-annotated pin locations from prior compilation
- Verified I/O placement and constraints using I/O Assignment Analysis

END OF EXERCISE 5 (PIN PLANNER)

CONTINUE TO THE NEXT PAGE TO EXPERIMENT WITH INTERFACE PLANNER (TIME PERMITTING)

Using the Interface Planner

*In this part of the lab, you'll get a chance to play with the Interface Planner. To do this, you'll work with a more complicated project design that includes two DDR3 external memory interfaces (EMIFs). The project is stored as a Intel Quartus archived project file (**.qar**) in the class files installation directory.*

Step 1: Prepare project for Interface Planner

____ 1. In the main Intel Quartus Prime window, from the **Project** menu, select **Restore Archived Project**.

____ 2. Click the browse button for the **Archive name** and go to **<*lab_install_directory*>**\QPF17_1\InterfacePlanner\.

____ 3. Open the **.qar** file named **ddr3_x72_2inst.qar**.

____ 4. Back in the **Restore Archived Project** dialog, edit the **Destination Folder** name to point to the same location: **<*lab_install_directory*>**\QPF17_1\InterfacePlanner\.

____ 5. Click **OK**.

____ 6. When the new project opens, perform a full **Analysis & Synthesis** from the toolbar, the **Processing** menu, or the Compilation Dashboard. This should only take a minute or two.

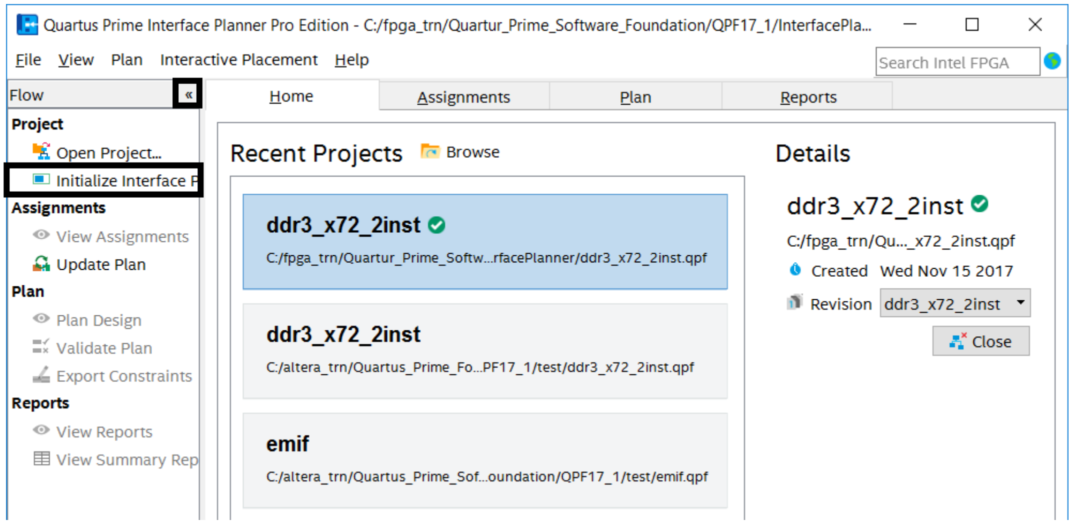
The design uses Platform Designer, Intel's easy-to-use system design tool, mentioned earlier, to create and parameterize the two EMIFs, connect them together, and generate HDL code for synthesis. Platform Designer is also the basis for the IP Parameter Editor, which you used earlier.

Step 2: Start the Interface Planner design flow

You'll now go through the Interface Planner flow to set up the Interface Planner project and create a Interface Planner plan.

____ 1. Start the Interface Planner from the **Tools** menu or the toolbar . Choose to close the Intel Quartus Prime software when prompted.

____ 2. Click **OK** to close the welcome dialog if it appears.



Quartus Prime Interface Planner Pro Edition - C:/fpga_trn/Quartur_Prime_Software_Foundation/QPF17_1/InterfacePla...

Recent Projects

ddr3_x72_2inst
C:/fpga_trn/Quartur_Prime_Softw...rfacePlanner/ddr3_x72_2inst.qpf

ddr3_x72_2inst
C:/altera_trn/Quartus_Prime_Fo...PF17_1/test/ddr3_x72_2inst.qpf

emif
C:/altera_trn/Quartus_Prime_Sof...oundation/QPF17_1/test/emif.qpf

Details

ddr3_x72_2inst
C:/fpga_trn/Qu..._x72_2inst.qpf
Created Wed Nov 15 2017
Revision ddr3_x72_2inst

____ 3. With the **ddr3_x72_2inst** project selected, click **Initialize Interface Planner** in the **Flow** window. If you don't see this window, click the arrows in the upper-left to reveal the window. Click **OK** when the process completes.

This first step checks the Intel Quartus project and imports any I/O-related assignments that have already been created in the Intel Quartus Prime software, such as those created in the Pin Planner.

____ 4. Click **View Assignments** in the **Flow** window or switch to the **Assignments** tab.

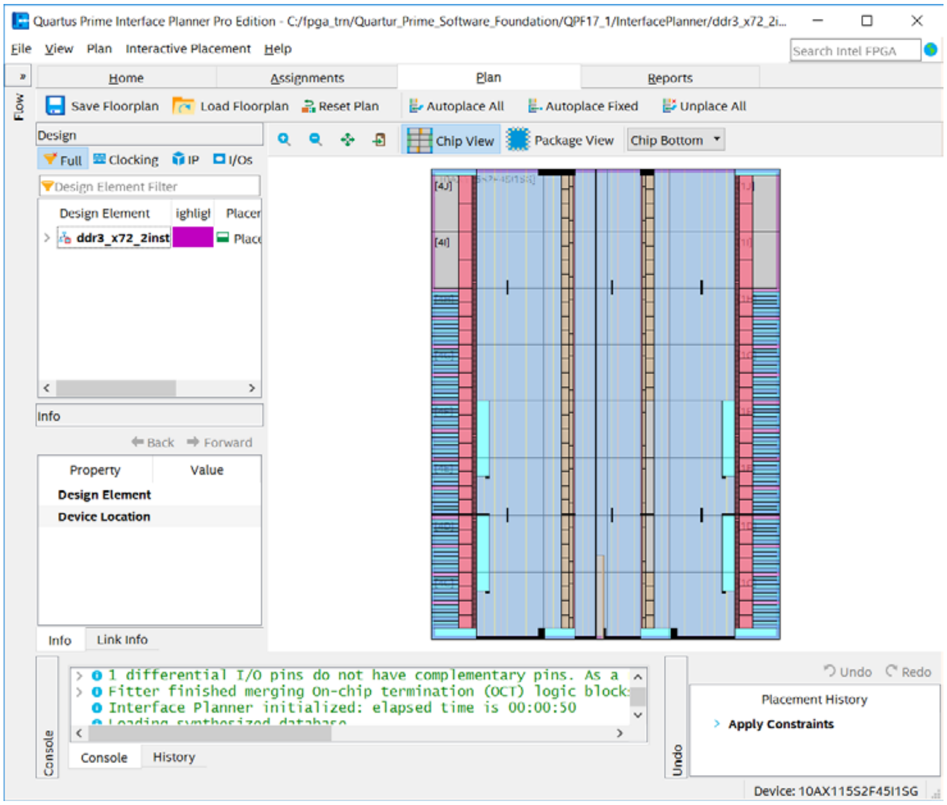
At this point in the flow, you would normally choose whether you want to maintain assignments created in the Intel Quartus Prime software or allow Interface Planner to override those assignments. You would do this by disabling Intel Quartus assignments on this tab that you want to override. There were no assignments in this project, so we'll skip this step. We will, however, come back later to see assignments after they are created by Interface Planner.

____ 5. From the **Flow** window or the **Plan** menu, click **Update Plan**.

Updating the plan incorporates Intel Quartus assignments, if there were any enabled ones, and gets you to the main function of Interface Planner: creating the plan.

Step 3: Create the Interface Planner plan

____ 1. Switch to the **Plan** tab or click **Plan Design** in the **Flow** window.


Quartus Prime Interface Planner Pro Edition - C:/fpga_trn/Quartur_Prime_Software_Foundation/QPF17_1/InterfacePlanner/ddr3_x72_2i...
File View Plan Interactive Placement Help
Search Intel FPGA
Home Assignments Plan Reports
Flow
Save Floorplan Load Floorplan Reset Plan Autoplace All Autoplace Fixed Unplace All
Design
Full Clocking IP I/Os
Design Element Filter
Chip View Package View Chip Bottom
Info
Back Forward
Info Link Info
Console History
Undo Redo
Placement History
Apply Constraints
Undo

Design Element	ighligl	Placer
ddr3_x72_2inst	🟪	Plac

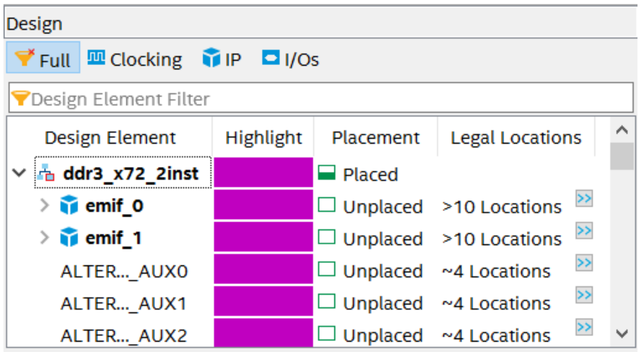
Property	Value
Design Element	
Device Location	

```
1 differential I/O pins do not have complementary pins. As a
Fitter finished merging On-chip termination (OCT) logic block:
Interface Planner initialized: elapsed time is 00:00:50
Loading synthesized database
```

Device: 10AX115S2F45I1SG

There are 2 EMIFs, so you'll make assignments using each of the two views available in Interface Planner, starting with the Chip View. The Chip View is very similar to the Intel Quartus Chip Planner. As you can see in the view, there are two thick vertical bars near the middle of the chip. These represent the I/O columns where most of the hard resources for external memory interfaces reside.

____ 2. In the **Design** window on the left, expand the top-level design element named **ddr3_x72_2inst**. Optionally, click the **IP** filter to enable it.



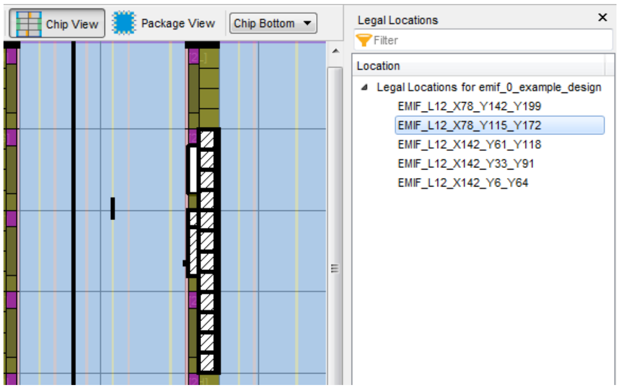
Design Element	Highlight	Placement	Legal Locations
ddr3_x72_2inst	🟪	Placed	
emif_0	🟪	□ Unplaced	>10 Locations
emif_1	🟪	□ Unplaced	>10 Locations
ALTER..._AUX0	🟪	□ Unplaced	~4 Locations
ALTER..._AUX1	🟪	□ Unplaced	~4 Locations
ALTER..._AUX2	🟪	□ Unplaced	~4 Locations

Here you can see all the design elements that make up the design. You can use Interface Planner to place everything you see in the list, but we're going to focus on the two memory interfaces named ***emif_0*** *and* ***emif_1****.*

____ 3. In the **Legal Locations** column, click the arrows ».

The design processes for a moment, and the ***Legal Locations*** *window opens. This lists the 10+ legal locations available for this interface that were indicated in the Design Elements list. You can see the legal locations highlighted in the Chip View. Zoom into the Chip View using the buttons above it to see the locations.*

____ 4. Click any of the listed legal locations to select it.



Location

Legal Locations for emif_0_example_design
- EMIF_L12_X78_Y142_Y199
- EMIF_L12_X78_Y115_Y172
- EMIF_L12_X142_Y61_Y118
- EMIF_L12_X142_Y33_Y91
- EMIF_L12_X142_Y6_Y64

All of the resources required to implement the EMIF at that location are highlighted.

____ 5. Double-click any of the locations in the Legal Locations window to assign **emif_0** to that location.

You can also make the assignment by clicking and dragging ***emif_0*** *to one of the highlighted locations in the Chip View. Notice that the Design Elements list now indicates that* ***emif_0*** *is placed. You can hover over the design element to see it highlighted in the Chip View or double-click in the Design window's* ***Highlight*** *column to select a different color for the placed design element.*

____ 6. Click the arrows » in the Legal Locations column for **emif_1**. Do not assign **emif_1** to a location yet.

Now notice that there is only 1 legal location for the second interface depending on where you placed the first interface. The placement of ***emif_0*** *used resources that prevent another EMIF from being placed in the same I/O column. Interface Planner updates as you make assignments to only display legal locations based on your selections.*

____ 7. Switch to the **Package View** using the button above the Chip View.

As you can see, the Package View is similar to the Intel Quartus Pin Planner, allowing you to make assignments based on I/O pin availability instead of internal device resources. You can see the 2 or 3 available legal locations, indicated by highlighted I/O pins, representing starting points for placing the second interface.

____ 8. Click to select the remaining legal location.

All of the I/O pins required to implement the second interface get highlighted.

____ 9. Double-click one of the remaining legal locations to place the interface.

Again, you can change highlight colors if you like.

____ 10. With the two interfaces placed, switch to the **Reports** tab.

____ 11. Experiment with reports by double-clicking any report task in the **Tasks** window.

Reports can help during the planning process to see what design elements have yet to be placed and to get more information about all elements, placed or not. Clicking blue links in reports highlights the selected element back on the Plan tab, making it easy to continue making assignments for unplaced elements or to make changes to placed elements.

____ 12. Switch back to the **Plan** tab.

____ 13. Examine the remaining design elements that still require placement.

The remaining elements include required power pins, EMIF status signals (listed as CLUSTERS, which we'll look at in a moment), and clock networks. For clocks, Interface Planner gives you the ability to easily assign clock device resources, a task that normally would need to be performed in the Assignment Editor.

____ 14. Click **Autoplace All** at the top of the Interface Planner window.

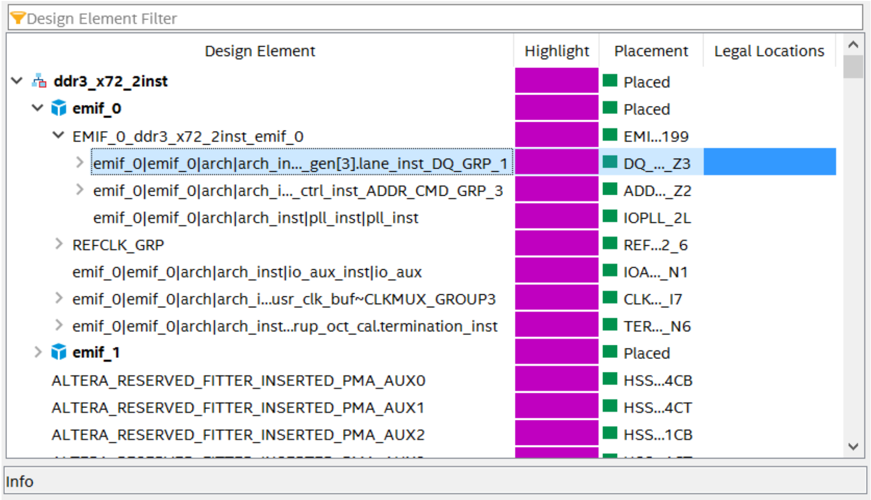
As expected, this places all remaining design elements. Before bringing these assignments to the Intel Quartus Prime software, let's do a little fine tuning.

Step 3: Adjust memory data pin locations

Let's say your board designer requests that some of the memory interface's data (DQ) pins be moved to different locations, if possible, to help with signal integrity. You can make changes like this and guarantee the changes' legality easily in Interface Planner.

____ 1. In the Design Element list, expand the **ddr3_x72_2inst** for either **emif_0** or **emif_1**.

____ 2. Directly below one of the instances, expand the next level of the hierarchy that starts with **EMIF_x_ddr3_x72_2inst…** (where **x** is **0** or **1**). Then, expand the next level below that which ends with **...lane_inst_DQ_GRP_1**.



Design Element Filter

Design Element	Highlight	Placement	Legal Locations
ddr3_x72_2inst	🟪	🟩 Placed	
emif_0	🟪	🟩 Placed	
EMIF_0_ddr3_x72_2inst_emif_0	🟪	🟩 EMI...199	
emif_0|emif_0|arch|arch_in..._gen[3].lane_inst_DQ_GRP_1	🟪	🟩 DQ_..._Z3	
emif_0|emif_0|arch|arch_i..._ctrl_inst_ADDR_CMD_GRP_3	🟪	🟩 ADD..._Z2	
emif_0|emif_0|arch|arch_inst|pll_inst|pll_inst	🟪	🟩 IOPLL_2L	
REFCLK_GRP	🟪	🟩 REF...2_6	
emif_0|emif_0|arch|arch_inst|io_aux_inst|io_aux	🟪	🟩 IOA..._N1	
emif_0|emif_0|arch|arch_i...usr_clk_buf~CLKMUX_GROUP3	🟪	🟩 CLK..._I7	
emif_0|emif_0|arch|arch_inst...rup_oct_cal.termination_inst	🟪	🟩 TER..._N6	
emif_1	🟪	🟩 Placed	
ALTERA_RESERVED_FITTER_INSERTED_PMA_AUX0	🟪	🟩 HSS...4CB	
ALTERA_RESERVED_FITTER_INSERTED_PMA_AUX1	🟪	🟩 HSS...4CT	
ALTERA_RESERVED_FITTER_INSERTED_PMA_AUX2	🟪	🟩 HSS...1CB	

Info

*A little background if you are not familiar with DDR-based external memory interfaces: groups of data (DQ) pins are associated with a data strobe (DQS) pin that acts like a clock for the DQ pins. All the DQ pins and the DQS pin are bidirectional, so the DQS is used to clock in data at either the controller on the FPGA (for reads) or on the external memory (for writes). These groupings are referred to as lanes. In Intel Arria 10 devices, these lanes are implemented using hard resources referred to as tiles in the I/O columns. In the Design Element list, you can now see the 8 DQ pins (**dq[0]** through **dq[7]**) that are associated with **dqs[0]** (along with a data mask pin, **dm[0]**). In the Design Element list, each pin is referred to as a cluster, since each I/O is made up of a number of hard resources. You can filter the Design Element list to show the clusters as pins using the **I/Os** filter above the list. You can expand any of the clusters to see their resources, but you don't need to go deeper into the hierarchy to make the following adjustments to the floorplan.*

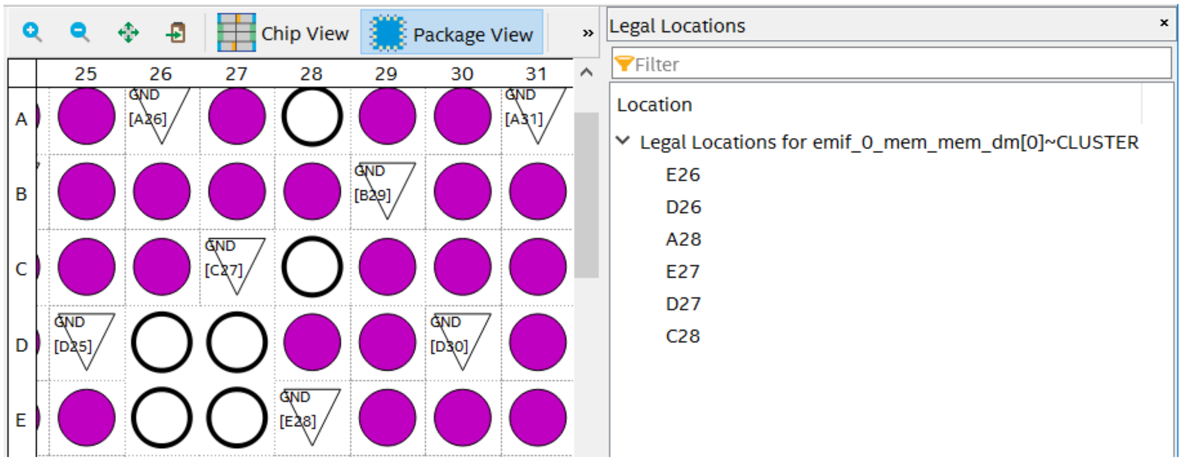
____ 3. Control-click any number of the **dq[x]** clusters to select them.

You can select as many of the 8 as you want, but selecting fewer will save some time and make the changes we're about to make more obvious. Notice again that hovering over these design elements highlights the assigned pin in the Package View.

____ 4. Right-click the selected clusters and select **Zoom to Selected**. Right-click them again and select **Unplace Selected**.

The clusters now have a placement status of unplaced and you can see the assignments removed from the Package View.

____ 5. De-select the now unplaced clusters. Click the arrows » in the Legal Locations column for one of the unplaced clusters.



Location

Legal Locations for emif_0_mem_mem_dm[0]~CLUSTER

- E26
- D26
- A28
- E27
- D27
- C28

Just like when placing interfaces in device resources, the Legal Locations window displays all the legal pin locations for the unplaced DQ clusters.

____ 6. Double-click a location to assign the selected cluster to that location.

Again, you can try using drag and drop instead of double-clicking legal locations if you'd like.

____ 7. Continue re-placing the rest of the unplaced clusters.

Depending on what clusters you unplaced and where they are located, you may see the list of legal locations change as you display them for each cluster. Once again, Interface Planner updates the legal locations based on your previous selections.

*Don't worry about making mistakes in Interface Planner. You can undo any assignments you create using the **Placement History** in the lower-right corner of the Interface Planner window. You can also start creating the plan all over again by clicking **Unplace All** and quickly place all design elements to legal locations by clicking **Autoplace All**, both found at the top of the Interface Planner window.*

____ 8. When you are satisfied with the floorplan, save it as a **.plan** file by clicking **Save Floorplan** at the top of the Interface Planner window or by selecting it from the **File** menu. Choose any name you'd like for the file.

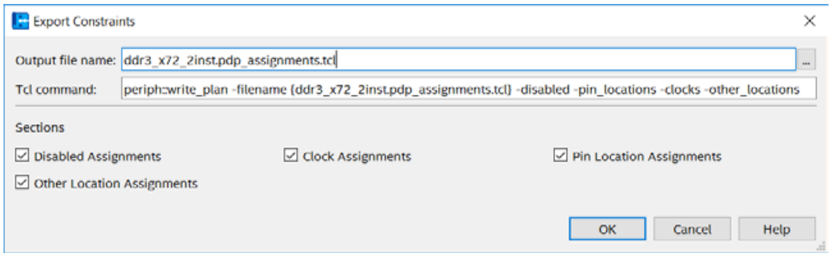
*Creating a **.plan** file saves your floorplan so you can continue working on it later. It does **not** bring your plan assignments into the Intel Quartus Prime project. We'll do that next.*

Step 4: Verify plan and transfer assignments to the project

____ 1. In the **Plan** menu or the **Flow** window, select **Validate Plan**.

*The plan should pass validation quickly. If there are any issues, check the **Console** window for information and fix the issues. If you are running out of time, simply **Unplace All** and then **Autoplace All** (followed by **Validate Plan**) to quickly create a valid plan.*

____ 2. In the **Plan** menu or the **Flow** window, select **Export Constraints**.



Export Constraints

Output file name: ddr3_x72_2inst.pdp_assignments.tcl

Tcl command: periph::write_plan -filename {ddr3_x72_2inst.pdp_assignments.tcl} -disabled -pin_locations -clocks -other_locations

Sections

☑ Disabled Assignments ☑ Clock Assignments ☑ Pin Location Assignments

☑ Other Location Assignments

____ 3. Name the **Output file name** whatever you'd like and click **OK**. Click **OK** again when the tool indicates the export was successful.

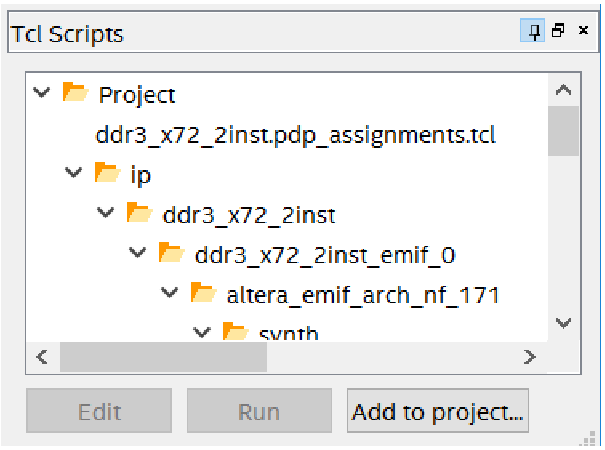
*Optionally, you could choose to only include certain types of Interface Planner-created assignments in the output **.tcl** file using the checkboxes here, but we're just going to use all the assignments.*

____ 4. Close Interface Planner. You already saved a **.plan** file, so select **Discard and Close** in the **Close Project** dialog.

____ 5. Reopen the Intel Quartus Prime Pro edition software and reopen the project, **ddr3_x72_2inst.qpf**. It should be listed right at the top of the **Recent Projects** list on the **Home** tab.

____ 6. From the **Tools** menu, select **Tcl Scripts**.

____ 7. Select the script you exported.



The script is full commented, including instructions for executing it from a command line.

____ 8. Click **Run**. Click **OK** when the script finishes.

____ 9. From the Assignments menu, open the Pin Planner and the Assignment Editor.

*Notice that the locations selected in Interface Planner now appear in the Pin Planner. They also appear in the Assignment Editor with a tag (**Tag** column) indicating that the assignments were created using Interface Planner. You would now continue the Intel Quartus flow: perform a full compilation and timing analysis, generate a programming file, and program the design into an Intel Arria 10 device.*

You can see just how easy it is to create these assignments using Interface Planner.

The following steps are optional, but you can follow them if you have time and want to see how existing project assignments affect Interface Planner usage.

____ 10. *(Optional; time permitting)* Reopen and re-initialize Interface Planner.

____ 11. Switch to the **Assignments** tab.

*Notice that all the assignments created in Interface Planner and incorporated into the project's **.qsf** file through the Tcl script now appear here.*

____ 12. Experiment with disabling some of the assignments and continuing through the Interface Planner flow to see how existing assignments affect Interface Planner's listing of legal locations.

Exercise Summary

- Followed the Interface Planner flow to create legal I/O assignments
- Created a Tcl script for those assignments
- Imported Interface Planner assignments back into the Intel Quartus project for storage in the project's **.qsf** file

END OF EXERCISE 5

Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.