XAPP789 (v1.0) July 31, 2012

7 Series FPGAs AXI Multi-Port Memory Controller Using the Vivado Tools

Author: Khang Dao, Sikta Pany, and Ricky Su

Summary

A multi-port memory controller (MPMC) is used in applications where multiple devices share a common memory controller. An MPMC is a common requirement in many video, embedded, and communications applications where data from multiple sources moves through a common memory device, typically DDR3 SDRAM.

This application note demonstrates how to create a basic DDR3 MPMC design using a 7 series FPGA and the Vivado™ Design Suite [Ref 1]. The MPMC is created by combining the Memory Interface Generator (MIG) core and the AXI Interconnect IP, both of which are provided in the Vivado tools.

AXI is a standardized IP interface protocol based on the Advanced Microcontroller Bus Architecture (AMBA® 4) specification [Ref 2]. This reference design uses the AXI4, AXI4-Lite, and AXI4-Stream interfaces as described in the AXI4 specification. These interfaces provide a common IP interface protocol framework for building the system.

Overview

The example design in this application note is a full working hardware system on the KC705 evaluation board in the Xilinx Kintex™-7 FPGA KC705 Evaluation Kit [Ref 3].

The design implements a basic video system where data from a video test pattern generator (TPG) loops in and out of memory multiple times before being sent to the high-definition multimedia interface (HDMI technology) connector on the board. The DDR3 memory acts as a multi-ported memory being shared by multiple video frame buffers. The video frame buffers are controlled by two AXI Video Direct Memory Access (AXI VDMA) cores. Each AXI VDMA takes AXI4-Stream data carrying video information and moves the data to or from memory over AXI4 interfaces. The AXI Interconnect acts as an arbitrated switch to multiplex AXI4 transactions to the shared memory controller, thus creating an AXI MPMC system. A clock generator block supplies clocks throughout the system, and AXI4-Lite master cores generate the necessary configuration commands to set up the AXI VDMA after reset. Figure 1 provides a block diagram of the system and illustrates the basic data flow in the system.

Note: In addition to the Vivado tools logic design flow project, an equivalent Xilinx Platform Studio (XPS) project is also provided for reference. The XPS tools delivered in the Vivado Design Suite: System Edition (EDK) might provide a higher level of automation than building the equivalent system using the Vivado tools logic design flow.

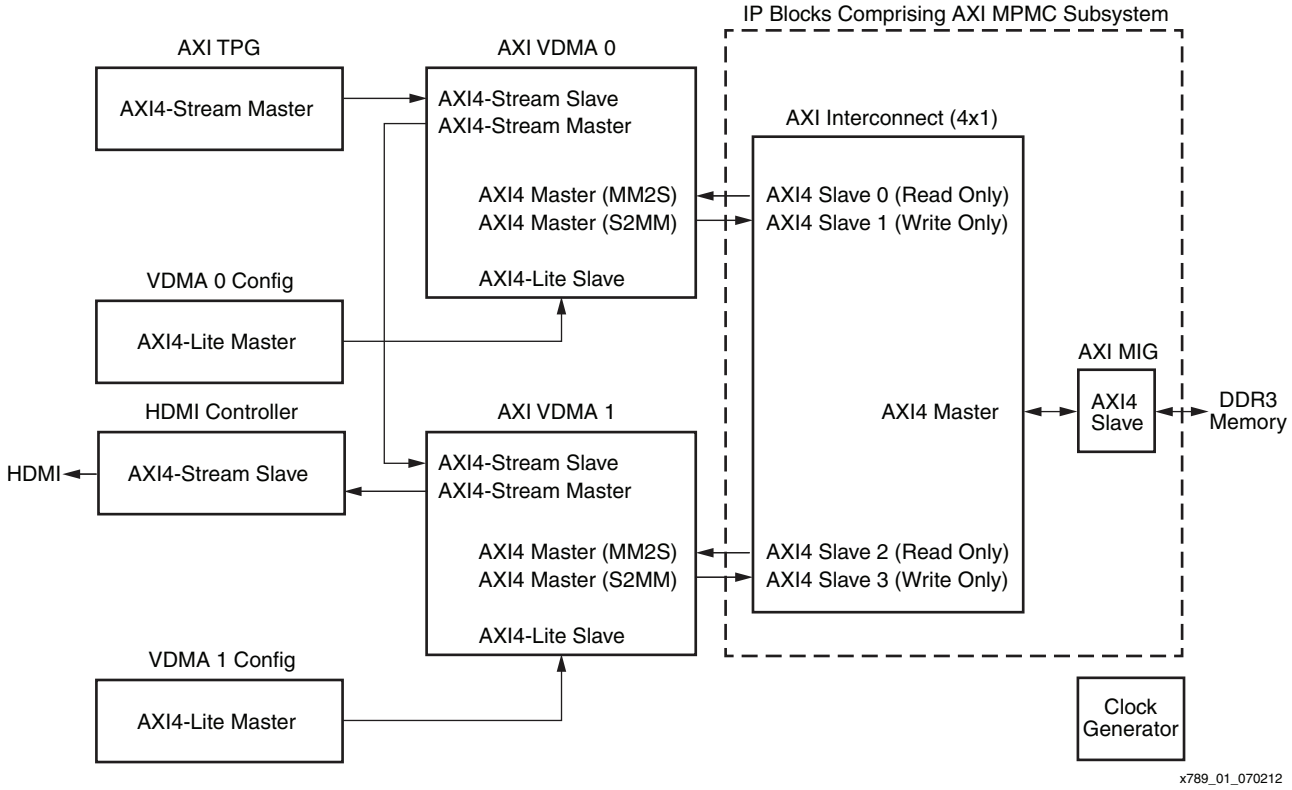


Figure 1: **Overview of the AXI MPMC System**

Quick Start

This section provides the steps to build the design starting from the complete project file set and also shows how to run the demonstration design on the KC705 board.

Note: Instructions to build the design from a new Vivado tools project are covered in Creating the AXI MPMC Design from a New Vivado Tools Project, page 11.

Steps to Open and Rebuild the Design

To open and rebuild the design:

1. Install the Vivado Design Suite 2012.2 (requires Logic Edition at a minimum).
2. Unzip the reference design files accompanying this application note into a local folder (referred to as `<design_dir>`).

To run the pre-generated bitstream in hardware without rebuilding the design, go to step 8. Otherwise, continue with these instructions to rebuild the design:

3. Open the Vivado tools in Windows by selecting **Start > Xilinx Design Tools > Vivado 2012.2 > Vivado** or enter the **`vivado`** command in Linux after setting up the Xilinx tools.

4. In the Vivado tools, select **Getting Started > Open Project** (Figure 2).

Figure 2: **Vivado Tools Getting Started Window (Open Project)**

5. Select `<design_dir>/vivado/project_1.xpr` and click **OK** (Figure 3).

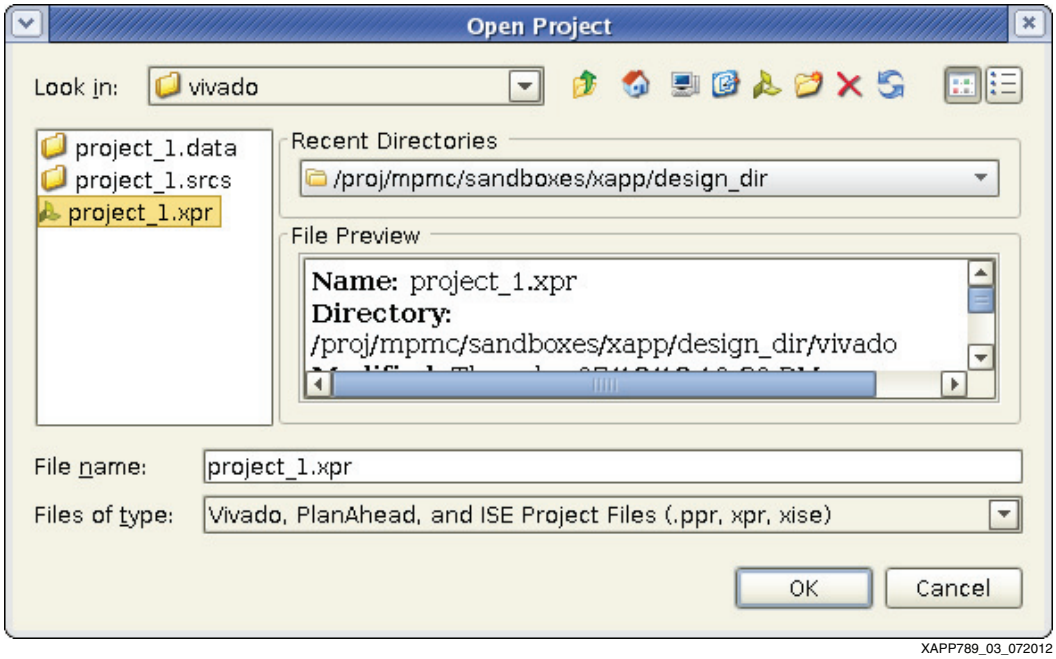


Figure 3: **Open Project Window**

6. Select **Flow > Generate Bitstream** (Figure 4) or press the icon next to **Program and Debug > Generate Bitstream** in the Flow Navigator window pane (Figure 5).

   Note: If a dialog is displayed asking to run Synthesis and Implementation, click **Yes**. The process to run synthesis and implementation on this design can take one hour or longer to complete.

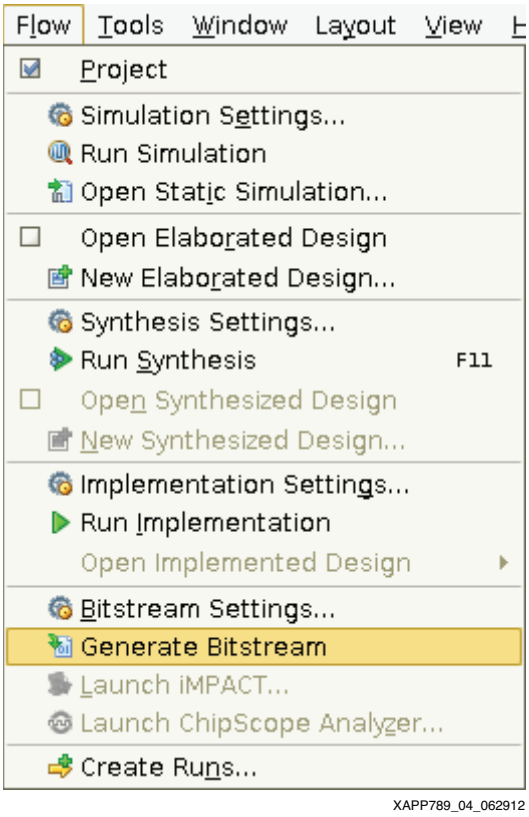


Figure 4: **Starting Bitstream Generation Using Menu Options**

Figure 5: **Starting Bitstream Generation Using Icon**

7. Synthesis, implementation, and bitstream generation operations are performed on the design. When the message “Bitstream Generation Completed” appears, click **OK** (Figure 6).

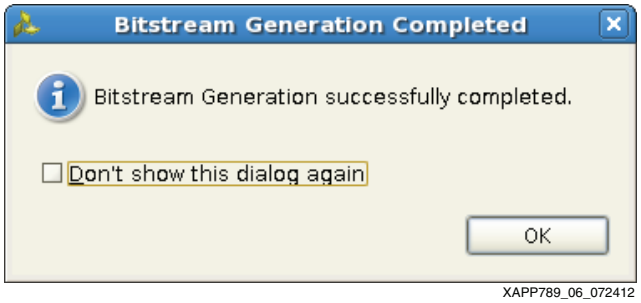


Figure 6: **Bitstream Generation Completed Dialog**

KC705 Board Setup

8. The reference design runs on the KC705 board (Rev D), as shown in Figure 7.

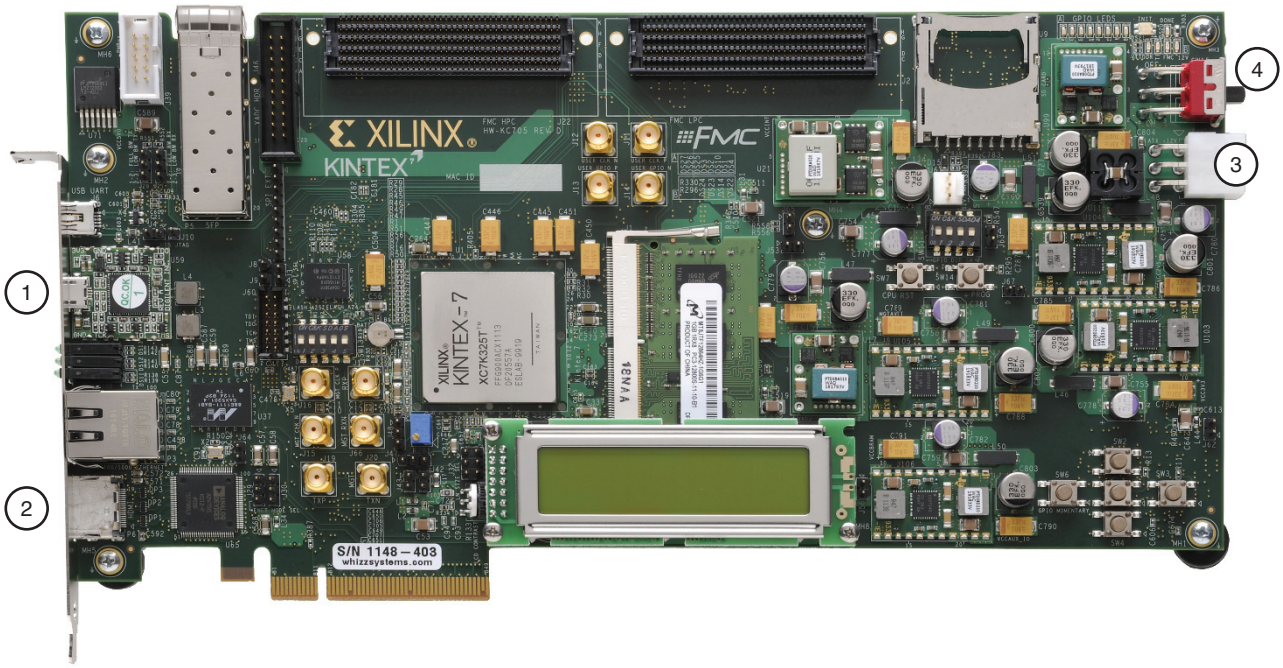


Figure 7: **KC705 Board Photo**

9. Connect the USB cable (provided with the board) from the host PC to the USB JTAG port (1 in Figure 7) of the KC705 board. Ensure that the appropriate device drivers are installed.
10. Connect the KC705 board's HDMI connector to a video monitor capable of displaying a 1280 x 720p, 60 Hz video signal (2 in Figure 7).
11. Connect the power supply cable to the KC705 board (3 in Figure 7).
12. Turn on the power to the KC705 board (4 in Figure 7).

Download and Run Bitstream

If the design was not rebuilt and the KC705 board is to be programmed using the pre-generated bitstream included in the application note, perform step 13. If the design has already been rebuilt in the Vivado tools and the project is open, go to step 14.

13. Open iMPACT directly by selecting **Start > Xilinx Design Tools > ISE Design Suite 14.2 > ISE Design Tools > (32 or64)-bit Tools > iMPACT** or enter `impact` at the command line in Linux. Go to step 15.
14. Run iMPACT by selecting **Flow > Launch iMPACT** from the Vivado tools menu bar (Figure 8) or click the icon next to **Program and Debug > Launch iMPACT** (Figure 9).

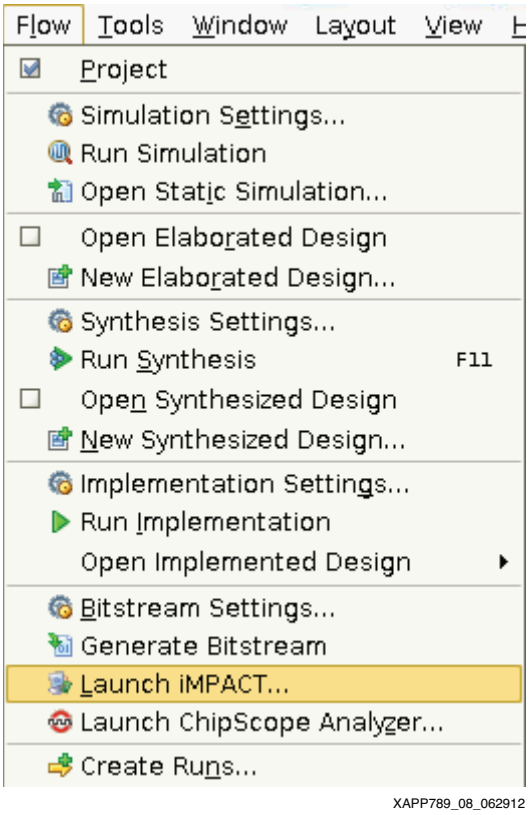


Figure 8: **Running iMPACT Using Menu Options**

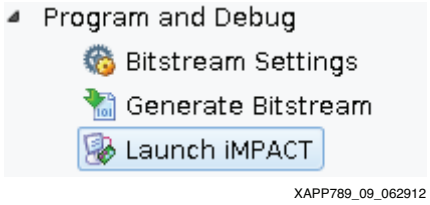


Figure 9: **Running iMPACT Using Icons**

15. When the iMPACT tool opens, select **Edit > Launch Wizard** (Figure 10).

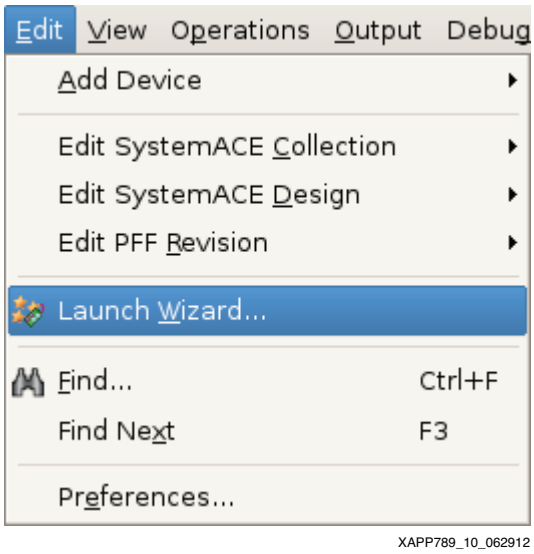


Figure 10: **Launching iMPACT Wizard**

16. In the Welcome to iMPACT window, select **Configure devices using Boundary-Scan (JTAG)** and click **OK** (Figure 11).

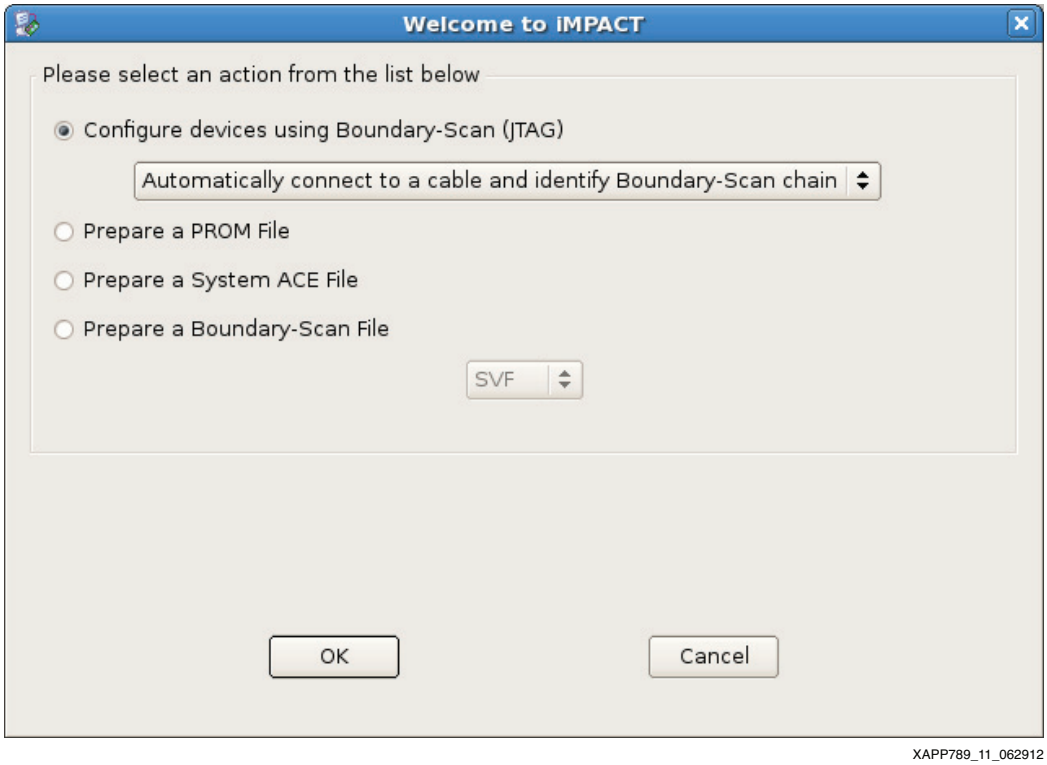


Figure 11: **Welcome to iMPACT Dialog**

17. In the Auto Assign Configuration Files Query Dialog window, click **Yes** (Figure 12).

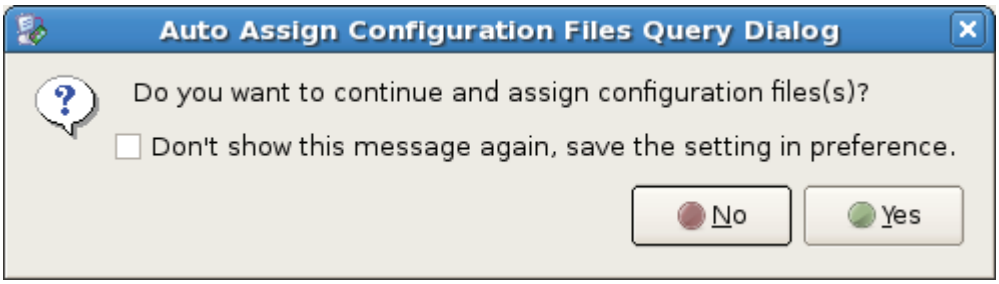


Figure 12: **Auto Assign Configuration Files Query Dialog**

18. If the design was rebuilt in the Vivado tools, browse to `<design_dir>/vivado/project_1.runs/impl_1, select system.bit`, and click **Open** (Figure 13).

    Note: If the design was not rebuilt using the Vivado tools, download the pre-generated bitstream file by opening `<design_dir>/ready_to_download/system.bit`.

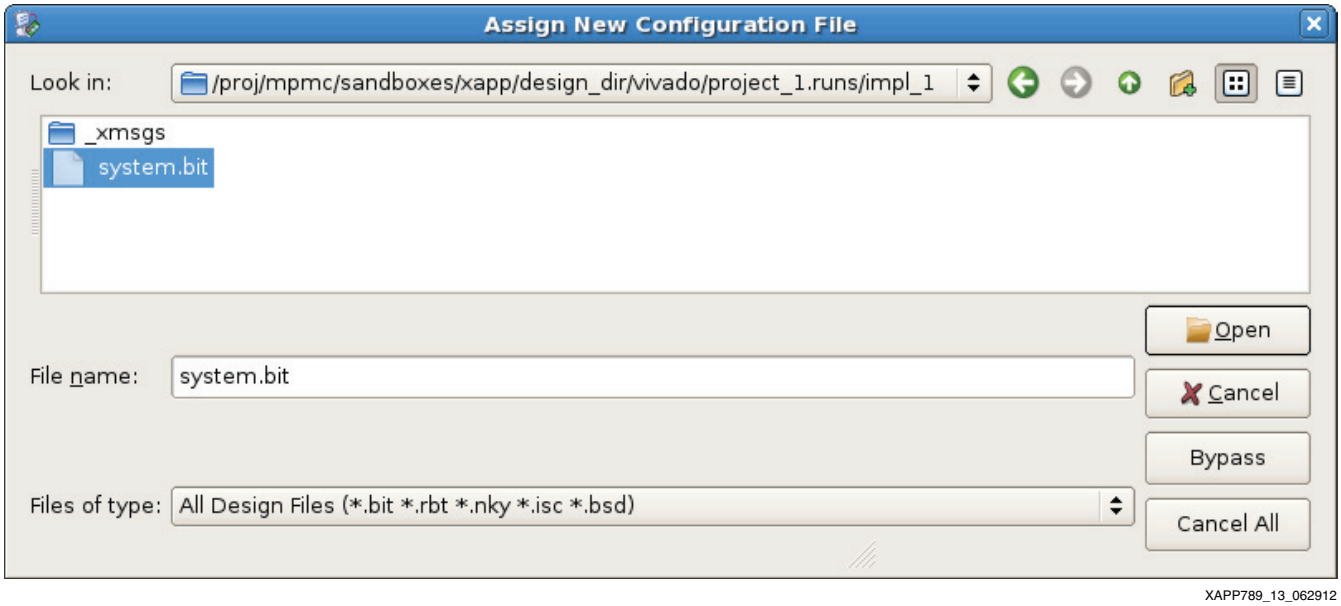


Figure 13: **Assign New Configuration File Dialog**

19. For SPI or BPI PROM options, click **No** (Figure 14).

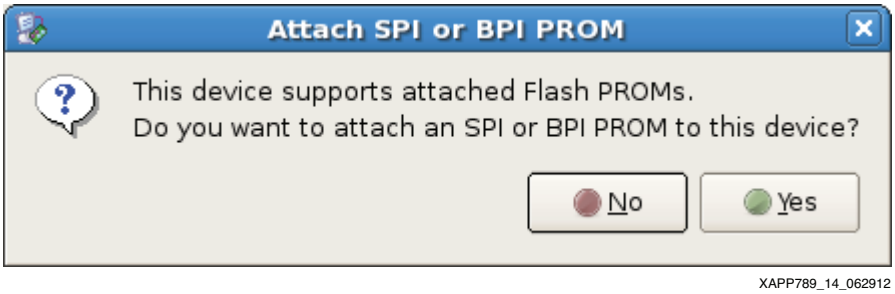


Figure 14: **Attach SPI or BPI PROM Dialog**

20. In the Device Programming Properties window, click **OK** (Figure 15).

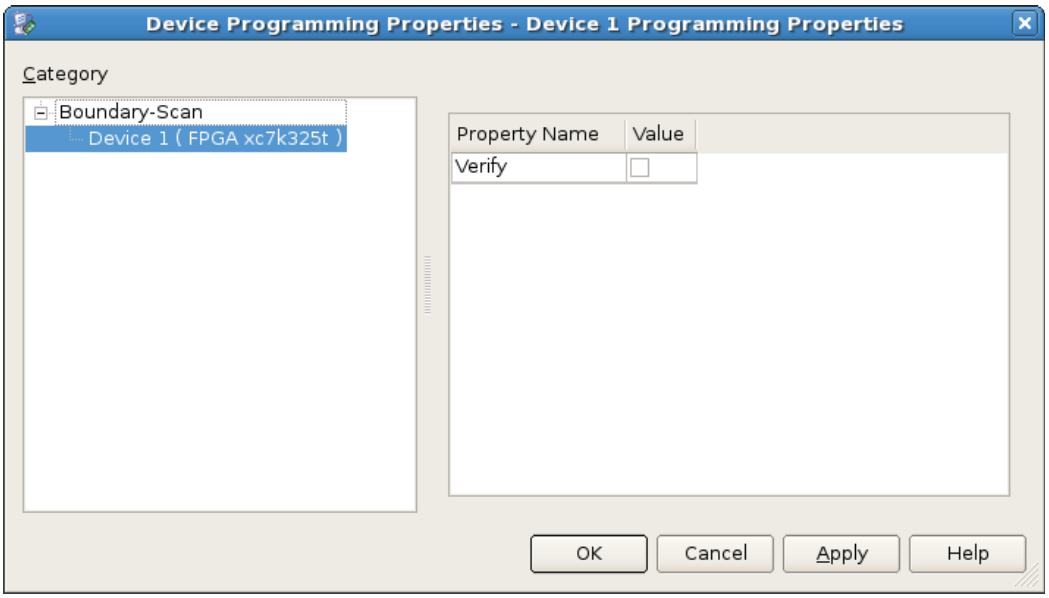


Figure 15: **Device Programming Properties Dialog**

Program the FPGA

21. Right-click the xc7k325t device and select **Program** (Figure 16).

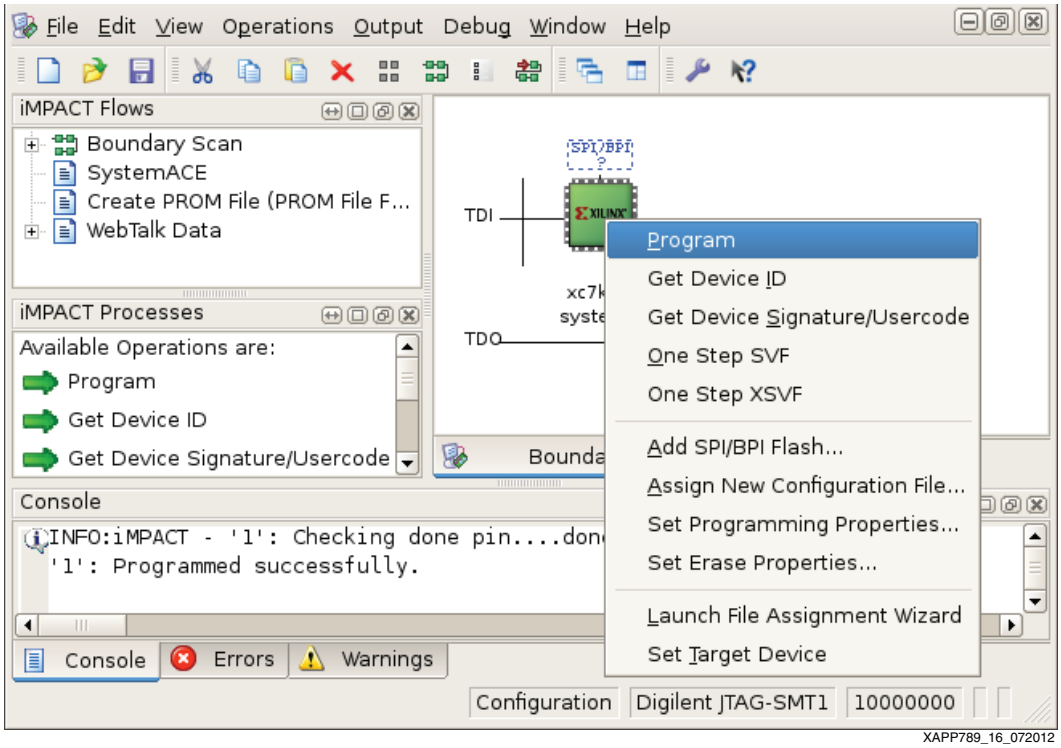


Figure 16: **Programming the FPGA**

After programming completes, the Program Succeeded message is displayed (Figure 17).

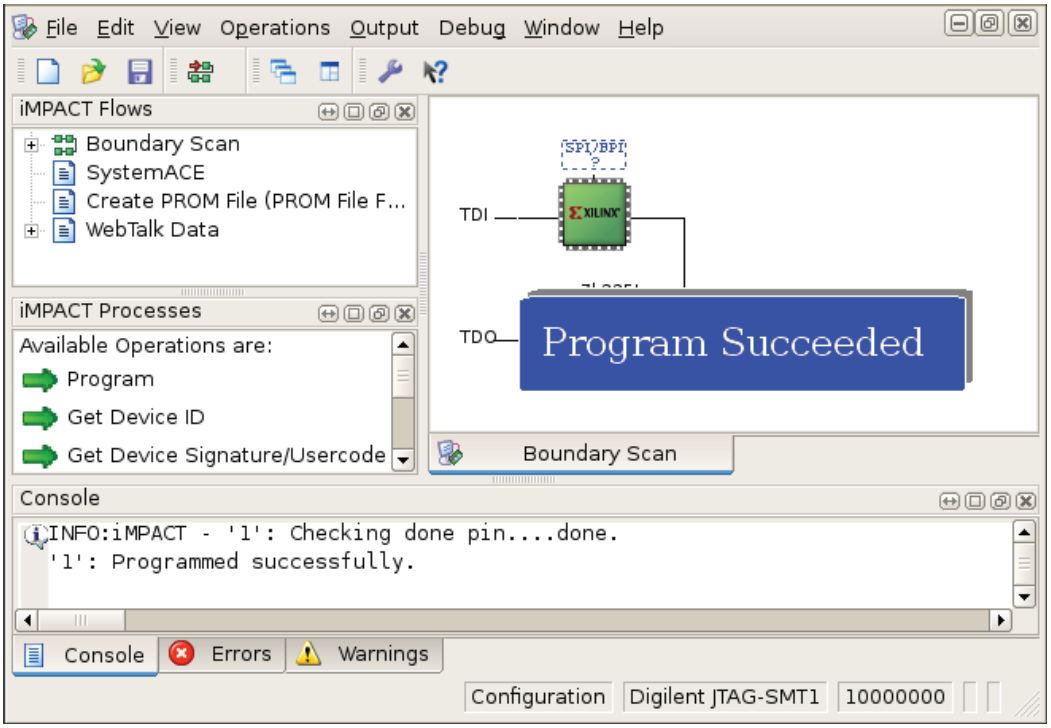


Figure 17: **Programming Succeeded Message**

After the design is downloaded, the video monitor shows a number of white circular ripple patterns that slowly move outward (Figure 18). This demonstrates a live AXI MPMC system in hardware moving multiple frames of video data through DDR3 memory controlled by two AXI VDMA IP blocks.

XAPP789_18_062912

Figure 18: **Video Image Displayed on Monitor When KC705 Board is Programmed**

Creating the AXI MPMC Design from a New Vivado Tools Project

This section provides the steps to build the design starting from a new Vivado tools project. It describes how to use the Vivado Design Suite to add or configure IP blocks to the project, connect them together into a system, and implement the design to a bitstream.

Note: Figures might show additional configuration or GUI options than those listed in the instructions. In these cases, leave the option at its default value unless instructed to change the setting.

Start a New Vivado Project and Set the Project Options

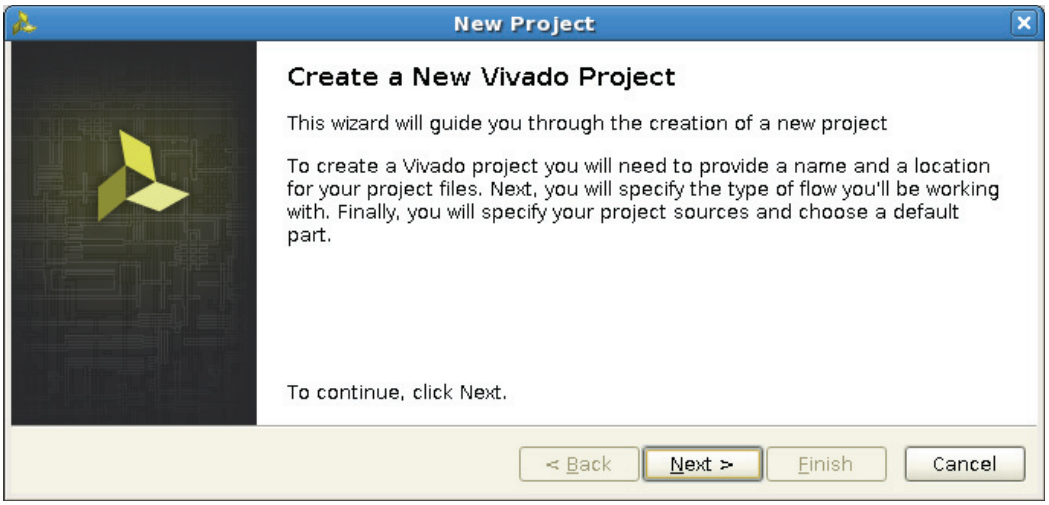
1. Install the Vivado Design Suite 2012.2 (requires Logic Edition at a minimum).
2. Unzip the reference design files into a local folder (referred to as `<design_dir>`).
3. Open the Vivado tools in Windows by selecting **Start > Xilinx Design Tools > Vivado 2012.2 > Vivado** or enter the **`vivado`** command in Linux after setting up the Xilinx tools.

4. Create a new project by selecting **Getting Started > Create New Project** (Figure 19).

XAPP789_02_072012

Figure 19: **Vivado Tools Getting Started Window (Create New Project)**

5. In the New Project window, click **Next** (Figure 20).



XAPP789_20_062912

Figure 20: **New Project Window**

6. Enter the project name **`project_1`** and select a directory for the project. The selected directory for the new project is referred to as `<user_dir>` (Figure 21).

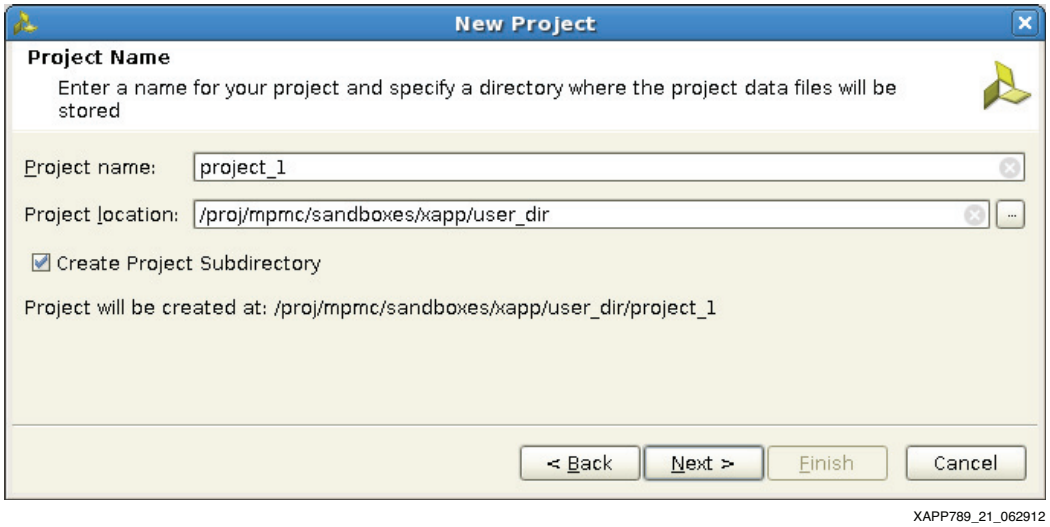


Figure 21: **New Project Window (Project Name)**

7. Select **RTL Project** in the Project Type window (Figure 22).

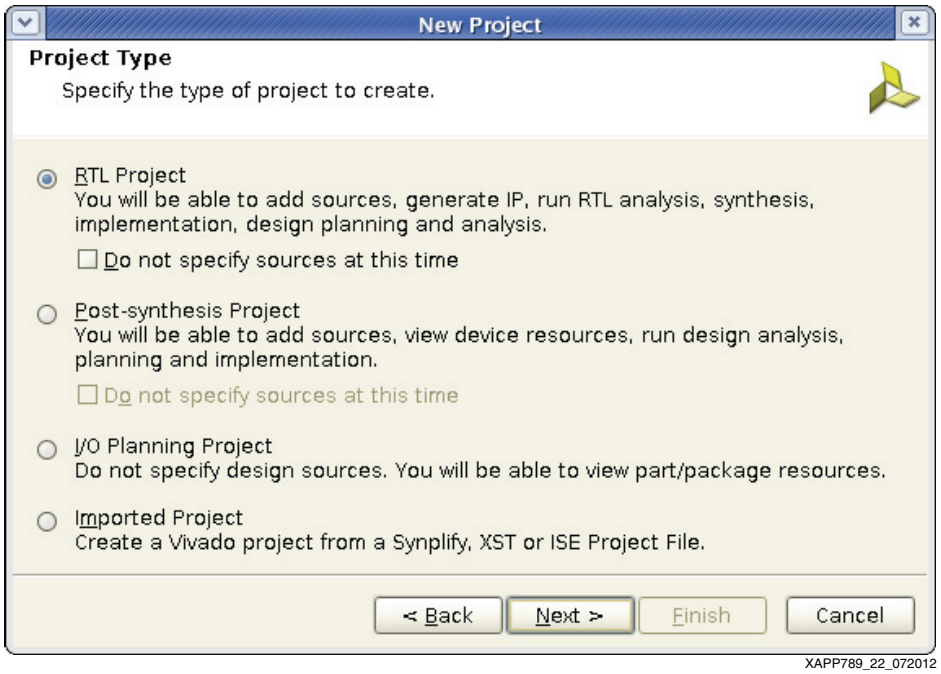


Figure 22: **New Project Window (Project Type)**

8. Click **Next** in the Add Sources window to create an empty project first (Figure 23).

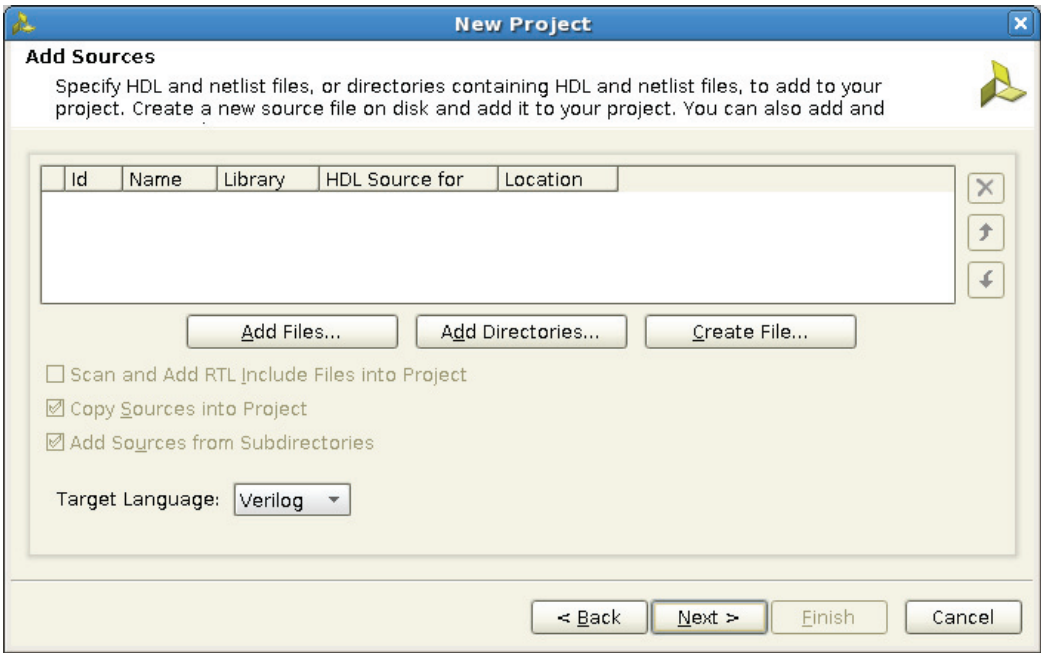


Figure 23: **New Project Window (Add Source)**

9. Click **Next** in the Add Existing IP window to create an empty project (Figure 24).

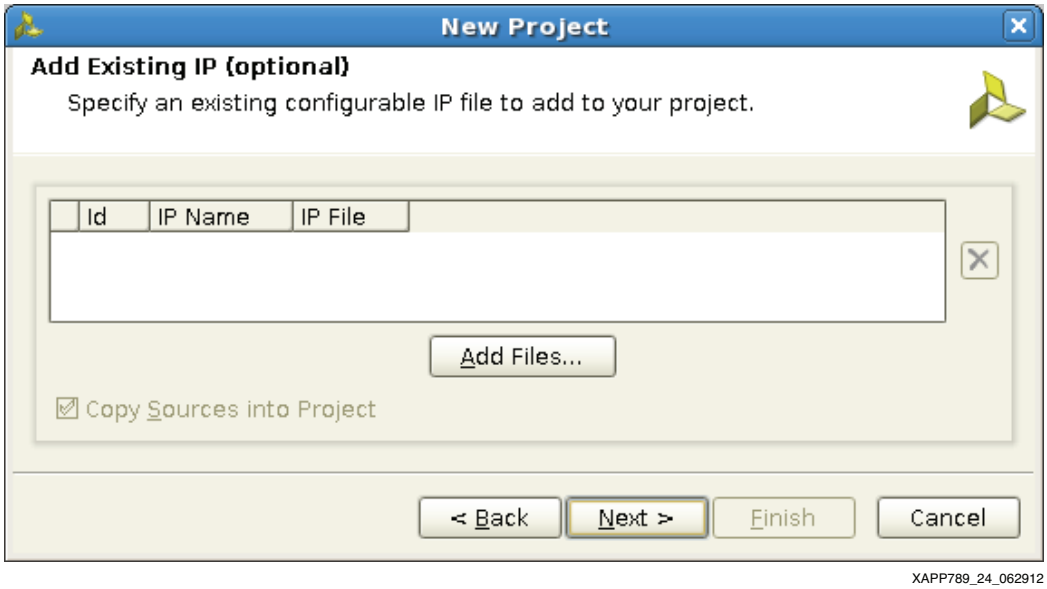


Figure 24: **New Project Window (Add Existing IP)**

10. Click **Next** in Add Constraints window (Figure 25).

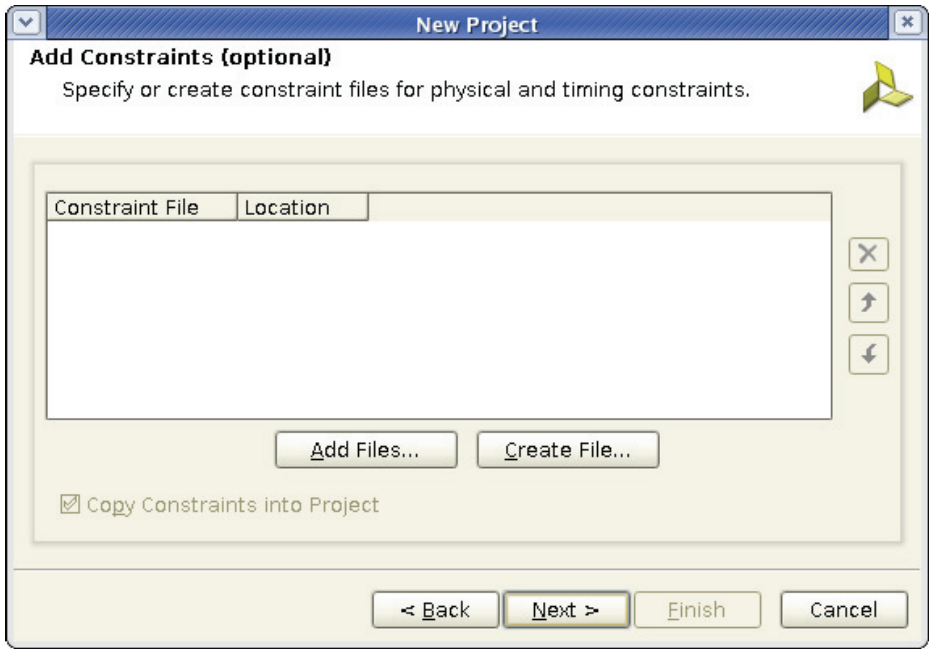


Figure 25: **New Project Window (Add Constraints)**

11. Select **xc7k325tffg900-2** in the Default Part window by entering the part number in the search text box and click **Next** (Figure 26). Alternatively, the part can be selected by choosing the Kintex-7 KC705 Evaluation Platform by selecting **Specify > Boards and Filter > Family > Kintex-7**.

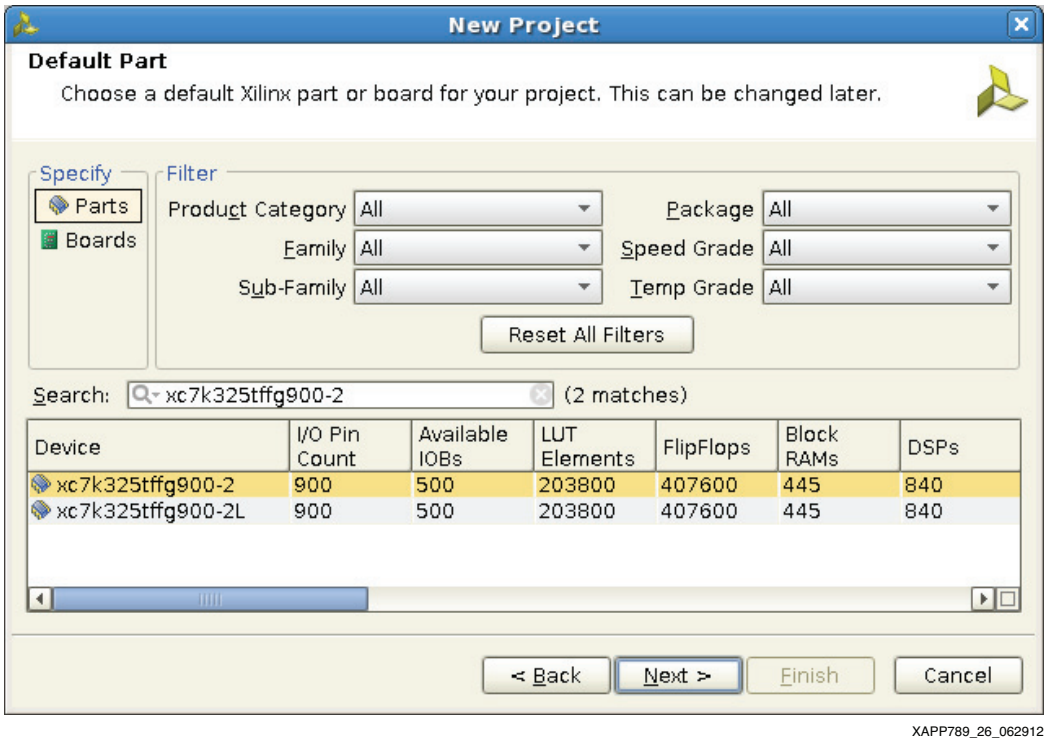


Figure 26: **New Project Window (Default Part)**

12. View the New Project Summary and click **Finish** (Figure 27).

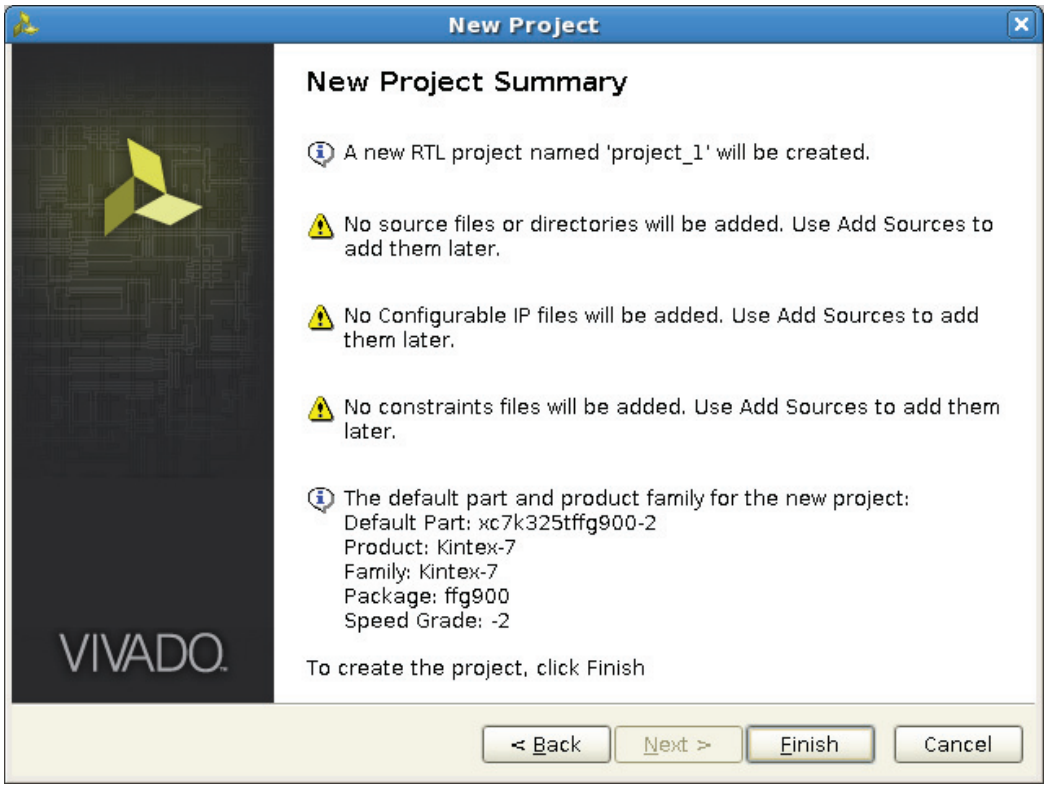


Figure 27: **New Project Window (New Project Summary)**

Adding the Memory Controller (MIG IP Core) to the Design

13. Select **Project Manager > IP Catalog** or click the **IP Catalog** button on the left side of the Flow Navigator window to bring up the IP Catalog.
14. From the IP Catalog tab, select **Memories & Storage Elements > Memory Interface Generators > MIG 7 Series (Version 1.6)**.
15. Right-click and select **Customize IP** to start the configuration for the memory controller (Figure 28). It might take a few minutes to launch the IP customization GUI.

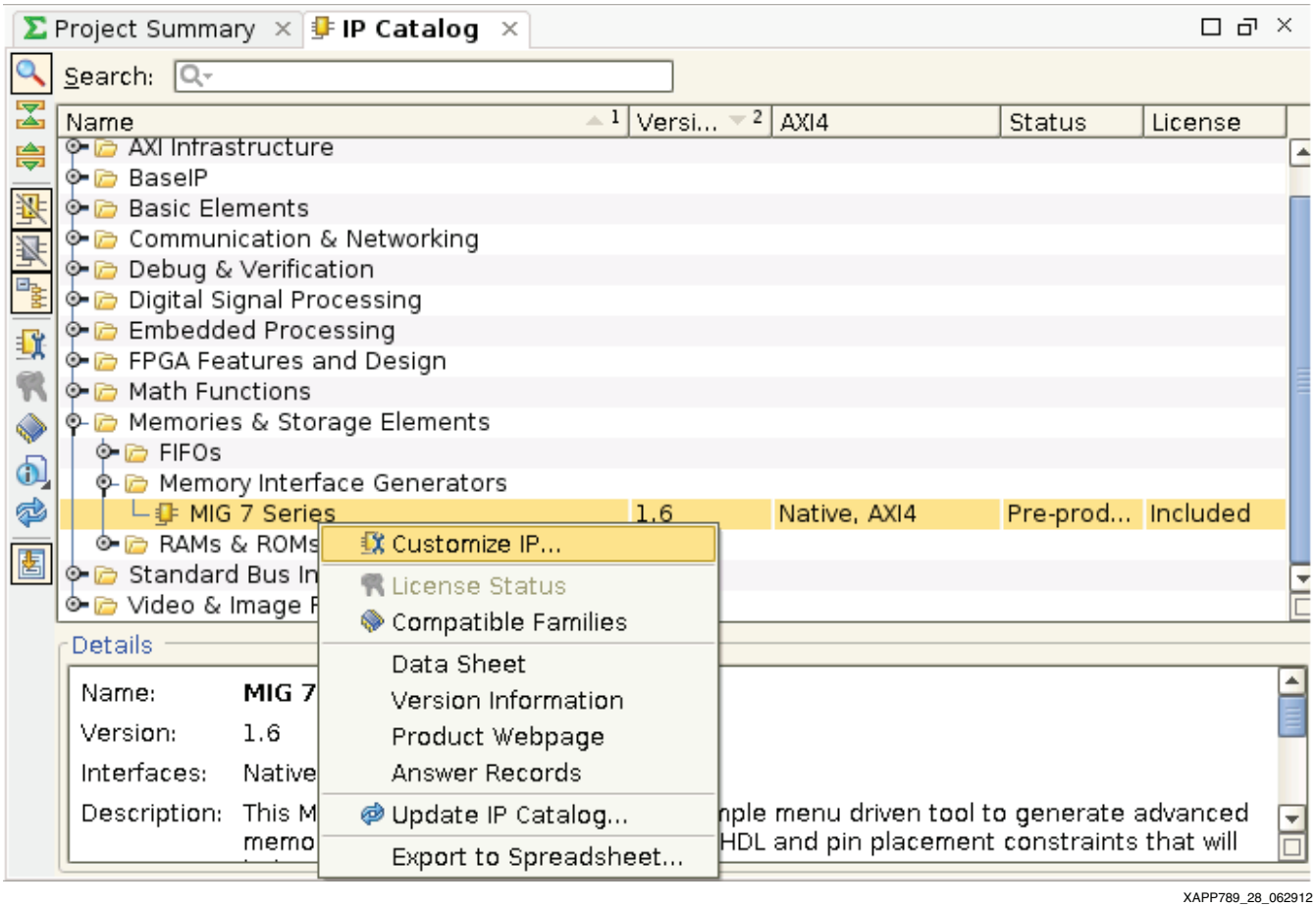


Figure 28: **IP Catalog Menu**

16. In the MIG GUI, review the core options and click **Next** (Figure 29).

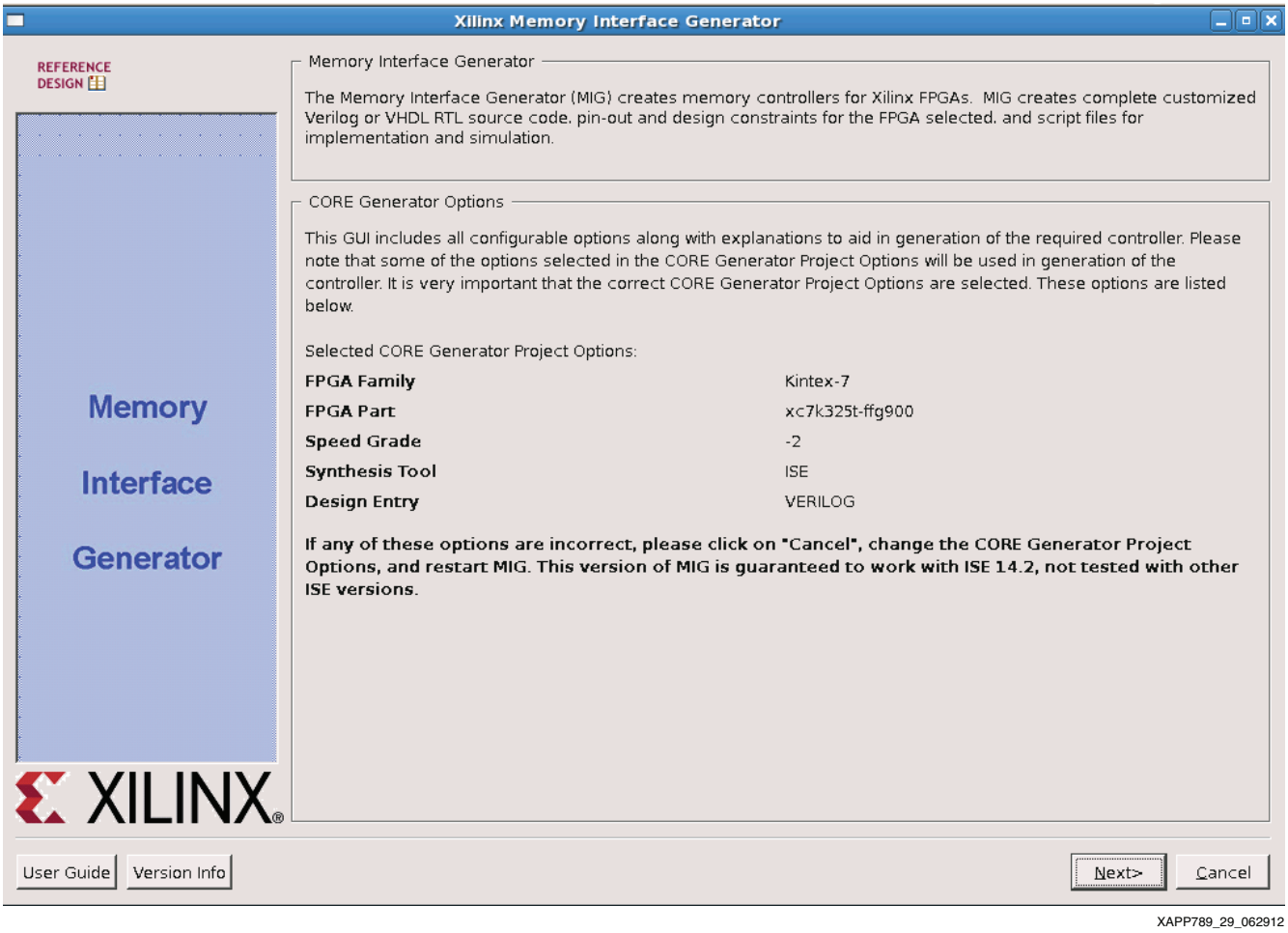
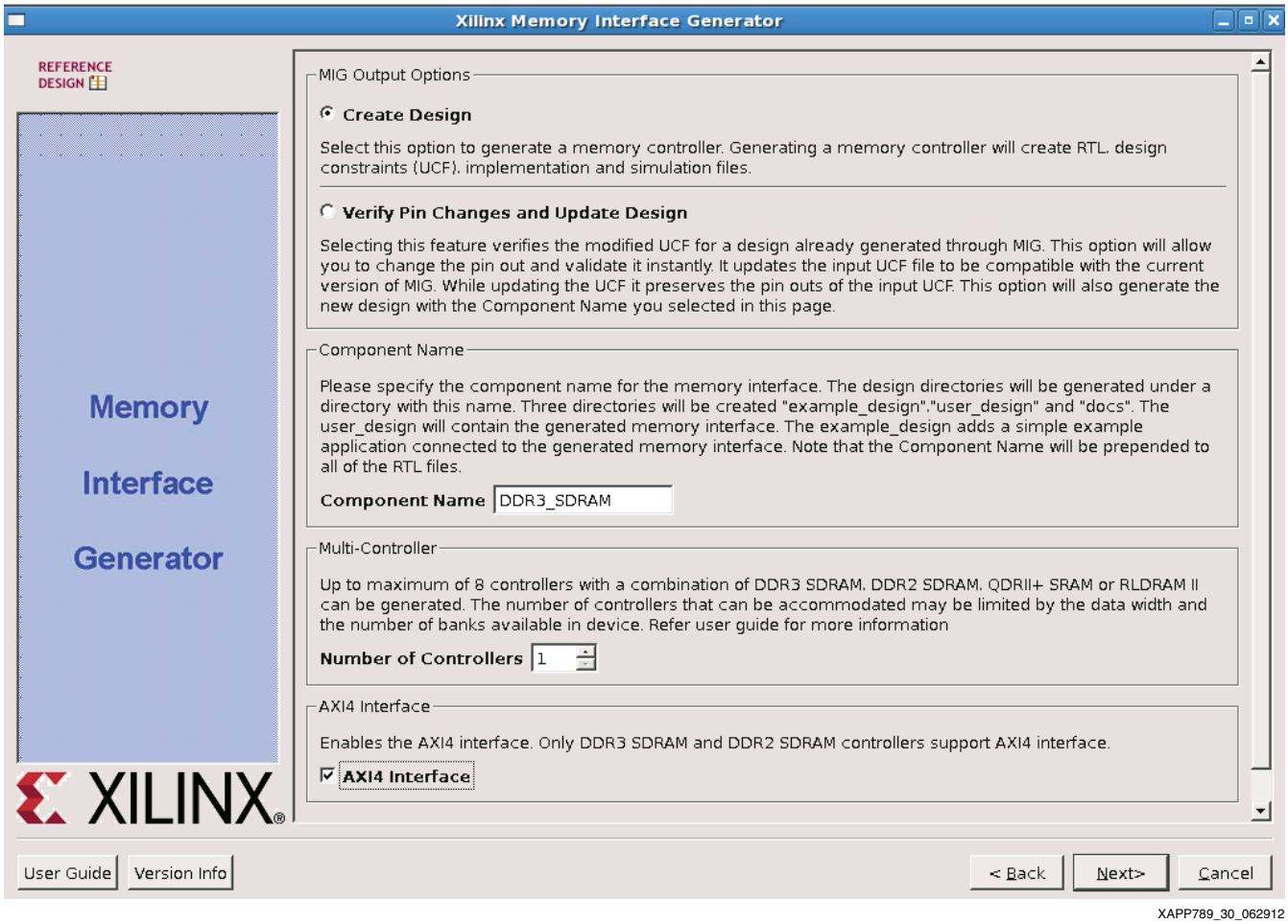


Figure 29: **Kintex-7 FPGA Memory Interface Generator (MIG) Front Page**

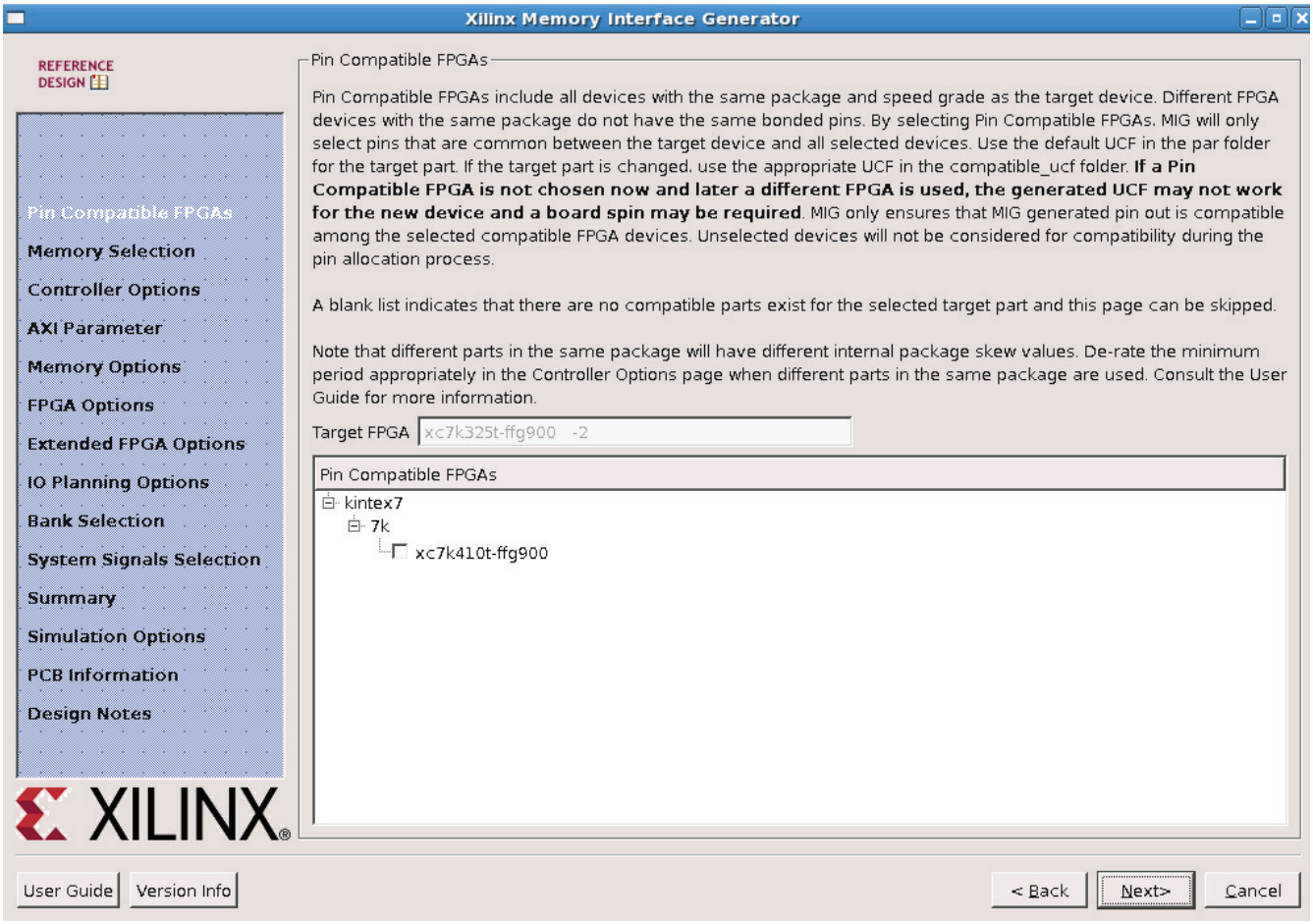
17. Select **Create Design** and specify the Component Name as **DDR3_SDRAM**. Enable the AXI Interface by checking **AXI4 Interface** and click **Next** (Figure 30).



XAPP789_30_062912

Figure 30: **MIG Output Options**

18. For Pin Compatible FPGAs settings, click **Next** (Figure 31). This page is normally used for a new board pinout definition that is desired to be compatible across multiple devices in the same package.



XAPP789_31_062912

Figure 31: **MIG Pin-Compatible Kintex-7 FPGAs**

19. For Memory Selection, select **DDR3 SDRAM** and click **Next** (Figure 32).

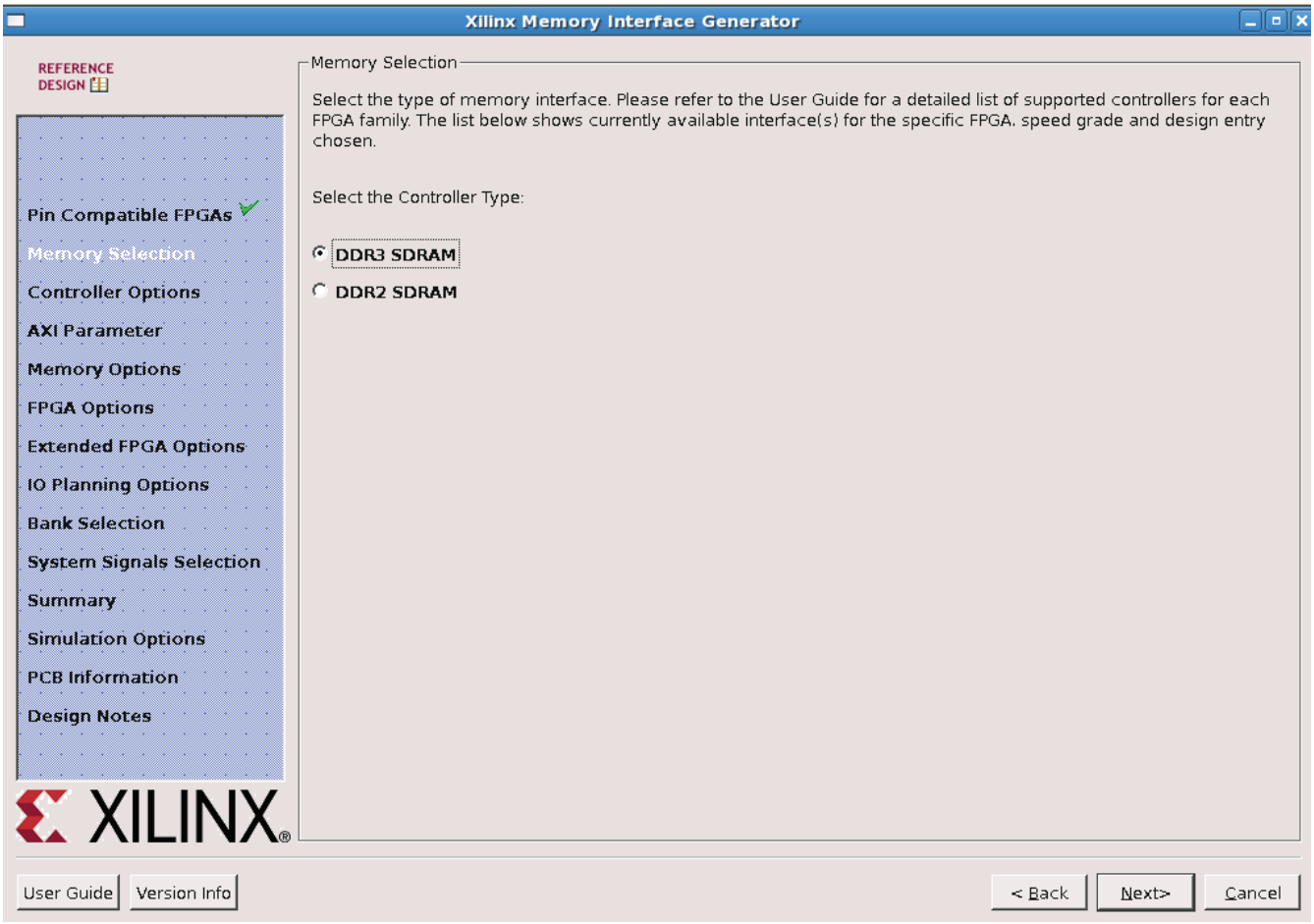
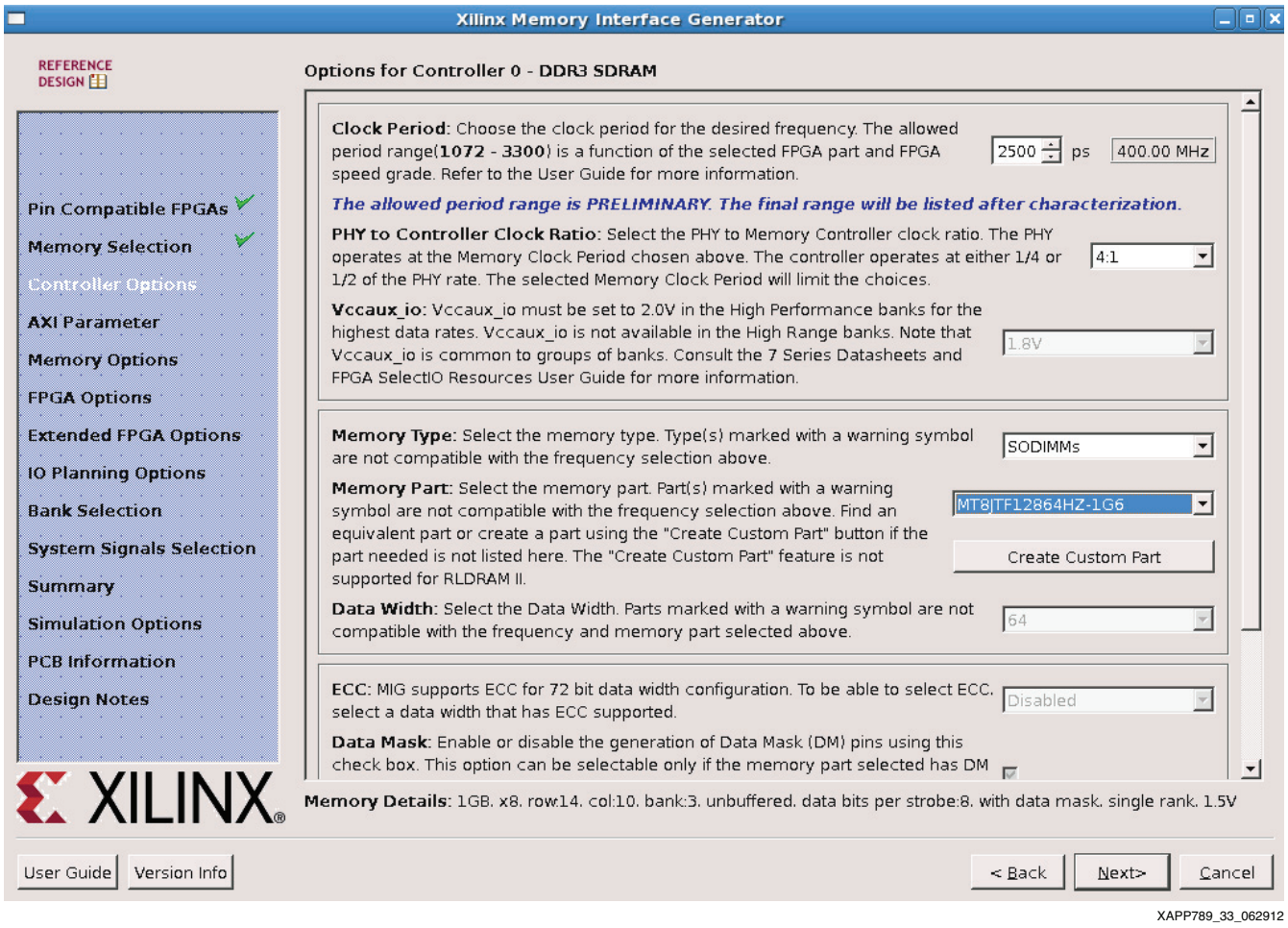


Figure 32: **MIG Memory Type and Controller Selection**

20. For Controller Options (Figure 33), set the clock frequency to **2500 ps** for a 400 MHz memory clock, which is a clock frequency supported for the device and speed grade of the FPGA on the KC705 board. Set the memory information to match the KC705 board:
    - Memory Type: **SODIMMs**
    - Memory Part: **MT8JTF12864HZ-1G6**
    - Data Width: **64**



XAPP789_33_062912

Figure 33: **MIG Controller Options Page**

21. Scroll down the page as needed to ensure that the Data Mask option is checked (default) and Ordering is set to **Normal** (default). Click **Next** (Figure 33).

22. For AXI Parameter settings, make the following selections, then click **Next** (Figure 34):
    - AXI Data Width: **512**.

      This sets the AXI Interface data width to match the native data width of the MIG data paths.
    - Narrow Burst Support: **0**.

      This disables support for AXI narrow width transactions to reduce area and latency. Narrow width transactions are generally unused by Xilinx IP, as described in the *Xilinx AXI Reference Guide*. [Ref 4].
    - AXI ID Width: **4**.

      This value is set depending on the network of masters connected to this memory controller. The value is generally a system-level parameter described in the ARM AXI4 documentation [Ref 4] and is determined by the AXI Interconnect or IP block connected to the memory controller. In this design, a setting of 4 matches the default configuration used by the AXI Interconnect. However, for a custom user system, this value might need to be adjusted to meet the requirements of the AXI system connected to the AXI Interconnect, especially if the number of devices or AXI Interconnect configuration is changed significantly.

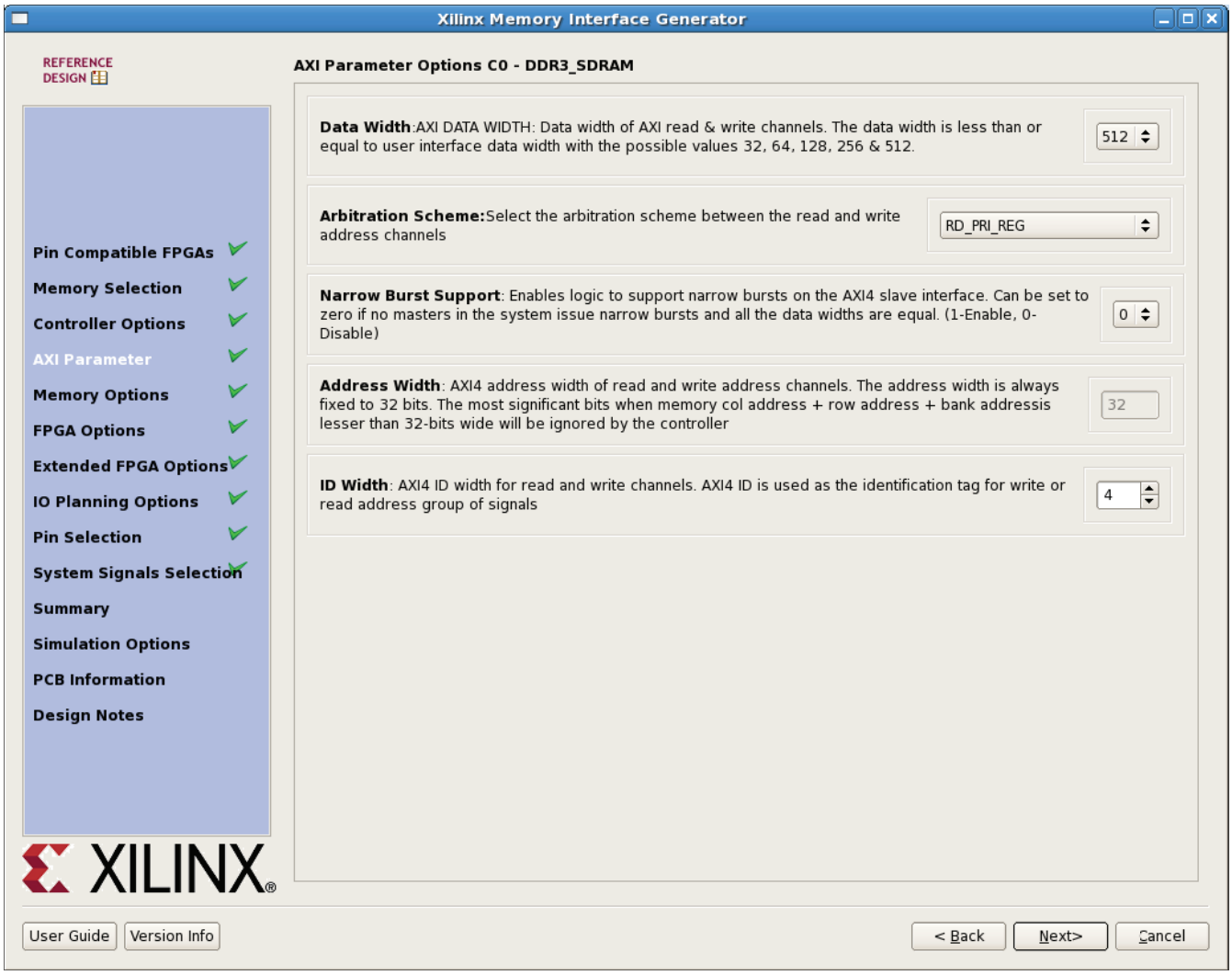


Figure 34: **MIG AXI Parameter Options**

23. For Memory Options, make the following selections, then click **Next** (Figure 35):
    - Input Clock Period: **5000 ps (200 MHz)**.

      This is the clock frequency available in the KC705 board.
    - Read Burst Type: **Sequential**.

      This is a recommended general setting given that the masters in the system are video devices generally using sequential bursts.
    - Output Driver Impedance Control: **RZQ/7**.

      This matches the design requirements of the KC705 board.
    - RTT (nominal) - On Die Termination (ODT): **RZQ/6**.

      This matches the design requirements of the KC705 board.
    - Memory Address Mapping Selection: **Bank/Row/Column**.

      Either the bank, row, or column settings can be used. Because the video devices in the system have long sequential access patterns, similar results are likely to be observed. The Bank/Row/Column mapping is chosen because, in theory, it can allow multiple masters to access different banks, potentially keeping them open longer.

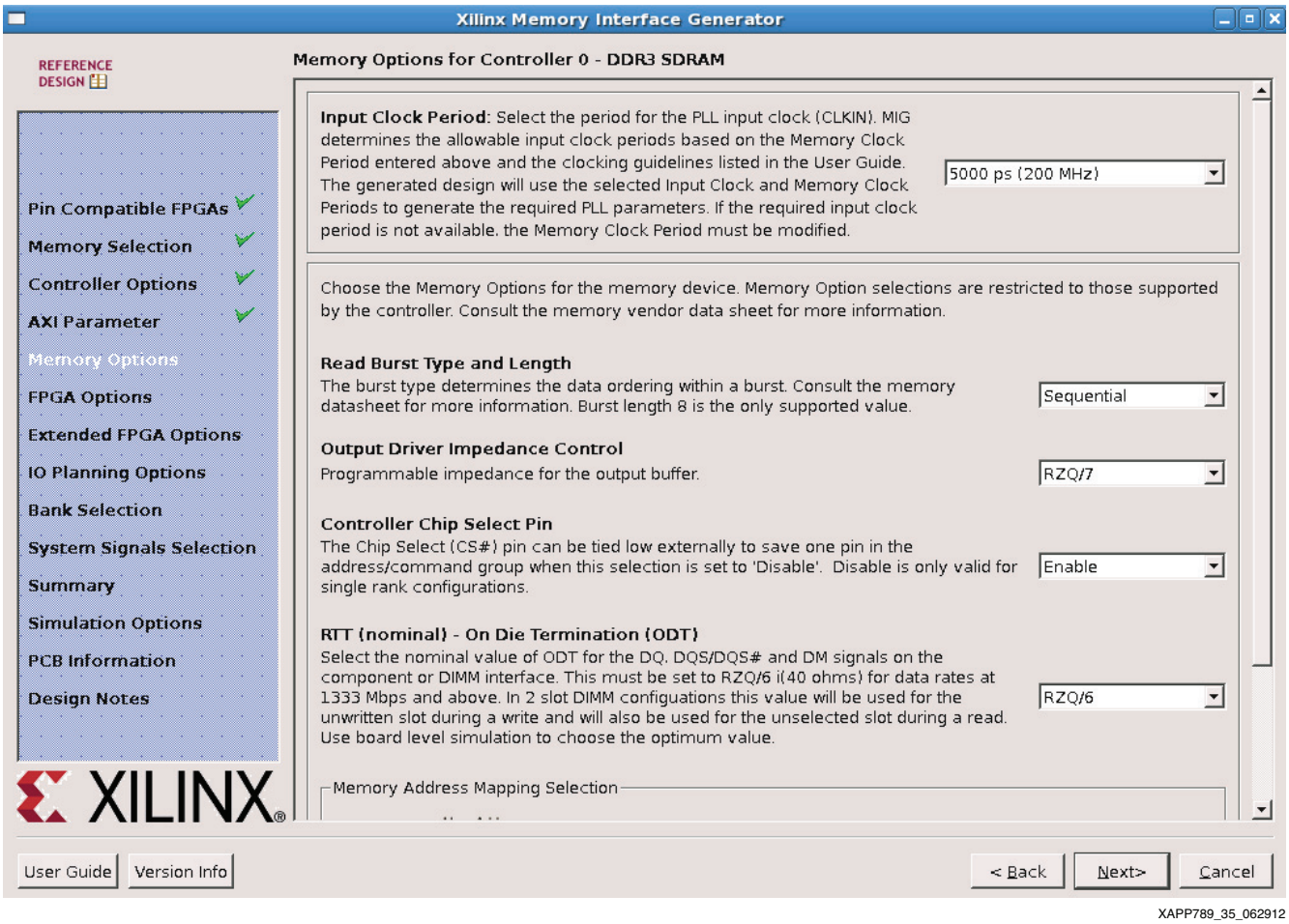


Figure 35: **MIG Memory Mode Options**

24. For FPGA Options, make the following selections, then click **Next** (Figure 36):
    - System Clock: **Differential**.

      This is the default setting.
    - Reference Clock: **Use System clock**.

      This allows the system clock to be shared as the reference clock source.
    - Debug Signals for Memory Controller: **OFF**.

      This is the default setting.
    - Internal Vref: (Unchecked).

      This is the default setting.
    - IO Power Reduction: **ON**.

      This is a new feature to reduce I/O power and is on by default. This matches the normal configuration of the KC705 board.

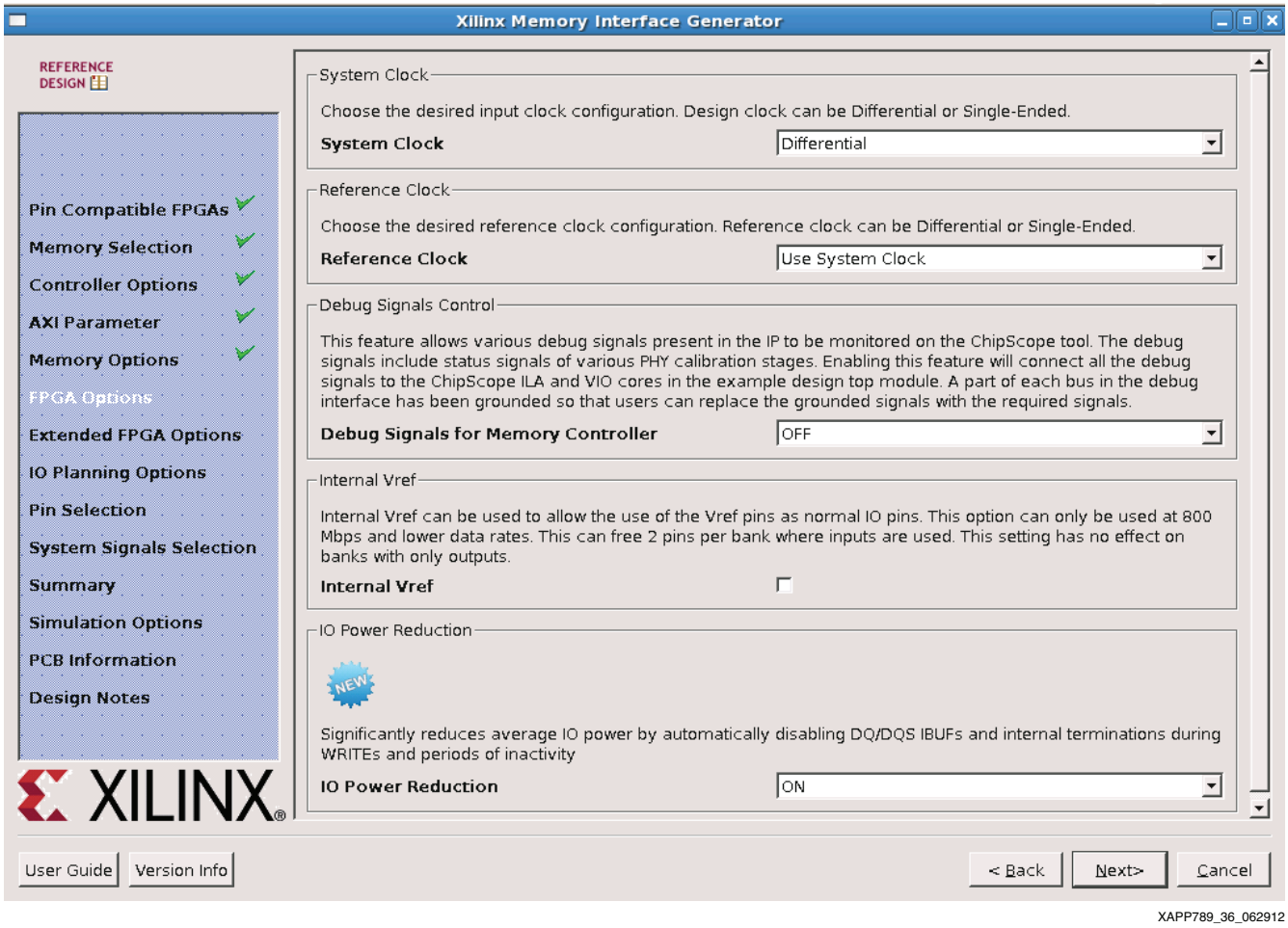


Figure 36: **MIG FPGA Options**

25. For Extended FPGA Options, set Internal Termination Impedance to **50 Ohms** and check the **DCI Cascade** option. This matches the requirement for the KC705 board. Click **Next** (Figure 37).

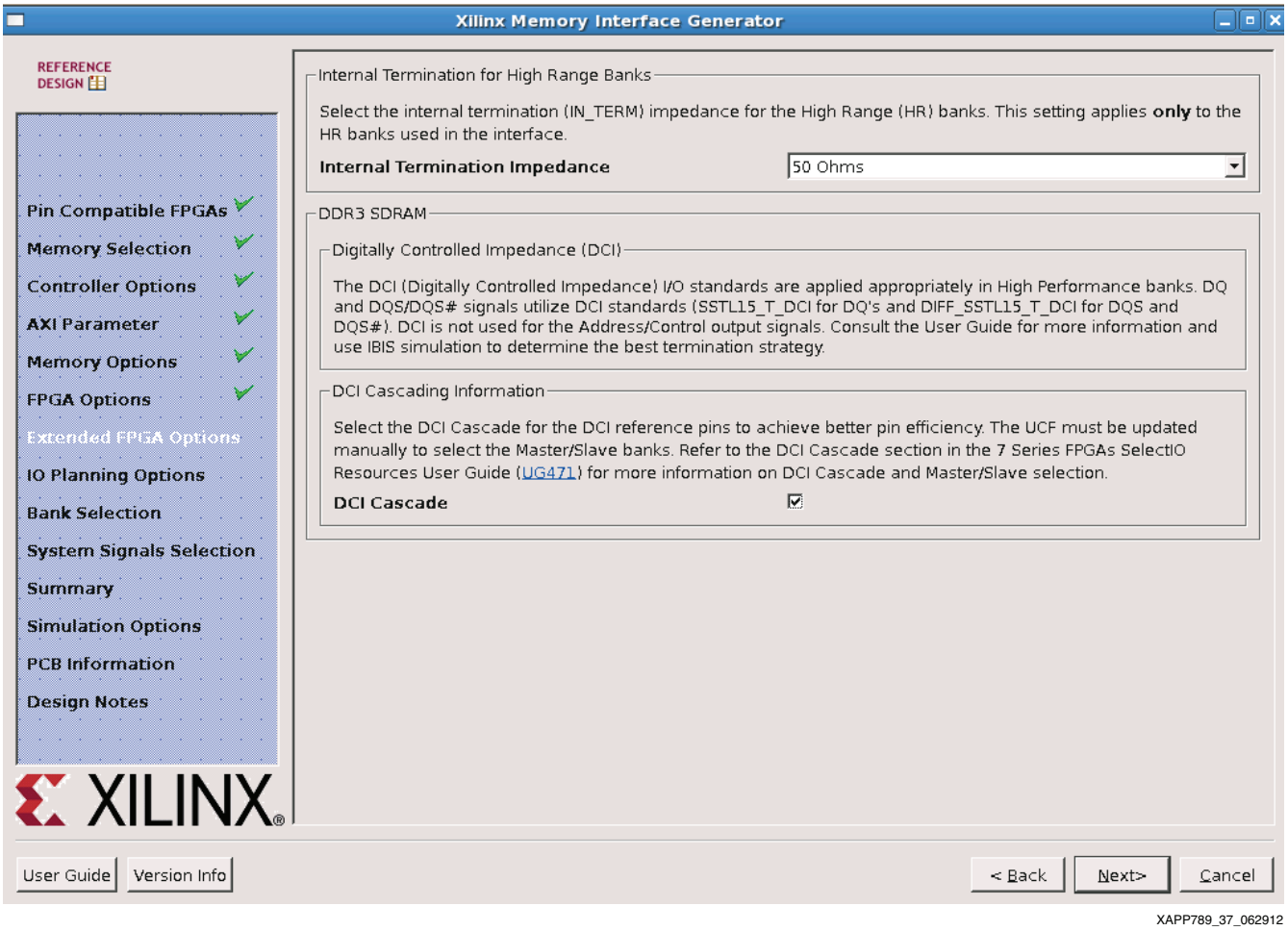


Figure 37: **MIG Extended FPGA Options**

26. For Pin/Bank Selection Mode, select **Fixed Pin Out** and click **Next** (Figure 38). This allows the user to enter predefined pinout information for an existing board like the KC705.

    Note: The other option is for a new board design in which the user desires to have the tool select a pinout.

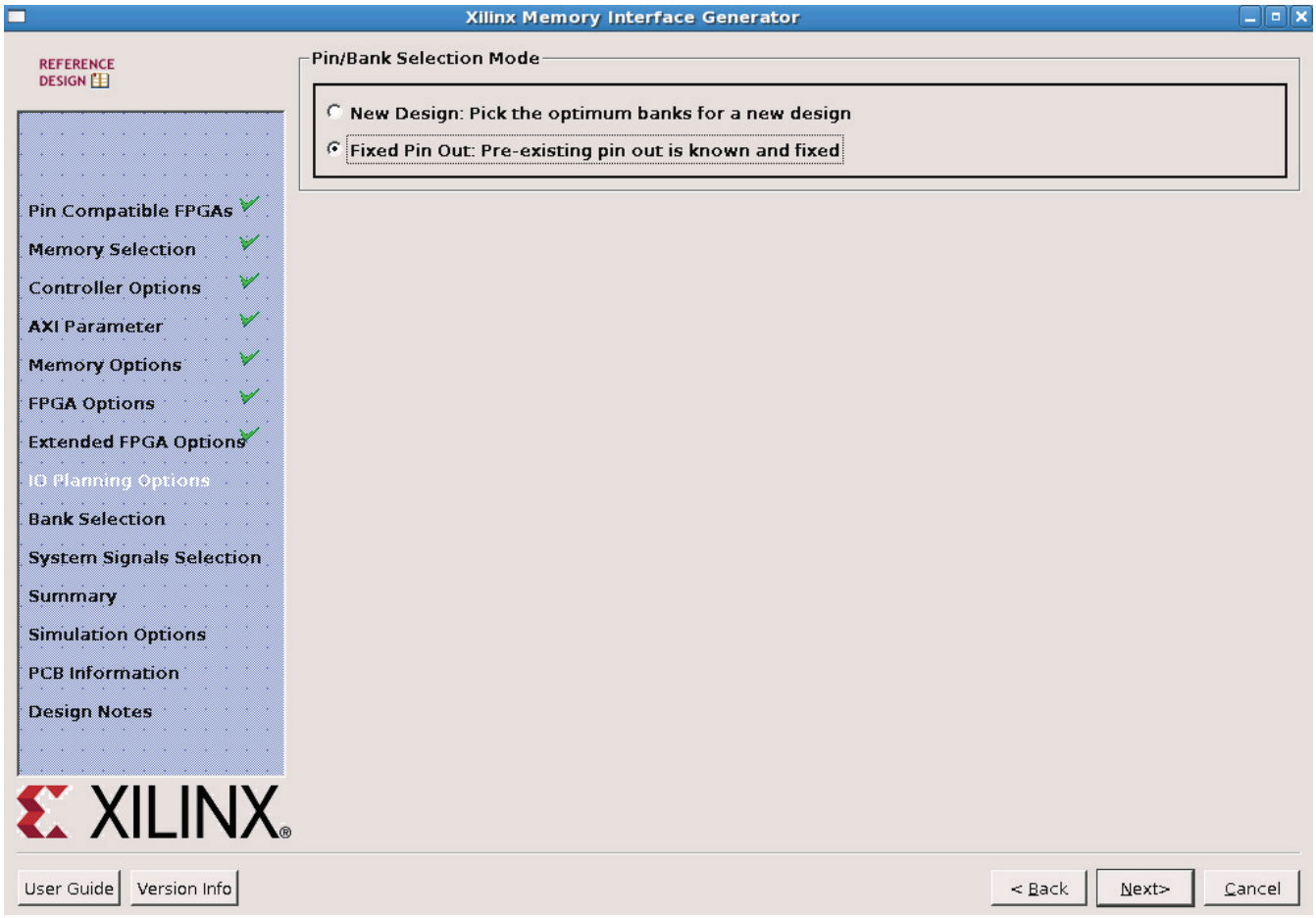


Figure 38: **MIG I/O Planning Options**

27. For Pin Selection, a user would normally pull down menus in the GUI to enter the pinout information for I/Os required by the memory interface. To save time, the MIG GUI has an option to read a UCF file to import the pinout information. The format of this file is described in the MIG documentation. To prevent having to enter each pin location for this design, click **ReadUCF** and enter the name of a file pre-generated for the KC705 board. Browse and select `<design_dir>/src/ip/DDR3_SDRAM/user_design/constraints/DDR3_SDRAM.ucf`, then click **Open** to load the KC705 pinout information file. This returns the user to the Pin Selection menu. Click **Validate** to have the MIG tool check the pinout, then click **Next** (Figure 39). Click **OK** to the DRC Validation Message Log, which confirms the pinout is valid for the chosen part and IP configuration.

Note: The MIG tool only uses a UCF file to save and read back memory interface pinout information. This file is not used for I/O constraints in the implementation tools. The Vivado tools use XDC files generated by the MIG tool to specify I/O constraints for the implementation tools.

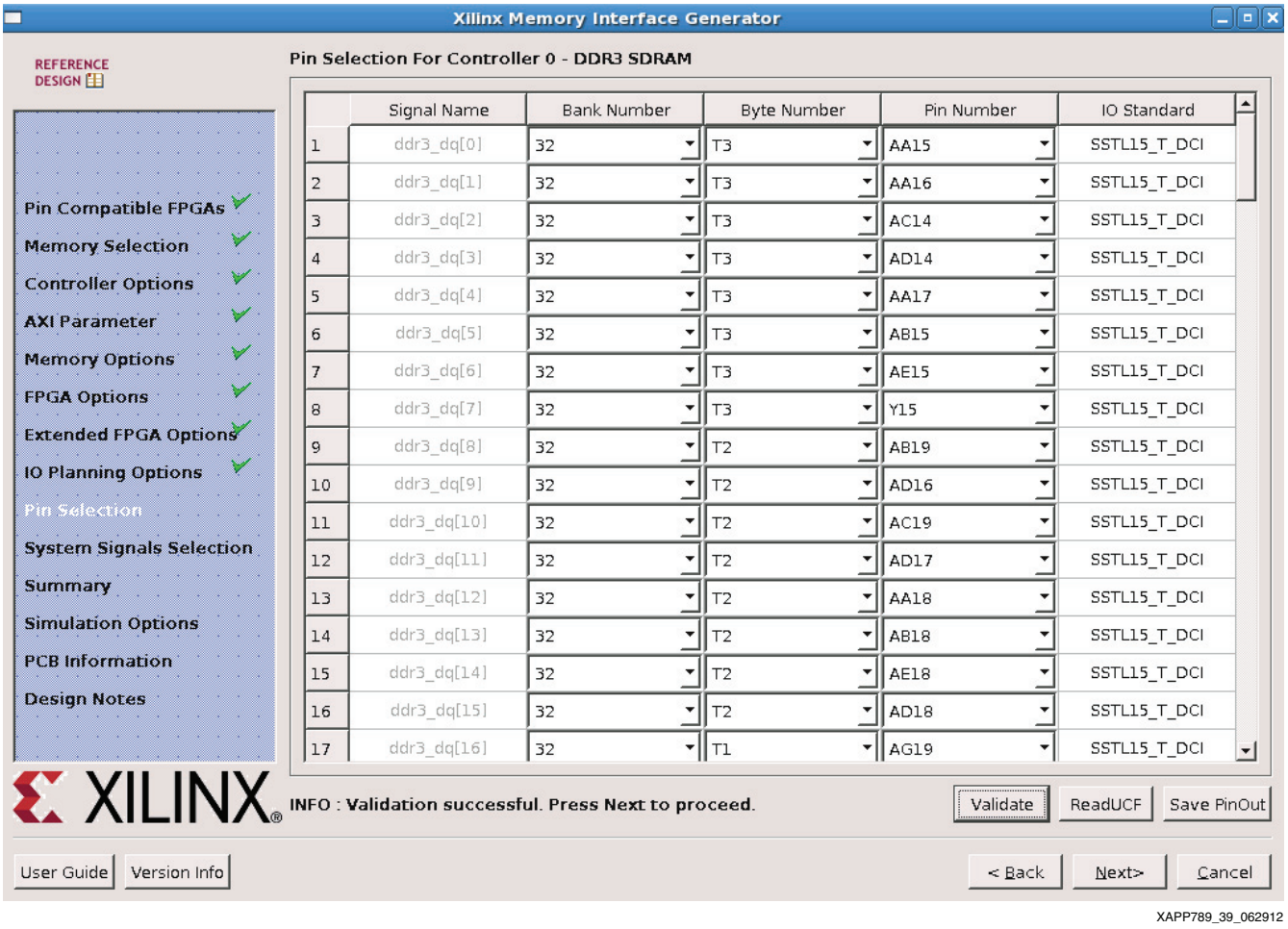
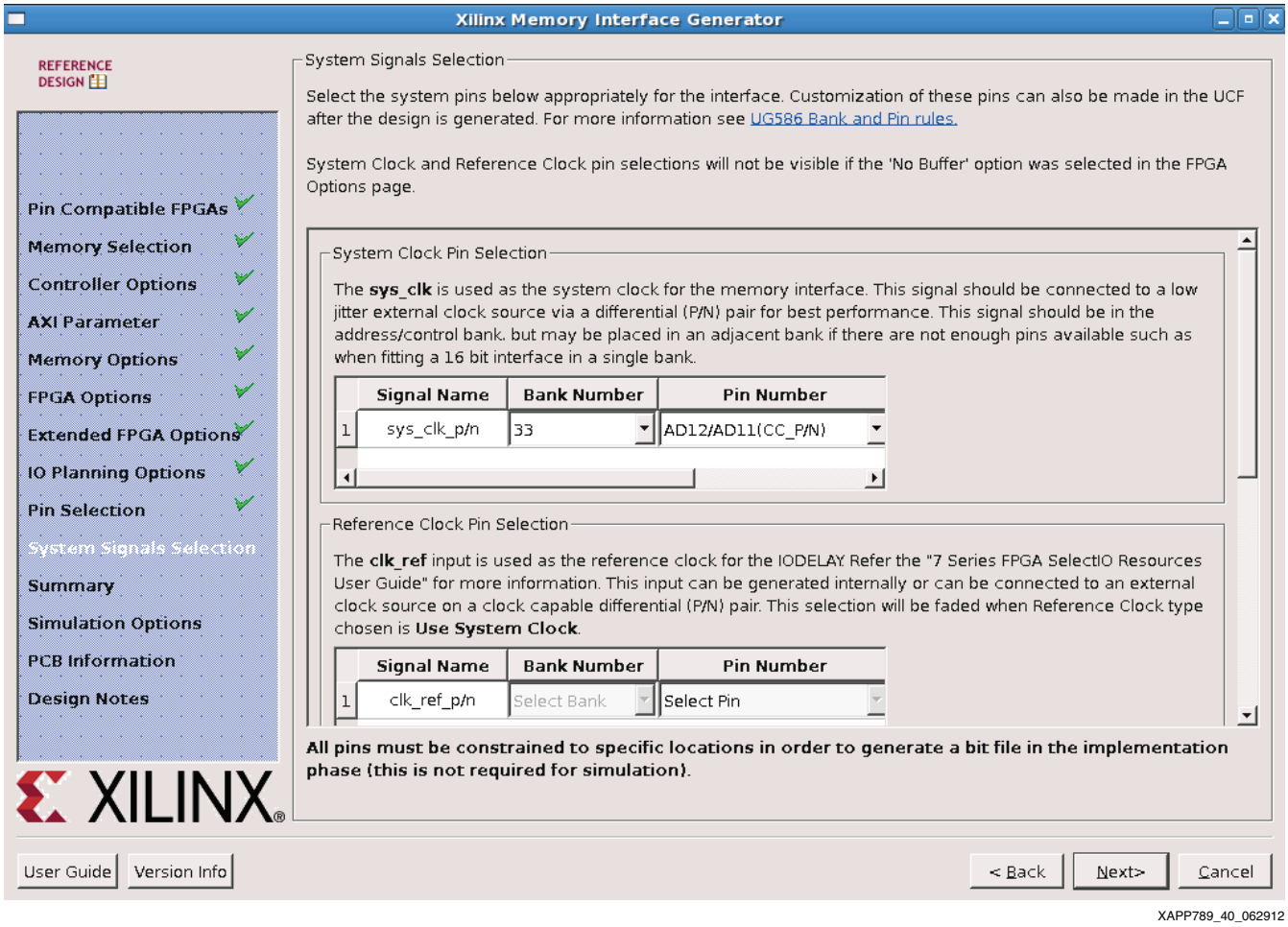


Figure 39: **MIG Pin Selection Using an Existing Pinout**

28. For the System Signals Selection, set the sys_clk_p/n signal to **AD12/AD11**, then click **Next** (Figure 40).



XAPP789_40_062912

Figure 40: **MIG System Signals Selection**

29. The Summary page shows the overall IP configuration settings. Click **Next** to continue (Figure 41).

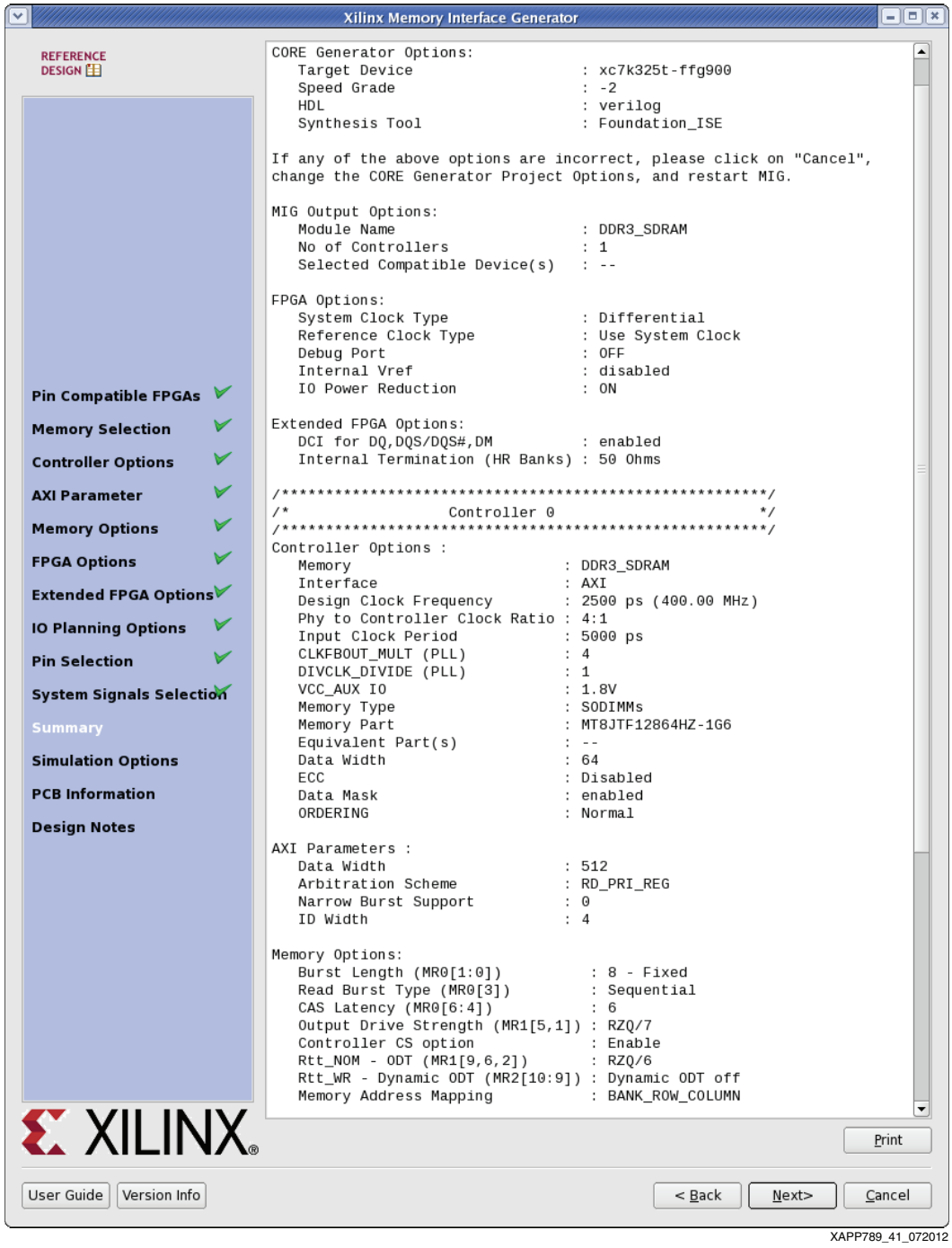


Figure 41: **MIG Summary**

30. Review the License Agreement, click **Accept**, and then **Next** to have the Vivado tools deliver a memory simulation model for use in creating a simulation testbench (Figure 42). If the License agreement is declined, a separate model must be obtained to perform simulations with this IP core.

    Note: This application note does not demonstrate the simulation flow.

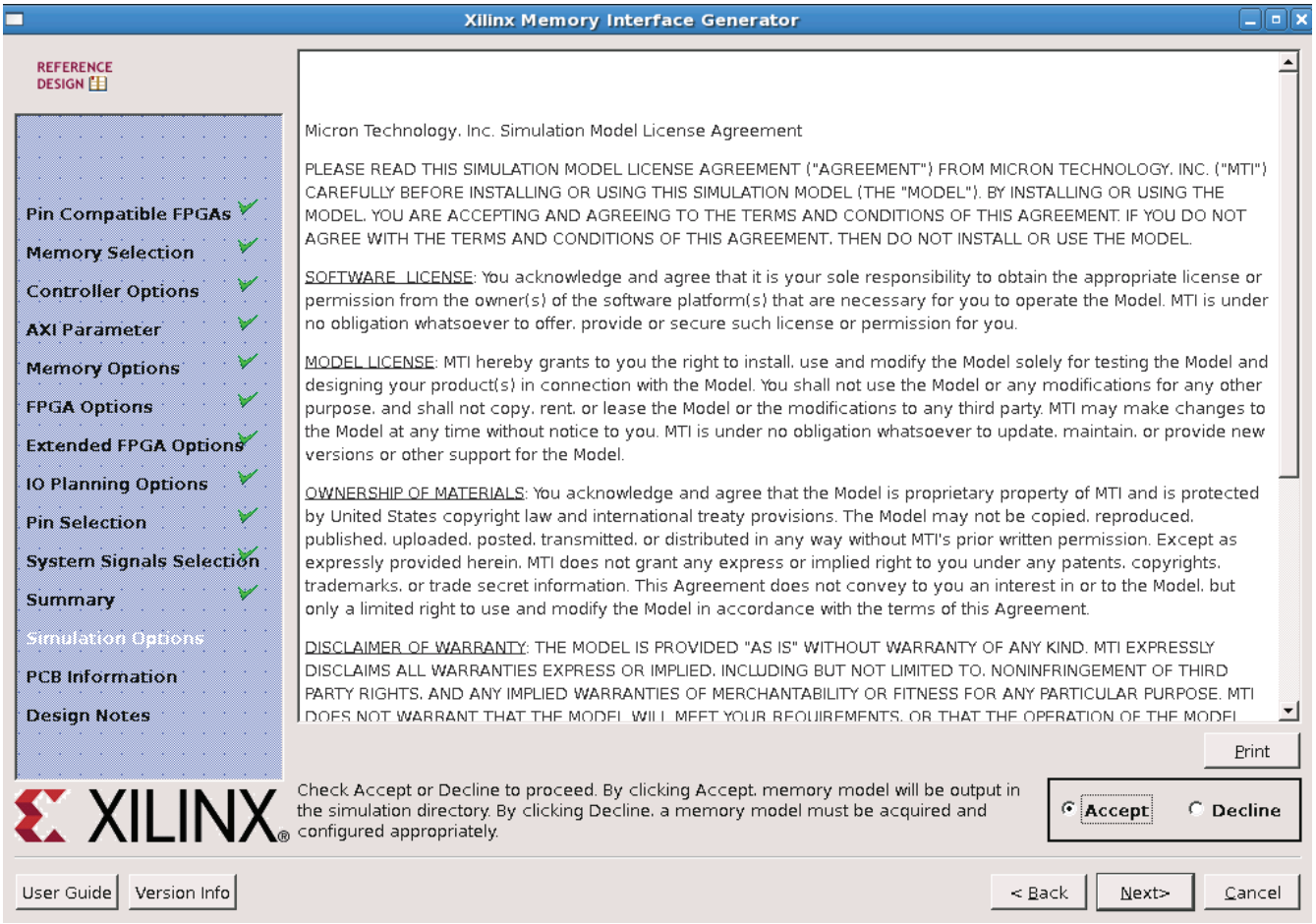


Figure 42: **MIG Micron Model License Agreement**

31. This window mentions that important PCB information is contained in the User Guide accessible by clicking the button in the lower-left corner. Click **Next** to continue (Figure 43).

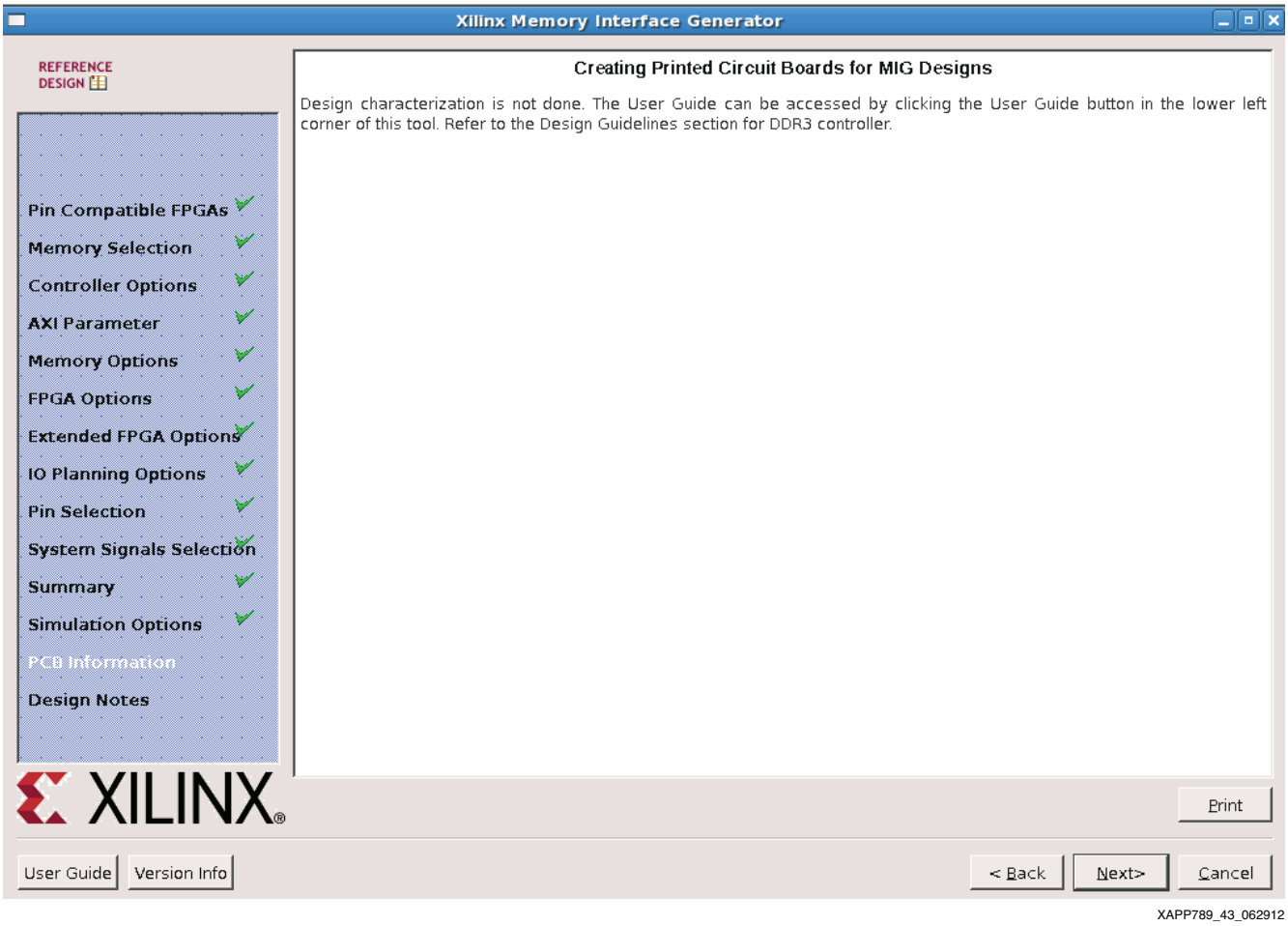


Figure 43: **MIG PCB Information**

32. Review the Design Notes and click **Generate** (Figure 44). The MIG IP files are generated as the IP customization GUI is running to create the underlying HDL and constraint files. After several seconds for the generation process to complete, control is returned to the Vivado tools. This creates a memory controller for the DDR3 memory on the KC705 board that has an AXI Interface to later connect to the AXI Interconnect IP.

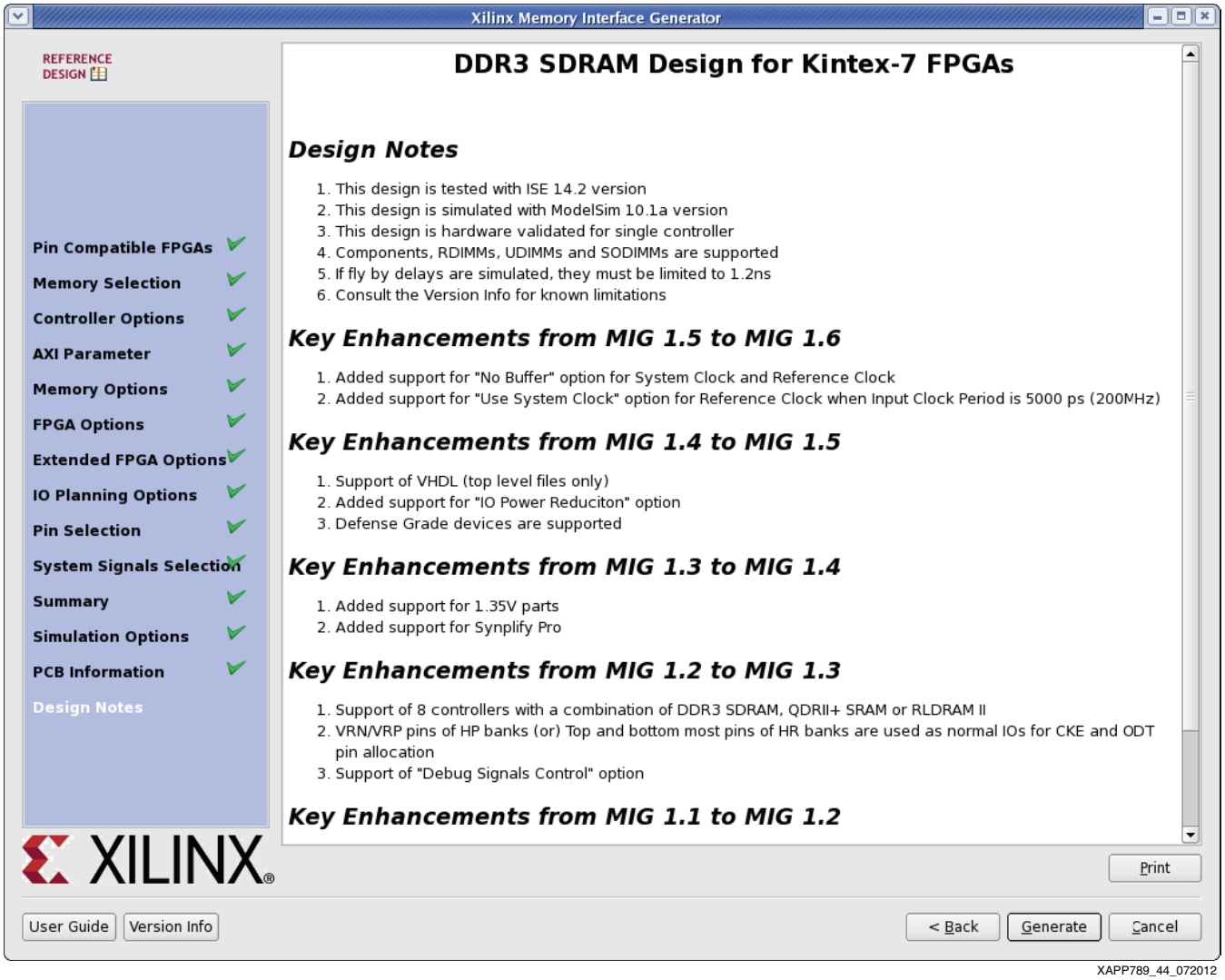


Figure 44: **MIG Design Notes**

Adding an AXI Interconnect to the Design

This section describes the steps to add the AXI Interconnect IP block to the project. Detailed information about the AXI Interconnect IP core is described in the *LogiCORE IP AXI Interconnect Data Sheet* [Ref 5]. The configuration of the AXI Interconnect and the process of optimizing AXI systems is described in the "AXI System Optimization: Tips and Hints" chapter of the *AXI Reference Guide* [Ref 4].

33. Return to the IP Catalog in the Vivado GUI or click **Project Manager > IP Catalog** to open the IP Catalog window.
34. Find the AXI Interconnect IP in the AXI Infrastructure IP directory or type the name in the Search bar. Double click the IP name to configure this IP (Figure 45).

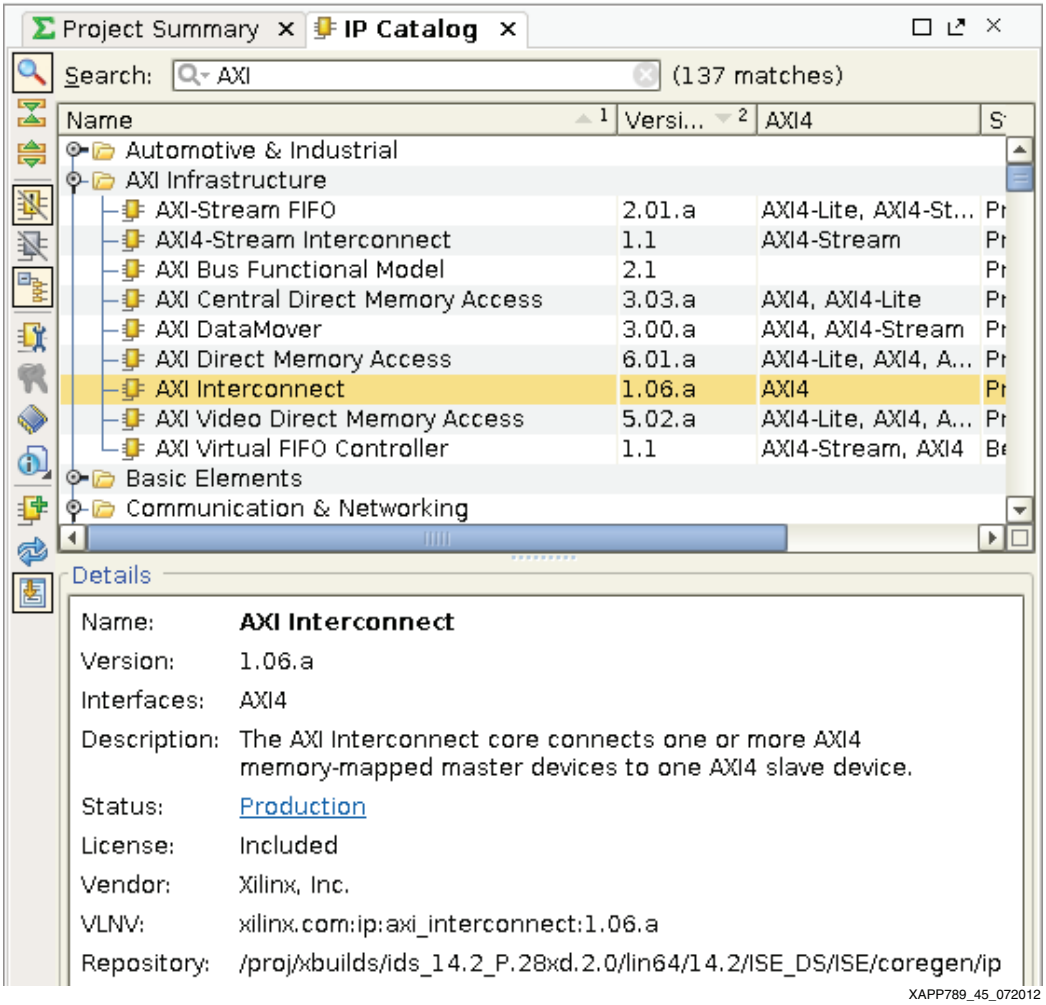


Figure 45: **Open IP Catalog**

35. In the Global Settings tab of the AXI Interconnect IP configuration GUI, configure the interconnect to support connections from four master endpoint IP cores to one slave. Each AXI_VDMA contains a separate read-only and write-only interface, resulting in a total of four masters between both AXI_VDMAs. The following settings should be used for this configuration (Figure 46):
    - Component Name: **axi4**.
    - Number of Slave Interfaces: **4**.

      This allows four AXI master endpoints to be connected to the Interconnect.
    - Slave Interface Thread ID Width: **0**.

      This is selected because the AXI_VDMA master does not generate IDs.
    - Address Width: **32**.

      This value should be selected to match the MIG tool's address width setting.
    - Interconnect Internal Data Width: **512**.

      This value should be selected to match the MIG tool's AXI Interface data width.

      Note: Setting this value lower than 512 bits causes a throughput bottleneck between the MIG tool and the AXI Interconnect. Setting this value above 512 bits increases area unnecessarily.
    - Period of INTERCONNECT_ACLK in PS: **10000**.

      This value is selected to operate the interconnect at 100 MHz, which is the same clock frequency as the MIG tool's AXI interface.

    The AXI Interconnect is configured to match the 512 bit x 100 MHz interface of the AXI MIG that the AXI Interconnect will be connected to. This optimizes the performance of the system and reduces the need for width and clock conversion.

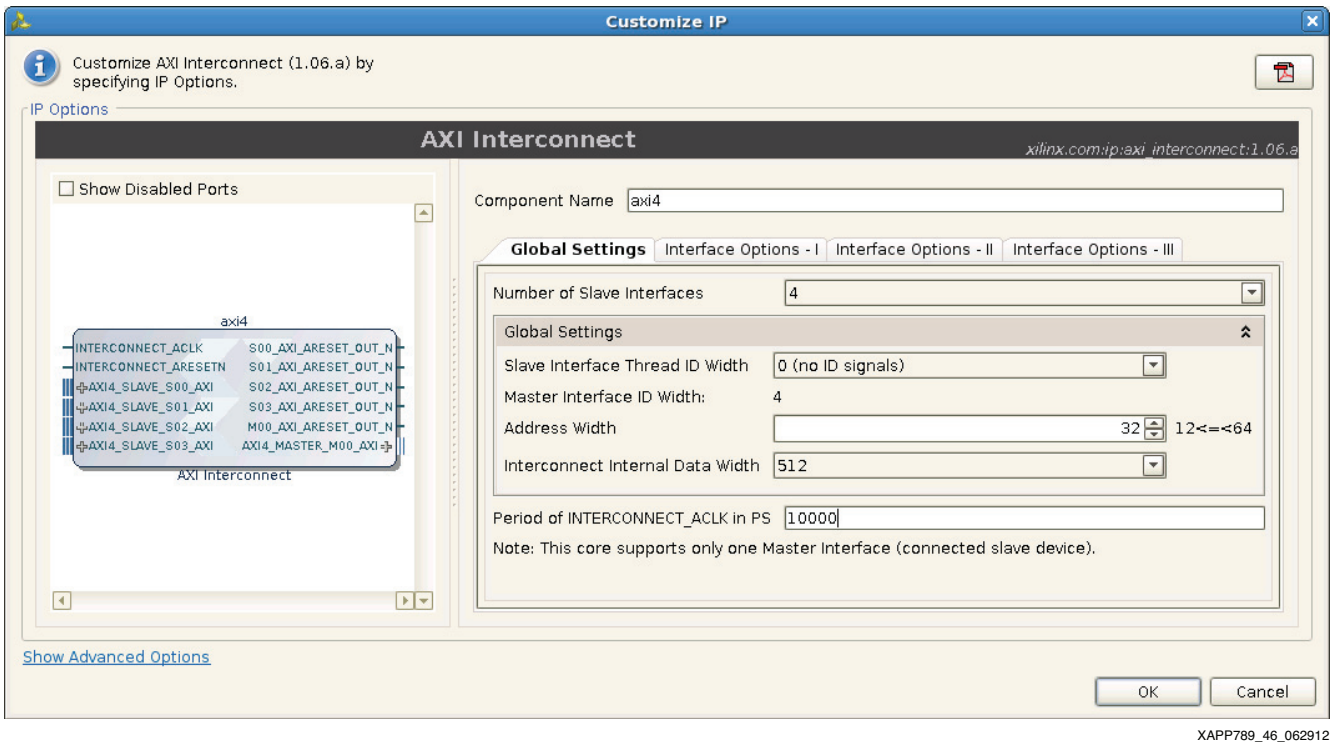


Figure 46: **AXI Interconnect Global Settings**

36. Click the **Interface Options – I** tab of the AXI Interconnect IP Configuration and make the following settings (Figure 47):
    - Master Interface Data Width: **512**.

      This matches the width of the MIG slave AXI interface.
    - Slave Interface 0–3 Data Width: **32**.

      This matches the width of each AXI_VDMA master AXI interface.
    - Slave Interface 0–3, Is Aclk Async: (Checked).

      Clock converters are required on the slave interfaces (enable corresponding checkboxes) to the AXI VDMA masters because they run at 75 MHz, whereas the AXI Interconnect runs at 100 MHz.
    - Register Slices: (Unchecked).

      This is because the system is relatively simple, but if timing is not met, the pipelining of register slices might be required.

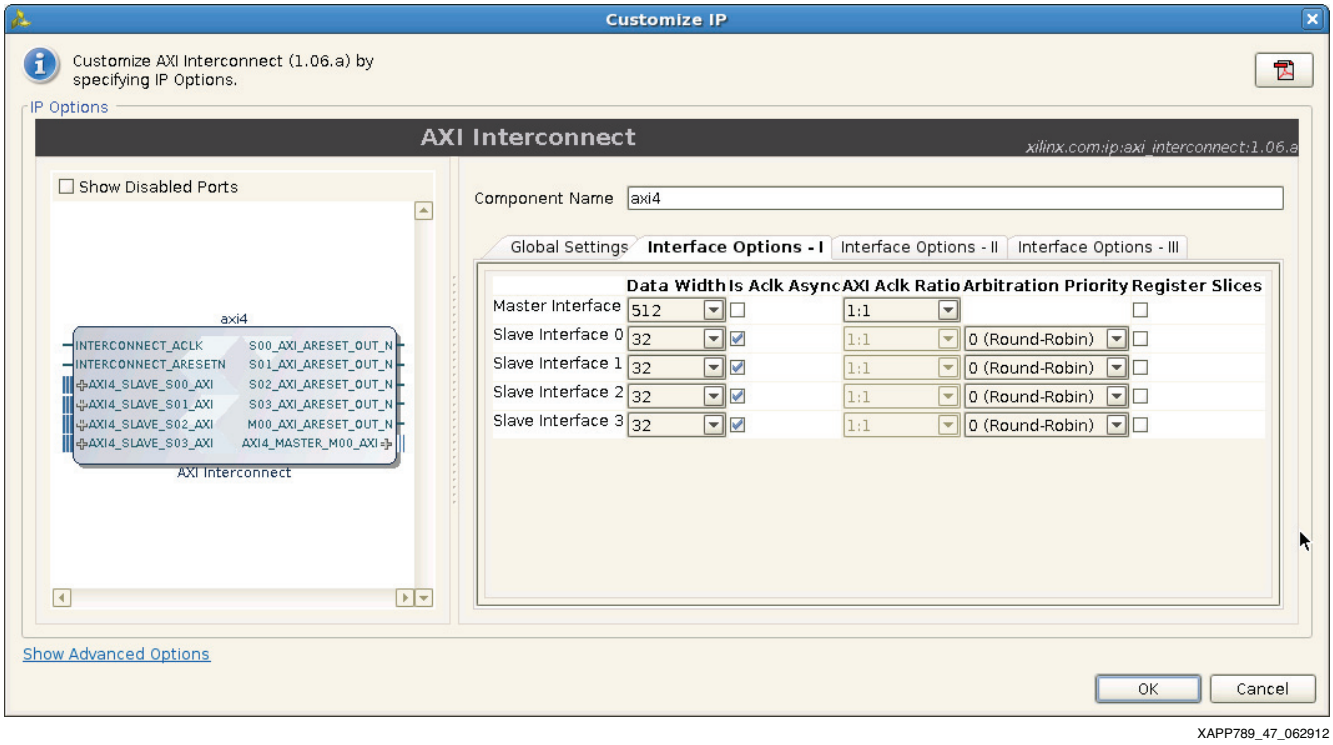


Figure 47: **AXI Interconnect Width and Clock Settings**

37. Click the **Interface Options – II** tab of the AXI Interconnect IP Configuration and make the following settings (Figure 48):
    - Slave Interfaces 0 and 2: **READ-ONLY**.

      These are required to connect to the memory map to slave (MM2S) AXI interfaces of the AXI_VDMA.
    - Slave Interfaces 1 and 3: **WRITE-ONLY**.

      These are required to connect to the slave to memory map (S2MM) AXI interfaces of the AXI_VDMA.
    - Slave Interfaces Read/Write Acceptance (where not grayed out): **2**.

      As described in the "AXI Optimization" chapter of the *AXI Reference Guide* [Ref 4], the AXI_VDMA uses a relatively long burst so that an acceptance value of 2 minimizes area while allowing support for pipelined transactions.
    - Master Interface Read/Write Acceptance: **4**.

      On the master interface to the memory controller, a higher acceptance value of 4 allows it to pipeline several transactions to the memory controller to maximize throughput.

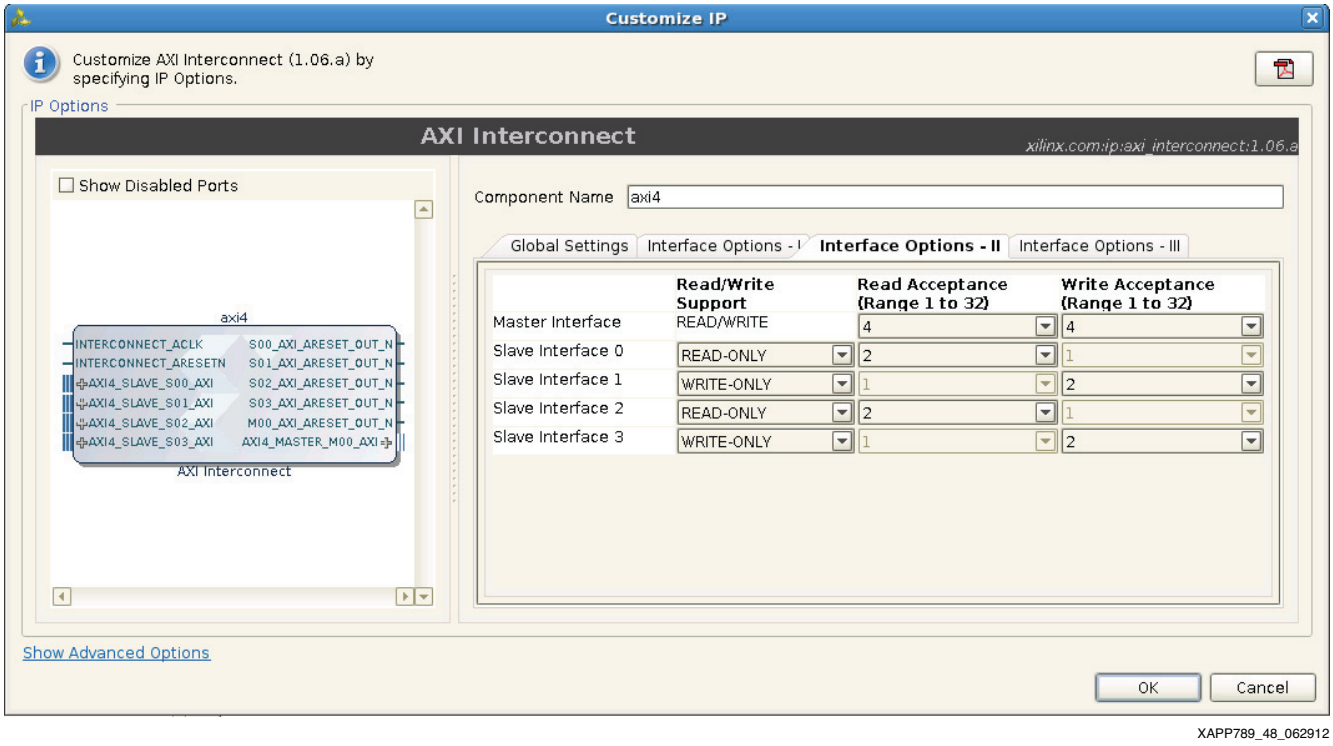
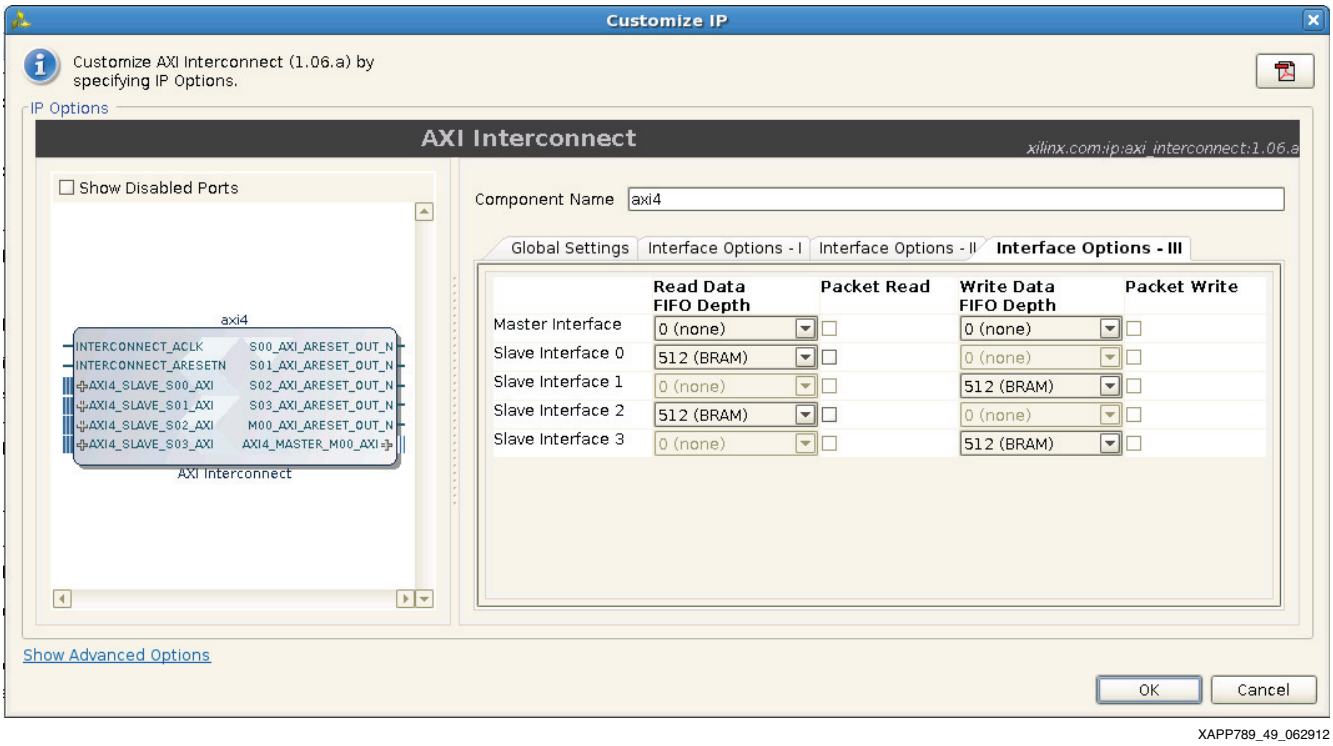


Figure 48: **AXI Interconnect Channel Data Settings**

38. As described in the “AXI Optimization” chapter of the *AXI Reference Guide* [Ref 4], block RAM FIFOs on the slave interfaces should be enabled to improve throughput. Click the **Interface Options – III** tab of the AXI Interconnect IP Configuration and make the settings listed below, then click **OK** to generate the core (Figure 49). After clicking **OK**, the IP customization GUI generates the HDL and constraint files for the IP. After several seconds for the generation process to complete, control is returned to the Vivado GUI:
    - Master interface Read Data FIFO Depth: **0 (none)**.
    - Master interface Write Data FIFO Depth: **0 (none)**.
    - Slave Interface 0 and 2 Read Data FIFO Depth: **512 (BRAM)**.
    - Slave Interface 1 and 3 Write Data FIFO Depth: **512 (BRAM)**.
    - Packet Read/Write: (Unchecked).

      Leave the default disabled settings for packet FIFO modes because this design does not require packet read/write.



XAPP789_49_062912

Figure 49: **AXI Interconnect Packet Read/Write Settings**

Adding Clocking Wizard to the Design

The AXI MIG core contains its own clock generation and reset logic, leaving only the 75 MHz video clock to be generated with the Clocking Wizard from the Vivado tools IP Catalog. The MIG tool provides a 100 MHz clock output via the ui_clk signal, which can be used as the input to the Clocking Wizard.

39. Launch the Clocking Wizard customizing GUI from the IP Catalog in the Vivado tools by selecting **FPGA Features and Design > Clocking > Clocking Wizard**.
40. Name the component **clock_generator**.
41. Click the **Clocking Features/Input Clocks** tab and make the following settings (Figure 50):
    - Clocking Features: **Frequency synthesis**, **Phase alignment**
    - Jitter Optimization: **Balanced**.
    - Input Clock information (primary input clock):
      - Input Frequency (MHz): **100**.
      - Input jitter: **0.01**.
      - Source: **No buffer**.

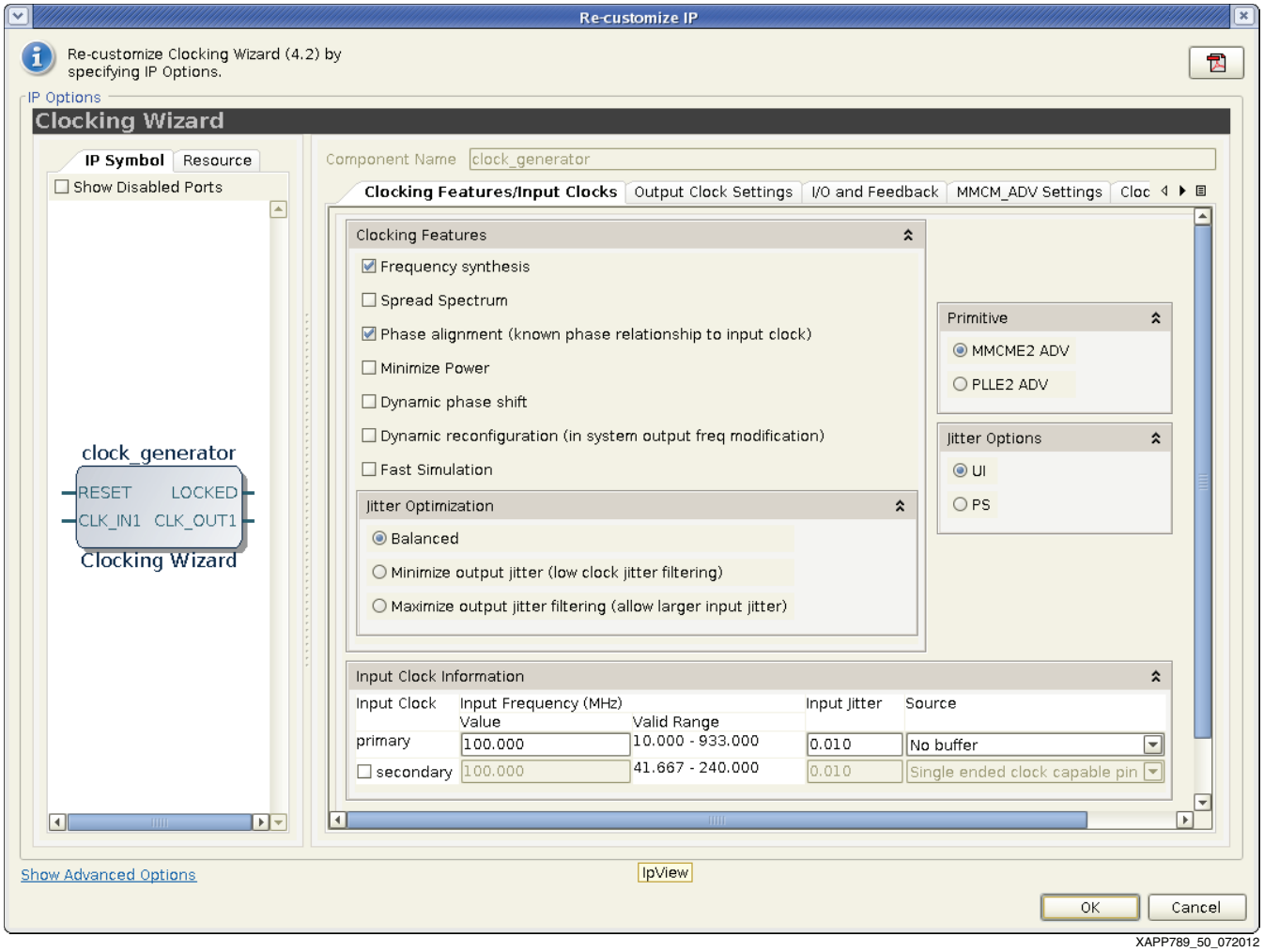


Figure 50: **Clocking Wizard Features and Input Clocks Settings**

42. Click the **Output Clock Settings** tab and make the following settings (Figure 51):
    - Output Clock CLK_OUT1:
      - Output Freq (MHz): **75**.
      - Duty Cycle (%): **50**.
      - Drives: **BUFG**.

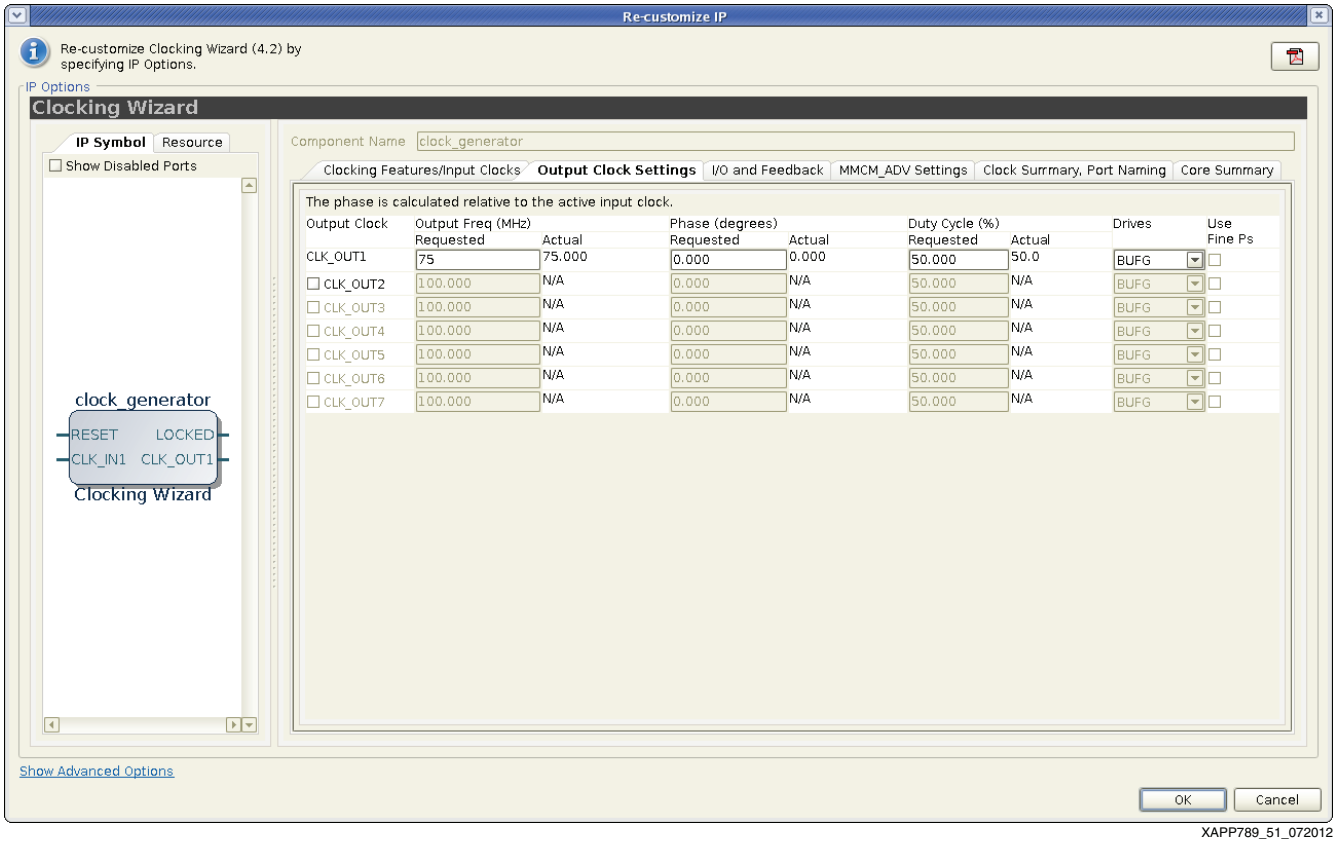


Figure 51: **Clocking Wizard Output Clock Settings**

43. Click the **I/O and Feedback** tab and verify that the following settings are correct (Figure 52):
    - Optional Inputs/Outputs:
      - RESET: (Checked).
      - LOCKED: (Checked).
      - INPUT_CLOCK_STOPPED: (Unchecked).
      - POWER_DOWN: (Unchecked).
      - CLKFBSTOPPED: (Unchecked).
    - Clock Feedback Source:
      - Automatic control on-chip: (Checked).

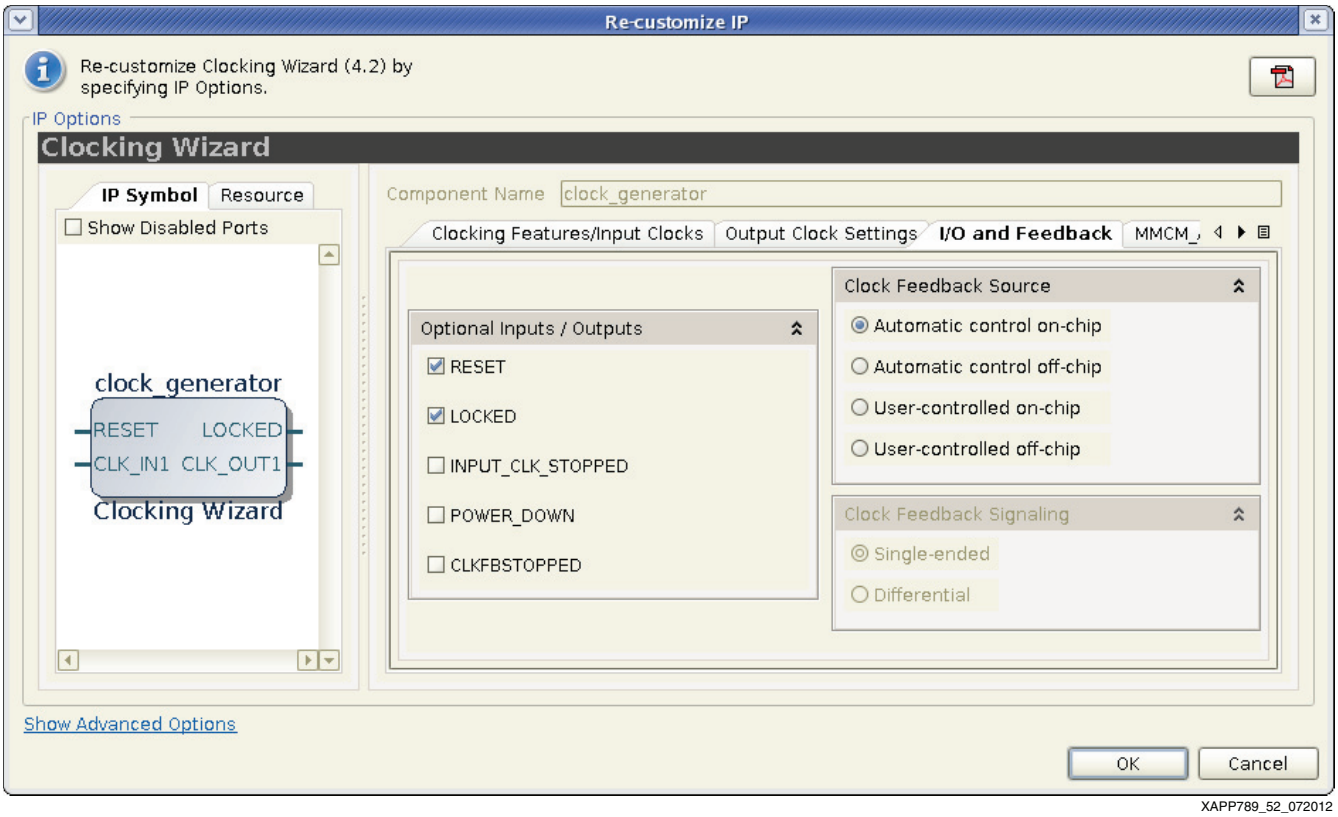


Figure 52: **Clocking Wizard I/O and Feedback Settings**

44. Click the **MMCM_ADV Settings** tab and review the settings (Figure 53).

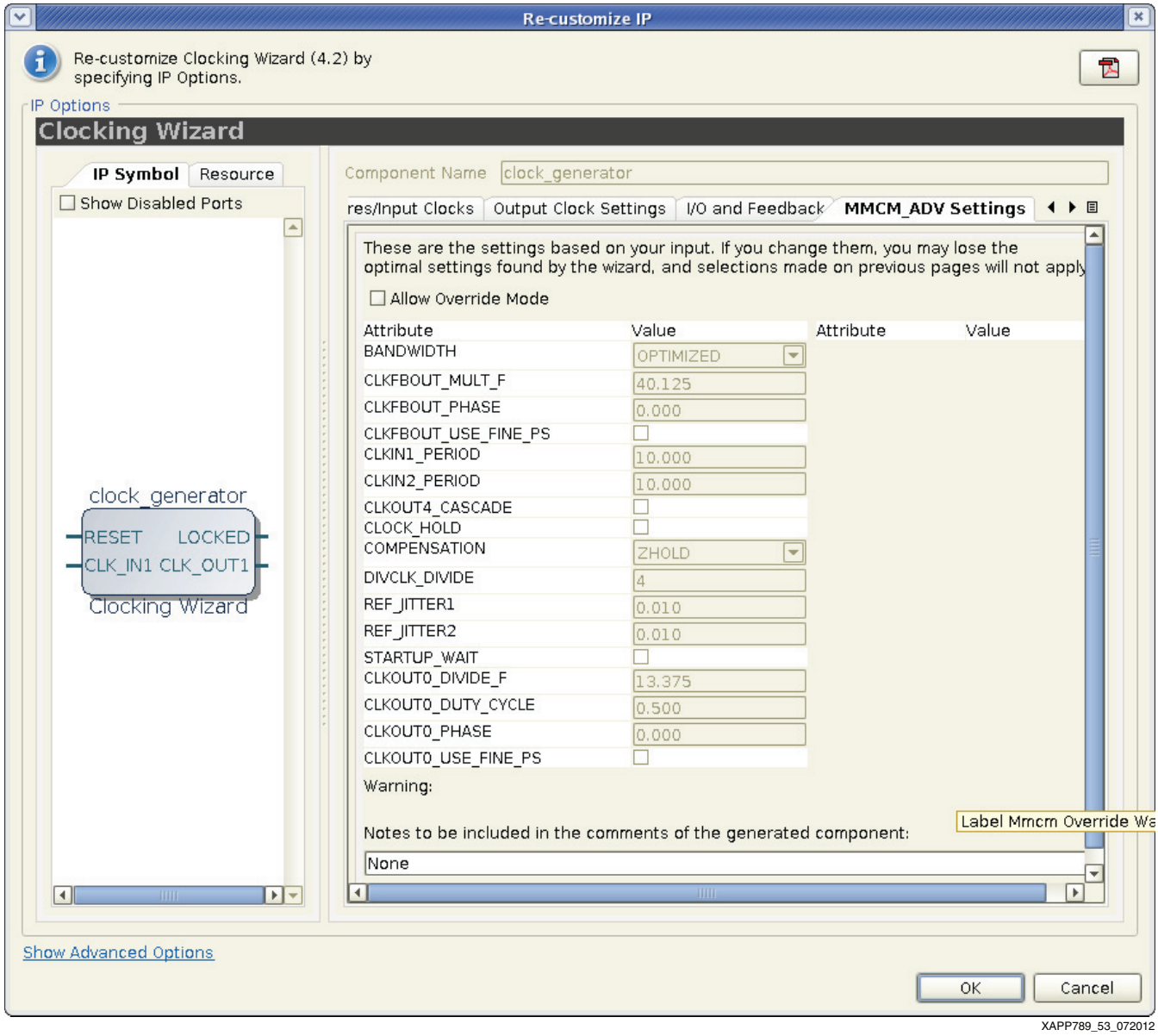


Figure 53: **Clocking Wizard MMCM_ADV Settings**

45. Click the **Clock Summary, Port Naming** tab and verify that the primary input clock port name is **CLK_IN1** (Figure 54).

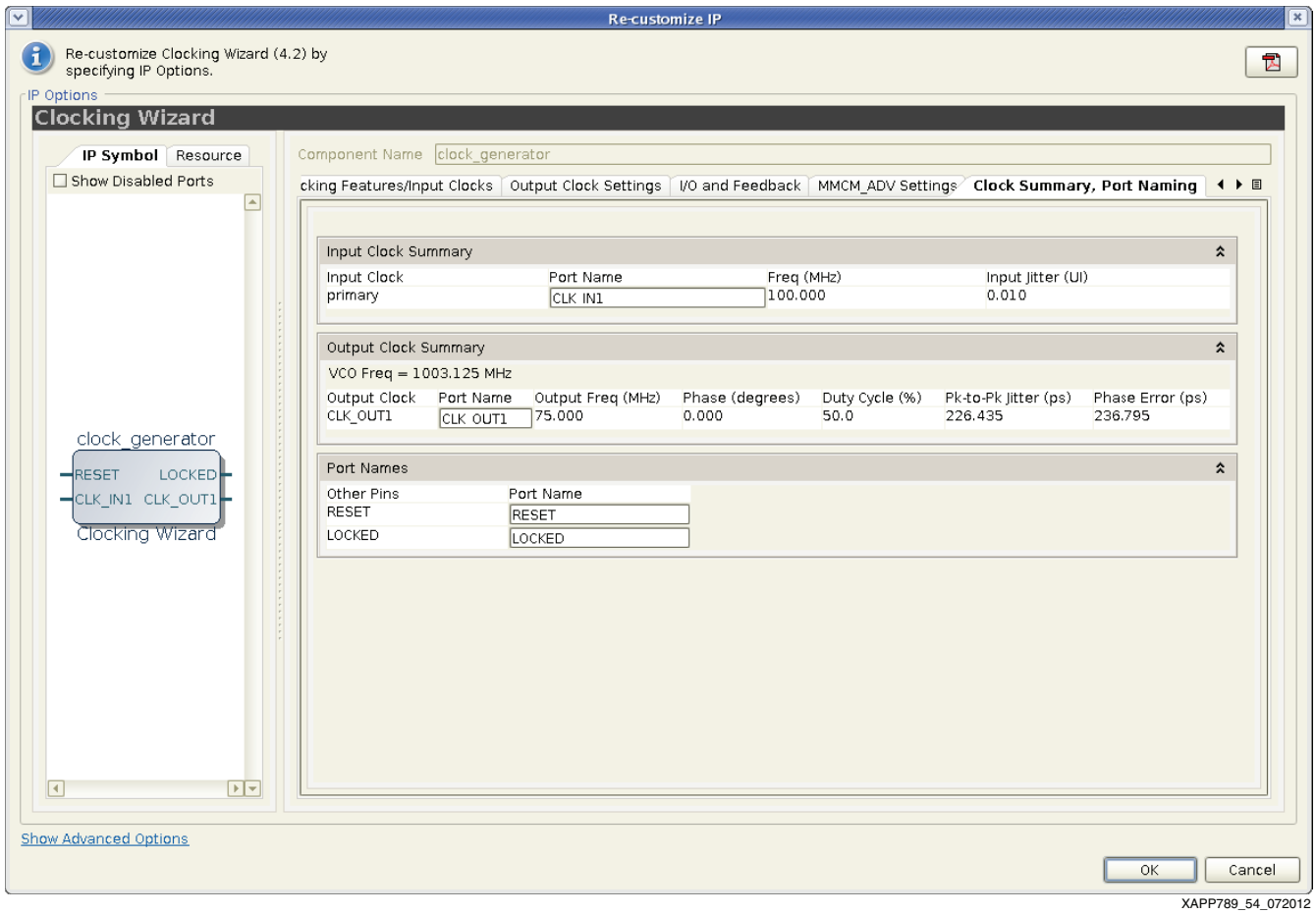


Figure 54: **Clocking Wizard Clock Summary and Port Naming Settings**

46. Click the **Core Summary** tab and review the settings. Click the **Resource** tab on the left to review the resource utilization for this clocking design (Figure 55).

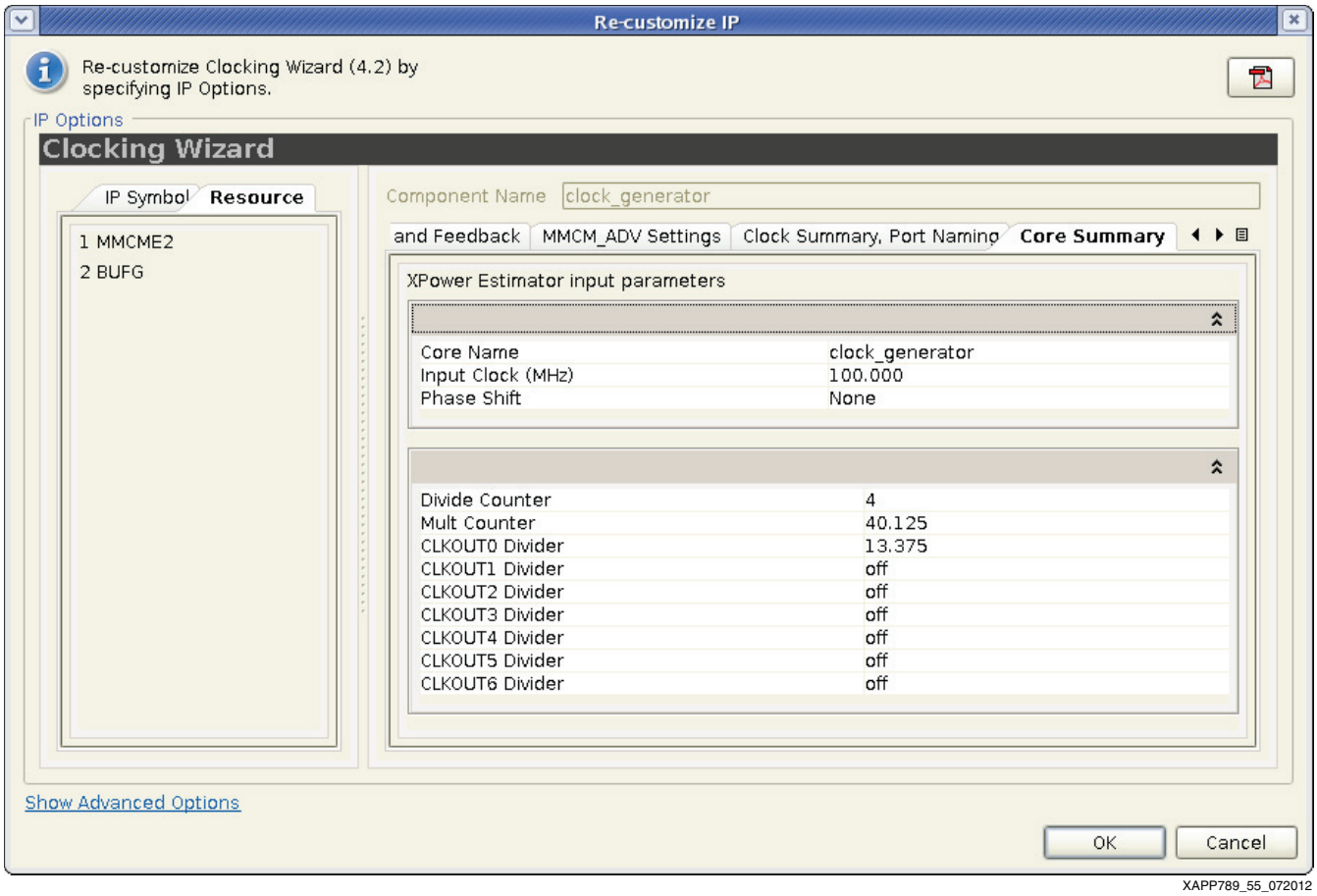


Figure 55: **Clocking Wizard Core Summary**

47. Complete the wizard by clicking **OK**.

Adding AXI VDMA to the Design

This section describes the generation of both AXI_VDMA cores into the design. VDMA_0 receives video data from the master_example core and places it into DDR3 memory. VDMA_0 reads data back out and loops the video data over AXI4-Stream channels to VDMA_1. VDMA_1 writes the data to memory, and then reads it back out and presents it to the slave_example for display. Both AXI VDMA cores are configured identically in this design.

48. Launch the IP Customization GUI of the AXI_VDMA core from the IP catalog in the Vivado tools by selecting **IP Catalog > AXI Infrastructure > AXI Video Direct Memory Access**.

    The VDMA is configured to act as a video frame buffer with settings to enable reasonable performance. Use these steps to configure the AXI VDMA (Figure 56).

49. Make the following settings in the VDMA Basic tab (Figure 56):
    - Component Name: **vdma**.
    - Set Frame Stores: **3**.

    This sets the maximum number of frame buffer ranges needed, and generates associated configuration registers for each. For this design, three frame buffers provide sufficient buffering to allow the MM2S channel to read data and the S2MM channel to write data without the channels overwriting or reading stale data.
    - Memory Map Data Width and Stream Data Width: **32**.
    - Set Line Buffer Depth for both Read Channel and Write Channel: **2048**.

    This enables the VDMA's line buffers and sets the depth to 2048 bytes. The line buffers allow some elasticity between the AXI4-Stream and associated AXI4 interfaces providing sufficient buffering to prevent push-back from the VDMA on the AXI4-Stream.

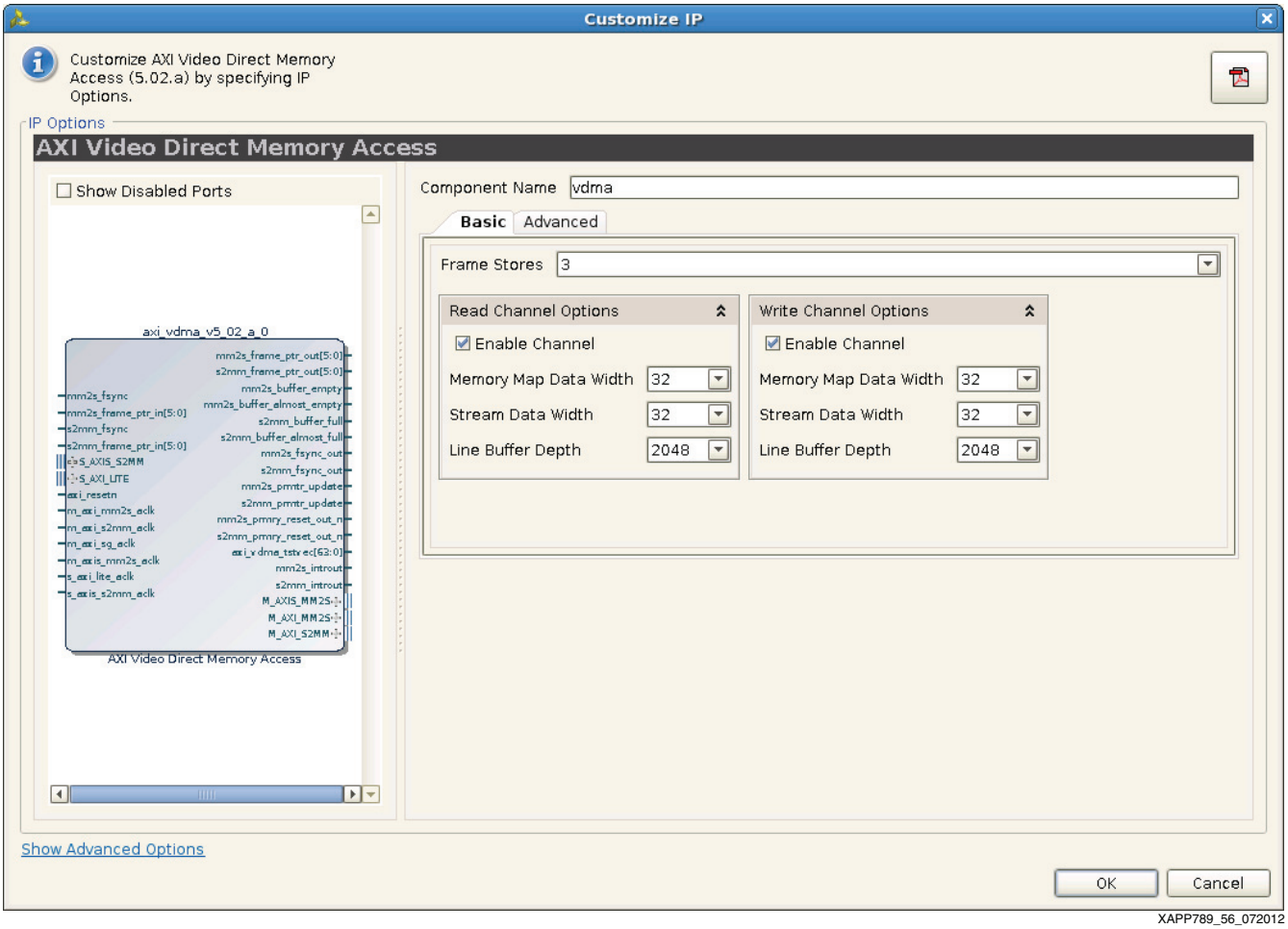


Figure 56: **AXI VDMA Basic Settings**

50. Make the following settings in the VDMA Advanced tab, then click **OK** to generate the core. (Figure 57):
    - Enable Scatter Gather Engine: (Unchecked).

      Scatter/gather operation generally requires a processor or additional state machine to provide and maintain buffer descriptors. In this design, a simple AXI4-Lite Master configures the VDMA directly through its AXI4-Lite slave interface. After the video transfer parameters of hsize, frame delay, stride, and start addresses are written, vsize is written starting the video transfers. Transfers continue indefinitely with frame synchronization signals keeping each interface from overwriting or reading stale data.
    - Enable Asynchronous Mode: (Unchecked).

      All functions of the VDMAs in this design run off the same 75 MHz clock. The asynchronous clock crossing to the 100 MHz DDR3 AXI interface occurs in the AXI Interconnect. Running the VDMAs in synchronous mode reduces FPGA resource utilization of the VDMA.
    - Use Fsync for Read and Write Channels: (Checked).

      This feature enables the VDMA frame rates to be synchronized to the video endpoint IP they are connected to.
    - GenLock Mode:
      - Read channels: **Slave**.
      - Write channels: **Master**.

      Genlock is used to maintain frame level synchronization between the read and write channels preventing each interface from overwriting data or reading stale data.
    - Read/Write Burst Size: **128**.

      This sets a large AXI4 burst size for the VDMAs. A setting of 128 provides enough bandwidth to prevent video data overruns or underruns. Choosing a large burst size also improves DDR3 memory utilization. The DDR3 memory has a 512-bit wide AXI4 slave interface, which requires long bursts to be efficient. However, using a large burst length can increase latency of other masters in the system. This design trade-off is discussed in *AXI Reference Guide* [Ref 4].
    - Allow Unaligned Transfers: (Unchecked for both Read and Write channels).

      In this design, all video lines and frames are aligned to 32-bit word boundaries so the data realignment engines in the VDMA are not needed. Leaving the realignment engines disabled reduces FPGA resource utilization.

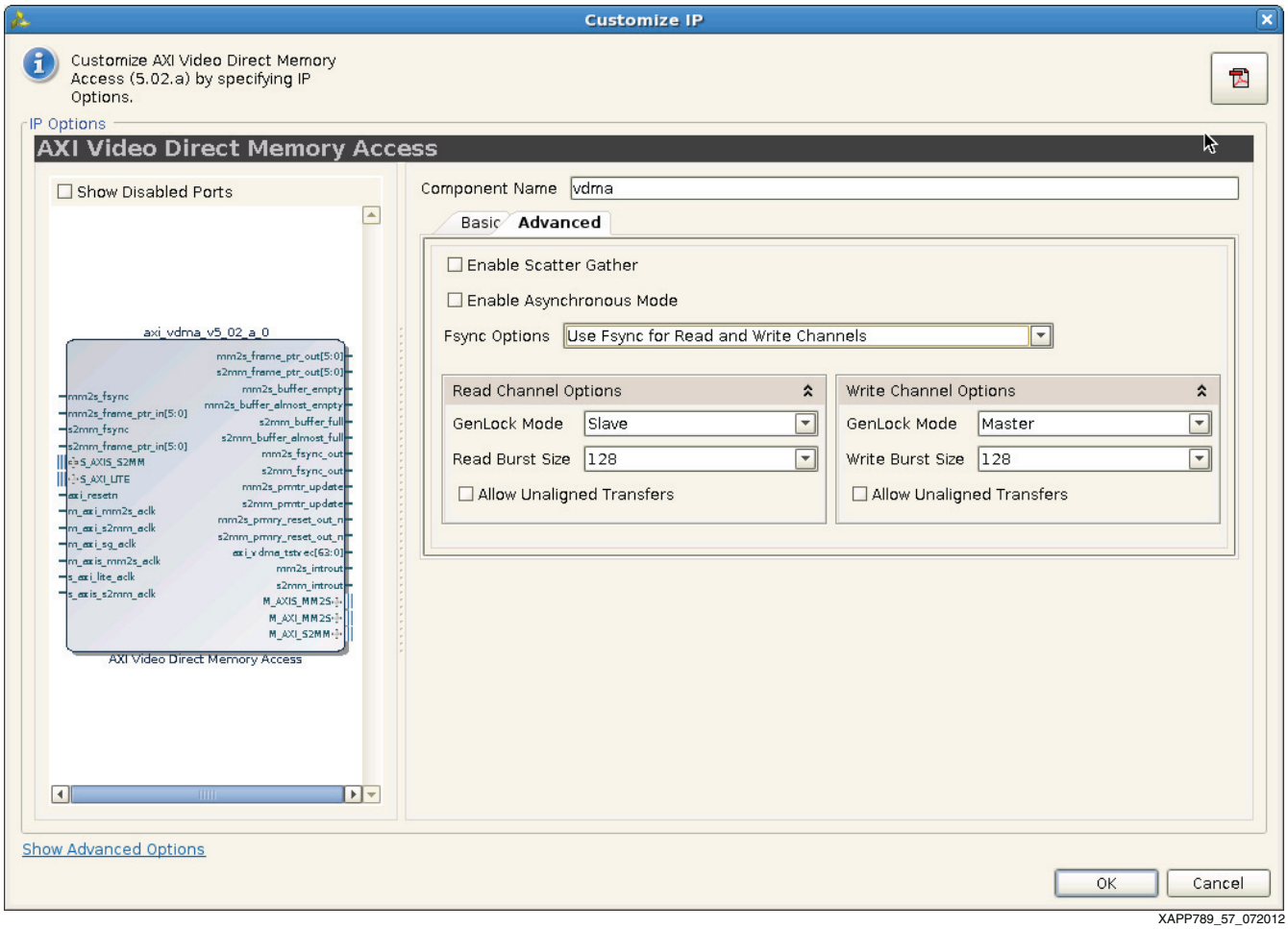


Figure 57: **AXI VDMA Advanced Settings**

Adding Other Ready-Made Modules to the Project

In addition to the IP generated by the Vivado tools for basic building blocks, the system requires some customized logic to achieve the intended design goals:

- Configure the DMA engine
- Generate test video patterns
- Display the patterns on an HDMI/DVI display port

The following section describes the functions of these additional modules and the process of adding them to the project.

AXI4-Lite Masters to Configure AXI VDMAs

Each AXI VDMA must be configured at start-up to act as a circular frame buffer for 1280 x 720p video frames. Each AXI VDMA has an AXI4-Lite interface to configure its control registers. In the system, two simple write-only AXI4-Lite masters are instantiated. Each AXI4-Lite master has an internal Verilog state machine that executes a fixed sequence of write commands to properly configure the AXI VDMA blocks. A separate AXI4-Lite master IP block is connected directly to each AXI VDMA control interface.

AXI4-Stream Test Pattern Generator

The AXI4-Stream source driving AXI VDMA 0 is a video test pattern generator (TPG) with a timebase generator (to generate video frame sync signals). The TPG block is delivered as a

fixed netlist. The fixed netlist is configured to drive a 1280 x 720p, 60 Hz video pattern into AXI VDMA 0 using an AXI4-Stream protocol.

AXI4-Stream DVI/HDMI Display Controller

AXI VDMA 1 drives an output AXI4-Stream into an HDMI/DVI Display Controller with a timebase generator (to generate video frame sync signals). The DVI/HDMI Display Controller block is delivered as a fixed netlist and also includes an IIC driver to configure the video chip on the KC705 board. The fixed netlist is configured to display a 1280 x 720p, 60 Hz video signal on the KC705 board using an AXI4-Stream protocol from the AXI VDMA 1.

The files related to these modules can be found in the `<design_dir>/src` directory. The file names and their functions are listed in Table 1.

Table 1: **Custom Module File Name**

File Name	Module	File Type
`axi_lite_master_vdma_0.v`	AXI4-Lite Master to Configure AXI VDMAs	Verilog implementation source files
`axi_lite_master_vdma_1.v`	AXI4-Lite Master to Configure AXI VDMAs	Verilog implementation source files
`axis_master_example.ngc`	AXI4-Stream Test Pattern Generator	Netlist implementation
`axis_master_example.v`	AXI4-Stream Test Pattern Generator	Netlist based HDL simulation model and HDL black-box definition of the IP block
`axis_slave_example.ngc`	AXI4-Stream DVI Display Controller	Netlist implementation
`axis_slave_example.v`	AXI4-Stream DVI Display Controller	netlist based HDL simulation model and HDL black box definition of the IP block

Add the files listed in Table 1 to the Vivado tools project using the following steps:

51. Select **Project Manager > Add Source > Add or Create Design Sources > Add Files**.
52. In the Add Source window, browse to the `<design_dir>/src/` directory and add the files listed in Table 1 (Figure 58).

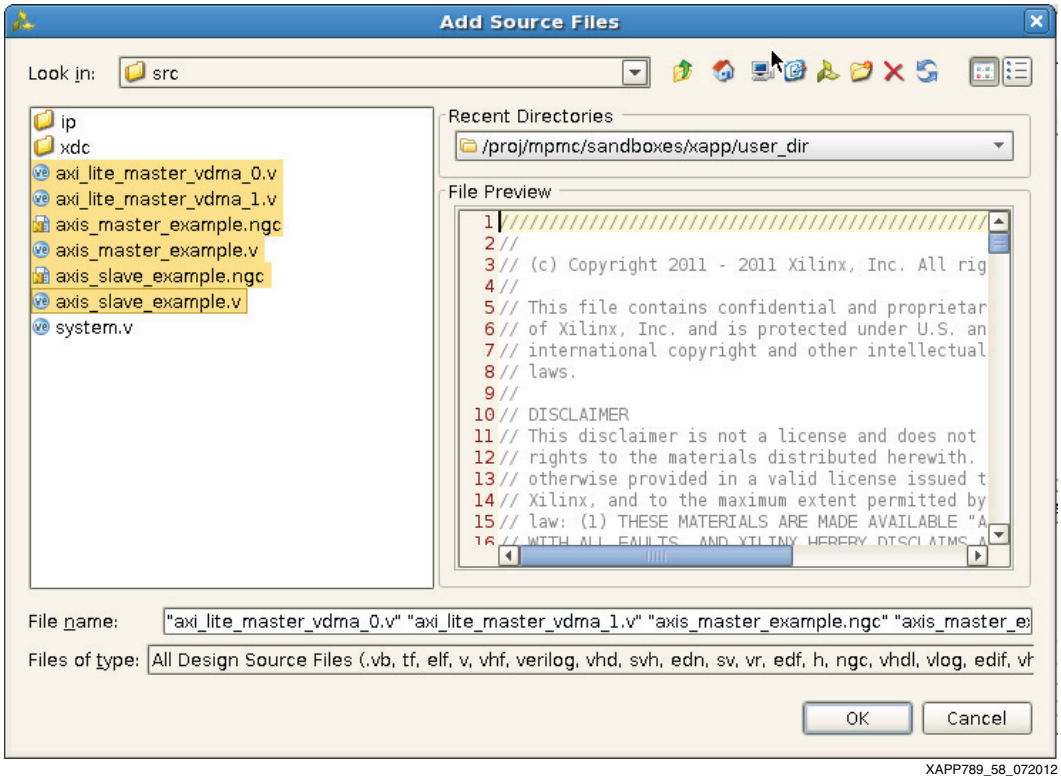


Figure 58: **Add Source Files Browser**

53. Click **OK**.
54. Associate these files as shown in Figure 59 and click **Finish**.

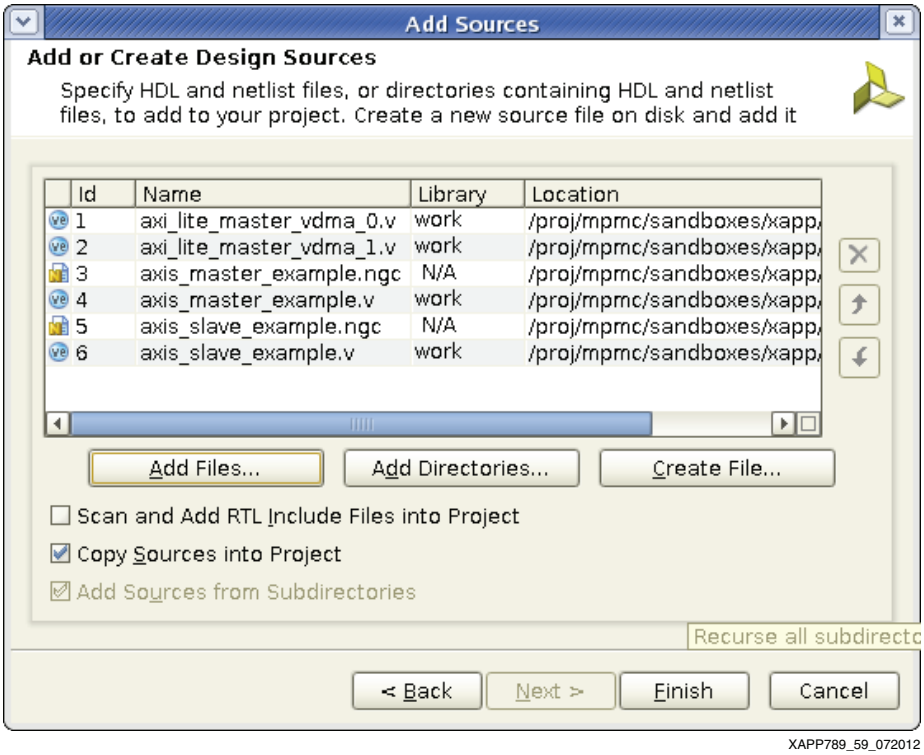


Figure 59: **New Source File Association Selection**

Module Instantiation and System Connection

After the necessary system modules are added to the project, they must be instantiated and connected together at the top level.

55. In the Project Manager view, Sources window, IP Sources tab, right click **clock_generator IP** and select **Generate…** (Figure 60).

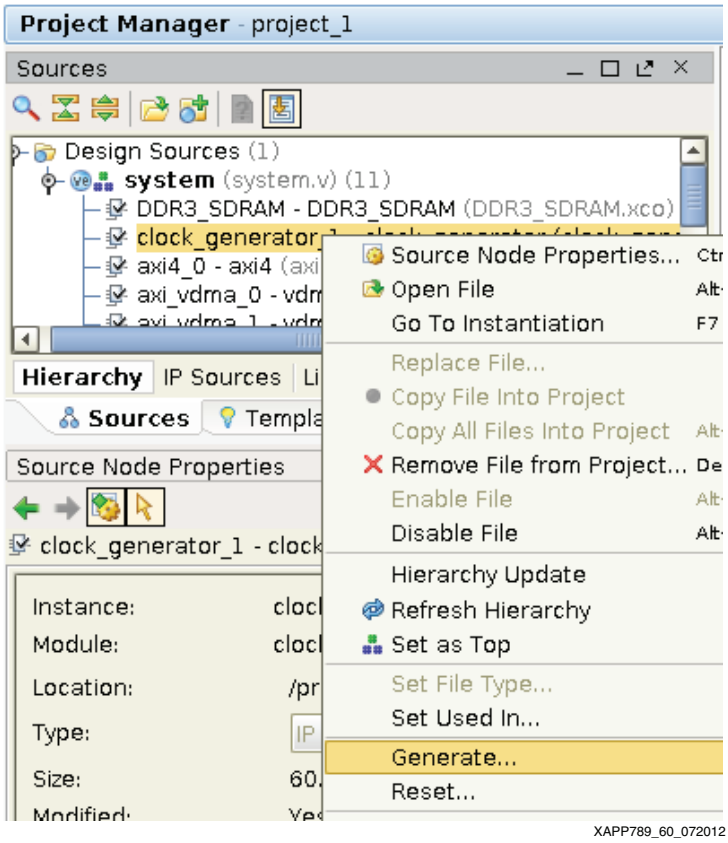


Figure 60: **Generate IP Block Instantiation Template**

56. Select **Instantiation Template** in the Generate dialog box and click **OK** (Figure 61).

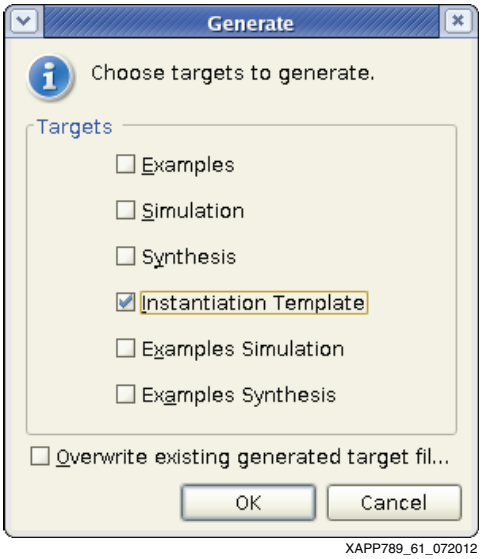


Figure 61: **Generate Dialog Box**

57. Open the `clock_generator > Instantiation Template > clock_generator.veo` file in the IP Sources tab of the Sources window to view the instantiation template (Figure 62).

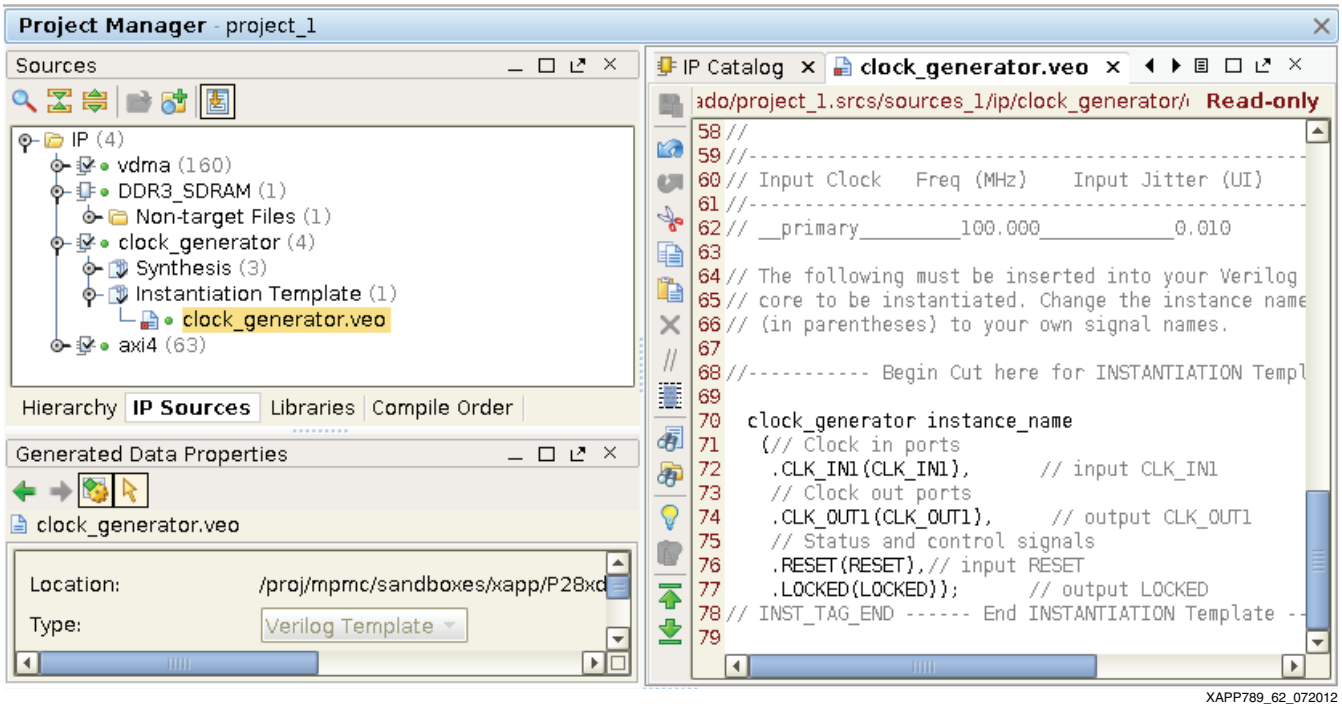


Figure 62: **View Instantiation Template**

58. Repeat step 55 to step 57 for all other IP cores and submodules to generate instantiation templates.
59. Create a new file called `system.v` as the top module of the project.
60. Copy and paste the newly created IP instantiation templates into `system.v` and connect their ports.

    Note: To save time, add the completed top-level `system.v` file to integrate all of the submodules.
61. Select **Project Manager > Add Source > Add or Create Design Sources > Add Files**.
62. In the Add Source dialog box, browse to the `<design_dir>/src/` directory and add `system.v`, then click **OK**.
63. Associate `system.v` with the `work` library and set the HDL Source for **Synthesis & Simulation**.

Reset

The MIG DDR3 controller contains its own clock generation and reset logic, and provides a reset output via ui_clk_sync_rst that can be used to reset the rest of the system. The ui_clk_sync_rst is connected to the reset input of the clock_generator instance. The LOCKED output is used to reset the AXI Interconnect core to ensure a stable clock is available before operation. The AXI Interconnect then drives the synchronized reset signals to all the other cores that are clocked from the 75 MHz signal, including the AXI4-Stream example cores.

Note: These reset connections are already implemented in the completed `<design_dir>/src/system.v` file.

Pin Constraints

In this section, a `system.xdc` file will be created and edited to add the necessary I/O constraints for the newly added top-level I/O ports.

64. Select **Layout > I/O Planning** to get to the I/O Planning view (Figure 63). This view can be used to set the location constraints for non-DDR3 related I/O pins in a custom board. Because the KC705 board is used, the DDR I/O pin locations have been preset in the MIG tool. However, to save time, the information can also be entered via XDC commands in a constraint file added to the project.

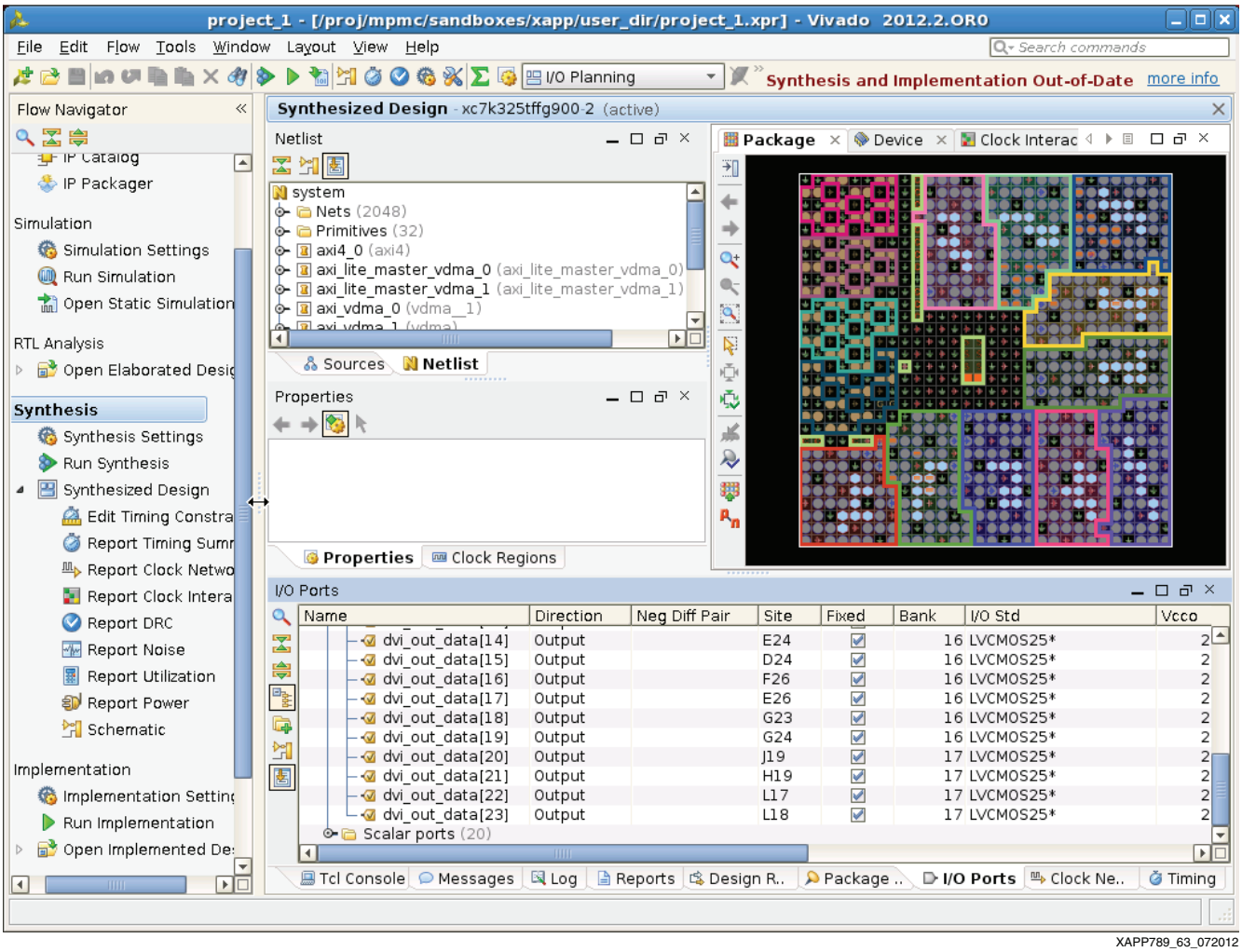


Figure 63: **I/O Planning Layout**

Note: Use the completed `<design_dir>/src/xdc/system.xdc` file, which contains the necessary I/O constraints for the KC705 board. Use **Add Sources > Add or Create Constraints > Add Files** in the Vivado tool to add this completed `system.xdc` file to the project (Figure 64).

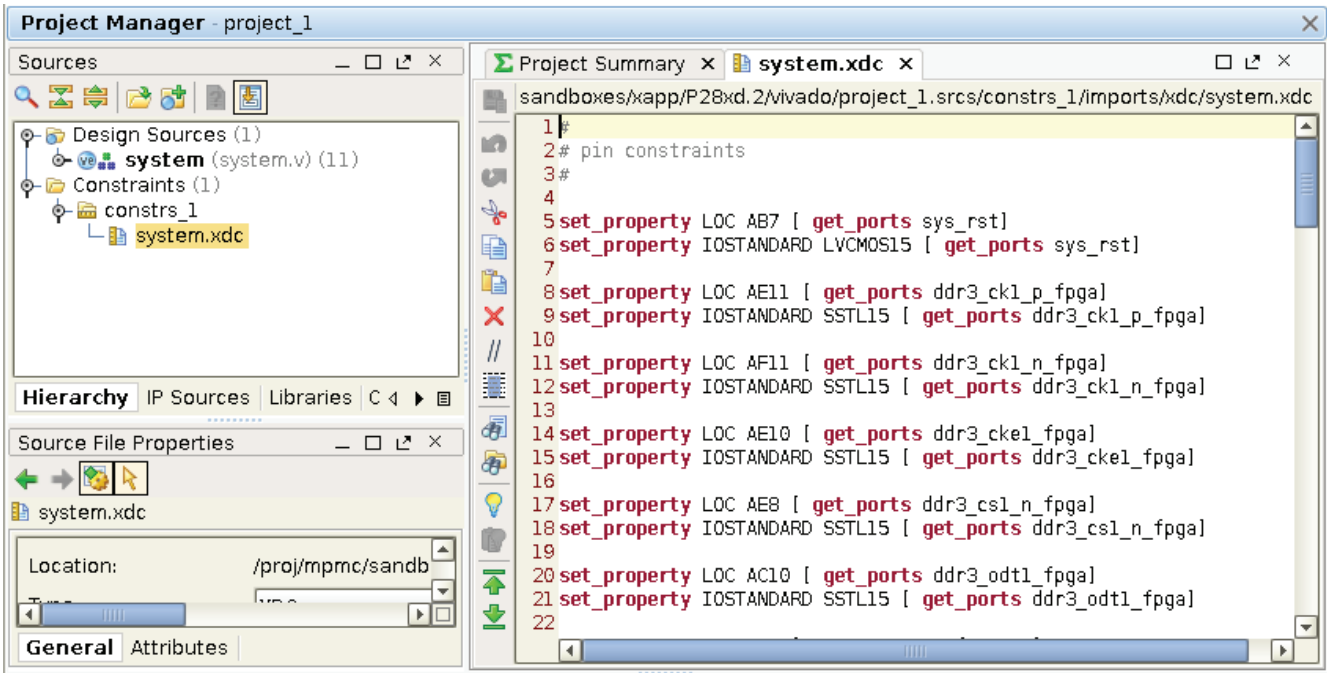


Figure 64: **Pin Constraints Added into `system.xdc`**

65. Set timing constraints.

    This design has only one input clock. All internal clocks are derived from this input clock via the MMCM. The input clock timing constraint is generated by the MIG core and the MMCM takes care of all generated or propagated clocks. No user timing constraints need to be added manually, except the XDC command **`set_false_path -through [get_nets DDRX_PHY_INIT_DONE]`**. This command is used to disable timing checks on the net that drives the GPIO[0] LED on the KC705 board, indicating that the DDR3 memory controller has successfully been initialized.

66. The design is now completed and can be implemented to bitstream and run on the board. Go to step 6 for instructions on how to generate the bitstream and download it to the board.

Simulation Testbench

This design does not support simulation. Because this is a video design with long frame times, simulations of multiple video frames would be impractical.

Equivalent XPS Design

For reference, an equivalent Xilinx Platform Studio (XPS) design added as an embedded source to the Vivado tools is provided in the reference design ZIP file. The Vivado tools project with XPS submodule tool flow is in `<design_dir>/vivado_xps`.

The XPS design has the same IP and overall functionality as the Vivado tools HDL flow design, but is delivered as a complete XPS design that can be built to a bitstream. Figure 65 shows the block diagram of the XPS design. This design differs slightly from the Vivado tools HDL design by its use of `proc_sys_reset` and `clock_generator` blocks native to XPS and the extra AXI Interconnect blocks for the AXI4-Lite connections. The AXI Interconnect blocks connecting the AXI4-Lite masters to each AXI VDMA are required for XPS modeling of AXI connections, but are essentially passive.

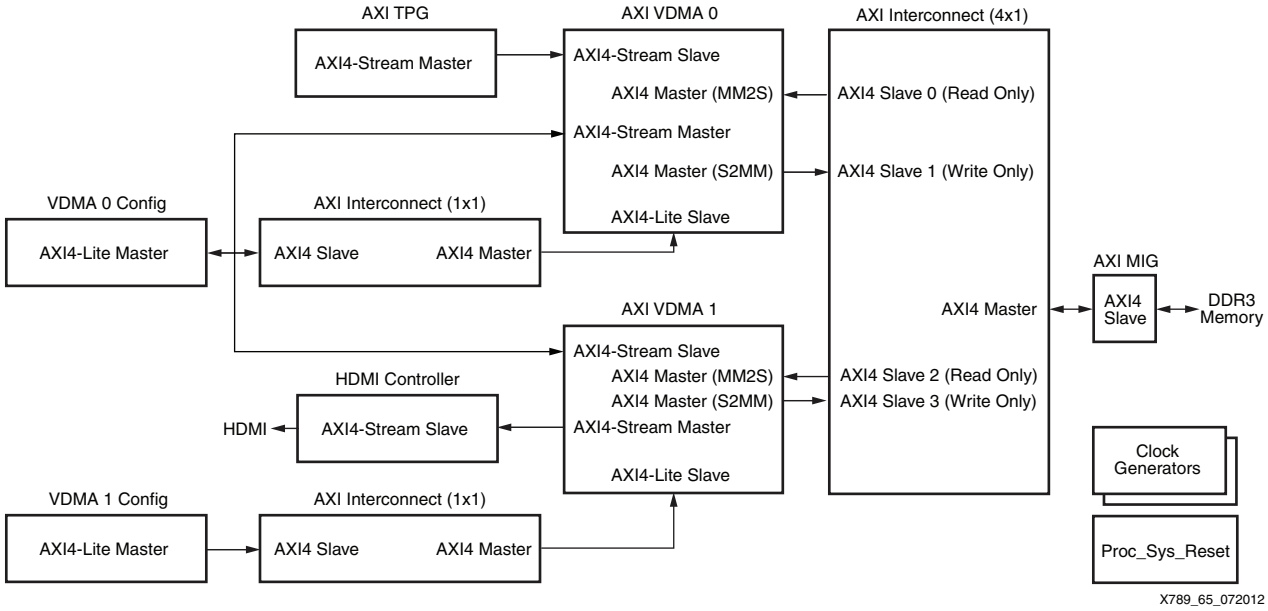


Figure 65: **Block Diagram Overview of AXI MPMC System (EDK)**

The XPS design can be rebuilt using normal build flows in the XPS tools when invoked from the Vivado tools. See *EDK Concepts, Tools, and Techniques: A Hands-On Guide to Effective Embedded* System Design [Ref 6] for more information about the XPS tools. For reference, the EDK project file for running in an XPS standalone flow under the ISE tool flow is stored in `<design_dir>/edk/system.xmp`.

Reference Design

The reference design files for this application note can be downloaded at:

https://secure.xilinx.com/webreg/clickthrough.do?cid=191117

The reference design checklist is shown in Table 2.

Table 2: **Reference Design Checklist**

Parameter	Description
General	
Developer Name	Xilinx
Target Devices (Stepping Level, ES, Production, Speed Grades)	Kintex-7 FPGAs
Source Code Provided?	Yes (except two blocks delivered as a netlist)
Source Code Format	Verilog
Design Uses Code or IP from Existing Reference Design, Application Note, 3rd party, or Vivado Software?	Reference designs provided for HDL and XPS flows based on the Vivado tools
Simulation	
Functional Simulation Performed?	N/A (simulation not supported)
Timing Simulation Performed?	N/A (simulation not supported)
Testbench Provided for Functional and Timing Simulations?	N/A (simulation not supported)
Testbench Format	N/A (simulation not supported)

Table 2: **Reference Design Checklist *(Cont'd)***

Parameter	Description
Simulator Software and Version	N/A (simulation not supported)
SPICE/IBIS Simulations?	N/A (simulation not supported)
Implementation	
Synthesis Software Tools and Version	Vivado Synthesis 2012.2
Implementation Software Tools and Version	Vivado Design Suite 2012.2
Static Timing Analysis Performed?	Yes
Hardware Verification	
Hardware Verified?	Yes
Hardware Platform Used for Verification	KC705 Board Rev D

The resource utilization and clock frequency of the system are summarized in Table 3. The system is designed to fit the FPGA resources and speed grade of the XC7K325TFFG900-2 FPGA on the KC705 board. It has not been characterized for other FPGA devices or speed grades.

Table 3: **Resource Utilization**

Parameters	Specification/Details	
Maximum Frequency (by speed grade)	-2	100 MHz (AXI), 400 MHz (Memory Clock)
Device Utilization	Slices	11,281
	LUTs	27,553
	Registers	31,085
	GCLK Buffers	4
	Block RAMs	44
HDL Language Support		Verilog
DDR3 Memory Configuration		64-bit, 400 MHz DDR3 SDRAM
Video Clock Frequency		75 MHz
AXI Interconnect and MIG Main Clocking		100 MHz

The AXI MPMC Interconnect and MIG tool are configured for medium to high performance. The system is not area optimized, but is optimized for throughput. The AXI MPMC portion of the system has more available throughput than the two AXI VDMA masters can consume. The extra available throughput can be used for expansion of the system to include additional AXI4 masters. See the AXI Reference Guide [Ref 4] for more information about optimizing the AXI MPMC for different area, timing, throughput, latency, and ease of use trade-offs.

Device resource utilization is detailed in Table 4 for each IP core in Figure 1.

Note: The information in Table 4 is taken from the Vivado tools by selecting **Open Implemented Design** and then clicking **Tool > Report Utilization** (Figure 66).

Table 4: **Module Level Resource Utilization**

IP Core	Instance Name	Slice LUTs	Slice Registers	Block RAMs
MIG	DDR3_SDRAM	12,898	7,379	0
AXI Interconnect	axi4_0	10,790	15,493	34
VDMA 0 Config	axi_lite_master_vdma_0	38	12	0
VDMA 1 Config	axi_lite_master_vdma_1	30	12	0
AXI VDMA 0	axi_vdma_0	2,532	3,127	4
AXI VMDA 1	axi_vdma_1	2,475	3,099	4
AXI TPG	axis_master_example_0	1,097	1,221	2
DVI Controller	axis_slave_example_0	594	742	0
Clock Generator	clock_generator_1	0	0	0

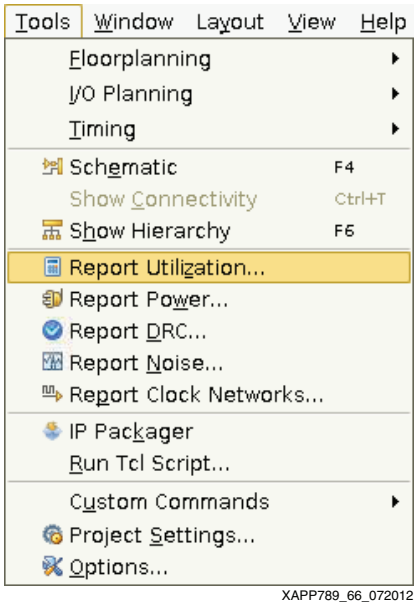


Figure 66: **Generating a Resource Utilization Report**

The report shows the resource utilization by resource type and design hierarchy (Figure 67).

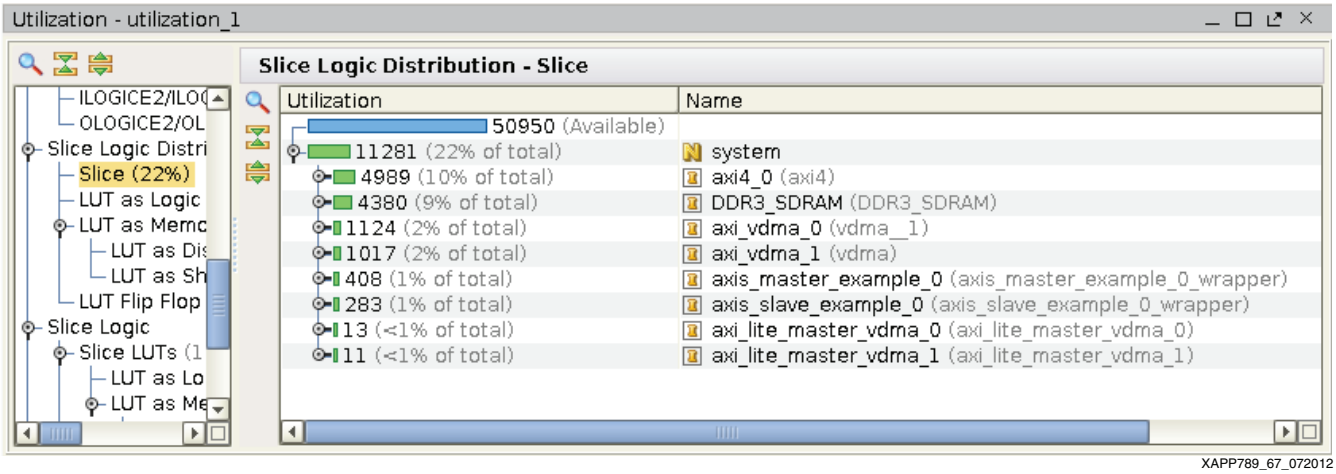


Figure 67: **Obtaining Module Level Resource Utilization**

The utilization information is approximate due to cross-boundary logic optimizations and logic sharing between modules.

Conclusion

This application note describes the steps to build a high-performance AXI MPMC (memory controller and AXI Interconnect) system using the Vivado tools. It illustrates the steps and design considerations necessary to create a working system in hardware. The AXI MPMC system is exercised by AXI VDMA IP cores moving video frames through DDR3 memory from a video test pattern generator IP block to an HDMI Display IP block. This system can be used as a template and as training information for a new design.

References

This document uses the following references:

1. Vivado Design Suite 2012.2 Documentation
   http://www.xilinx.com/products/design-tools/vivado/index.htm
2. Advanced Microcontroller Bus Architecture (AMBA) ARM AXI4 specifications
   http://www.amba.com
3. Kintex-7 FPGA KC705 Evaluation Kit
   http://www.xilinx.com/products/boards-and-kits/EK-K7-KC705-G.htm
4. UG761, *AXI Reference Guide*
5. DS768, *LogiCORE IP AXI Interconnect Data Sheet*
6. UG683, *EDK Concepts, Tools, and Techniques: A Hands-On Guide to Effective Embedded System Design*
7. UG586, *7 Series FPGAs Memory Interface Solutions User Guide*

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
07/31/12	1.0	Initial Xilinx release.

Notice of Disclaimer

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.

Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.