



XAPP1099 (v1.0) March 2, 2016

# Local Partial Reconfiguration Using Embedded Processing for 3D ICs

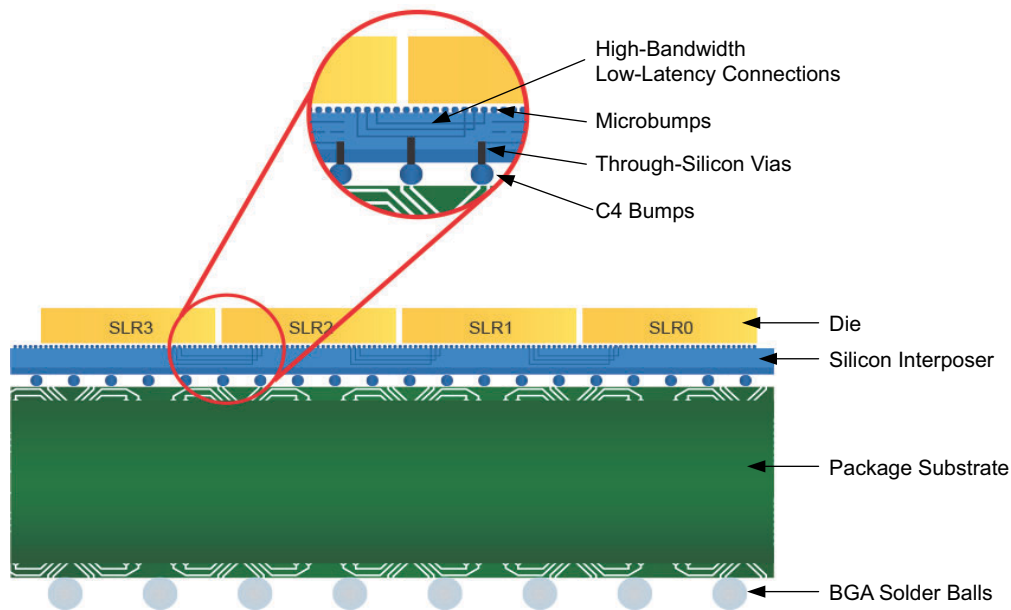
Author: George Duffy

## Summary

Xilinx All Programmable 3D ICs have a powerful capability called partial reconfiguration (PR) that gives designers the ability to time-multiplex device resources in a manner similar to when a microprocessor switches tasks. This technology gives the benefit of both a flexible software implementation and the performance afforded by a hardware implementation. The potential design applications of this technology are numerous. Examples of common applications, along with various mode and capability options, are described in the *Vivado Design Suite: Partial Reconfiguration User Guide* (UG909) [Ref 1].

The 3D ICs utilize stacked silicon interconnect (SSI) technology to combine, early in the product life cycle, multiple smaller programmable devices into a single larger package to maximize capacity and performance. See the *Large FPGA Methodology Guide, Including SSI Technology* (UG872) [Ref 2]. The Virtex-7 2000T device (Figure 1) consists of four connected super-logic regions (SLR). An SLR is a single die slice contained in an 3D IC.

Although the combination of these two technologies offers unprecedented flexibility and performance in design functionality, there are limitations that need to be managed in the design process. This application note describes and gives examples on managing designs using partial reconfiguration and SSI technology.



X15829-012516

Figure 1: Virtex-7 2000T FPGA Enabled by SSI Technology

One method of performing partial reconfiguration uses the master internal configuration access port (ICAP). The ICAP interface provides the programmable logic with access to the 7 series configuration system. However, there are restrictions when using ICAP that must be followed.

- When using any initial configuration method other than SPI, the ICAP located in the master SLR (SLR1) must be used for partial bitstream delivery, unless the reconfigurable partition is in the same SLR as the slave ICAP being targeted.
- When configuring in SPI modes, the master ICAP cannot access configuration memory in slave SLRs.
- When the mode pins are set to JTAG mode, the master ICAP cannot access slave SLRs.
- When configuring in SPI x2 or SPI x4 modes, you cannot use the ICAP in the slave SLRs. Local ICAP access is available for all other configuration modes.
- In SPIx2 or SPIx4 configuration modes, you cannot use SEM IP (error correction).

Refer to the *7 Series FPGAs Configuration User Guide* (UG470) [Ref 3] for more information on ICAP access to configuration memory for SSI devices.



---

**IMPORTANT:** For 7 series devices, the PR regions cannot cross the SLR boundaries and must be entirely confined within a single SLR. This restriction is resolved in UltraScale™ architecture-based devices.

---

One approach that addresses the restrictions imposed by using only the SPI x1 configuration mode is to implement a MicroBlaze design controlling an ICAP on each SLR. You can apply the relevant partial bitstream to the correct ICAP. When using the IP Integrator (IPI), a MicroBlaze design can be added to existing processor designs without significant overhead. An alternative or complementary approach to the embedded solution presented in this application note uses the Xilinx partial reconfiguration controller IP, which is included in the Vivado™ Design Suite, to manage the design reconfiguration. The *Partial Reconfiguration Controller (v1.0) Product Guide* (PG193) [Ref 4] describes this core.

This application note describes the methodology for creating an IPI design. It can control up to four ICAPs and can reconfigure modules local to a specific ICAP. The reference design provides the resources necessary to launch a Vivado software-based partial reconfiguration design. For more information about the partial reconfiguration design flow, refer to the *Vivado Design Suite: Partial Reconfiguration User Guide* (UG909) [Ref 1]. The reference design is a guideline for developing partial reconfiguration solutions on your board. It implements a MicroBlaze system connected to an ICAP on each SLR and targets a Virtex-7 2000T device (XC7V2000T, FHG1761 package, -1 speed specification). Although the design approach can target any Xilinx SSI device, including UltraScale devices, there are some different capabilities as new features are added. For example, in UltraScale devices a reconfigurable module can cross SLR boundaries. For a full list of capabilities and any limitations of partial reconfiguration on different device families, refer to the *7 Series FPGAs Configuration User Guide* (UG470) [Ref 3].

---

## Introduction

The IPI design described in this application note consists of a MicroBlaze™ system, a clocking wizard, a UART Lite interface, and four axi\_hwicap IPs (one for each SLR). For an example of building a MicroBlaze-based IPI design, refer to the *Vivado Design Suite Tutorial: Embedded Processor Hardware Design* (UG940) [Ref 5].

The MicroBlaze system is used to manage all the operations of this design, from selecting the SLR to be reconfigured to loading the chosen partial bitstream into the ICAP. In this example, partial bitstreams are transferred byte-by-byte using the UART interface to the MicroBlaze system, which then performs the writes to the ICAP. While this setup is for demonstration purposes only, the bitstream delivery could be used on board resources such as SPI, BPI, SD, or Ethernet to retrieve the partial bitstreams.

---

## Design Flow Overview

Implementing a partially reconfigurable design is similar to implementing multiple non-PR designs that share common logic. Partitions are used to ensure that the common logic between the multiple designs is identical.

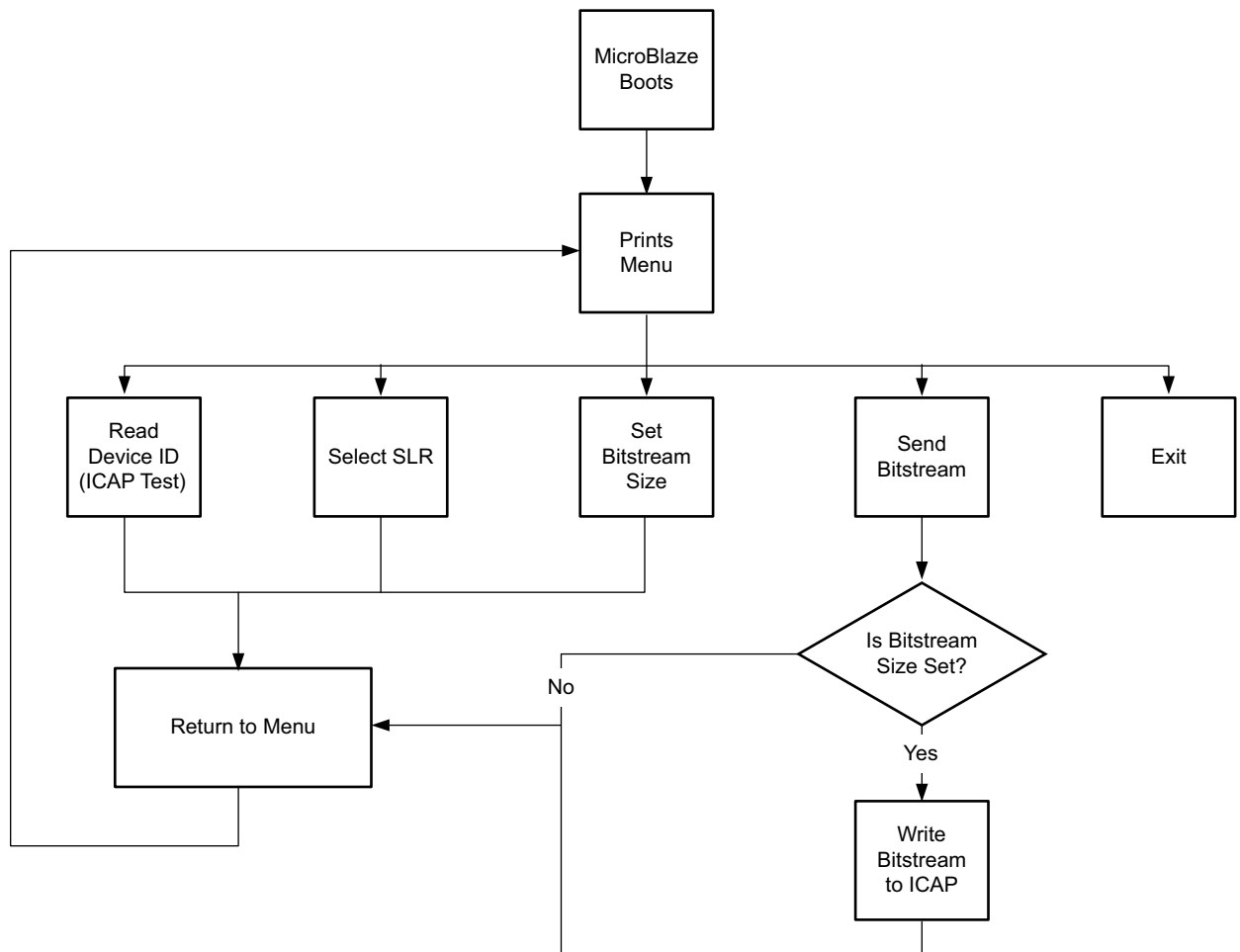
The HDL sources for each module must be synthesized separately and are then added to the static logic netlist. The appropriate netlists are implemented in each design to generate the full and partial bit files for that configuration. The static logic from the first implementation is shared among all subsequent design implementations. A detailed description of the full partial reconfiguration flow is available in the *7 Series FPGAs Configuration User Guide* (UG470) [Ref 3]. More examples of using this flow are found in the *Vivado Design Suite Tutorial: Partial Reconfiguration* (UG947) [Ref 6].

The following is a high-level description of the partial reconfiguration flow.

1. Synthesize the static and reconfigurable modules separately.
2. Create physical constraints (Pblocks) to define the reconfigurable regions.
3. Set the HD.RECONFIGURABLE property on each reconfigurable partition.
4. Implement a complete design (static and one reconfigurable module per reconfigurable partition) in context.
5. Save a design checkpoint for the fully routed design.
6. Remove reconfigurable modules from this design and save a static-only design checkpoint.
7. Lock the static placement and routing.
8. Add new reconfigurable modules to the static design and implement this new configuration.
9. Repeat [step 8](#) until all reconfigurable modules are implemented.
10. Run a verification utility (pr\_verify) on all configurations.
11. Create bitstreams for each configuration.

## Application Software

The application software is compiled for a MicroBlaze system using the Xilinx Software Development Kit (SDK) [Ref 7] and includes binaries and source code in the workspace. A high-level flow diagram is shown in Figure 2. You set the bitstream size and select the SLR that contains the reconfigured module for loading. When you are ready to send a bitstream, the processor is signaled and an acknowledgment of readiness is returned. You use the file transfer function on the Tera Term [Ref 8] to deliver the bitstream to the processor, which then writes to the selected ICAP, and partially reconfigures the device.



X15844-010616

Figure 2: Application Flowchart

The MicroBlaze system continues to write received words to the ICAP until the number of bytes written matches the input bitstream size.

---

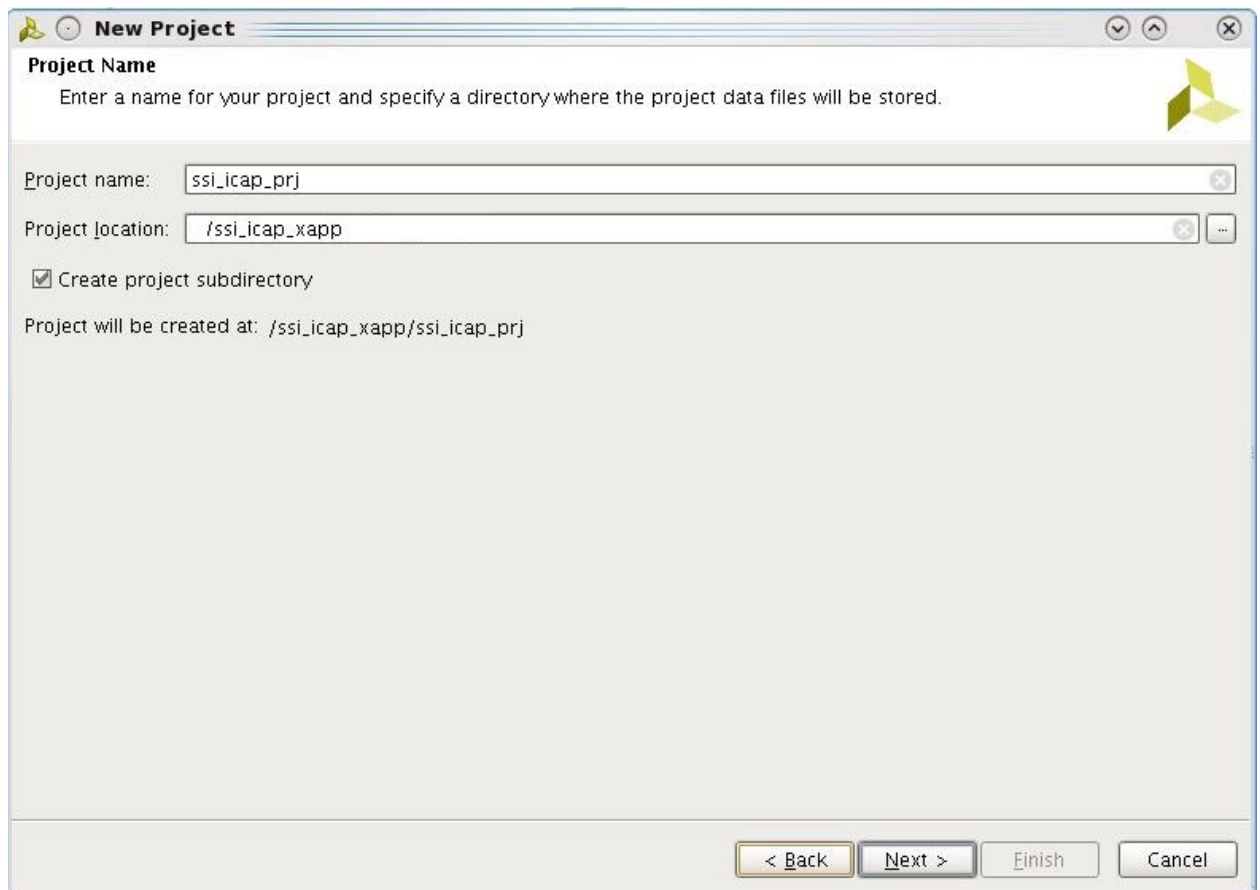
## Creating Your Processor System Project

Download the [Reference Design](#) file from the Xilinx website. Extract the ZIP file contents to any write-accessible location. The unzipped ssi\_icap\_xapp data directory is referred to in this application note as the <Extract\_Dir>. The project portion of the design creates the IPI portion

of the design and synthesizes the static portion of the design. Bottom-up synthesis is already used on two reconfigurable blocks to create the provided netlists. One of these netlists shifts six LEDs left, while the other shifts them right. There is also a pushbutton in the design that selects the reconfigurable partition that drives the LEDs. This allows for visual verification of a successful reconfiguration on each SLR by multiplexing the output.

## Creating a New Project

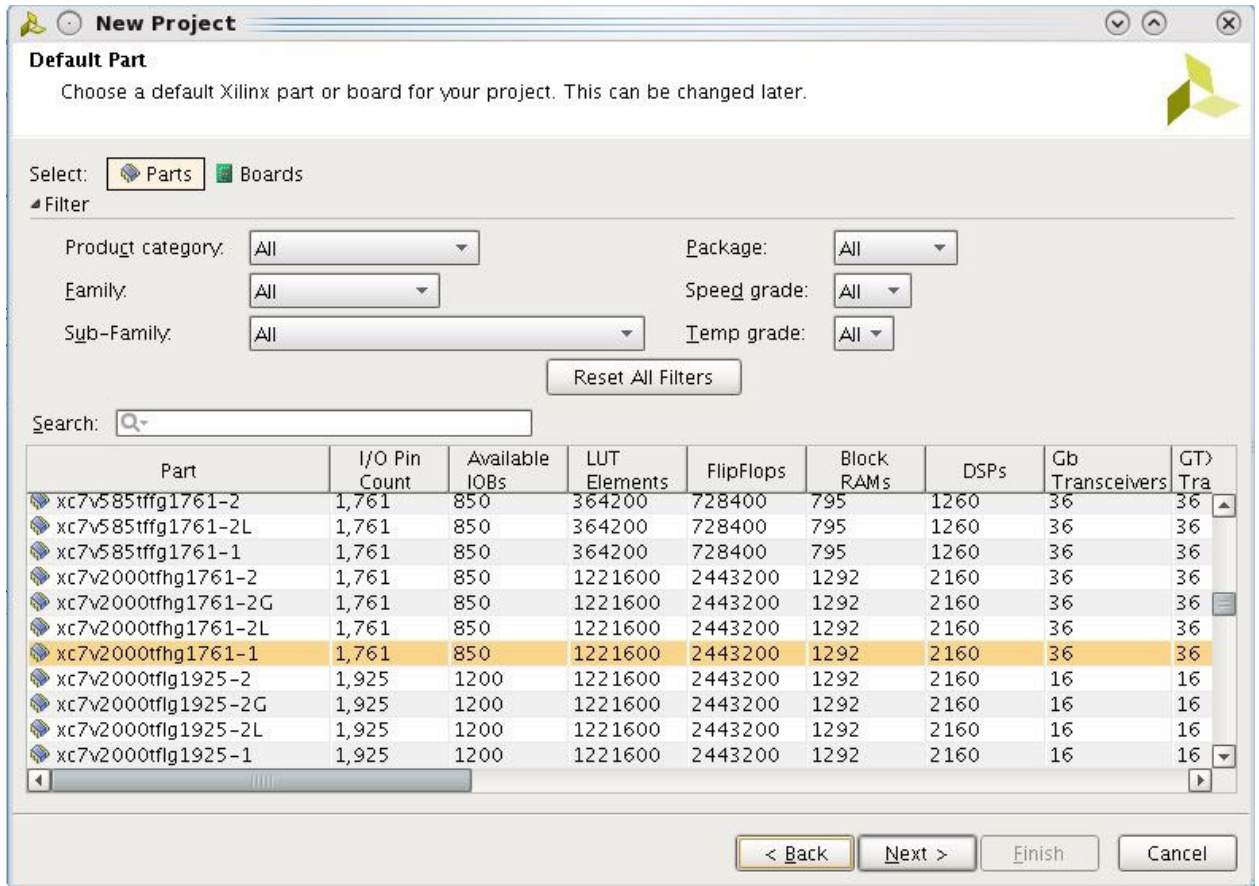
1. Open the Vivado Design Suite version 2015.3 or later.
2. The following steps detail the creation of a new project. There is an already created project under <Extract\_DIR>/ssi\_icap\_base. This project contains all the required files with the design flow already completed. In this case, it is assumed that the device is fitted to a VC707 board and the I/O constraints do not match any custom boards. You must provide your own I/O constraints for the design. Do not use the included bitstream files.
3. Open the Vivado Design Suite from the Getting Started page and select *Create a New Project*. Alternatively, you can source <Extract\_DIR>/scripts/project.tcl to generate the project and skip to [step 22](#). Click *Next*.
4. As shown in [Figure 3](#), enter the project name, *ssi\_icap\_prj*, and place the project in the location <Extract\_Dir>. The *Create project subdirectory* must be checked. Click *Next*.



X15831-012516

Figure 3: Create a New Project Wizard

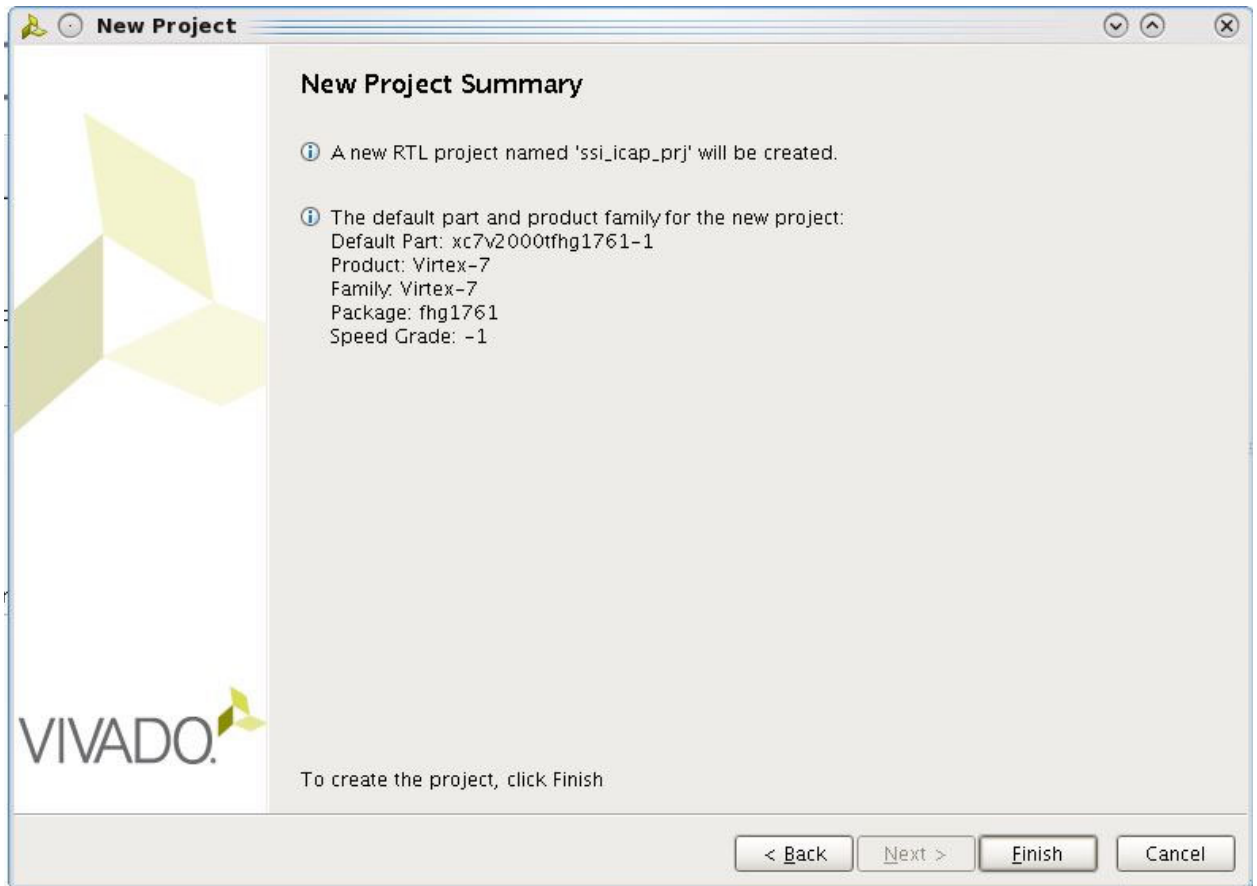
5. For an *RTL Project* → Do not specify sources at this time → click *Next*.
6. In the part selection window (Figure 4), select the xc7v2000tfhg1761-1. If your example targets a different SSI device, fill in the specific part and click *Next*.



X15832-012516

Figure 4: Device Selection GUI

7. On the *New Project Summary* page (Figure 5), verify the selected options are correct, and click *Finish*.

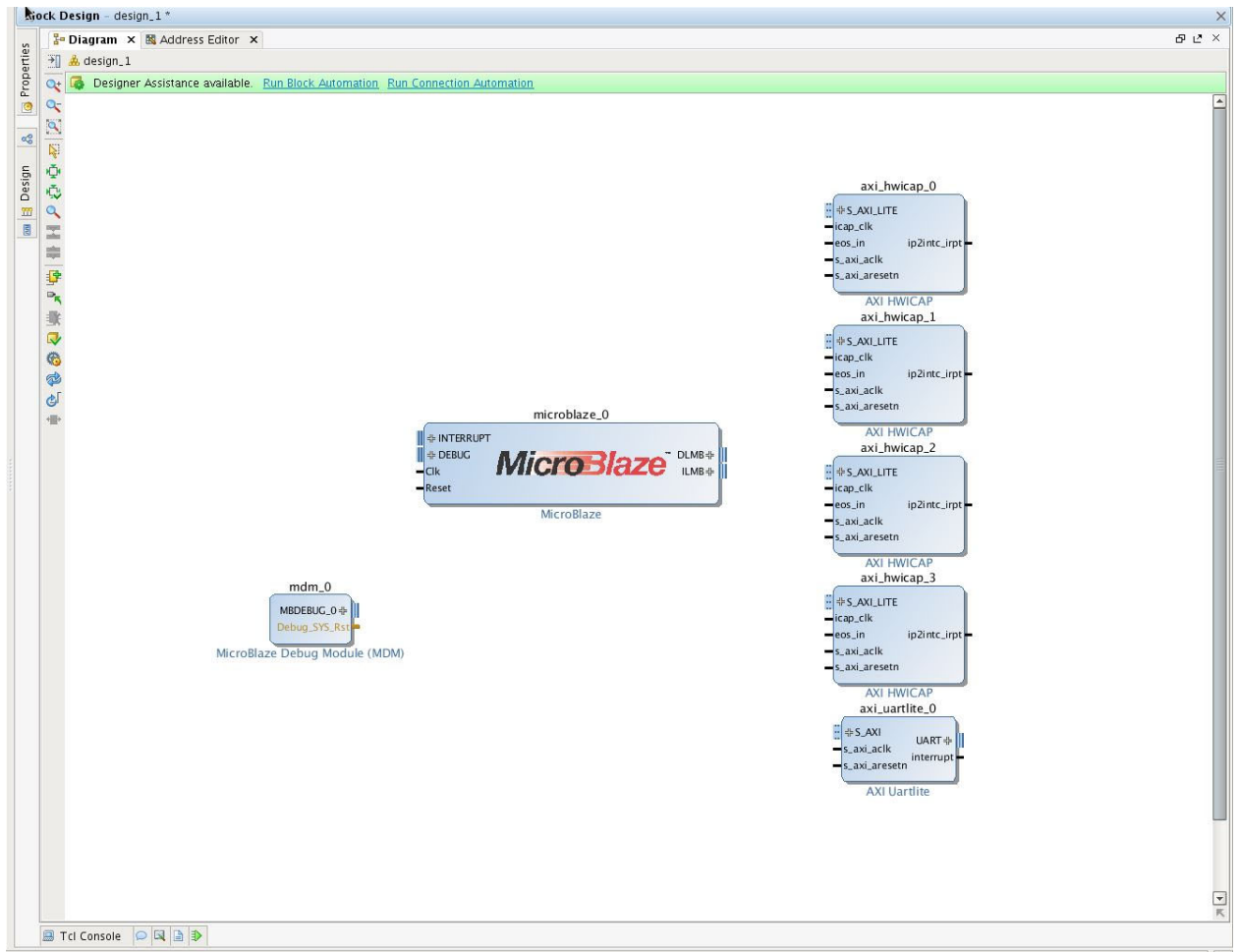


X15833-012516

Figure 5: Project Summary Page

8. When the project initialization is complete, select *create block design* in the project manager and name it *hwicap\_design1*. Leave the directory local to the project and set the source as *Design sources*. Alternatively, you can source `<Extract_DIR>/scripts/build_bd_design.tcl` to build the block design for you and skip to [step 22](#). Click *OK*.

- In [Figure 6](#), right click on the empty diagram to select *add IP*. Add a MicroBlaze processor, a MicroBlaze debug module, four AXI hardware ICAP IPs, and an AXI universal asynchronous receiver transmitter (UART) Lite IP.

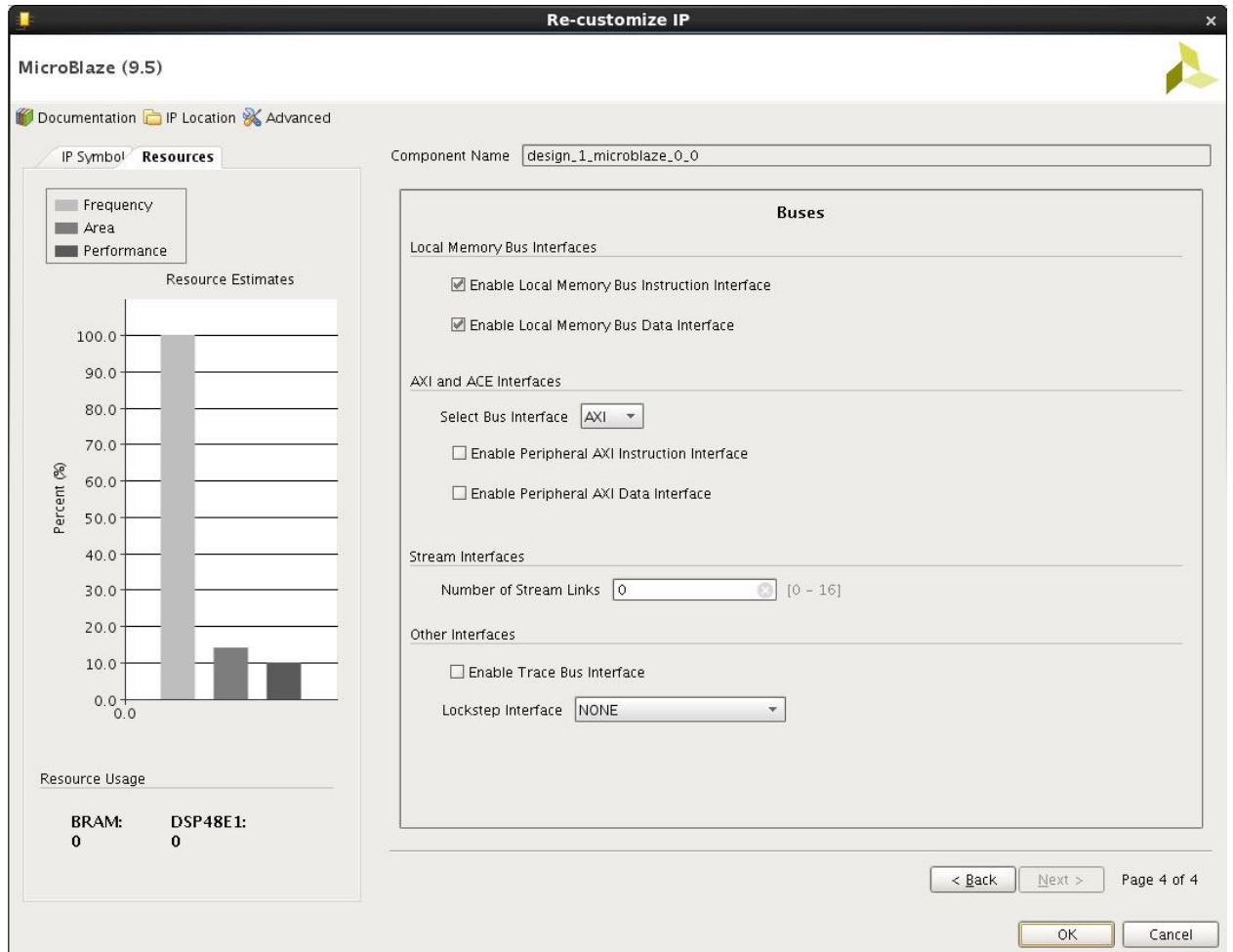


X15834-012516

Figure 6: Initial IPI Project



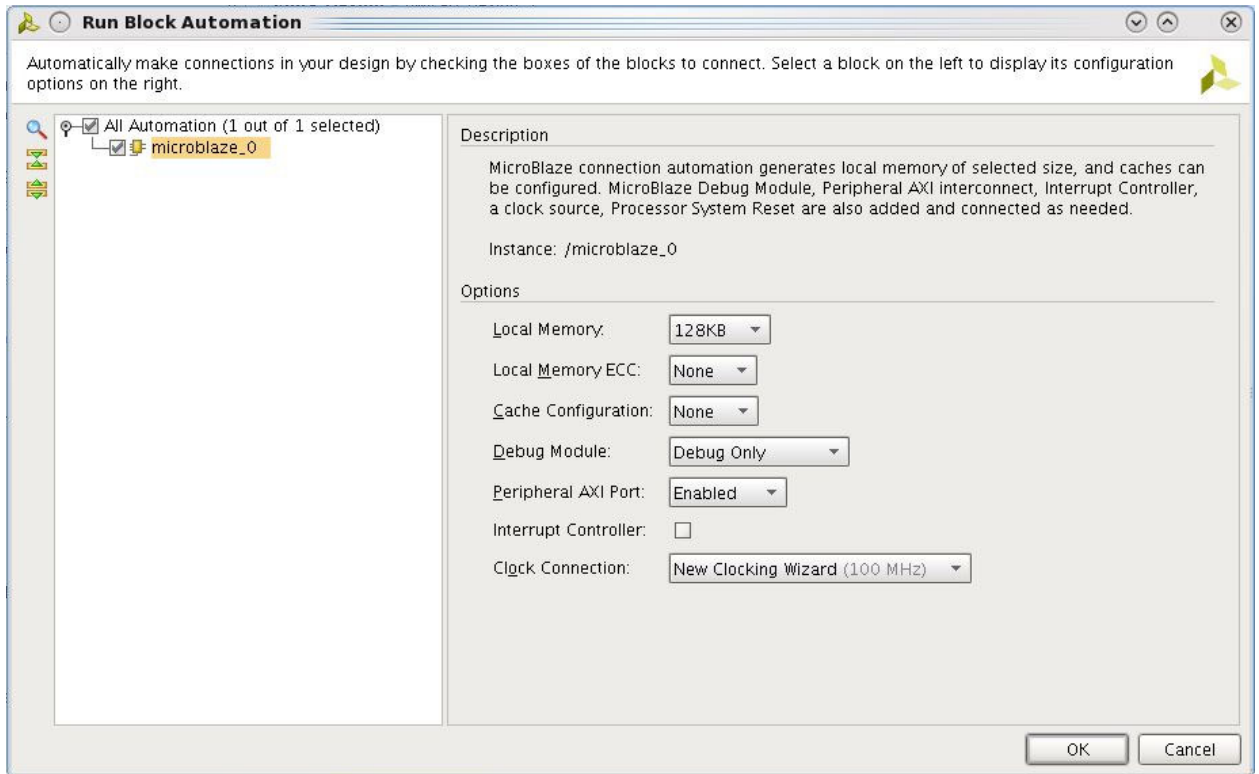
- Double click the MicroBlaze page to begin customization. Check the *Enable MicroBlaze Debug Module Interface*. Click *Next* until the fourth page (Figure 7). Select to enable both the local memory bus instruction and data interfaces. Click *OK* to finish customization.



X15835-012516

Figure 7: MicroBlaze Customization Wizard

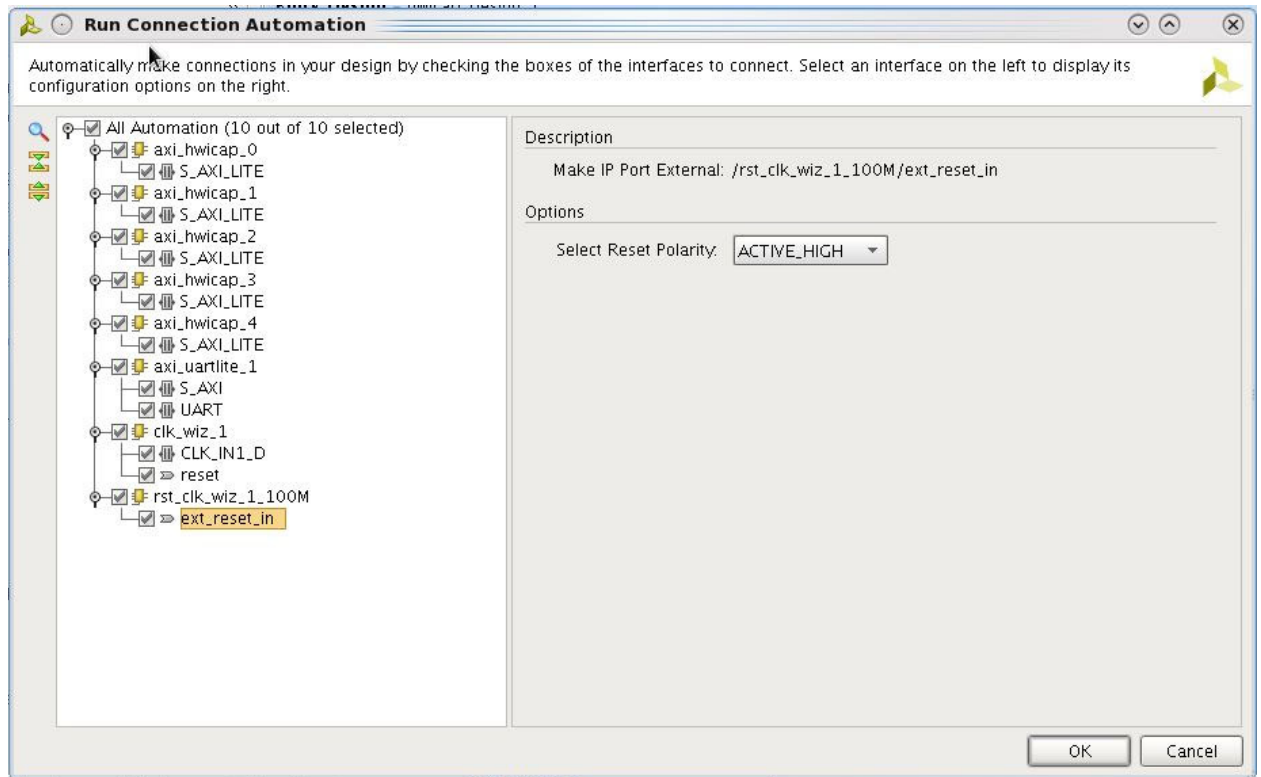
- Where it says that *Designer Assistance available*, Click to get the *Run Block Automation* page (Figure 8). Change *Local Memory* to 128 KB and *Cache Configuration* to *None*. Leave all other values at their defaults. Click *OK*. This step automatically adds a clocking wizard IP and the block RAM resources for the processor.



X15836-012516

Figure 8: Block Automation Page

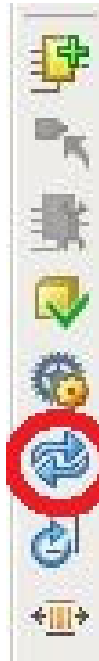
- Click to get the *Run Connection Automation* page and check the *All Automation* box (Figure 9). Select the *ext\_reset\_in* port for the *rst\_clk\_wiz* IP and change the polarity to ACTIVE\_HIGH. Click OK.



X15837-012516

Figure 9: Connection Automation

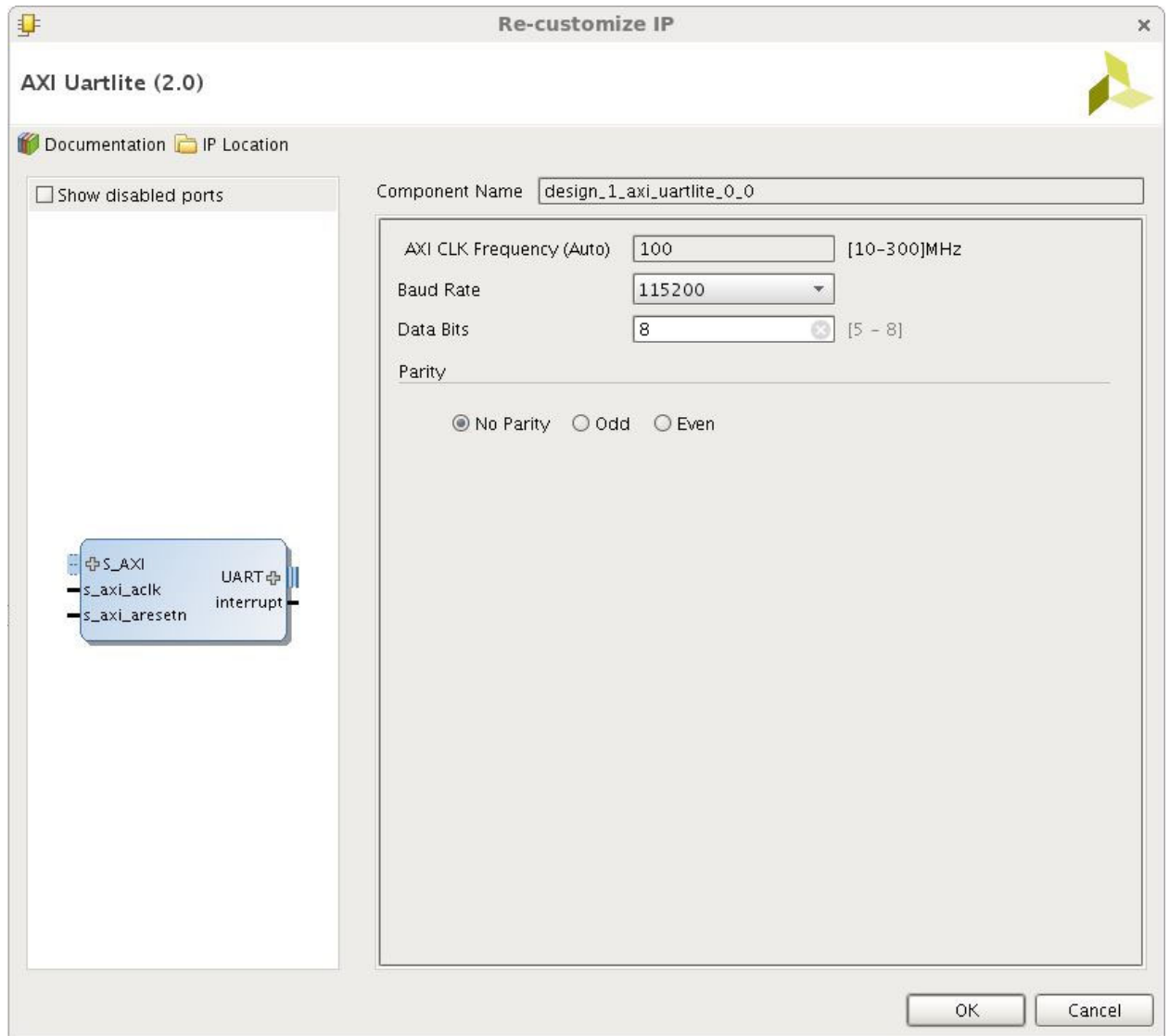
13. Click the regenerate layout button (Figure 10) to automatically tidy up the design layout.



X15838-012516

Figure 10: **Regenerate Layout Button**

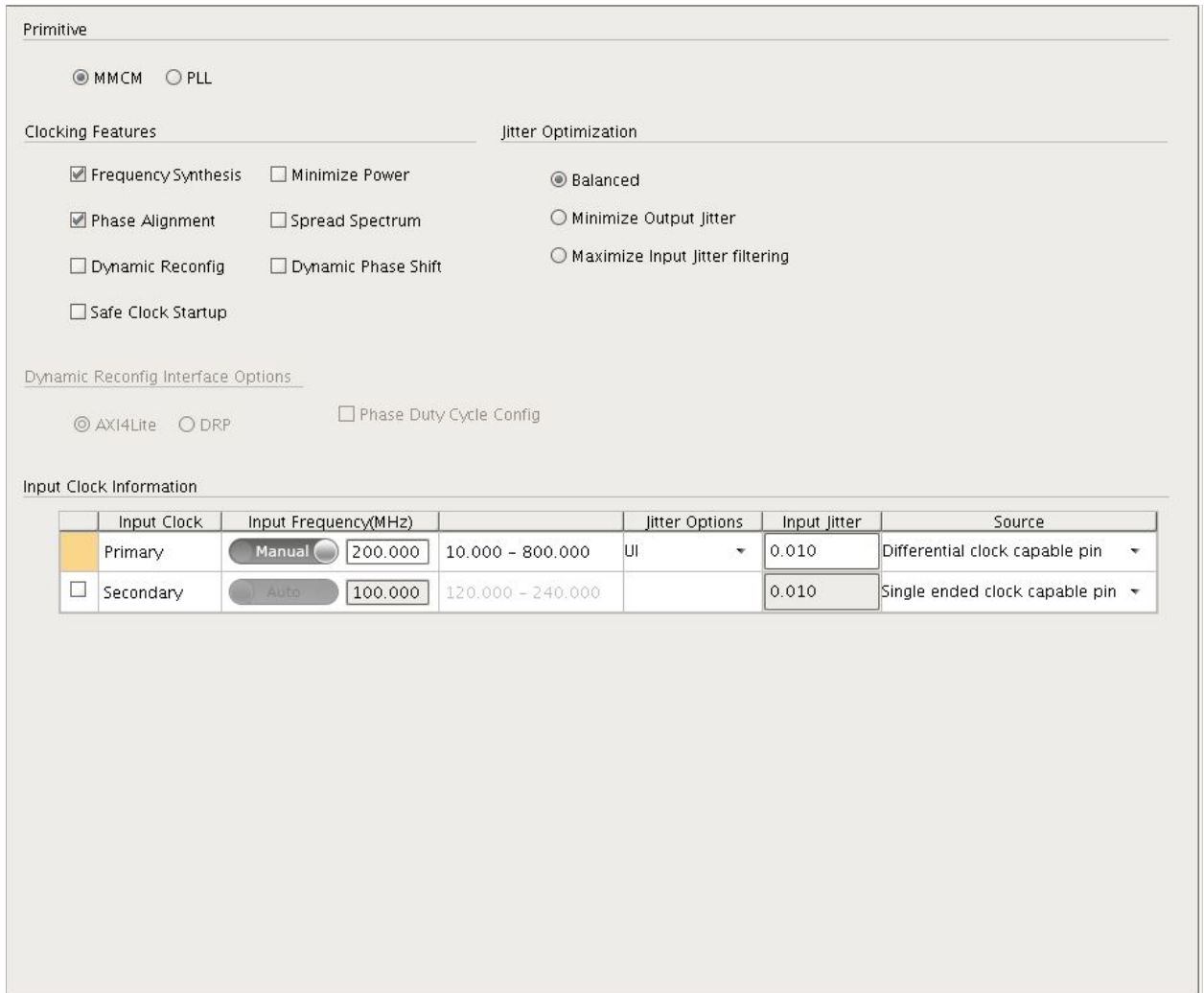
- Double click the AXI UART Lite (Figure 11). Set the baud rate to 115200 with eight data bits and no parity. Click OK.



X15839-012516

Figure 11: AXI UART Lite GUI

- Double click on the clocking wizard (Figure 12). This design uses a 200 MHz differential input clock. A custom board or a different clock input needs you to input the correct input frequency and select the appropriate source type. Open the output clock tab and make sure the output is set to 100 MHz. Click OK.

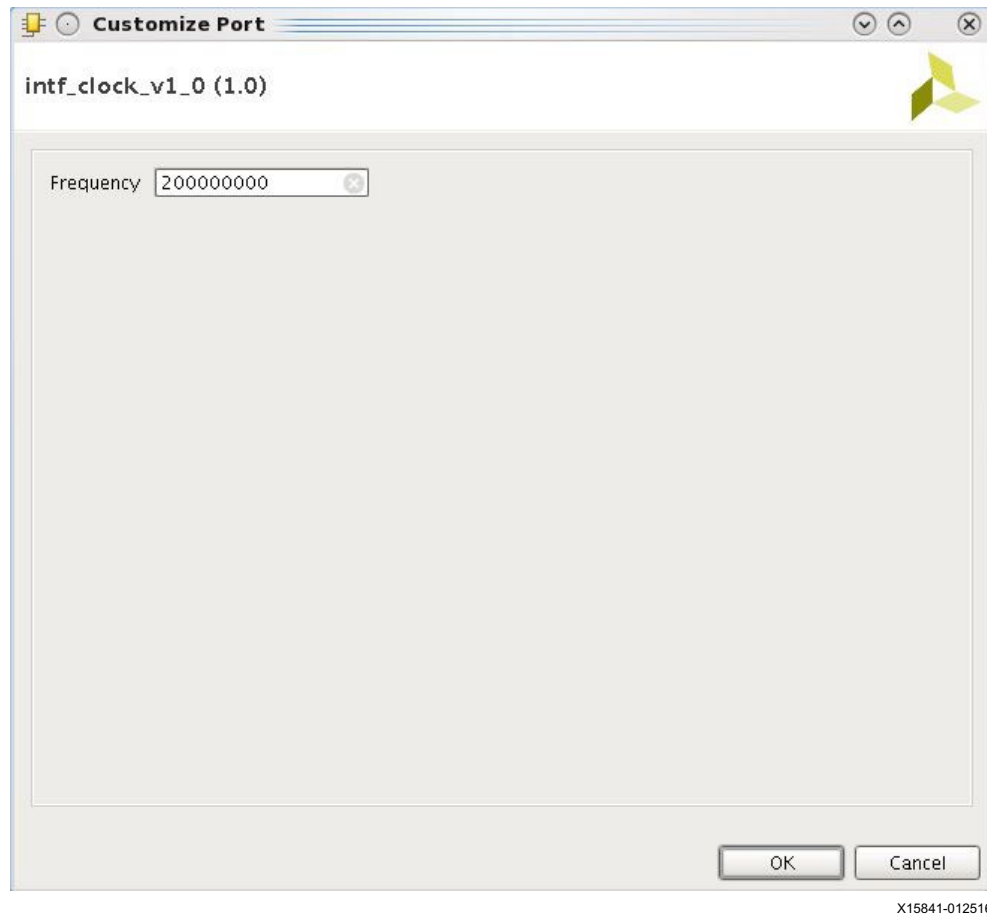


X15840-012516

Figure 12: Clocking Wizard Customize GUI

- Connect the `clk_out1` port from the clocking wizard to the `icap_clk` port on the AXI HWICAP blocks. Right click on the `clock_out1` port on the clocking wizard and select `Create_Port`. Leave the default selections. Click OK.

17. Double click the *diff\_clock\_rtl* port and set the frequency to match your input clock (Figure 13). Click OK.

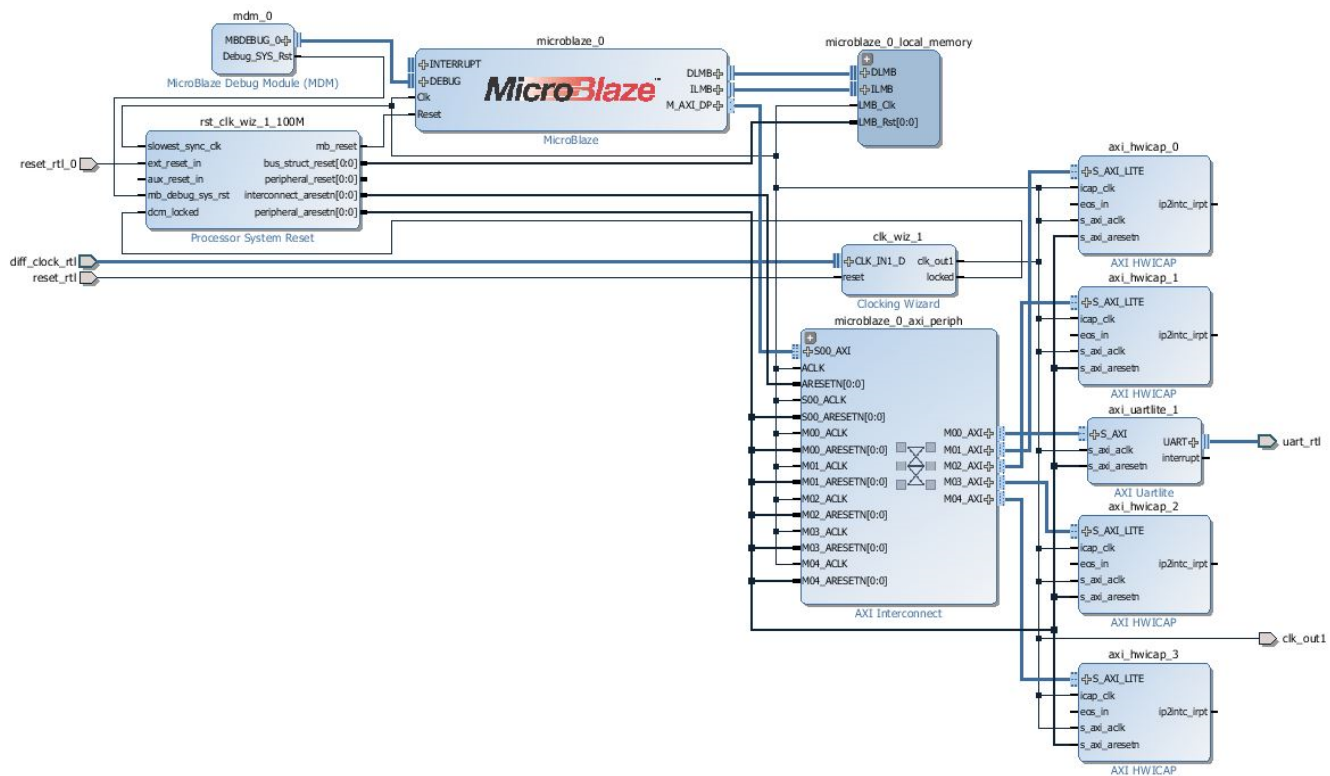


X15841-012516

Figure 13: Input Clock Frequency Setting

18. Click the *Validate\_design* button. This ensures that your IPI design is fully legal and has no port mismatches.

19. Your completed IPI design should look like Figure 14.



X15842-012516

Figure 14: Example of a Completed IPI Design

20. The next step is to import the Verilog source files for the top module. It will instantiate the block design and the reconfigurable modules used in the design. Right click in the sources tab and click *Add Sources*. Select to add or create design sources and click *Next*. Select the files:

```
<Extract_DIR>/Sources/hdl/hwicap_design_1_wrapper.v
<Extract_DIR>/Sources/hdl/shift_wrapper.v
<Extract_DIR>/Sources/hdl/debounce.v
```

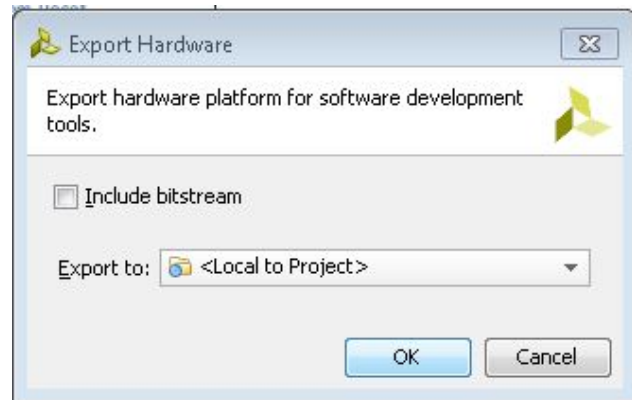
Also ensure the *Copy sources into project box* is checked.

21. Add sources again, but this time select to *add or create constraint sources*. Add the file `<Extract_DIR>/Sources/xdc/hwicap_const.xdc` to the project. This contains constraints for the different p-blocks and LOC constraints for the ICAP sites, putting them all on a separate SLR. The Virtex-7 2000T is not on a Xilinx evaluation board, there are no clock or I/O constraints to match your design setup. To run the design on hardware, edit the file and add these constraints for your own application. The XDC contains all the necessary constraints. The set package pins and clock period constraints are all that need to be changed.

22. Open the sources tab and right click on `hwicap_design1.bd`. Click *Generate Output Products*. Click *Generate*.



23. Now that this portion of the design is complete, it is necessary to export the hardware specification for use later when creating the software application in the Xilinx SDK. Click *File* → *Export* → *Export Hardware*. Select to export *Local to Project* and click *OK*. This creates a folder in the project directory called `ssi_icap_prj.sdk`, which contains the HDF file required by the tools to create an application.



X15843-012516

Figure 15: Export Hardware

24. To run the partial reconfiguration implementation flow, source `<Extract_Dir>/scripts/design.tcl` in the TCL console. This script synthesizes the design and then runs through the partial reconfiguration design flow and generates all the full and partial bitstreams needed to run the design on hardware. The bitstreams are in `<Extract_Dir>/ssi_icap_prj/implement/`. These bitstreams can be combined with the MicroBlaze system application ELF file embedded in the block RAM init values using the `updatemem` command. This allows the application to be run as soon as configuration is complete. Only the initial full bitstream needs this process. Details are further documented in chapter 6 of the *Vivado Design Suite Tutorial: Embedded Processor Hardware Design* (UG940) [Ref 5].

## SDK Steps

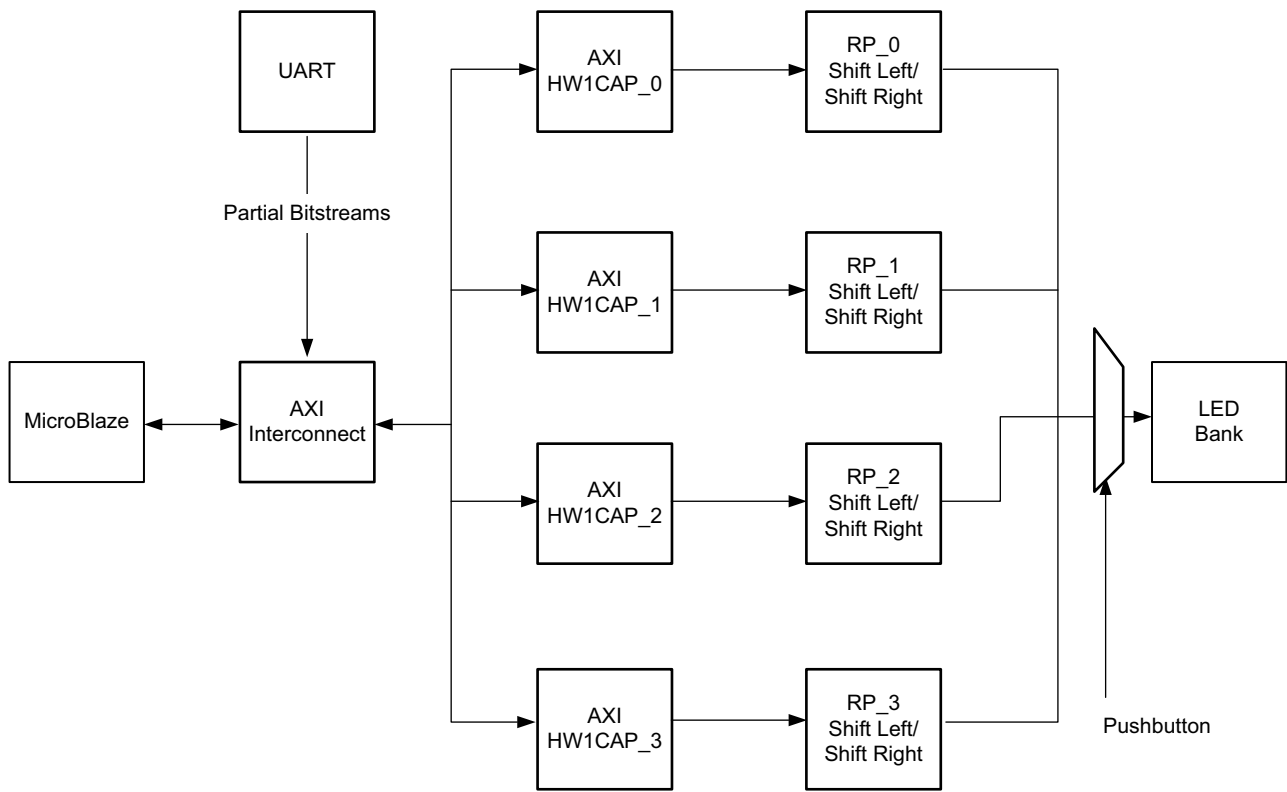
A pre-prepared software workspace with compiled binaries is in `<Extract_Dir>/sw/workspace`. It is included for reference to observe application control of the multiple ICAP instances. To create a similar workspace from scratch, refer to the steps outlined in the *Vivado Design Suite Tutorial: Embedded Processor Hardware Design* (UG940) [Ref 5] and use the hardware platform specification exported from your earlier Vivado project.

## Running the Design on Hardware

To run the reference design on hardware, you need the Silicon Labs drivers that are available on the Silicon Labs website for *CP210x USB-to-UART Bridge VCP Drivers* [Ref 9]. The terminal emulation application, such as Tera Term, is available on the *Tera Term Home Page* [Ref 8].

This design consist of four reconfigurable partitions (RP). A single High bit is shifted to the left or to the right along six bits that are output to LEDs on the board. Each RP is placed in its own SLR on the device. The outputs of the RPs are multiplexed so that only one SLR output is

showing on the LEDs at any one time. The two leftmost LEDs indicate, in binary, the current SLR output. You can select the output RP using the connected push button. A block diagram of the design is shown in [Figure 16](#).

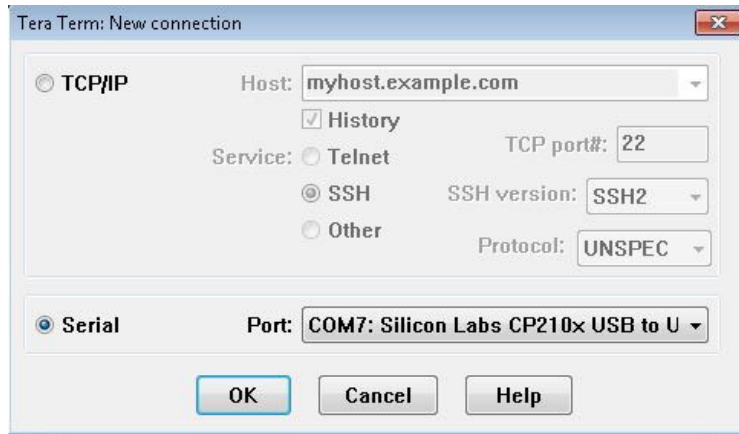


X15844-012516

**Figure 16: Block Diagram of Full Design**

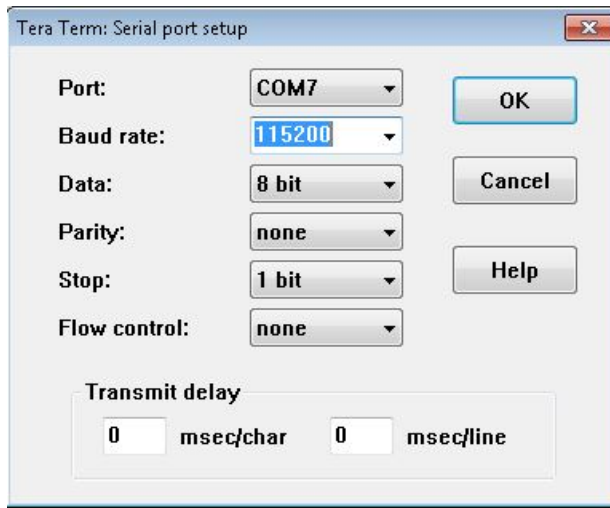
To observe a successful reconfiguration, use the selected local ICAP and monitor the LED pattern as it changes from a left shifting to a right shifting pattern (or a right to left-shifting pattern). Follow these steps to run this design on hardware.

1. Open a Tera Term serial connection at the relevant port (Figure 17) and baud rate (Figure 18).



X15845-012516

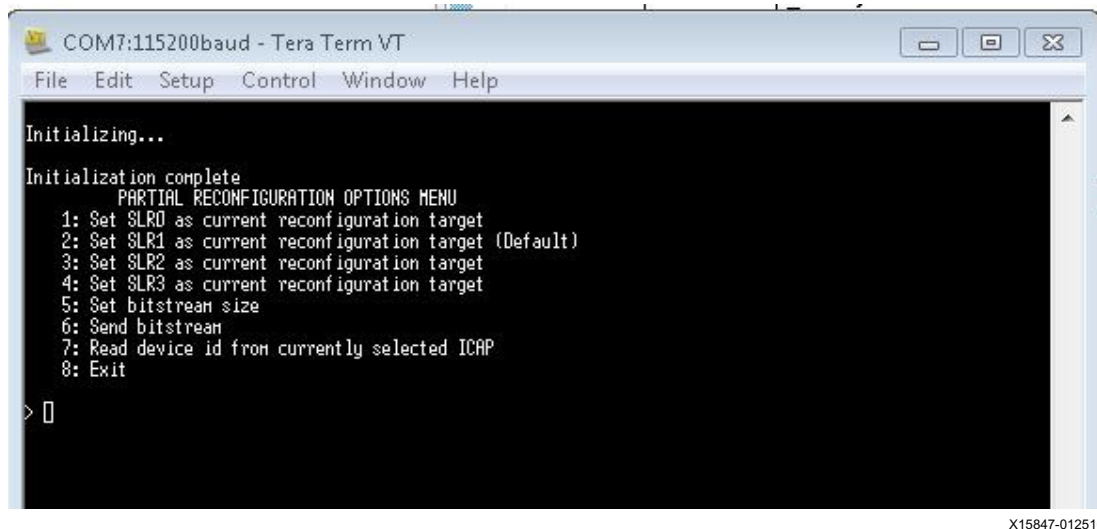
Figure 17: Selecting the Communications Port in Tera Term



X15846-012516

Figure 18: Setting Baud Rate for Communications Port

- Using the Vivado hardware manager, load the first configuration bitstream onto the device. Run the application software as demonstrated in *Vivado Design Suite Tutorial: Partial Reconfiguration (UG947)* [Ref 6]. The software menu is printed to the terminal (Figure 19).



```

COM7:115200baud - Tera Term VT
File Edit Setup Control Window Help
Initializing...
Initialization complete
PARTIAL RECONFIGURATION OPTIONS MENU
1: Set SLR0 as current reconfiguration target
2: Set SLR1 as current reconfiguration target (Default)
3: Set SLR2 as current reconfiguration target
4: Set SLR3 as current reconfiguration target
5: Set bitstream size
6: Send bitstream
7: Read device id from currently selected ICAP
8: Exit
>
  
```

X15847-012516

Figure 19: User Application Menu Printed to Tera Term

- In Tera Term, follow the menu to set the SLR to reconfigure and enter the partial bitstream size.
- Select *Send Bitstream* from the menu and a message appears instructing you to use Tera Term to send the partial.

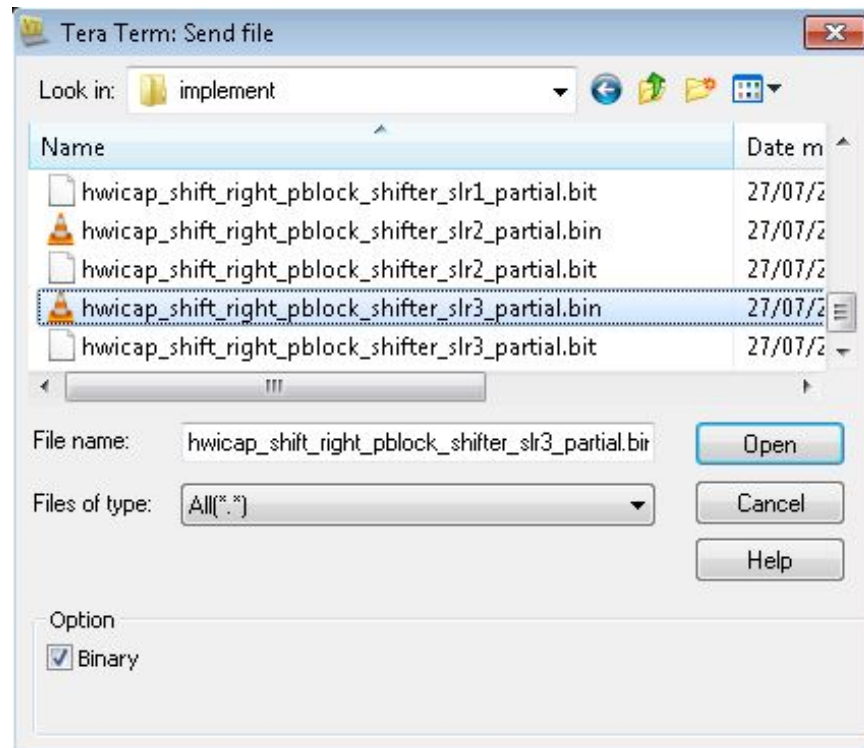


**CAUTION!** Do not press any more keys at this stage. Any bytes sent over the serial port will be written to ICAP and will invalidate your reconfiguration. If this occurs, power cycle the board and restart hardware testing.



**TIP:** This same software approach can be used for designs targeting UltraScale devices, either using a clear and partial bitstream concatenated together or setting the bitstream size for each and individually loading.

5. Select File → Send file in Tera Term, then select your partial BIN file and ensure the binary box is checked (Figure 20).



X15848-012516

Figure 20: Sending Bin File over UART Connection

6. Select Open and a file transfer status box appears.
7. The transfer is complete when the MicroBlaze system application states Full file sent and reprints the user menu.
8. You can visually verify that the correct partial is loaded by inspecting the LEDs and observing the first RP lights in a left-shifting pattern while the second RP lights in a right shirting patten.

## Reference Design

You can download the [Reference Design Files](#) for this application note from the Xilinx website.

**Table 1** shows the reference design matrix. It indicates the tool flow and verification procedures used for the reference design.

**Table 1: Reference Design Matrix**

Parameter	Description
<b>General</b>	
Developer name	Xilinx
Target devices	XC7V2000T
Source code provided	Yes
Source code format	VHDL/Verilog
Design uses code and IP from existing Xilinx application note and reference designs or third party	Vivado Design Suite
<b>Simulation</b>	
Functional simulation performed	No
Timing simulation performed	No
Test bench used for functional and timing simulations	No
Test bench format	
Simulator software/version used	
SPICE/IBIS simulations	No
<b>Implementation</b>	
Synthesis software tools/versions used	
Implementation software tools/versions used	
Static timing analysis performed	No
<b>Hardware Verification</b>	
Hardware verified	Yes
Hardware platform used for verification	

## Resource Utilization

The FPGA resources utilized by the reference design are summarized in [Table 2](#). Numbers outside the brackets are for a single block. Numbers inside the brackets are for a design with four modules.

*Table 2: Resource Usage Between Static and Reconfigurable Partitions*

Resources	Static Partition	Shift Left Single RP (Total for Four RPs)	Shift Right Single RP (Total for Four RPs)
LUTs	2876	12 (48)	11 (44)
Flip-Flops	5201	33 (132)	33 (132)
Block RAMs	36	0	0

---

## References

1. *Vivado Design Suite: Partial Reconfiguration User Guide* ([UG909](#))
2. *Large FPGA Methodology Guide, Including SSI Technology* ([UG872](#))
3. *7 Series FPGAs Configuration User Guide* ([UG470](#))
4. *Partial Reconfiguration Controller (v1.0) Product Guide* ([PG193](#))
5. *Vivado Design Suite Tutorial: Embedded Processor Hardware Design* ([UG940](#))
6. *Vivado Design Suite Tutorial: Partial Reconfiguration* ([UG947](#))
7. The [Xilinx Software Development Kit \(SDK\)](#)
8. The [Tera Term Home Page](#) (English)
9. The Silicon Labs website for [CP210x USB-to-UART Bridge VCP Drivers](#)

---

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/02/2016	1.0	Initial Xilinx release.

---

## Please Read: Important Legal Notices

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 Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; 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 such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.

© Copyright 2016 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.