Summary

This application note describes how to implement security- or safety-critical designs using the Xilinx® Isolation Design Flow (IDF) with the Xilinx Vivado® Design Suite. Design applications include information assurance (single-chip cryptography), avionics, automotive, and industrial applications. This document explains how to:

- Implement isolated functions in a Xilinx UltraScale+™ device or a Zynq® UltraScale+™ MPSoC
- Verify the isolation using the Xilinx Vivado Isolation Verifier (VIV)

To add additional security to your design, purchase the Security Monitor IP developed by Xilinx. If you embed this IP, modifications to the steps in this document must be made. For more information, see Integration and Verification of Security Monitor 3.0 for Zynq UltraScale. Also refer to the Aerospace and Defense Security Monitor IP Core Product Marketing Brief. Contact your local Xilinx® representative for more information and access to these documents. If the target application requires mask control, a defense-grade (XQ) device might be needed.

This application note specifically covers UltraScale+ devices and Zynq UltraScale+ MPSoC devices using Vivado Design Suite 2018.3, and builds on earlier IDF concepts.

An example design is provided in the Isolation Design Example for Zynq UltraScale+ MPSoC Application Note (XAPP1336). For more information see Isolation Design Example.

Introduction

The flexibility of programmable logic affords security-critical and safety-critical industries many advantages. However, before Isolation Design Flow (IDF) was developed, in applications, such as information assurance, government contractors and agencies could not realize the full capability of programmable logic due to isolation, reliability, and security concerns, and were therefore forced to use multichip solutions.

To address these concerns, the IDF was developed to allow independent functions to operate on a single chip. Examples of single chip applications include, but are not limited to, redundant Type-I cryptographic modules, or resident safety- and non safety-critical functions.

The successful completion of the Xilinx® Isolation Design Flow allows Xilinx to provide new technology for the information assurance (IA) industry, as well as, provide safety-critical functions in avionics, automotive, and industrial applications.
Isolation Design Flow Overview

Developing a safe and secure single chip solution containing multiple isolated functions in a single FPGA is made possible through Xilinx® isolation technology. Special attributes, such as HD.ISOLATED and the features it enables, are necessary to provide controls to achieve the isolation needed to meet certifying agency requirements.

To better understand the details of the Isolation Design Flow (IDF), the designer should have a solid understanding of the hierarchical design flow (see Hierarchical Design User Guide (UG905)). Many of the terms and processes in the partition flow are used in the IDF. Areas that are different supersede the partition design flow and are identified in this application note.

Common Terminology

Throughout this document, the terms ownership, function, logic, region, and fence are used extensively. These terms are defined as follows:

- **Ownership (physical/logical):** The concept of physical versus logical ownership is an important concept to understand when using the IDF. This concept is described below under "Trusted Routing".

- **Function:** A collection of logic that performs a specific operation (for example, an AES encryptor).

- **Logic:** Circuits used to implement a specific function (for example, flip-flop, look up table, and RAM).

- **Isolated Region/Pblock:** A physical area to place logic.

- **Fence:** The concept of physical versus logical ownership is an important concept comprised of a set of unused tiles in which no routine or logic is present.

- **Trusted Routing:** The routes that satisfy the rules outlined in the following figures. Trusted routing is automatically enabled after the HD.ISOLATED attribute is set to TRUE on at least one isolated module. These routes are a subset of existing routing resources that meet the following restrictions:
  - No entry or exit point in the fence between isolated regions
  - One source and one destination region
  - Its entirety stays contained in the source/destination regions
  - It does not come within one fence tile from another unintended isolation region

These rules act as a filter to all available routes in a given design. An example of routes that would be filtered are shown in the following figures. Example routes excluded for programmable interconnect points (PIPs) outside the intended isolation regions or proximity to unintended isolation regions are also shown.
Rules

A secure or safety-critical solution can be achieved while using FPGA design techniques and coding styles with only moderate modifications to the development flow. Xilinx® Isolation Design Flow (IDF) development requires the designer to consider floorplanning much earlier in the design process to ensure that proper isolation is achieved in logic, routing, and I/O buffers (IOBs). In addition to early floorplanning, the development flow is based on hierarchy. That is, each function you wish to isolate must be at its own level of hierarchy. Although this flow requires additional steps, the hierarchical approach has certain advantages.

There are a few unique design details that must be adhered to achieve an FPGA-based IDF solution. Carefully consider all aspects of the design details explained in subsequent sections of this application note. These considerations include:

- Each function to be isolated must be in its own level of hierarchy.
- A fence must be used to separate isolated functions.
• IOBs must be instantiated inside isolated modules for proper isolation of the IOB. This can be achieved by manual user instantiation or automatically by the tools.

**Note:** Automatic logical inferencing by the tools is unique to the Vivado® Design Suite.

• On-chip communication between isolated functions is achieved through the use of trusted routing (Tools automatically choose trusted routes along coincident physical borders).

### Top Level Logic

Isolated designs must keep the amount of top level logic to a minimum. In a typical Isolation Design Flow (IDF) design, the only logic at the top level should be clock logic. Any component that is not part of an isolated module in the design hierarchy is optimized to the top level. Because isolation is defined by the HD.ISOLATED attribute being set on a hierarchical module, all top logic is, by default, not isolated. This has the following implications:

• There are no placement constraints on top level logic other than it will not be placed in the fence.

• Top level logic can be placed in any isolated region.

• There are no routing restrictions on top level logic other than it will not violate the fence with used programmable interconnect points (PIPs).

• Top level routes can route to, from, and through any isolated region.

### Reference Design

For clarity, an example single-chip 2 channel functional safety design is used throughout this application note to describe the design details and tool flow. The following figure shows the floorplan for the lab design as implemented in an XCZU5CG-SFVC784-1-e device. It consists of five isolated regions. In addition, this design has been implemented with Vivado® Design Suite 2018.3, verified by the Vivado Isolation Verifier (VIV) 2.0, and provided to the designer as a reference.

**Figure 3:** IDF Design Floorplan
Access the isolation design for this application note and the *Isolation Design Example for Zynq Ultrascale+ MPSoC Application Note* (XAPP1336) from the Functional Safety lounge.

**Architecture Overview**

**MPSoc Chip Layout**

Xilinx® MPSoC devices are made up of several sections connected to each other. The processor system (PS) is a monolithic block which is connected to the programmable logic (PL) through a set of interconnect tiles. The PS is made up of columns of tiles organized into clock regions as shown in the following figure for the ZU5 architecture.

*Figure 4: ZU5 Layout*

![ZU5 Layout Diagram]

**Clock Regions and Columns**

Each MPSoC is built using similar architecture. The ZU5 programmable logic (PL) uses four rows of three clock regions. Each clock region contains multiple columns of PL and dedicated block. Each column is composed of a single type of tiles. The number of tiles per column depends on the height of the tile type. A column of configurable logic block (CLB) tiles has 60 tiles.

The PCIe® tile is an example of the the highest tile used. A column of PCIe has one tile. The column width changes based on the tile type for that tile.

It is important to understand that the column width for clock regions vertically arranged is constant. In other words, a column of CLB tiles in the blue clock region extends to the same column in the red clock region below it all the way to the last clock region going vertically down so the same width of the column is maintained from clock region to clock region.
Clock Spines and Interconnect Columns

The clock spine for each clock region is located in the center of each clock region as shown in the following figure. The clocks are distributed vertically using the columns of interconnect tiles which then drive local logic devices in other tiles.

Figure 5: Clock Spine Location

Interconnect tiles are used to route connections between logic tiles. These interconnect tiles are the equivalent to route channels in ASIC designs, and also provide a convenient method to isolate logic regions.

Pblocks and Programmable Units

A Pblock is a region made up of logic blocks called programmable units (PUs). Pblock are used to define an isolated logical unit.

A PU is the smallest logic building block that can be assigned to a Pblock. A Pblock is created by selecting multiple PUs, which create large regions of logic.

Programmable Unit Sizes

It is important to include enough required programmable unit (PU) resources in a Pblock to support the assigned hierarchy. This is verified by reviewing the resources table when the Pblock is drawn.

Each PU has a height, defined by how many PUs can fit in a clock region column, and a width defined by how many columns are needed to support the PU function. A Pblock can be any shape. The shape is defined by the combination if PUs are selected.

Table 1: PU Unit Sizes

<table>
<thead>
<tr>
<th>Programmable Unit (PU)</th>
<th>User Tile Description</th>
<th>Fence Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLICEL (CLBL)</td>
<td>Configurable Logic Block. The key logic unit of an FPGA.</td>
<td>Vertical: 1 PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Horizontal: 1 PU</td>
</tr>
<tr>
<td>SLICEM (CLBM)</td>
<td>Configurable Logic Block. The key logic unit of an FPGA.</td>
<td>Vertical: 1 PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Horizontal: 1 PU</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor. A programmable math function (DSP tile is two DSP48E1 slices; analysis was done on DSP48E1 slice; Vivado® tools only allow selection of a DSP tile).</td>
<td>Vertical: 1 PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Horizontal: 1 PU</td>
</tr>
</tbody>
</table>
Table 1: PU Unit Sizes (cont’d)

<table>
<thead>
<tr>
<th>Programmable Unit (PU)</th>
<th>User Tile Description</th>
<th>Fence Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAM</td>
<td>Block RAM. User-accessible high speed RAM (BRAM tile is RAMB36 which is two RAMB18 blocks; Vivado tools only allow selection of a BRAM tile).</td>
<td>Vertical: 1 PU Horizontal: 1 PU</td>
</tr>
<tr>
<td>URAM</td>
<td>High density block RAM. User-accessible high speed RAM</td>
<td>Vertical: 1 PU Horizontal: 1 PU</td>
</tr>
<tr>
<td>HDIO</td>
<td>General purpose I/O block</td>
<td>Vertical: 1 PU Horizontal: 1 PU</td>
</tr>
<tr>
<td>52 HPIO</td>
<td>High Performance I/O block</td>
<td>Vertical: 1 PU Horizontal: 1 PU</td>
</tr>
<tr>
<td>4 Channel GTX/GTY</td>
<td>High speed transceiver. The GTX or GTY Quad tile is made up of four channels. PR Units in different Pblocks are allowed to abut as long as one channel along the abutment is not used.</td>
<td>Vertical: 1 PU Horizontal: 1 PU PU GTX/Y Channel</td>
</tr>
<tr>
<td>PCI-E</td>
<td>PCI Express® Endpoint Block</td>
<td>Vertical: 1 PU Horizontal: 1 PU</td>
</tr>
<tr>
<td>SYSMON</td>
<td>System Monitor. Contains analog-to-digital converters (ACDs).</td>
<td>Vertical: 1 PU Horizontal: 1 PU</td>
</tr>
<tr>
<td>CMAC</td>
<td>Centralized Media Access Control Block</td>
<td>Vertical: 1 PU Horizontal: 1 PU</td>
</tr>
<tr>
<td>INTERLAKEN</td>
<td>High Speed Chip to Chip pack transfer port</td>
<td>Vertical: 1 PU Horizontal: 1 PU</td>
</tr>
</tbody>
</table>

Programmable Unit Size Examples

The smallest programmable unit (PU) is the CLB PU unit shown in the following figure. The CLB PU unit is 1/60th of a clock region high and two columns wide.

Figure 6: CLB PU
The next smallest programmable units are the DSP PU UNIT and the BRAM PU UNIT shown in the following figures.

**Figure 7: DSP PU**

**Figure 8: BRAM PU**
The URAM PU shown in the following figure.

The highest PUs are:

- **52 HPIO PU**: One clock region high and two columns wide
- **4 Channel GTH PU**: One clock region high and three columns wide
- **SYSMON PU**: One clock region high and also three columns wide

Other one clock high PU units include PCIe® and CMAC.
Figure 10: 52 HPIO PU
Figure 11: 4 Channel GTH PU
Figure 12: **SYSMON PU**
Figure 13: HDIO PU
Fences

A fence is defined as a continuous line of unprogrammed logical resources (unprogrammed user tiles) between a Pblock. These resources also use programmable units (PUs). The fence, like a Pblock, can be any shape shown as the translucent area as shown in the following figure. The fence is formed by drawing Pblocks that are separated by at least one PU. Except for GT programmable units, any Pblock that abut to another Pblock will create a fence violation. When GT PUs abut, one I/O channel from either GT PU cannot be used at the abutment edge. This unused I/O channel becomes part of the fence.

A fence width or height needs to be only one PU high or wide. Routing resources across the fence become more limited if the fence is wider than one PU.

Referring to the following figure, two Pblocks are defined (each outlined in purple and red). The translucent tiles are fence.

*Figure 14: Pblocks and Fences*
Design Process

The following figure shows the recommended process for a design using the Isolation Design Flow (IDF). These steps are key to achieve single-fault hardware isolation required in Information Assurance and Functional Safety designs.

**Figure 15: Design Flow Chart for IDF**

**IP Integrator**

The IP integrator is where the design is captured in the Vivado® Design Suite. After the design is captured, a functional hierarchy is established based on your isolation strategy with respect to how data is flowing in your design. These create the logical boundaries which will define the physical isolation as the design flow is implemented. This step is critical as all other activities are based on this hierarchy.

After the hierarchy is established, an additional wrapper is required. This additional wrapper is required to enable port splitting that might not be allowed by tool generated items, such as the hierarchy created by the triple modular redundancy (TMR) management tool (see MicroBlaze Triple Modular Redundancy (TMR) Subsystem Product Guide (PG268)).
Isolation Design Flow for UltraScale+ Devices and the Zynq UltraScale+
MPSoC

Figure 16: **IP Integrator Tool Generated Design Example**

![IP Integrator Tool Generated Design Example](image)

Figure 17: **MicroBlaze™ Hierarchy for TMR Manager**

![MicroBlaze™ Hierarchy for TMR Manager](image)

Figure 18: **MB_SYSTEM Hierarchy**

![MB_SYSTEM Hierarchy](image)
Figure 19: MicroBlaze TMR Completed Automation

Figure 20: Updated MB_SYSTEM Hierarchy with TMR Implemented

Figure 21: MB_SYSTEM Hierarchy Removed to Flatten Design for Wrappers
Simulation
This step is used to verify your functional expectation of your design. No additional actions are required to comply with the isolation design flow (IDF).

Elaboration
In Isolation Design Flow (IDF), the HD.ISOLATED property is added and enabled for each user-created wrappers that need to be isolated from each other. All other hierarchies that do not require isolation do not get this property added. Such hierarchies will get optimized to top level and have no place and route restrictions beyond keeping fence intact. At this point in the design flow, this property is added on each wrapper, so synthesis knows which hierarchy is isolated, preventing optimization across isolation boundaries.
Figure 23: Adding Property HD.ISOLATED
Enabling HD.ISOLATED

This step creates the actual logic that is used to implement your design. Before an isolated design can be implemented, a floorplanning strategy must be manually entered. This placement restriction uses Pblocks, which are drawn on the device in the Device window. For logic that is outside an isolated hierarchy (such as top level logic), this logic can be in any Pblock.

Synthesis
Floorplanning

FPGA arrays are made up of multiple clock regions with columns of logic resources within each clock region. Each logic resource has a fixed height which are stacked one on top of another to make up a column as shown in the following figure. Pblocks span these regions, but if a resource, such as a PCI™ block, which takes an entire clock region, is not enclosed by a single Pblock, it will not be included in the Pblock. Before creating each Pblock, map out where each Pblock will be located.

Figure 25: FPGA Layout
Drawing Pblocks for Floorplanning

Floor planning is accomplished using Pblocks. Drawing Pblocks is a multi-step process which is outlined here. This is the recommended process, but not the only method.

On a synthesized design:

1. In the NETLIST window, right-click a WRAPPER entry. For this example, select a wrapper entry under design_1_i.

2. Select Floorplanning → Draw Pblock as shown in the following figure.

3. In the Device window, draw a rectangle that defines the initial Pblock as shown in the following figure.
Once a Pblock is created, the Statistics Window in Pblock Properties is updated, which outlines the percentage the Pblock resources that satisfies the synthesized requirements of the isolated region hierarchy as shown in the following figure.

4. To add more resources to a Pblock, in the Device window, select the Pblock to highlight the region.

5. Right-click the highlighted region, and select **Add Pblock Rectangle**. As you grow the size of the Pblock, the Statistics window will update showing added resources. Pblocks can be any shape.

6. To delete the Pblock completely, in the Window drop down menu under Physical Constraints, right-click the Pblock entry, and click **Delete** as shown in the following figure.
When drawing Pblocks, you will see shaded resources that show you which resources are included in the Pblock and which are not. All resources not in a Pblock (non-shaded) are fences. In the following figure, the red shaded resources are allocated to one Pblock, the green shaded resources are allocated to a second Pblock, and the blue shaded resources are allocated to a third Pblock. The regions outlines are only guidance artifacts from creating the Pblock and do not define resources. All non-shaded resources are fences, which means those resources are not be used.
Pblocks can be any shape and size as shown in the following figure. This design enables green, red, and blue Pblocks to easily route to each other. The orange Pblock has all of the top level logic, and the yellow Pblock routes the orange Pblock.

7. To complete the design and make sure your Pblock includes the required I/O Bank, assign pins. In this example, the orange Pblock is top level that also contains the design I/Os.

   In the Window drop down menu, select **IO Ports** to display the I/O Ports window.

   **Note:** In the following figure, the target I/O block is part of the orange Pblock, which is assigned to the top level logic. Make sure each Pin Assignment is fixed and has the correct I/O standard; otherwise, the final design rule check (DRCs) ran under Generate Bitstream will fail.
Vivado IDF Verifier (VIV) Checks 1, 2, 3, and 4

In Vivado® Design Suite 2018.3 and later versions, Vivado Isolation Design Verifier (VIV) 2.0 must be enabled using the `<set_param hd.enableIDFDRC true>` command entered in the Tcl Console. This enables the tool. After all of the flooring planning is complete the Vivado Isolation Design Verifier is run to check for Pblock and fencing rules.

- IDF_VIV2-1 checks for the provenance of the design.
- IDF_VIV1-2 checks for I/O bank assignment violations.
- IDF_VIV2-3 checks for package pin assignment violations.
- IDF_VIV2-4 checks for Pblock floorplan violations.
1. To run this tool, open the Report DRC window in the Reports drop-down menu as shown in the following figure.

2. In the Rules window, under Isolation, select **Provenance** and **Constraints**, and click **OK**.
The result displays in the DRC window as shown in the following figure.

![DRC Window](image)

After this report is clean, run Implementation.

**Implementation**

This step performs resource placement, which uses the Pblock isolation rules, followed by routing which uses only trusted routes that avoids using routing resources in the fences.

**Vivado IDF Verifier (VIV) Checks 5 and 6**

After implementation completes clean, run the Vivado® Isolation Design Verifier to check for placement and routing violations. IDF_VIV2-5 checks for placement violations. IDF_VIV2-6 checks for routing violations.

To run this tool, open the Report DRC window in the Reports drop down menu, and select Implementation under Isolation as shown in the following figure.
The result is displayed in the DRC window as shown in the following figure.

Isolation Design Example
An isolation design example is provided in the Isolation Design Example for Zynq Ultrascale+ Application Note (XAPP1336) and is used throughout this application note to describe the design details and tool flow. For detailed information on the Isolation Design Flow (IDF), see the IDF website at www.xilinx.com/idf.

Conclusion
The focus of this document is to help design engineering professionals understand the new features and limitations of implementing the Xilinx® Isolation Design Flow (IDF) technology with UltraScale+™ architecture and Zynq® UltraScale+™ MPSoC programmable logic (PL). Implement the IDF technology as a part of your overall layers of protection strategy used in functional safety and security design architectures. Although, the impact of using IDF is not discussed in this document, you are encouraged to discuss such solutions with your Xilinx Field Application Engineer or other Xilinx resources.
References

2. *Vivado Isolation Verifier User Guide (UG1291)*
7. *UltraScale Architecture Memory Resources User Guide (UG573)*
10. *Isolation Design Example for Zynq UltraScale+ MPSoC Application Note (XAPP1336)*
11. *Calculating Zynq UltraScale+ Programmable Logic Failure Rates for Functional Safety Applications (XAPP1334)*
12. *Calculating Zynq-7000 Failure Rates for Functional Safety Applications (XAPP1279)*
13. *Calculating Artix-7 Failure Rates for Functional Safety Applications (XAPP1310)*
14. *Calculating Spartan-7 Failure Rates for Functional Safety Applications (XAPP1325)*

*Note:* References 11 through 14 are available from the *Functional Safety lounge.*

Revision History

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Section</th>
<th>Revision Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary and Conclusion</td>
<td>Added coverage of UltraScale+ devices.</td>
</tr>
<tr>
<td>04/15/2019 Version 1.1</td>
<td></td>
</tr>
<tr>
<td>Initial Xilinx release.</td>
<td>N/A</td>
</tr>
<tr>
<td>02/15/2019 Version 1.0</td>
<td></td>
</tr>
</tbody>
</table>

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 https://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 https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS “XA” IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE (“SAFETY APPLICATION”) UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD (“SAFETY DESIGN”). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright

© Copyright 2019 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, ISE, Kintex, Spartan, Versal, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. PCI, PCIe, and PCI Express are trademarks of PCI-SIG and used under license. All other trademarks are the property of their respective owners.