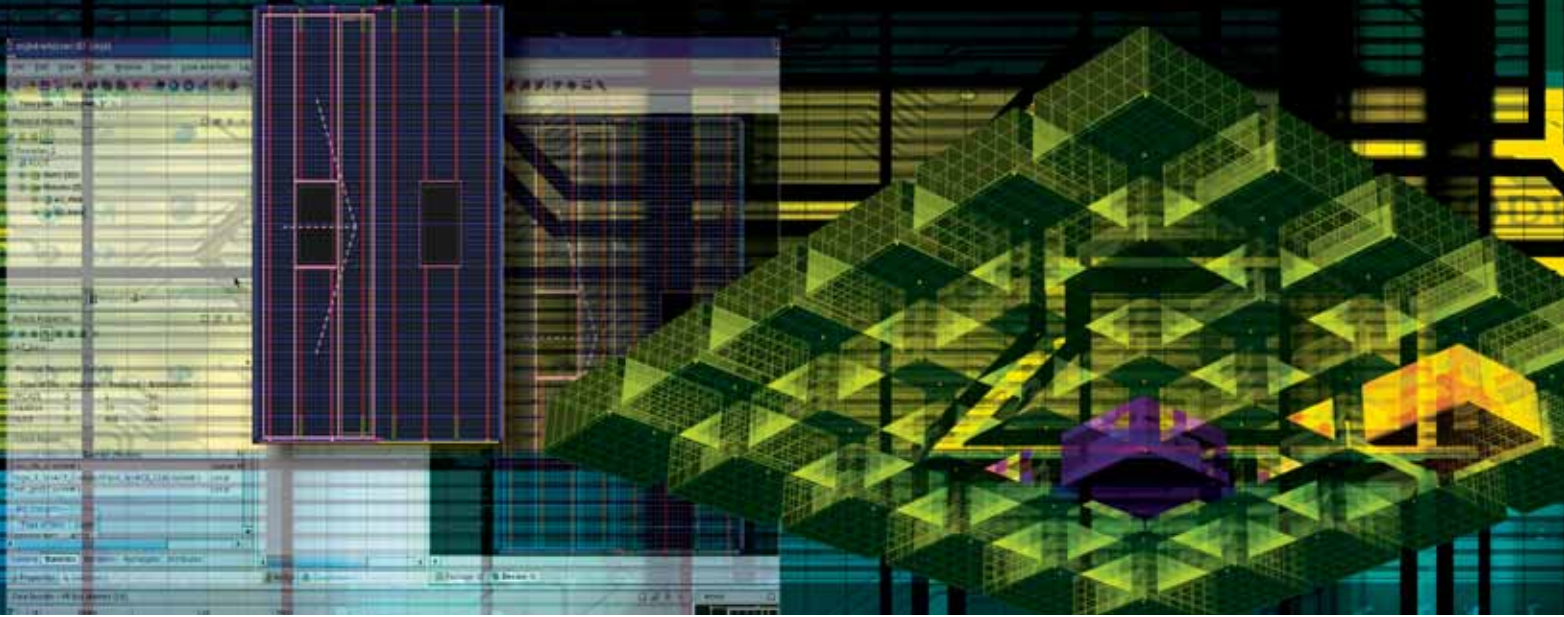


PlanAhead Software as a Platform for Partial Reconfiguration

PlanAhead software delivers a streamlined environment to reduce space, weight, power, and cost.



by Nij Dorairaj
Senior Staff Engineer
Xilinx, Inc.
nij.dorairaj@xilinx.com

Eric Shiflet
Technical Marketing Engineer
Xilinx, Inc.
eric.shiflet@xilinx.com

Mark Goosman
Product Marketing Manager
Xilinx, Inc.
mark.goosman@xilinx.com

The architecture of the Xilinx® Virtex™ platform family of FPGAs allows design modules to be swapped on-the-fly using a partial reconfiguration (PR) methodology. This powerful capability allows multiple design modules to time-share resources on a single device, even while the base design operates uninterrupted.

Partial reconfiguration is a process of device configuration that allows a limited, predefined portion of an FPGA to be reconfigured while the remainder of the device continues to operate. This is especially valuable where devices operate in a mission-critical environment that can not be disrupted while some subsystems are being redefined.

Using partial reconfiguration, you can dramatically increase the functionality of a single FPGA, allowing for fewer, smaller devices than would otherwise be needed. Important applications for this technology include reconfigurable communication and cryptographic systems.

Virtex Configuration Background

Partial reconfiguration is supported in both the Virtex and Spartan™ families. In this article, we will focus on implementing the partial reconfiguration

methodology as it applies to Virtex-II and Virtex-II Pro FPGAs.

All user-programmable features inside a Virtex FPGA are controlled by memory cells that are volatile and must be configured on power-up. These memory cells are known as the configuration memory, and define the look-up table (LUT) equations, signal routing, input/output block (IOB) voltage standards, and all other aspects of the design.

To program configuration memory, instructions for the configuration control logic and data for the configuration memory are provided in the form of a bitstream, which is delivered to the device through the JTAG, SelectMAP, serial, or ICAP configuration interface.

A programmed Virtex FPGA can be partially reconfigured using a partial bitstream. You can use partial reconfiguration to change the structure of one part of an FPGA design as the rest of the device continues to operate.

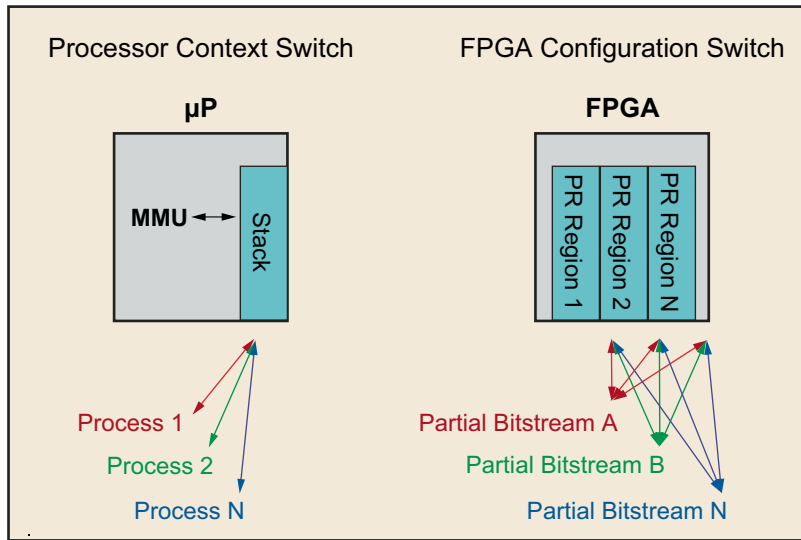


Figure 1 – The analogy between microprocessor context switching and FPGA partial reconfiguration regions

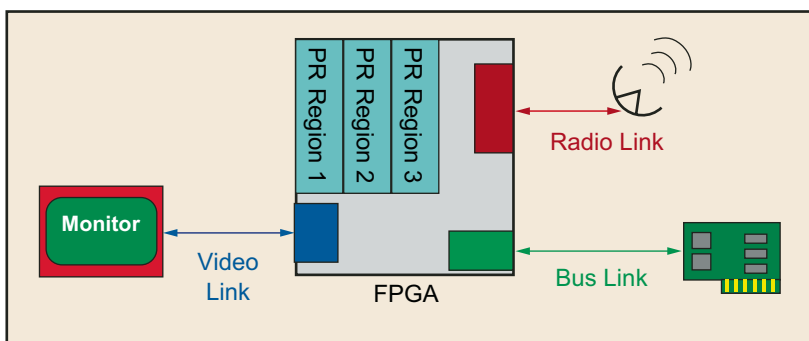


Figure 2 – Maintaining real-time links during partial reconfiguration

Use Cases

Partial reconfiguration is useful for systems with multiple functions that can time-share the same FPGA device resources. In such systems, one section of the FPGA continues to operate, while other sections of the FPGA are disabled and partially reconfigured to provide new functionality. This is analogous to the situation shown in Figure 1, where a microprocessor manages context switching between software processes. Except in the case of partial reconfiguration of an FPGA, it is the hardware – not the software – that is being switched.

Partial reconfiguration provides an advantage over multiple full bitstreams in applications that require continuous operation not otherwise accessible during full reconfiguration. One example, illustrated in Figure 2, is a graphics display that utilizes horizontal and vertical synchronization. Because of the environment in which this application operates,

signals from radio and video links need to be preserved – but the format and data processing format require updates and changes during operation. With partial reconfiguration, the system can maintain these real-time links while other modules within the FPGA are changed on the fly.

Methodology

To implement a partial reconfiguration design successfully, you have to follow a strict design methodology. Here are some of the guidelines to follow:

- Insert bus macros between modules that need to be swapped out (called partial reconfiguration modules, or PRMs) and the rest of the design (static logic)
- Follow synthesis guidelines to generate a partially reconfigurable netlist
- Floorplan the PRMs and cluster all static modules

- Place bus macros
 - Follow PR-specific design rules
 - Run the partial reconfiguration implementation flow
- PlanAhead™ software provides a single environment (or platform) to manage the preceding guidelines.
- Here is a list of the steps in which you can use PlanAhead design tools to implement a partial reconfiguration design:
- Netlist import
 - Floorplanning for partial reconfiguration
 - Design rule checks
 - Netlist export
 - Implementation flow management
 - Bitstream size estimation

Netlist Import

PlanAhead software works with any synthesized netlist, such as XST or Synplify. You can import any hierarchical netlist (single edf/ngc or multiple edf/ngc files). Follow the regular guidelines to import the design into PlanAhead design tools and create a floorplan for the design.

Floorplanning for Partial Reconfiguration

This is an important step in the partial reconfiguration flow. Floorplanning within PlanAhead software is based on design partitions referred to as physical blocks, or Pblocks. A Pblock can have an area (such as a rectangle) defined on the FPGA device to constrain the logic. You can define Pblocks without rectangles and ISE™ software will attempt to group the logic during placement. Netlist logic placed inside of Pblocks will receive AREA_GROUP constraints.

The key floorplanning tasks for partial reconfiguration are:

1. Setting up a PRM:
 - Assign an area for the PRM by creating a Pblock with an area defined within the fabric
 - Assign RANGES for the Pblock

- Define the MODE=RECONFIG attribute on the PRM (Pblock property->attribute section)

2. Setting up static logic:

- Every top-level module, other than PRMs, should be grouped together in a single Pblock. This is called a static logic block. This block should not have

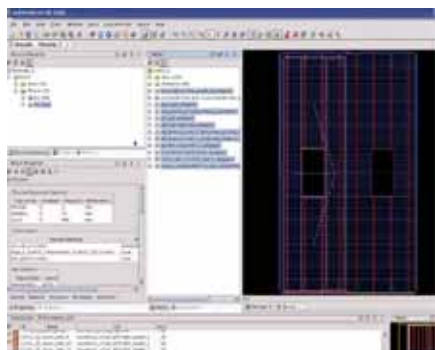


Figure 3 – All static logic is grouped in a single Pblock AG_base.

a RANGE defined; this will cluster the static logic together in a single Pblock. Select all top-level modules (except the PRM) and assign it to a Pblock. Figure 3 shows the static logic grouped in a Pblock named AG_base.

- When you have completed the floor-planning in PlanAhead design tools, the resulting physical hierarchy will be organized as shown in Figure 4.

3. Bus macro placement:

- Bus macros are physical ports that connect a PRM to static logic. Any connection from a PRM to static logic should always go through a bus macro. Bus macros are instantiated as black boxes in RTL and are filled with a pre-defined routing macro in the form of an .nmc file. Bus macros are placed on the PRM boundary. Static logic connected to PRMs will migrate towards the bus macro during placement.

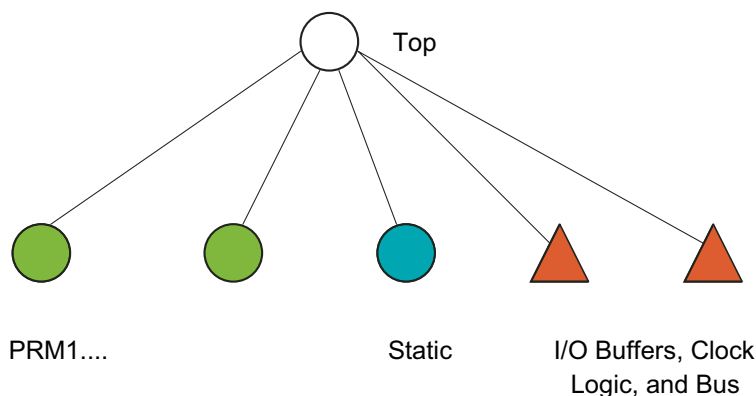


Figure 4 – The hierarchy for a partial reconfiguration design

Design Rule Checks

Given the complexity of the flow, it is common for mistakes to be introduced in the original RTL and during the floor-planning process. PlanAhead-PR design tools check for design violations. Also integrated into this feature is the PR-Advisor, which provides feedback on how to improve your design.

The PR design rule checks are:

1. Bus macro DRC. This provides verification for all design rules related to bus macro connectivity and placement. One example of a bus macro DRC is the PRBP check. This DRC checks for all rules that should be followed for bus macro placement. Figure 5 shows an example of a design that failed the PRBP DRC. In this case, the interleaved/nested macro should be placed at SLICE_X41Y*.
2. Floorplanning DRC. This covers floor-planning rules. Clock objects (global clock buffers, DCM) and I/Os should be placed and static logic clustered.
3. Glitching logic DRC. This verifies glitching logic elements (SRL and distributed RAM) above and below PRM regions.
4. Timing Advisor/DRC. This provides a check for timing-related issues. One example of timing DRC is the PRTP check. The static module is implemented before the PRM during the implementation phase. Regular timing constraints do not cover the paths that cross between the static and a PRM. This does not present a problem, provided that the bus macro is synchronous. However, if it is an asynchronous bus macro, the static module does not know about the propagating of asynchronous paths, as shown in Figure 6. This could be important if these paths are timing-critical. One way to pass this information to the static module is to specify a TPSYNC constraint on the bus macro output net. PlanAhead software will recommend a TPSYNC constraint that can be added to the .UCF file.

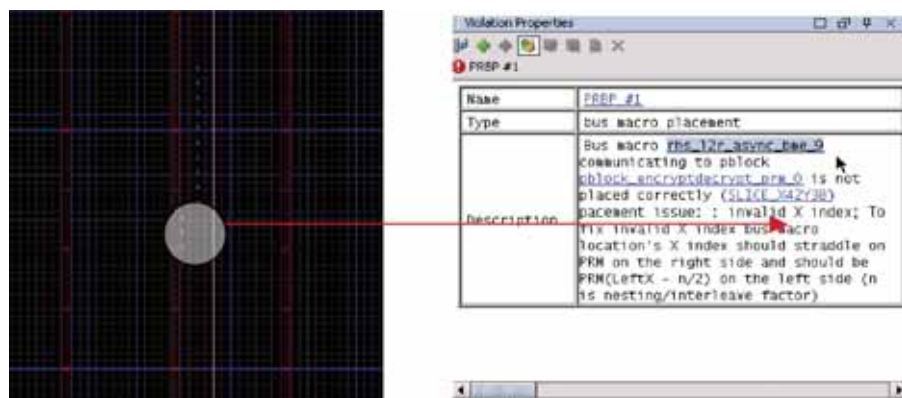


Figure 5 – The PRBP DRC verifies all rules that should be followed for bus macro placement.

Netlist Export

Once the design is floorplanned and passes the DRC checker, it is ready to be exported. PlanAhead design tools take care of exporting the original hierarchical netlist

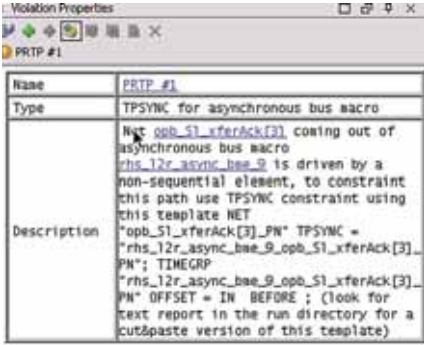


Figure 6 – Special consideration should be given to timing-critical paths, which include an asynchronous path.

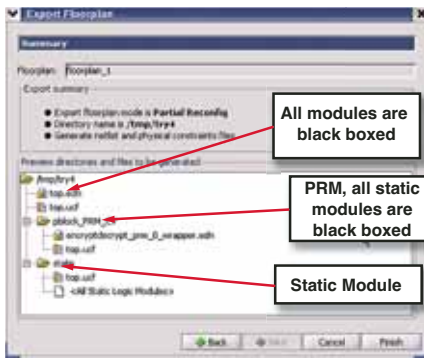


Figure 7 – Upon successful completion of the floorplanning DRC processes, all modules will be displayed in black text within the Export Floorplan dialog box.



Figure 8 – The PlanAhead partial reconfiguration flow wizard provides an easy-to-follow flow.

into a PR-style netlist that has a specific format (static and PRM in separate directories). The export directory will appear as shown in Figure 7.

Implementation Flow Management

The partial reconfiguration flow wizard, shown in Figure 8, runs the partial reconfiguration implementation on the exported design. It will produce a full bitstream for the complete design and a partial bitstream for each of the PRMs. The implementation steps are:

- Initial budgeting
- Static module implementation
- PRM implementation (one implementation for each version of every PRM)
- Assembly and bitstream generation (results are stored in the merge directory)

To run the tools, start the flow wizard by selecting “Tools > Run Partial Reconfig.” This wizard will guide you through each of the implementation steps. You can either run the implementation or just generate the scripts, which can be used from a command line outside of the PlanAhead user interface.

Bitstream Size Estimation

The Pblock statistics report includes a section that reports PRM bitstream size (Figure

9). This information can be used for estimating the size of configuration memory storage such as external flash and DDR. This information can also be used to calculate how long it will take to swap the module based on your bitstream memory interface.

Conclusion

PlanAhead software is the first graphical environment for partial reconfiguration. Using PlanAhead design tools as a platform for partial reconfiguration applications can greatly simplify the complexities of juggling the dynamic operating environment of these cutting-edge applications, allowing a single device to operate in applications that previously required multiple FPGAs.

The methodology offered by PlanAhead software can dramatically increase productivity and decrease time-to-solution for designers using partial reconfiguration.

The capability of designs to leverage partial reconfiguration opens doors to a whole host of applications. By providing a platform that leverages the advantages of partial reconfiguration for designs targeting Virtex-II and Virtex-II Pro FPGAs, PlanAhead design tools allow you to dramatically extend the functionality of your design. With similar support for designs targeting the Virtex-4 multi-platform FPGA in the near future, the application space for partial reconfiguration is practically limitless.

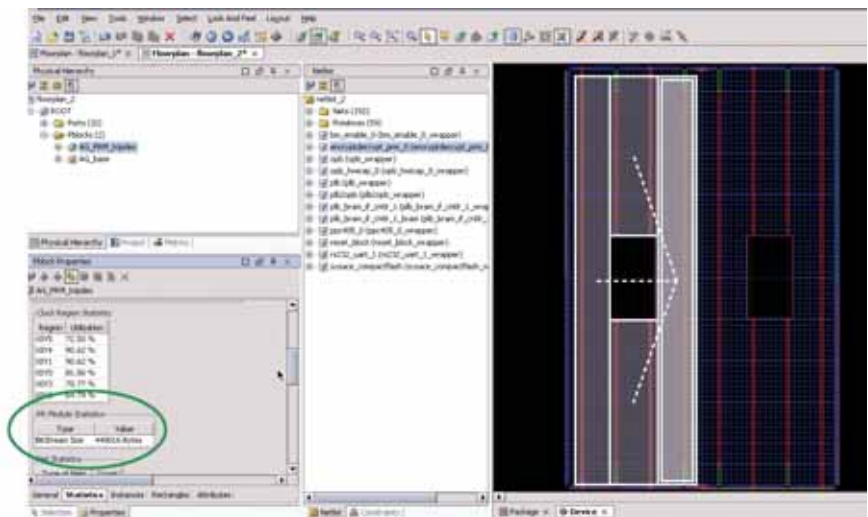


Figure 9 – The display of Pblock properties includes the estimated size of the bitstream.