FPGA Run-Time Reconfiguration: Two Approaches

Introduction

Run-time reconfiguration for FPGA designs is an increasingly important requirement for many user markets, particularly military users who must adapt quickly to different threats and evolving communications waveforms. Run-time reconfiguration is defined as the ability to modify or change the functional configuration of the device during operation, through either hardware or software changes. This is an FPGA feature that is important to communications, military, and consumer applications as an approach to reducing component count and power consumption, by reusing the same FPGA for several functions.

Examples of run-time reconfiguration applications include:

- Software-Defined Radio (SDR): The ability to adapt to varying waveforms is a paramount goal in military SDR.
- Field testing: By designing with reconfigurable components, applications can be reconfigured in the field without needing to go back to the lab for re-engineering.
- Airborne applications: Susceptibility to radiation-induced single event upsets (SEUs) makes it important to monitor and reconfigure devices with bit failures.
- Remote sensors: The ability to reconfigure devices that are difficult to reach physically makes it important to be able to update “over the air.”

This white paper examines two approaches to run-time reconfiguration. FPGA partial reconfiguration (PR) is a Xilinx design flow that attempts to create reconfiguration regions in an FPGA device, so that one region can be reconfigured while the remainder of the FPGA continues to operate in system. Software programmable reconfiguration (SPR) is the designed-in capability to modify digital logic flows through internal or external software commands.

Advantages to Run-Time Reconfiguration

FPGA-based designs in all markets are beginning to incorporate more and more features, missions, and waveforms. This means larger designs, in larger FPGAs. This, in turn, translates into higher power requirements and longer design times. FPGA device sizes can only increase as fast as the silicon process technology will allow. Therefore, in order to make better use of logic resources, expanding engineering designs need other approaches to meet customer requirements and power budgets. One of the primary tools in this “requirements chase” is run-time reconfiguration.

The primary advantages of run-time reconfiguration in devices are reduced power consumption, hardware reuse, obsolescence avoidance, and flexibility. The costs of run-time reconfigurability are in design and implementation complexity-both in architecture definition and in coding and test. To metrically assess the advantages of the run-time reconfiguration approaches, the improvements in power reduction and flexibility should be compared to the corresponding increases in design complexity.

Power Sensitivity

Size, weight, and power (SWaP) have always been limiting factors directly effecting device use in military devices, particularly SDR applications. Recently there has been an increased focus to manage and reduce these limitations across all types of SDRs. Increased operational expectations for longer mission life and reduced cost, along with the need to add functions that increase efficiency, performance, flexibility, and features have driven a demand for SWaP reduction. In addition, the market for smaller, lighter, and less-expensive SDR applications has grown significantly in all sectors.

Due to the duty cycle of radio modes, standby operation typically dominates radio usage, so the high static power consumption of FPGAs has traditionally limited them to a coprocessing role in SDR applications. As a result, developers often select alternative devices, using FPGAs only when necessary. However, with the introduction of next-generation devices like Altera's Cyclone® III FPGAs, new standards have been set. Low static power consumption and the processing power of multiple digital signal processing (DSP) devices in a single FPGA device allow developers to process multiple signal data streams at a lower cost per channel than competing devices. The role of FPGAs in SDR applications has grown to the point where they can now be used as primary processing devices. The addition of run-time reconfiguration-either through device-dynamic reconfiguration or software like run-time methodology-is the next step in achieving low-power solutions.

Hardware Reuse

One of the most significant cost drivers in both military and communications infrastructure is the reuse of hardware, from FPGA intellectual property (IP) to boards and chassis, which enables radios to operate more than one waveform. FPGAs allow not only the reuse of boards from one application to the next, but also the reuse of FPGAs on any given board for several functions within the same application. This makes run-time reconfiguration an essential tool in streamlining product designs and achieving time-to-market milestones.

Obsolescence Avoidance

One of the most touted advantages of FPGAs in military designs is the ability to "future proof" applications through the careful application of hardware and software design, and the careful use of third-party application program interfaces (APIs) and design applications. Appropriate up-front software and hardware architecture design for run-time reconfiguration can also lay the groundwork for technology insertion into a design, forestalling obsolescence, increasing portability, and extending useful military design life.

Application Portability

One of the goals of a run-time configurable system is to encapsulate the reconfigurable system into a portable application or avionics pod. Complicated chip load and reload schemes both negate portability and lower the reliability of a system where chip reconfiguration relies on time-critical interaction with external processors.

Components of Run-Time Reconfiguration

Several portions of Altera® FPGA designs are already available in run-time reconfiguration modes. In Cyclone III devices, device phase-locked loops (PLLs) such as clock speed settings, phase information, and PLL bandwidth can be reconfigured dynamically without reconfiguring the entire FPGA. The capability to reconfigure user I/Os dynamically is well documented for Altera devices. This is accomplished through the JTAG boundary-scan test chain. Altera's MorphIO application enables I/O settings to be run-time configurable without reconfiguring the FPGA. Most importantly, device run-time reconfiguration schemes have the ability to reconfigure logic blocks and data paths in the user design.

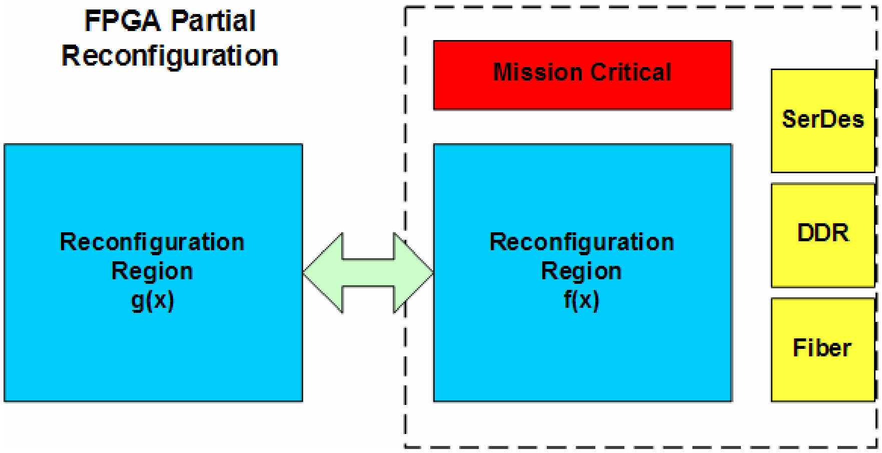
Two Approaches to Run-Time Reconfiguration

Currently, there are two FPGA solutions to run-time reconfiguration. The more ambitious PR approach necessitates FPGA architecture designed to support reconfiguration zones. In SPR, FPGA components are created as highly flexible building blocks controlled and manipulated through embedded software such as the Nios® II embedded processor, code running on a digital signal processor, or even through host software running on a general-purpose processor (GPP).

Partial Reconfiguration

FPGA reconfiguration can increase the functionality of a single FPGA device dramatically, allowing fewer and smaller devices, thus requiring less power than would otherwise be necessary. As shown in Figure 1, the architecture of the Xilinx Virtex family of FPGAs allows design modules to be swapped on the fly using the PR methodology. This capability allows limited resources within the Virtex device to be timeshared (reconfigured) while mission-critical or other base design requirements continue to operate.

Figure 1. FPGA PR as Implemented on Xilinx Devices


FPGA Partial Reconfiguration
Mission Critical
Reconfiguration Region g(x)
Reconfiguration Region f(x)
SerDes
DDR
Fiber

The FPGA device is divided into reconfiguration regions, and a partial bitstream must be created for programming each. The FPGA continues to operate mission-critical functions and meet external interface requirements while reconfiguration regions are reprogrammed to provide different functionality. An analogy can be drawn with a microprocessor that is context switching, where the current state is preserved in order to switch to a different process. FPGA reconfiguration, however, switches the functional hardware, not a software state. The main advantage of this methodology is that mission-critical operations can be preserved while only part of the FPGA device is reconfigured, as opposed to full reconfiguration of the FPGA device, which does not allow for uninterrupted operation.

Xilinx FPGAs using PR are typically limited to a single reconfiguration region that falls within logic column boundaries, adding significant additional timing constraints. Although Xilinx currently offers a design flow offering partial reconfigurability (available through FAEs only), there is no robust documented design methodology to guide a user through the implementation of the reconfiguration region.

In order to implement PR, a strict design methodology must be followed:

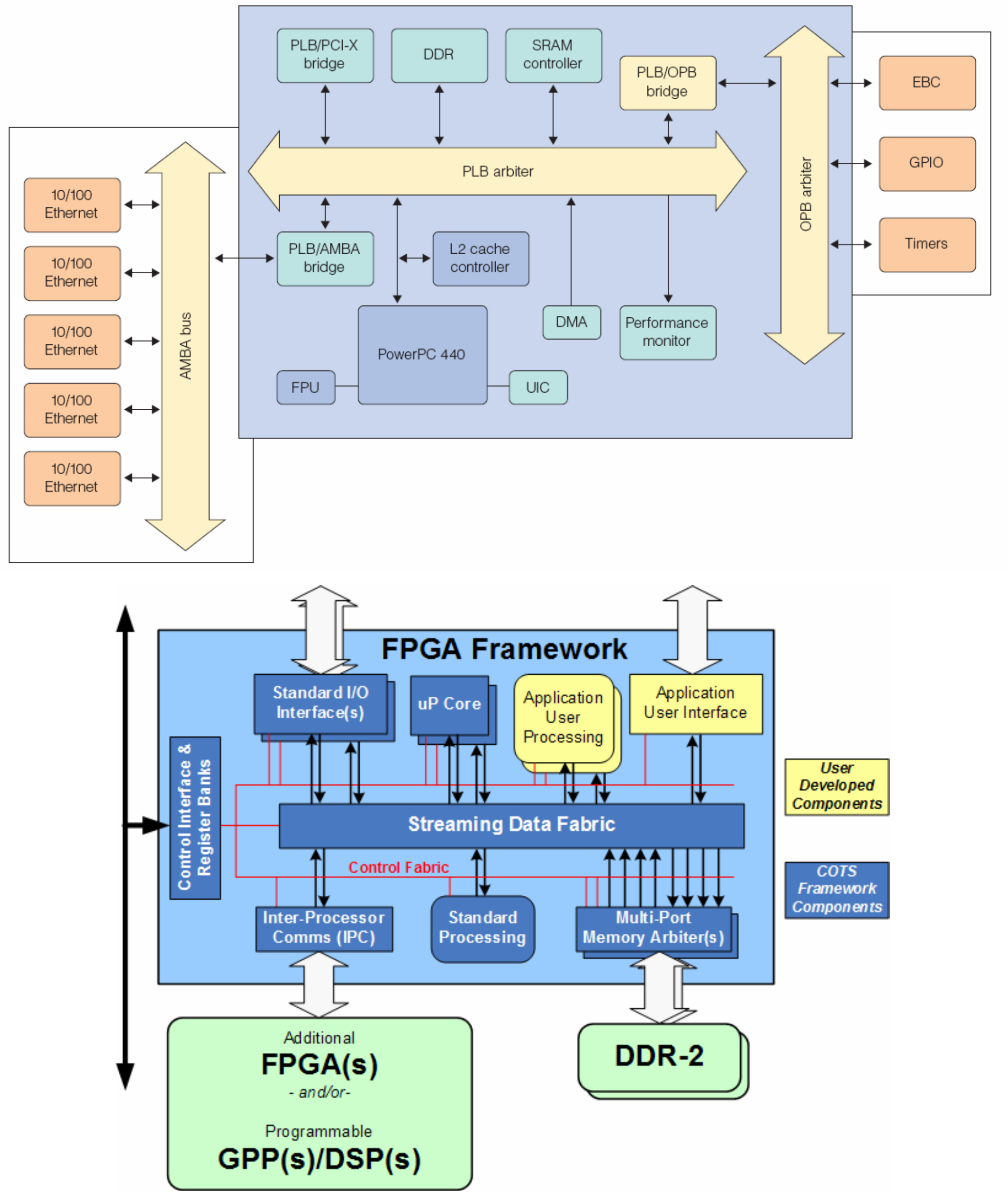
1. Insert bus macros between the PR modules and the static portion of the design.
2. Follow the PR synthesis guidelines to generate a partially reconfigurable netlist.
3. Create floor plans for all PR and cluster static modules.
4. Place bus macros.
5. Follow PR-specific design rules.
6. Run the PR implementation flow.

Xilinx offers a tool called PlanAhead to aid in the implementation of PR. However, due to the strict guidelines, requirements, and steep learning curve, significant effort is required to implement PR. The methodology required significantly complicates FPGA development efforts, lowering the abstraction level and forcing developers to focus their time and efforts on device-dependent designs close to the gates. In an effort to achieve more design flexibility and reduce risk, PR adds significant additional design risk to military programs. In addition, the PR approach is very dependent on tool version and device, limiting the designer’s selection of FPGA for size, I\O capability, and power. For example, there is no PR support for Spartan-class FPGA devices.

Software Programmable Reconfiguration

SPR is a methodology that leverages existing IP and design software to provide an FPGA reconfiguration solution superior to PR. By taking a more software-based approach to FGPA development, and looking at the FPGA as a system on a chip (SOC) with the peripheral infrastructure in place, the goals of FPGA reconfiguration can be achieved with all of the advantages stated earlier. Figure 2 shows the similarities between a SPR implementation and a standard microprocessor with peripheral support.

Figure 2. FPGA SPR Architecture

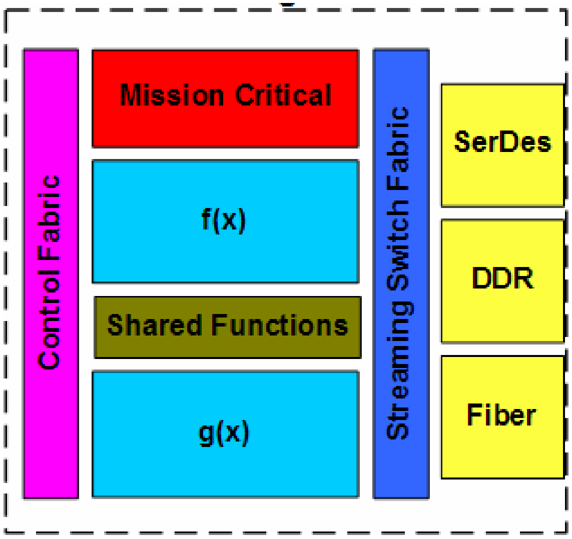


The functionality of an FPGA device can be increased significantly by using SPR, as the developer is able to raise the abstraction level to one similar to software. In addition, design and hardware reuse is promoted, allowing multiple waveforms to be implemented on one device or across multiple devices. The integration of a soft-core microprocessor into the design improves design exploration and test, providing a method of control, status, and flexible, real-time adaptive reconfiguration. As with microprocessors, this type of high-level design is scalable, allowing for an increase in application complexity, mapping directly to an ASIC flow, and resulting in a significant increase in overall design portability.

Instead of reconfiguring the device with a new bitstream, the whole application is implemented, integrating all components in a single device. A control plane provides a path for control, configuration, and status of the implemented components, allowing dynamic reconfiguration and feedback of each function. It also allows commands to be sent to the streaming data switch fabric, reconfiguring the date path and allowing other functions to operate.

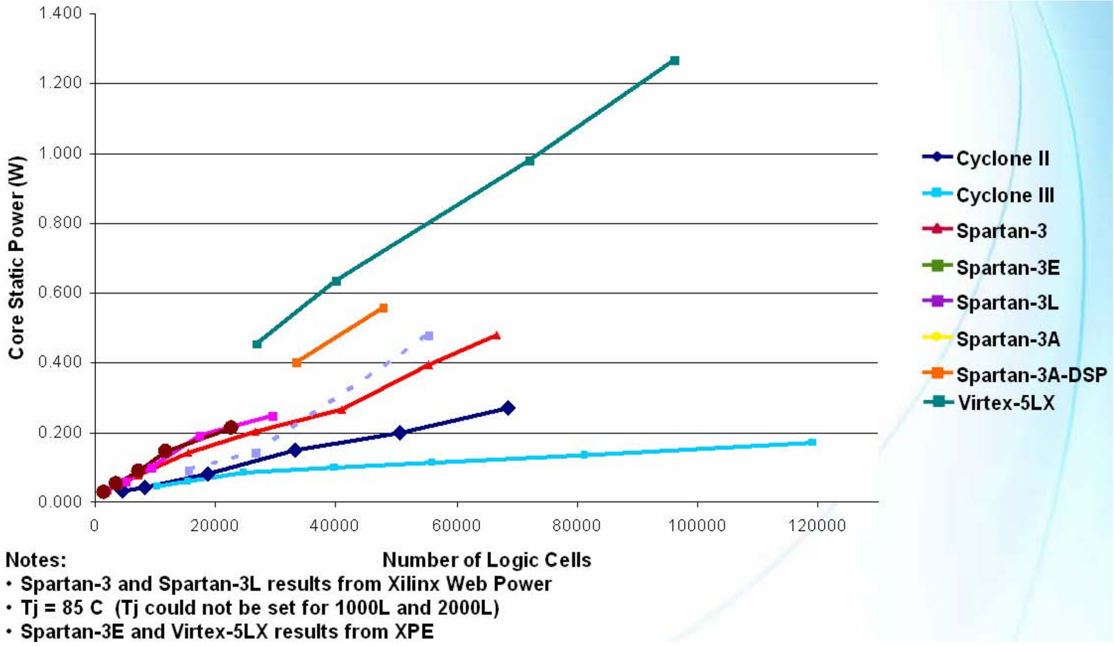
Similar components within each function can be shared by multiple functions to save resources. In addition, a mixed time-division multiplexed (TDM) mode can be applied that is not supported by PR. PR is function A or function B. The SPR-based methodology is not a true TDM function, in that f(x) can be operating while g(x) also operates, sharing only the resources necessary, as shown in Figure 3.

Figure 3. FPGA SPR Configurable Regions



However, in order for SPR to meet SWaP requirements using this methodology, FPGA devices must themselves be low power, providing a significant amount of computational resources. By leveraging the low-power, high-density FPGAs provided by Altera, the goals of FPGA reconfiguration can be achieved. The core static power curve shown in Figure 4 illustrates the significant power savings per logic block available in Altera devices. The Cyclone III FPGA provides more logic cells than any low-cost Xilinx device, even more than the Virtex-5LX at one-sixth the static core power consumption.

Figure 4. Static Power Curves for Several FPGAs on the Market



Notes:
- Spartan-3 and Spartan-3L results from Xilinx Web Power
- Tj = 85 C (Tj could not be set for 1000L and 2000L)
- Spartan-3E and Virtex-5LX results from XPE

One drawback is that SPR has no significant capability to detect and recover from SEUs as a properly designed PR scheme could theoretically achieve. In addition, Altera FPGAs have memory-scanning capabilities, so SEUs propagated through such a system may require the device to be reconfigured, interrupting operations. However, it should be emphasized again that for systems using a handful of different waveforms, SPR holds significant promise in single-FPGA implementation, with a path to large-scale implementation on HardCopy® ASIC devices. As Moore's law continues to hold true, SPR implementations are likely to become more popular.

Conclusion

Current FPGA capabilities in the area of run-time reconfiguration are maturing to meet the needs of military and wireless communications customers. As the capabilities of programmable logic devices grow, there will be more demand for flexible reuse of FPGA resources. Advances in this area will continue to be made in device configuration and reconfiguration speed, built-in error detection and recovery, and ease of design of the reconfiguration modes.

Given the implementations of run-time reconfigurable technology for FPGAs today, the lowest risk and most power-efficient implementation is a SPR as implemented by BittWare Inc. It offers the most flexible and feature-rich ability to operate in a handful of different DSP modes. In addition, this approach holds the promise of higher reusability, portability, and scalability to structured ASIC device for further reductions in power and unit cost.

Designers are likely to continue developing sophisticated multi-faceted designs that require robust implementations of PR. For this reason, work will continue both in developing the capabilities and ease-of-use of PR, as well as SPR. As is the case with many FPGA capabilities, having the technology available today is not sufficient to compel usage; crossing over a "design usability" barrier is required. SPR has both the design usability and reconfigurable features for today's reconfigurability requirements.

Further Information

- *SPR Framework for Software Defined Radio*:
  www.altera.com/literature/cp/cp-01034-spr-framework-for-development-of-sdr.pdf
- *FPGAs Provide Reconfigurable DSP Solutions*:
  www.altera.com/literature/wp/wp_dsp_fpga.pdf
- BittWare Software Programmable Reconfiguration Paper and Development Platform:
  www.bittware.com/products/tools/prod_desc.cfm?ProdShrtName=SPRD
- *Designing with Confidence for Military Software Defined Radio Applications*:
  www.altera.com/literature/wp/wp-01020.pdf
- *MorphIO: An I\O Reconfiguration Solution for Altera FPGA Device*s:
  www.altera.com/literature/wp/wp_morphio_reconfig.pdf
- *Implementing PLL Reconfiguration in Stratix® II Devices*:
  www.altera.com/literature/an/an367.pdf
- *FPGAs: Solving "Future Proofing" in Military Applications via Technology Insertion*:
  www.mil-embedded.com/pdfs/Xilinx.July07.pdf

Acknowledgements

- J. Ryan Kenny, Technical Marketing Manager, Military and Aerospace Business Unit, Altera Corporation
- David Rupe, FPGA Product Manager, BittWare, Inc.

101 Innovation Drive
San Jose, CA 95134
www.altera.com