Bitstream Relocation with Local Clock Domains for Partially Reconfigurable FPGAs

Adam Flynn, Ann Gordon-Ross, Alan D. George
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering, University of Florida
Gainesville, FL
{flynn, ann, george}@chrec.org

Abstract—**Partial Reconfiguration (PR) of FPGAs presents many opportunities for application design flexibility, enabling tasks to dynamically swap in and out of the FPGA without entire system interruption. However, mapping a task to any available PR region (PRR) requires a unique partial bitstream for each PRR. This replication can introduce significant overheads in terms of bitstream storage and communication requirements. Previous research in partial bitstream relocation can alleviate these overheads by transforming a single partial bitstream to map to any available PRR. However, careful steps are necessary to ensure proper functionality of relocated partial bitstreams and may result in clock routing inefficiencies. These routing inefficiencies can be alleviated by using regional clock resources introduced in the Virtex-4 FPGAs to implement local clock domains. PRRs can internally drive local clock domains, enabling each PRR to vary its clock frequency with respect to a single global clock signal, as opposed to sending multiple global clock signals (one for each desired clock frequency) to each PRR. We introduce this novel local clock domain (LCD) concept, which provides enhanced PR design flexibility. However, integration of LCDs and partial bitstream relocation introduces new challenges. In this paper, we identify motivating application domains for this integration, analyze integration benefits, and provide a detailed integration methodology.**

Keywords-partial reconfiguration; relocatable; local clock domains (LCDs).

I. Introduction and Related Work

Module-based partial reconfiguration (PR) [12] enables finer reconfiguration granularity compared to traditional methods of full reconfiguration. In partially reconfigurable FPGAs, the FPGA fabric is partitioned into one static region and one or more partially reconfigurable regions (PRRs). This fabric partitioning enables reconfiguration of a single PRR without system interruption (the static region and other PRRs continue execution while only the reconfigured PRR halts). Signal interfaces between static and reconfigurable regions are maintained through designer-defined signal interfaces referred to as *bus macros*.

To exploit PR, applications must be decomposed into one or more functional modules/tasks (partially reconfigurable modules or PRMs), which collectively encompass the entire application behavior. PRMs can be mapped to a single PRR, and time-independent PRMs (mutually exclusive application tasks) can share PRRs, reducing the FPGA fabric area needed for a given application. Furthermore, PRMs can map to multiple PRRs, providing additional runtime flexibility by allowing a PRM to be mapped to any available PRR through *dynamic runtime PRM placement*.

Since PR restricts reconfiguration to a particular PRR, PR bitstreams (*partial bitstreams*) yield faster reconfiguration times and smaller bitstream sizes when compared to traditional *full bitstreams* (bitstreams to reconfigure the entire FPGA fabric) as partial bitstreams only contain configuration data for the target PRR. This reconfiguration isolation relaxes both bitstream storage and communication resource requirements. However, since a unique partial bitstream is needed for each PRM to PRR mapping and PRMs can map to multiple PRRs, multiple, nearly identical partial bitstreams must be stored and communicated (identical functionality but differing in FPGA fabric location).

Bitstream relocation provides a means to eliminate these redundant partial bitstream overheads. *Bitstream relocation* manipulates a single partial bitstream, enabling the placement of that bitstream into an arbitrary PRR (independent of the initial PRM to PRR mapping), eliminating redundant bitstream storage and communication. Figure 1(a) illustrates a sample PR system with two PRMs and four partial bitstreams per PRM

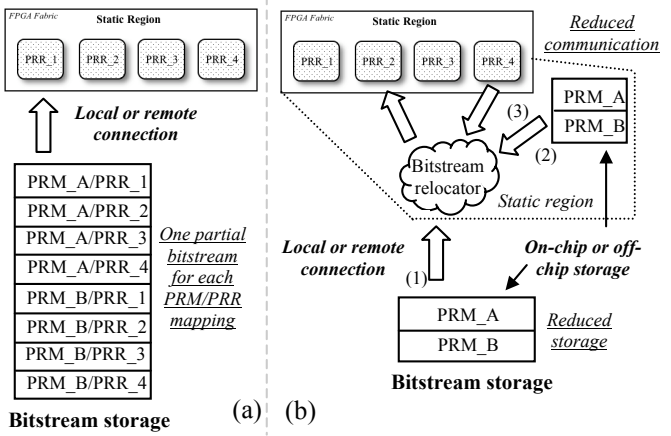

Figure 1. (a) PR system without bitstream relocation capability. (b) A bitstream relocation capable system with the benefits of both reduced storage and communication resources (smaller bitstreams may fit on-chip). The bitstream relocator would reside in the FPGA static region and would obtain PRMs from either (1) local or remote off-chip storage, (2) local on-chip storage, or (3) an existing PRR.

(eight partial bitstreams in total). These redundant partial bitstreams are necessary for dynamic runtime PRM placement, but impose unnecessary increases in storage and communication resource requirements.

Figure 1 (b) illustrates the advantages of bitstream relocation. The bitstream relocator resides in the FPGA's static region along with the partial bitstream storage. We point out that bitstream relocation does not *imply* that the partial bitstreams can be stored on the FPGA. Bitstream relocation simply reduces the storage requirements, making it more *likely* that the partial bitstreams would fit on the FPGA. The bitstream relocator obtains the partial bitstream to relocate from (1) off-chip bitstream storage, (2) on-chip bitstream storage, or (3) directly from a PRR. Previous work shows that both software and hardware bitstream relocation implementations require only milliseconds of processing time, making bitstream relocation amenable to dynamic runtime PRM placement.

The underlying algorithms from previous work [5] in bitstream relocation are similar throughout, though there are several fundamental differences in each work's implementation. Thus, the basis for our relocation algorithm draws upon these implementations, leveraging previous research to demonstrate the compatibility of local clock domains with partial bitstream relocation in implementing relocatable PRMs with locally driven clocks.

Unfortunately, in spite of the benefits and feasibility of partial bitstream relocation, the popular Xilinx Inc.'s Early Access (EA) PR design flow [11] is not compatible with partial bitstream relocation. Though the EA PR design flow produces partial bitstreams operational for a particular PRM to PRR mapping, relocated bitstreams are inoperable. Through further analysis, we concluded that unconstrained clock signal routing performed by the EA PR tools caused this inoperability. The EA PR design flow specifies that clock signals should not be routed through bus macros; the routing of clock signals is automatically handled by the EA PR tools and is transparent to the user. Modifying the EA PR flow to route clock signals through bus macros produced partial bitstreams compatible with bitstream relocation. However, this modification forces the clock signal to be routed through configurable logic blocks (CLBs), which are not intended for clock signals. This inappropriate routing can result in undesirable side effects such as higher clock skew, timing inconsistencies, and lower PRM operating frequency.

The main contributions of this paper are twofold. Our first contribution is a defined methodology for producing relocation-compatible partial bitstreams using dedicated clock resources rather than bus macros to eliminate the aforementioned undesirable side effects. Our second contribution is the enabling of each PRR and associated PRM to operate at its own independent clock frequency, which we refer to as *local clock domains (LCDs)*. LCDs leverage the *regional clock resources* introduced in Virtex-4 FPGAs [4][9] to both set the frequency and drive clock signals from within the PRM itself using a single global clock signal. This paper details standard EA PR design flow deviations required to implement each PRM as its own LCD. Currently, the implementation of *multiple clock domains* without LCD support requires multiple global clock signals, which could potentially span the entire FPGA. Since LCDs allow multiple clock domains to be implemented with a single global signal, LCDs reduce overall power consumption and top-level design complexity [7].

II. MOTIVATION

A. Motivating Application Domains

We have identified two domains that can benefit greatly from dynamic runtime PRM placement, namely reconfigurable fault tolerance (RFT) [2][8] for FPGA-based computing in space, and a novel virtual architecture (VA) for PR in FPGAs [5] (we refer the reader to the references for further details).

These application domains pose many challenges, two of which are addressed by the partial bitstream relocation methodology presented in this paper. The first challenge involves dynamic runtime PRM placement into arbitrary PRRs, providing increased PRM placement flexibility as well as reduced bitstream storage and communication costs. The second challenge involves providing a simple and power-efficient method for supporting multiple clock domains. Multiple clock domains may be needed to fine-tune a PRM's clock frequency to maximize performance, meet specific application timing requirements, reduce power consumption, etc. LCDs allow clock frequency specification at the PRM level, providing finer-grained control than global clock domains. Although our methodology also allows global implementation of multiple clock domains, LCDs provide a reduction in power consumption [7].

These domains' amenability to and potential benefits from relocatable PRMs with LCDs provide the motivating stimulus for pursuing the integration of prior research in partial bitstream relocation and LCDs.

B. Local Clock Domains (LCDs)

The Virtex-4 FPGA family introduced regional clock resources not present in earlier FPGA families [9]. These regional resources are integrated with global resources to form a multi-level clock distribution network and are accessed through instantiation of the *Regional Clock Buffer (BUFR)* primitive [4][9][14]. Advantages of this dedicated infrastructure include high clock frequency and low power consumption [9]. Depending on the FPGA size, 8 to 24 square clock regions are aligned to a rectangular grid within the FPGA. LCDs correspond directly to these clock regions. This infrastructure is intuitive for use with the FPGA fabric partitioning necessary for PR.

Xilinx documentation provides regional clock resource usage guidelines for PR on their EA PR tools' website [4]. The BUFR primitive drives *regional clock nets* within each clock region. Regional clock nets are confined to their respective regional clock region. Furthermore, the designer can specify the exact location of the BUFR within the FPGA, effectively including the BUFR in the bus macro interface necessary for partial bitstream relocation. The advantage of using the BUFR is that the clock signal is not routed through CLBs.

Multiple clock domains can be implemented with multiple global signals or LCDs. The Digital Clock Manager (DCM)

and Phase Matched Clock Divider (PMCD) primitives available in the Xilinx FPGAs support the implementation of multiple clock domains through multiple global signals. However, global signals must be routed through bus macros to be compatible with partial bitstream relocation. Leveraging regional clock resources for LCD implementation provides a convenient mechanism for efficient implementation of multiple clock domains. The BUFR primitive can divide the frequency of a single global signal, allowing a single global clock signal to drive multiple domains/PRRs. Furthermore, as Lamoureux and Wilton demonstrate the power efficiency of localizing clock signal distribution when implementing multiple clock domains, LCDs have the potential to reduce power consumption in applications requiring multiple clock domains [7].

Although [4] documents the guidelines for using regional clock resources within PR, to the best of our knowledge, no documentation of an actual PR application using regional clock resources exists. Furthermore, regional clock resources' structural amenability to PR raises the question of whether regional clock resources are compatible with bitstream relocation. We address this lack of documentation in Section IV.

III. METHODOLOGY

A. Partial Bitstream Relocation

The partial bitstream relocation methodology presented in this paper and the previous methodologies outlined in Section I leverage knowledge of the target FPGA's configuration data structure. The configuration data structure of Xilinx FPGAs evolves as the underlying architecture evolves. While many similarities exist between generations, the following discussion is specific to Virtex-4 FPGAs. Adaptation of our methodology to the Virtex-5 FPGA family requires minor modifications.

The Virtex-4 FPGA configuration data is arranged into tiled *frames* [10]. Each frame has a unique address within the FPGA; detailed documentation of the Virtex-4 addressing scheme can be found in [1][10]. Partial and full bitstreams for Xilinx FPGAs consist of a packet stream containing miscellaneous commands and configuration data. Each packet type is assigned a unique identification header. Two packet types are of particular interest to partial bitstream relocation. The first is the *Frame Address Register (FAR)* packet, indicating where an incoming configuration data packet should be located within the FPGA. The second is the *Cyclic Redundancy Check (CRC) Register* packet, whose purpose is to protect against bitstream file corruption.

Partial bitstream relocation is ultimately accomplished via FAR packet manipulation. Provided each PRR's configuration data address range is known a priori, the PRR targeted by a given partial bitstream can be dynamically determined by updating each FAR packet in the partial bitstream. Since updating each FAR changes the contents of the partial bitstream, the CRC register packets must also be dynamically recalculated and updated during relocation. Alternatively, it is possible to disable the CRC check and ignore the CRC register packets. Since bitstream corruption is not currently of concern, the CRC check is disabled by default in our implementation, but the CRC check can be enabled in applications where bitstream corruption is a concern, most notably RFT (Section II).

B. Local Clock Domains (LCDs)

We leverage regional clock resources present in Virtex-4 and Virtex-5 FPGAs for LCD implementation. Detailed documentation of regional clock resources can be found in [9] and guidelines for their usage within PR can be found in [4].

To implement LCDs, the BUFR primitive must be instantiated inside each PRM. The BUFR's support for multiple clock domains is realized with the BUFR_DIVIDE attribute. This attribute can be assigned any integer value between 1 and 8 or BYPASS. The propagation delay through the BUFR primitive is less for the BYPASS value than integer values. However, the propagation delay for integer values is small, less than 0.5 ns, imposing minimal impact on the clock signal.

The BUFR_DIVIDE parameter is part of the configuration data and can be reset during reconfiguration. The configuration of the BUFR primitive is independent of global clock primitives (e.g. DCM) and is isolated from potential issues associated with reconfiguration of such primitives.

C. Xilinx EA PR Design Flow Modifications

Substantial modifications to Xilinx Inc.'s EA PR design flow are necessary to implement partial bitstream relocation and LCDs. The baseline EA PR design flow is documented in [10]. In this section, we provide necessary design flow modifications to produce partial bitstreams compatible with bitstream relocation.

The baseline EA PR design flow allows resources from the static region to occupy unused resources in PRRs. Since a partial bitstream contains configuration data for an entire PRR, a partial bitstream from the baseline EA PR design flow may contain static resources not intended for relocation. Relocation

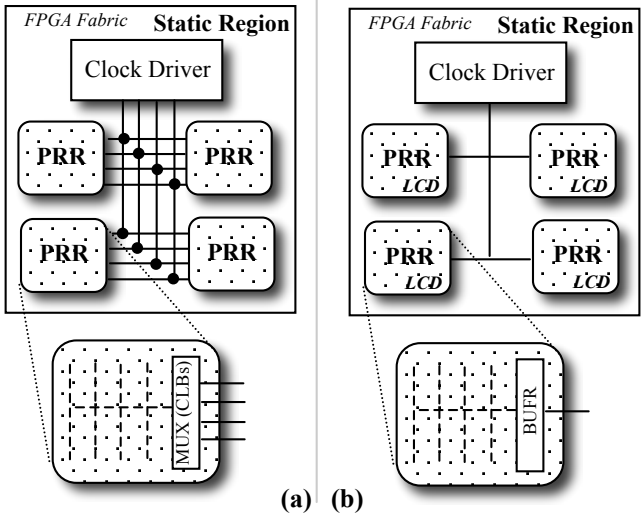

Figure 2. (a) Multiple clock domains provided to each PRR via multiple global clock signals. (b) Local clock domains (LCDs) within each PRR providing multiple clock domains with a single global clock signal and BUFR primitives.

of a partial bitstream generated by the baseline EA PR design flow is not deterministic and is not recommended. Placement of these static resources inside relocatable PRRs can be prohibited with the following UCF file constraint:.

```
AREA_GROUP "PRR_X" ROUTING=CLOSED.
```

It should be noted that this constraint may have several high-level implications, such as preventing PRRs from containing input/output buffers (IOBs). Additionally, the routing, area, and performance of the static region may be negatively affected. However, detailed investigation of these issues is outside the scope of this paper.

The preceding constraint is sufficient for partial bitstream relocation without LCDs and should be used during implementation of the entire design. To accommodate local clock domains, the ROUTING=CLOSED constraint should be used only during implementation of the static region. This constraint should be removed during PRM implementation, as routing errors occur when this constraint is included.

As described in [4], the appropriate BUFRs must be included in each PRR containing an LCD, using the UCF file constraint:

```
AREA_GROUP "PRR_X" RANGE=BUFR_X#Y#:BUFR_X#Y#,
```

where "#" represents valid BUFR X and Y coordinates. This constraint along with BUFR instantiation in the PRMs is sufficient for LCD implementation (no bitstream relocation). To maintain the consistent signal interface necessary for bitstream relocation, the relative location of BUFRs between PRRs and PRMs must be consistent.

IV. Results

We evaluate the benefits of our methodology by comparing partial bitstream relocation with LCDs to PR system designed using a *baseline PR* design flow. The baseline PR design flow refers to a PR system designed with the traditional EA PR design flow, excluding our modifications described in Section III.

Partial bitstream relocation for LCDs offers significant advantages to the RFT and VA application domains over the baseline PR design flow. Partial bitstream relocation benefits the VA domain through increases in scheduling flexibility. This benefit is application-dependent, difficult to quantify, and the focus of future work. Both RFT and VA domains benefit from partial bitstream relocation in bitstream storage requirements. The storage improvement is proportional to the number of PRRs in the system.

LCDs remove the routing inefficiencies and timing complications associated with routing clock signals through bus macros. Additionally, without LCDs, multiple global clock signals are needed to implement multiple clock domains, with each domain having a corresponding global signal. With dynamic runtime PRM placement, each global clock signal must be routed to each PRR to maximize placement flexibility. Figure 2 illustrates how LCDs provide the same flexibility with a single global clock signal, eliminating the routing overhead associated with multiple global clock signals.

This routing overhead also introduces power consumption overhead. Degalahal and Tuan state that clock signals are responsible for a significant portion of the power consumption [3] and Lamoureux and Wilton verify that power consumption increases with the number of domains [7]. Additionally, these works also cite the usage of LCDs as a means for minimizing this increase.

V. Conclusions and Future Work

In this paper, we enhance existing bitstream relocation research to support local clock domains (LCDs) and present a detailed methodology for the integration of these techniques. Application of this methodology can eliminate redundant partial bitstream storage and communication requirements as well as enhance dynamic runtime PRM placement. Furthermore, the potential exists for reducing timing complications and power consumption, increasing PRM clock frequencies, as well as providing finer-grained management of multiple clock domains.

Future work includes application case studies to quantify the benefits of LCDs with regards to maximum PRM operating frequency and overall power consumption.

Acknowledgment

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422. We gratefully acknowledge tools provided by Xilinx.

References

[1] C. Carmichael and C. W. Tseng, "Correcting Single-Event Upsets in Virtex-4 Platform FPGA Configuration Memory," Xilinx Application Note, XAPP988 (v1.0), March 13 2008.

[2] G. Cieslewski, C. Conger, A. George, and B. Kilpatrick, "Reconfigurable Fault Tolerance for FPGA-based Space Computing," *Proc. of Military and Aerospace Programmable Logic Devices Conference 2008 (MAPLD)*, Annapolis, MD, September 15-18, 2008.

[3] V. Degalahal and T. Tuan, "Methodology for High Level Estimation of FPGA Power Consumption," *Proc of 2005Asia and South Pacific Design Automation Conference,* Jan., 2005, pp. 657-660.

[4] E. Eto, "Support for BUFR in Partial Reconfigurable Modules," Xilinx White Paper, WP344 (v1.0), May 16 2008.

[5] F.V. Filho and E.L. Horta,, "Development Tools for Partial Reconfigurable Systems," *4th Southern Conference on Programmable Logic*, March 2008, pp.249-252.

[6] A. Jara-Berrocal and A. Gordon-Ross, "SCORES: A Scalable and Parametric Streams-Based Communication Architecture for Modular Reconfigurable Systems," *Proc. of Design, Automation, and Test in Europe (DATE) Conference*, Nice, France, Apr. 20-24, 2009.

[7] J. Lamoureux and S. J. E. Wilton, "FPGA Clock Network Architecture: Flexibility vs. Area and Power," *Proc. of 2006 Int'l. Symposium on FPGAs,* Monterey, CA, Feb. 22-24, 2006, pp. 101-108.

[8] D.P. Montminy, R.O. Baldwin, P.D. Williams, and B.E. Mullins, "Using Relocatable Bitstreams for Fault Tolerance," *Proc. of Second NASA/ESA Conference on Adaptive Hardware and Systems*, Washington, DC, Aug. 2007, pp. 701–708.

[9] Xilinx, Inc., "Virtex-4 FPGA User Guide," UG070 (v2.5), June 17 2008.

[10] Xilinx, Inc., "Virtex Series Configuration Architecture User Guide," XAPP151(v1.7) , October 2004.

[11] Xilinx, Inc., "Early Access Partial Reconfiguration User Guide," UG208(v1.1), March 2006.

[12] Xilinx, Inc., "ISE 8.1i Development System Reference Guide," Chapter 5, 2006.

[13] Xilinx, Inc., "Triple Module Redundancy Design Techniques for Virtex FPGAs," XAPP197 (v1.0.1), July 6, 2006.

[14] Xilinx, Inc., "Virtex-5 FPGA User Guide," UG190 (v4.2), May 9 2008