Programmable Logic Core Based Post-Silicon Debug For SoCs

Bradley R. Quinton
Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada
bradq@ece.ubc.ca

Steven J.E. Wilton
Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada
stevew@ece.ubc.ca

Abstract

Producing a functionally correct integrated circuit is becoming increasingly difficult. No matter how careful a designer is, there will always be integrated circuits that are fabricated, but do not operate as expected. Providing a means to effectively debug these integrated circuits is vital to help pin-point problems and reduce the number of re-spins required to create a correctly-functioning chip. In this paper, we show that programmable logic cores (PLCs) and flexible networks can provide this debugging capability. We elaborate on our PLC based debug infrastructure and summarize our current research. We address issues such as defining the debug architecture and debug methodology, determining the expected area overhead, optimizing the interconnect topology, creating a high throughput multi-frequency on-chip network and building efficient interfaces between the PLC and fixed-function logic. Finally, we outline a number of directions for ongoing research in this area.

1. Introduction

Advances in integrated circuit (IC) technology have made possible the integration of a large number of functional blocks on a single chip. There are many challenges associated with this high level of integration. One of these challenges is ensuring that the IC design is functionally correct. Although pre-fabrication simulation is used to help ensure that the IC performs as desired, the complexity of modern ICs prevents exhaustive verification. Design errors (bugs) are often found post-fabrication. For complex ICs, the process of verifying and debugging new devices is a significant cost and time investment [1]. As the level of IC integration continues to rise, this problem will be exacerbated as more functionality is combined into a single "black-box" that must be tested using only the external chip interfaces.

Debugging fixed function ICs presents a significant challenge. It is difficult to observe or control signals within fixed-function ICs. After fabrication, the chip cannot be easily modified to provide these signals to output pins where they can be observed. Even if the signals were identified before fabrication, providing external access to these signals at design-time is problematic since I/O resources are often limited and do not operate at the high speeds that would be required.

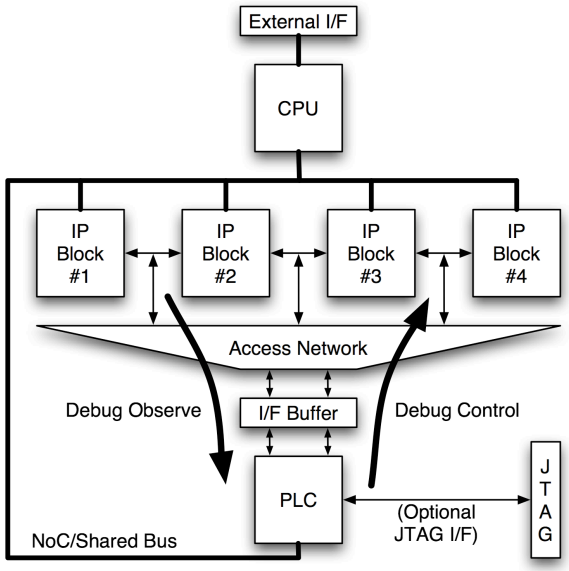

Figure 1 Debug Infrastructure

We propose a post-silicon debug infrastructure based on *programmable logic cores* (PLCs) and a *programmable interconnect network.* The post manufacturing re-configurability of the network and programmable logic core is a key aspect of the technique. If we consider the post-fabrication verification process, the region of the circuit being debugged may change over time, and in any case, it is not likely to be predictable during design time. The

flexible network allows the verifier to select the internal signals that are of interest for any given test, and the programmable logic core provides a means to process these signals in a manner that depends on the debugging task being performed.

Various methods have been used in the past to assist functional debug of post-silicon ICs. None of these have been systematic or general purpose. Some of these methods have been ad-hoc, such as controlled un-loading of DFT scan chains, and thermal emissions. Other methods are more formalized but directed at specific designs, such as in-circuit emulation (ICE) for stand-alone processors. This targeted mechanism is not general since it relies on keys characteristics of the processor. With the trend towards system-on-chip (SoC) designs, application-specific logic, high-speed interfaces and processors are being combined on a single die. In order to effectively debug this logic, flexible, full speed, real-time debugging is required.

Our infrastructure targets a generic SoC design containing multiple IP cores, as shown in Figure 1. Typically, at least one of these cores will be a processor, and often they will contain one or more high-speed interfaces. These cores can be connected either by fixed (direct) wires, a shared system bus, or Network-on-Chip (NoC).

A number of programmable debug/repair architectures have been described since our initial proposal of using programmable logic cores to facilitate post-silicon debug [2]. Abramovici, *et al.* (DAFCA) have described their reconfigurable design-for-debug infrastructure for SoCs [3]. Like our proposal, the DAFCA infrastructure is targeted at general-purpose digital logic in a SoC design. The DAFCA infrastructure is based on a distributed heterogeneous reconfigurable fabric. Unlike DAFCA, we propose using a centralized, generic programmable logic core. Therefore there are important differences between the two architectures. First, we make use of existing FPGA architectures and CAD tools to implement our debug circuits. This is an important difference since it leverages existing research on the architecture, synthesis, placement and routing of island style FPGAs [4]. Second, by centralizing the programmable logic we can make efficient use of an expensive resource. Each new debug circuit can reuse the same programmable logic. Third, by using generic programmable logic we ensure that debug circuits are fully flexible, and not restricted to certain functionality. For, example our infrastructure does not require any explicit trigger signals to be defined in advance, therefore it is possible to define the required triggering scheme at debug time. Fourth, by using a general-purpose core, and connecting it to the system bus, we enable the possibility of constructing error correction circuits or adding new features to the device after manufacturing. Finally, our centralized core imposes the requirement of a high-speed, global, configurable access network. This network and the interface between the network and a PLC are key aspects of our research in this area as outlined in Section 2.

Sarangi, *et al,* have described a proposal for using programmable hardware to help patch design errors in processors [5]. As part of their patching process they make use of programmable logic to detect specific conditions in the processor. In some cases, after the detection of these specific problem conditions, they can make use of existing processor features, such as pipeline flushes, cache re-fills, or instruction editing to correct the error; in other cases they can cause an exception to be serviced by the operating system or hypervisor. Their proposed architecture is distributed and is targeted at a specific type of design, namely modern processors. The programmable logic architecture that they use is more targeted than the one that we propose. They make use of a PLA-type fabric, which increases performance and lowers overhead, but is not able to implement arbitrary debug circuits. The primary motivation of their proposal is the in-field correction of processor design errors, and not post-silicon debug, however it is evident that their proposal could also be useful during the debug stage. Although our initial focus has been debugging, we have designed our infrastructure so it could also be used to detect and correct errors in a similar fashion. The PLC in our infrastructure can be configured to drive signals in the fixed function design. For example, the debug logic in the PLC could trigger an interrupt on the embedded processor if a specific error condition occurs in part of the design. The embedded processor could then take action to fix the problem.

2. Research Results

Our research is focused on the implementation of a post-silicon debug infrastructure based on programmable logic cores (PLC). We have addressed a number of key challenges including defining a basic architecture and debug methodology, determining the expected area overhead, optimizing the interconnect topology, creating a high throughput multi-frequency on-chip network and building efficient interfaces between the PLC and fixed-function logic. These research results will be summarized in the following sections.

3. Architecture, Methodology, Overhead

Our basic debug infrastructure is built around a programmable logic core (PLC) as shown in Figure 1 [2]. The key elements of this infrastructure are the PLC, the access network and I/F buffering. The infrastructure also makes use of an embedded processor and/or JTAG interface [6], which are likely to already be included in a SoC design. The PLC is connected to the rest of the chip using an interface buffer and access network. In addition, the PLC can communication with the on-chip CPU using a shared bus or NoC.

The configuration of our debug infrastructure at design time proceeds as follows. The SoC designer does not yet know whether the chip will fail, and if so, how it will fail. Thus, it is impossible to select a set of signals within the integrated circuit that will be directly connected to the PLC. Instead, the designer chooses a much larger set of signals (which we refer to as the *observable/controllable signals*) from the integrated circuit, and connects them to the input of the access network, as shown in Figure 1. Although our architecture allows a large number of signals to be observable/controllable, it will only cover a small percentage of the total internal signal in an integrated circuit. A subset of the total signals must be selected as observable/controllable signals. Although this process is somewhat design-dependent and manual, recent work has been done on the problem of automatic selection of important signals for debug [7]. For our work we assume that all signals that are used to connect IP blocks to other IP blocks at the top-level of the SoC are selected as observable/controllable signals. However, our technique will work equally well if the designer wishes to provide the ability to control/observe selected intra-IP block signals as well.

The debug process proceeds as follows. After the chip has been fabricated, and is deemed to require debugging, the designer (or verifier) can program debugging circuitry into the programmable logic core (PLC). The PLC is used to pre-process observed data before passing it to the on chip processor (or JTAG interface), and to provide high-speed control of the IC's internal signals based on directions from the on-chip processor (or JTAG interface). By allowing debug signals to be manipulated, the use of the PLC significantly reduces the off-chip communication requirements and allows for real-time, clock cycle accurate observation and control. For example, a counter may record the number of transitions on a signal over given period, and only return the total number of transitions. Another debug circuit could count the number of clock cycles between two signal assertions. Or, in another case, key bytes in a data stream could be recorded while other unimportant signals could be discarded.

In addition to programming the PLC, the designer or verifier can select signals from the set of observable/controllable signals to connect to the PLC. Since the number of pins on a PLC is typically smaller than the number of observable/controllable signals, an access network is used to select the observable/controllable signals that will drive and be driven by the PLC. The configuration of this network (i.e. which observable/controllable signals are connected to the PLC) can done at the same time as the PLC by programming memory bits that control the routing paths in the network.

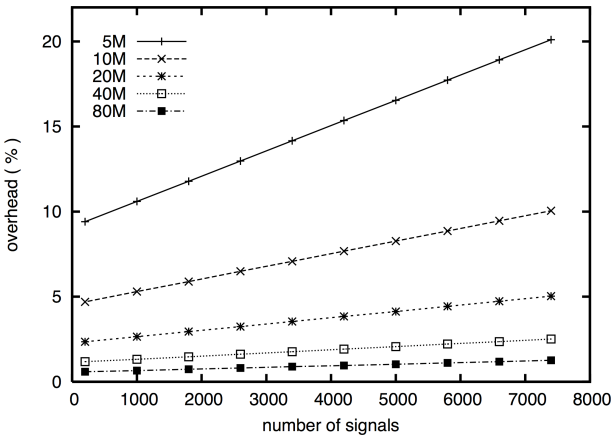

Figure 2 Area Overhead

To better understand the area overhead of our proposal, we created a parameterized model of SoC designs with and without our debug infrastructure. The model is parameterized based on the total number of gates in all IP cores and the number of observable/controllable signals. The area is calculated by summing the size of the IP cores themselves, a published area estimate of a specific programmable logic core and the area of standard cell implementations of the interface circuitry and access network. A 90nm technology and the STMicroelectronics Core90GPSVT standard cell library were assumed throughout [8]. The PLC we assumed was the 90nm PLC from IBM/Xilinx and we used the area estimate published for this core [9]. This PLC is equivalent to approximately 10,000 ASIC gates. We believe that this is more than enough capacity for the debug circuits we are considering. Therefore our overhead numbers can be viewed as conservative. The PLC has 384 available I/O ports. We assigned half of these I/O ports to be inputs and the other half were

assumed to be outputs. The interface to the NoC/shared network is implemented in the PLC logic and therefore required no additional area overhead.

For each integrated circuit size, the number of observable/controllable signals was varied from 100 to 7200, and the area overhead relative to the original integrated circuit was plotted as shown in Figure 2. The results show that for large IC designs, the cost of implementing the extra logic to facilitate post-silicon debug is modest. For example, on a 20 million gate ASIC it would be possible to observe and control 7200 internal signals for an overhead of 5%. Note that as the IC design becomes larger, the area of the PLC is amortized and its overhead becomes less significant. For these large ICs the cost of the access networks will be the important issue.

4. Interconnect Topology

The area cost and performance of the access network is an important factor in the efficiency of our proposal. The number of switches, total depth and interconnection of the switching (*i.e.* the topology) have a direct impact of these factors. We have demonstrated that when interfacing fixed logic functions to a PLC it is possible to reduce the cost of the network by taking advantage of the configurability of the logic core [10]. A *concentrator* is a network that provides full connectivity between the inputs and outputs of a network while removing restrictions on the ordering of the outputs. This removal of the ordering constraint matches the flexibility of a PLC since any physical port can be assigned to any logical functionality. We have shown the construction of a *concentrator* network that reduces the network depth and switch cost by 1/2 compared to a fully flexible network while still providing the full connectivity needed for debug. One of our possible new concentrator constructions is shown in Figure 3.

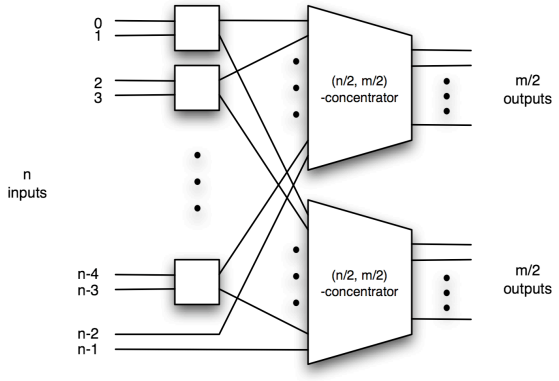

Figure 3 Concentrator Construction

5. Interconnect Implementation

A major challenge with our centralized PLC architecture is the requirement that the access network span the entire die and run at full speed. To address this we designed a two-stage concentrator network. The second stage of the network is responsible for aggregating the signals from all parts of the die. We investigated two different implementations of the second stage: a standard pipelined synchronous implementation and a novel asynchronous implementation [11]. The asynchronous interconnect was designed specifically to fit into a standard ASIC design flow. It can be built with only standard cells and optimized with current commercial CAD tools. The basic design structure is shown in Figure 4. The clock generation logic is based on discrete XOR logic to avoid clock glitching as shown in Figure 5. The asynchronous implementation has the major advantage of eliminating the need to generate a high-speed, low skew clock tree at the top-level of the SoC that would be required for synchronous pipelining flops.

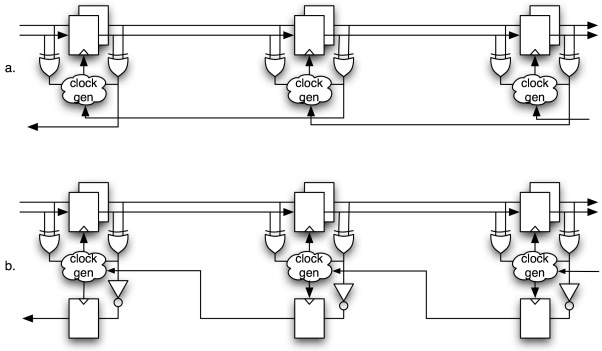

Figure 4 Asynchronous Pipeline Stages

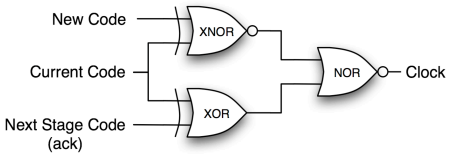

Figure 5 Clock Generation Logic

We measured the performance and efficiency of our asynchronous interconnect by performing placement and routing on a variety of trial IC configurations. The results showed, for example, that for a 5 million gate, 0.18µm IC with 64 design blocks, the synchronous network could run at speeds of greater than 250 MHz with no top-level pipeline flops, whereas the asynchronous network could run at speeds greater than 375. More recent results in 90nm

technology with an enhanced asynchronous pipelining procedure demonstrate a 14 million gate IC with 64 blocks could run at over 500 MHz synchronously, and 850 MHz asynchronously [12]. Details of these 90nm results are shown in Figure 6. Nine trial IC design configurations are represented on the x-axis. The scenarios represent cases with 16, 64 and 256 IP blocks, and die widths of 3830µm, 8560µm and 12090µm respectively. The results show that the asynchronous interconnect is able to manage high throughput signals while eliminating the need for synchronous pipeline flops.

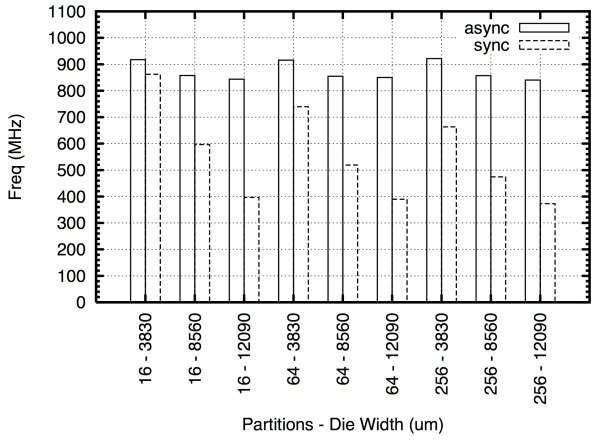

Figure 6 Throughput with no Pipeline Clock

6. Programmable Logic Interface

The architecture of our debug infrastructure requires interfacing high-speed fixed function logic to a programmable logic core. The primary challenge is managing the difference in timing performance between the fixed function logic and programmable logic. The performance of the programmable logic will inevitability be lower than that of fixed logic, therefore without careful consideration it may effect the performance of the entire IC. Initially, we focused on the interface between the system bus and the PLC [13]. Our proposal maintains the island-style structure of the PLC as shown in Figure 7. However, we propose integrating new configurable structures into the routing fabric and configurable logic blocks of the PLC to enhance the timing and efficiency of bus interfaces as shown in Figure 8. Although enhanced for bus interfaces, the modified PLC architecture maintains all the key attributes of a general purpose PLC. The standard FPGA CAD tools for placement, routing and static timing still work with only slight modification. In addition, designs that don't make use of our new enhancements incur an area overhead of less than 0.5%. For design that do make use of the new logic, our results show that, on average, our proposal would improve PLC interface timing by 36% while reducing the congestion and logic usage by 29% and 8%, respectively.

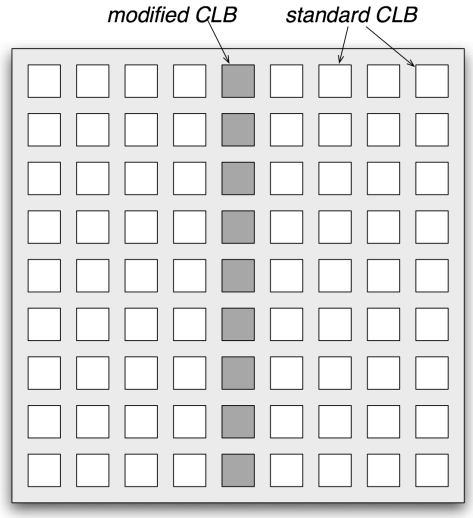

Figure 7 Modified PLC Architecture

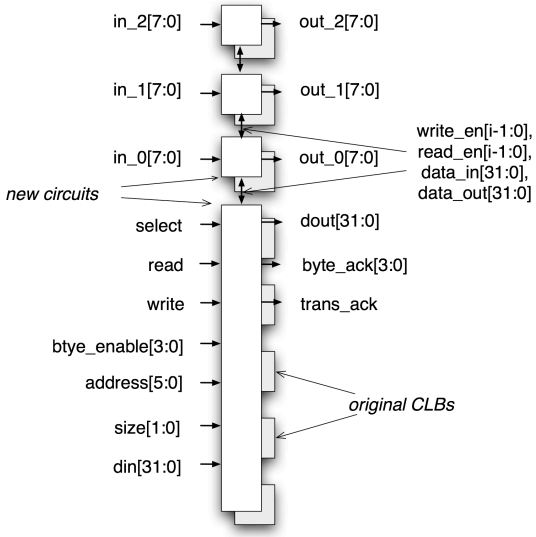

Figure 8 System Bus Shadow Clusters

7. Ongoing Work

Our ongoing work will continue to enhance our proposed debug infrastructure. We will focus on three areas. First, we intend to generalize our work on fixed-function to PLC interfaces to support generic synchronous and asynchronous interfaces. This will effectively pull the block labelled I/F Buffer in Figure 1 into the PLC structure. We anticipate that this will

improve interface efficiency and timing while enhancing the flexibility of the interface. Second, we intend to implement our infrastructure on an existing industrial SoC, and perform simulated post-silicon debugging. This will help us to further our understanding of the costs and requirements of our proposal. We also anticipate that we will be able to better understand the required size of the PLC and the number of internal signals required for effective debug. Thirdly, we intend to extend the functionality of our infrastructure and utilization methodology to address error correction and facilitate feature additions to existing SoCs.

References

[1] S. Sandler, "Need for debug doesn't stop at first silicon", *E.E. Times*, Feb. 21, 2005.

[2] B.R. Quinton and S.J.E. Wilton, "Post-Silicon Debug Using Programmable Logic Cores", *Proc. IEEE Int. Conf. on Field-Programmable Technology*, Dec. 2005.

[3] M. Abramovici, "A Reconfigurable Design-for-Debug Infrasctructure for SoCs", *Proc. Design Automation Conference*, July 2006.

[4] V.Betz, J. Rose, and A. *Marquardt, Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, 1999.

[5] S. Sarangi, et al., "Patching Processor Design Errors With Programmable Hardware", *IEEE Micro*, Jan-Feb 2007.

[6] IEEE Std. 1149.1-1990, IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture, *IEEE Computer Society*, 1990

[7] Y. Hsu, "Visivility Enhancement for Silicon Debug", *Proc. Design Automation Conference*, July 2006.

[8] STMicroelectronics, "CORE90 GP SVT 1.00V", *Databook*, Oct. 2004.

[9] P. Zuchowski, et al., "A Hybrid ASIC and FPGA Architecture", *Proc. IEEE/ACM Inter. Conf. on Computer-Aided Design*, pp. 187-194, 2002.

[10] B.R. Quinton and S.J.E. Wilton, "Concentrator Access Networks for Programmable Logic Cores on SoCs", *Proc. of the IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005.

[11] B.R. Quinton, M.R. Greenstreet, and S.J.E. Wilton, "Asynchronous IC Interconnect Network Design and Implementation Using a Standard ASIC Flow", *Proc. IEEE Int. Conf. on Computer Design*, Oct. 2005, pp. 267-274.

[12] B.R. Quinton, M. Greenstreet, S.J.E. Wilton, "Practical Asynchronous Interconnect Network Design", *in review for IEEE Transactions on VLSI.*

[13] B.R. Quinton and S.J.E. Wilton, "Embedded Programmable Logic Core Enhancements for System Bus Interfaces", *submitted to IEEE Conf. on Field-Programmable Logic and Applications*, 2007.