The Development of a Post-silicon Debug Infrastructure to Enhance the Validation of Integrated Circuits

Author: Brad Cameron

Introduction

1.1 Motivation

Integrated circuits are becoming increasingly complex. Today, state-of-the-art chips contain as many as 486 million transistors [], and that ~~is~~ number is expected to increase to 4.4 billion by 2015 []. This capacity ~~is~~ has enabled ~~these~~ integrated circuits (ICs) to provide the equivalent functionality of an entire system on a single device~~;, and~~ hence, these large ICs are often called Systems-on-a-Chip (SoC) []. However, ~~E~~ensuring that such complex devices operate as expected is challenging. Failures can occur because of design errors, fabrication errors, or impurities in the base silicon. Of these, design errors are often the most difficult to uncover.

The design of an integrated circuit typically follows a very structured methodology. Designs are specified using a variety of techniques, and Computer-Aided Design (CAD) tools are used to translate these designs to transistors and then to physical layout information that can be directly fabricated. ~~At all levels of the CAD flow, t~~The design is simulated in an attempt to identify as many design errors (bugs) as possible at all levels of the CAD flow. In large designs, as much as 50-70% of the pre-silicon design time is spent simulating the design []. In addition, formal verification techniques are often employed to prove **certain properties of the design (that the design has certain properties?)** [ref, one of Alan's papers]~~;.~~ ~~–~~when ~~Together,~~ simulation and formal verification **are combined (considered together?)**, it is termed *pre-silicon verification*. Once the designer has achieved a certain level of confidence that the design is correct, the chip is manufactured. Creating the lithography masks required for manufacturing can cost up to $1.2 million and take up to 16 weeks [].

Regardless of how careful a designer is, a certain number of bugs will escape the pre-silicon verification process. In the design of the Intel Pentium 4, researchers report that their simulation

effort required 6000 processors, each running for two years, and that their effective simulation coverage was less than 1 minute of real time operation []. Clearly, this is not sufficient to uncover all existing bugs ~~which may exist~~. Nonetheless, ~~Yet~~ it is important to find all these bugs before products are shipped; to not do so can be very costly. For example, ~~I~~in a well-publicized incident, a simple bug in the floating point division unit of the original Pentium chip cost Intel Corp. approximately $475 Million [].

The only way to find these bugs is to test manufactured chips. Unlike pre-silicon verification, testing manufactured chips can provide a much larger coverage~~,~~ because ~~since~~ the chip can run at-speed~~,~~ and be connected to other integrated circuits in a larger system. **Testing of the chip after fabrication (Post-fabrication testing of the chip)** is termed *post-silicon validation*~~.~~ ~~According~~ which ~~to~~ the 2007 International Technology Roadmap for Semiconductor (ITRS) describes (defines?) as:~~, "Post-silicon validation … is~~ "an area of verification which has grown rapidly in the past decade, mostly due to the large complexity of silicon-based systems…" []. Recent studies have shown that 35% of development time now occurs after the initial device fabrication **due to the need to employ post-silicon validation,** and this proportion continues to grow [].

When failures are observed ~~D~~during post-silicon validation, ~~when failures are observed,~~ it is critical to uncover the source of the failure in the original design. This process is termed *post-silicon debug*. Time-to-market pressure means that these design errors must be discovered quickly and~~. I~~it is ~~also~~ essential that as many errors as possible are uncovered before the chip is subject to another lengthy and costly manufacturing *spin*. Yet, finding these bugs is very challenging, mainly because of a lack of internal visibility. According to the same ITRS report quoted above, the key issue with post-silicon validation~~–~~ "is the limited visibility inside the manufactured silicon parts." []. As designers continue to integrate more functionality on an integrated circuit, the complexity of these devices will increase~~s~~ significantly. At the same time, the inter-component communications that could once be observed by probing traces on a printed circuit board (PCB)~~,~~ are moving inside the chip. This leaves the validation process in the position of having to manage more complexity with less visibility. In addition, as the likelihood of post-silicon bugs increases, the possibility that a bug will prevent extensive validation of the other portions of the chip **concurrently** increases. This puts chip designers in the unfortunate

position of having to potentially re-spin a device~~,~~ simply to enable further validation (which may, in turn, discover another bug).

Integrated circuit designers are attempting to address this issue by adding extra logic to their designs specifically to **assist** **(better facilitate)** the post-silicon debug process []. New post-silicon debug concepts are emerging.~~A~~ that are analogous to design-for-test (DFT) methodologies whereby (OR wherein) designers add dedicated logic to their design to **allow for** **(better facilitate?)** efficient manufacturing test procedures, ~~new post-silicon debug concepts are emerging.~~

While it is possible to achieve some level of post-silicon debug by reusing ~~re-using~~ existing DFT structures and/or software debug facilities, it has been shown that these techniques are not sufficient for modern, high-performance integrated circuits and SoCs []. Because of this, even designers of the latest high-volume, cost-sensitive, multi-core processors are willing to add extra logic to assist with the debug process. The AMD Opteron and IBM Cell BE both contain specialized chip-level *trace buffers* that are not tied to a single processor core or software debugger, but are intended to debug system level issues []. In both cases these debug resources have proved to be quite useful during the post-silicon validation phase. The Cell BE validation team described the successful debug of an intermittent cyclic redundancy check (CRC) error in the bus interface controller (BIC) that was caused by an hardware initialization failure []. Likewise, the AMD validation team described the successful debug of a dead lock scenario in an 8-node (4-(?) processor) configuration that was triggered after four days of continuous operation [].

Although these emerging techniques increase internal visibility to a degree, the current post-silicon debug solutions are limited in a number of ways. They are both quite design-specific~~,~~ and are not flexible enough to handle many unexpected debug situations: both the IBM and AMD proposals are limited to observing correctly formatted transactions on a specific internal interconnect, which significantly reduces the number of potential circuits that can be debugged on a given device~~,~~ as well as the number of scenarios for which the debug resources are useful. For instance, if a design block ~~–~~ ‘~~“~~locks-up’~~”~~, and ceases to generate new transactions, there will be no new transactions to observe. In this case, t~~T~~he root-cause~~, in this case,~~ is likely within the design block. However~~, with the current proposals,~~ these internal nodes cannot be observed with the current **proposals (techniques? proposed techniques? technique proposals? solutions?)**.

In addition, transactions are logged in dedicated, finite sized trace buffers and~~. A~~a limited set of configurable triggers and compression techniques are employed to make more efficient use of these buffers. ~~, h~~However, since the nature of the bug is not known in advance, these triggers and compression techniques are not always appropriate. Finally, since the debug information in these proposals is processed after the test, and not in real-time, there is no possibility ~~to~~ of taking action when an error is detected during normal operation.

In this thesis, we propose a new *reconfigurable* post-silicon infrastructure that can be used to assist the debug of any digital circuit. Similar to existing solutions, our proposal **provides visibility into (enhances the visibility of OR facilitates the visibility of)** the internal operation of the integrated circuit. However, the use of general-purpose programmable logic at the core of our infrastructure enables a number of key methodologies that are not available in other proposals. First, our ~~**proposal**~~ **infrastructure** has the flexibility to target arbitrary digital logic in a chip and is not restricted to specific circuits such as processor cores**, (delete comma if system busses are specific circuits)** or system busses. Rather, it ~~Instead our proposal~~ can be used to monitor key aspects of the internal design, such as the current **state (condition)** of a state-machine. The programmable logic **thus** allows us to build debug circuits to interpret any digital pattern. Second, our proposal allows for complex, scenario-specific, event triggers as well as ~~,~~ event filtering~~,~~ and trace compression, thereby making more efficient use of debug buffering~~;~~, enabling longer running debug scenarios~~;~~ and providing real-time monitoring. Third, our proposal enables the detection and potential correction of design errors during normal device operation by enabling the creation of new digital circuits in the programmable logic~~,~~ which can operate in parallel with the normal operation of the device. ~~, our proposal enables the detection and potential correction of design errors during normal device operation~~.

We believe the availability of this type of dedicated debug logic will provide a number of key benefits in ~~an~~ **integrated circuit development (OR the development of an integrated circuit),** including: 1) reduced time-to-market because of an increased ability to quickly isolate and understand unexpected behaviours, 2) decreased resources required for post-silicon validation because of an increase in the functional coverage of a given test, 3) elimination of design revisions **caused when (which result when)** one design error hides **a second (another)** error, and 4) increased customer satisfaction because of the enhanced ability to **provide quick ‘~~“~~work-arounds’~~”~~ to known bugs (quickly ‘work-around’ known bugs)**. We expect that "design-for-

debug" structures will become commonplace in all large integrated circuits[]~~;~~ and that structures such as ours will become a key part of this infrastructure.

1.2 Focus and Contributions of this Thesis

The focus of this thesis is the development of a new reconfigurable post-silicon debug infrastructure to enhance the validation of integrated circuits. This new infrastructure addresses **the (?)** key limitations of existing solutions and **provides a significant improvement on the state-of-the-art in this area (significantly improves on the state-of-the-art of debug infrastructures OR such infrastructures)**. We address the architecture, operation and implementation of this infrastructure**. Through the implementation we identify (In the implementation we identify OR The implementation allows us to identify)** three key areas for targeted research: 1) the development of low-~~-~~cost and low- depth network topologies for connecting fixed function circuits to circuits implemented in programmable logic, 2) the effective implementation of high-speed on-chip interconnects, and 3) the implementation of high-speed, rate-adaptive circuits in embedded programmable logic. We address each of these key areas in detail in this thesis. The results of these targeted research efforts both enable ~~both~~ the implementation of our infrastructure and provide results that can be extended to the implementation of integrated circuits in general. Although ~~W~~we ~~have~~ also considered the design of specialized programmable logic cores for the task of circuit debug [], ~~however in this thesis~~ we ~~have~~ decided to focus on the use of general-purpose programmable logic cores in this thesis. The research contributions of this thesis can~~,~~ therefore~~,~~ be grouped into four main areas~~,~~; each of which is summarized in the following subsections.

1.2.1 A Reconfigurable Post-Silicon Debug Infrastructure

The first major contribution of this thesis is the development of a new post-silicon debug infrastructure for complex integrated circuits and SoCs. As the level of SoC integration continues to increase, the difficulty of producing a functionally correct chip continues to grow []. A significant effort is made during the SoC development process to verify the correct behaviour of the design prior to the manufacturing of the device. Both simulation-based and formal methods are used for this *pre-silicon verification*, but ~~however,~~ devices are still manufactured which do not operate as expected []. These functional defects, or *bugs*, are discovered in the

post-silicon validation stage. The process of determining the root- cause of these bugs is becoming a large component of the overall development cost []. To address this, we propose a *reconfigurable* post-silicon debug infrastructure that enables the *observation and control* of internal signals. We use programmable networks and embedded programmable logic to create our infrastructure. ~~The~~ Its adaptive nature ~~of our infrastructure~~ is well suited to the problem of device debug since the bugs are, by their very nature, unexpected; ~~.~~ ~~U~~unlike existing solutions [], ~~our infrastructure~~ it can be reconfigured for each specific debug scenario. In addition, our reconfigurable infrastructure not only enable~~s, not only,~~ the diagnoses of bugs, but also allows the detection (and **potentially the correction OR the potential correction**) of errors in normal operation.

(Inserted Paragraph Break) In Chapter 2 we describe the architecture, operation and implementation of our new infrastructure, and~~. We~~ then analyze the area overhead of the infrastructure. The results show that it is possible to implement our reconfigurable post-silicon debug infrastructure with an area overhead of less than 10% for a large proportion of our target-integrated circuits. This work is significant because it demonstrates that it is possible to create a debug infrastructure with significant flexibility and that it is feasible to implement this infrastructure in an integrated circuit with a reasonably low area overhead.

1.2.2 Concentrator Network Topologies of SoCs

The second major contribution of this thesis is the construction of a new network topology that takes advantage of the flexibility of the programmable logic in our post-silicon debug infrastructure. We use this flexibility to decrease the area overhead and improve the network timing of our proposal. The area cost and performance of the programmable access network is an important factor in the efficiency of our overall infrastructure. The number, depth and interconnection of the switches (*i.e.* the topology) have a direct impact on these factors. We demonstrate that a class of unordered, non-blocking networks, **called** *a concentrator* ***(or a concentrator)***, is well suited to the task of connecting fixed function circuits to a programmable logic core. Concentrator networks have been studied previously []. In most cases the work has been theoretical with limited focus on the application of these networks. In our case, these networks can take advantage of the flexibility of the input and output assignments on the programmable logic cores, which removes the ordering constraint.

(Inserted Paragraph Break) In Chapter 3 we describe two new concentrator constructions. The first is shown to have **a lower area cost (OR lower areas costs?)** and lower switch **depth (depths?)** than previously described concentrators for network sizes that are of interest in our application. The second network construction is optimized specifically for the post-silicon debug infrastructure. It extends the first network construction to enable a network that can be partitioned into two levels of hierarchy. We show that this network can be implemented with low area overhead, resulting in as little as 2% overhead for most of our target implementations. This **work (quality, factor OR capability)** is significant for our post-silicon debug infrastructure since it enables a more efficient implementation. It is also salient ~~significant~~ to the more general problem of integrating embedded programmable logic into **fixed functions (fixed-function?)** integrated circuits~~, since~~ because it highlights an important element of flexibility on the interface between the two types of logic.

1.2.3 Asynchronous Interconnect Network Design and Implementation

The third major contribution of this thesis is the design and implementation of a new asynchronous interconnect network design that eliminates the need for top-level clock tree insertion while enabling high-speed operation for on-chip networks. A major challenge **with (to)** the centralized programmable logic in our post-silicon debug infrastructure is the requirement that the access network spans the entire device and run at full speed. Synchronous pipelining techniques have been proposed for this type of implementation [], but ~~. However, these techniques~~ they require the design of a low-skew clock tree that is distributed throughout chip. As ~~We present a~~ **the new asynchronous interconnect design and implementation we present (the new asynchronous interconnect design and implementation we present)** ~~that~~ does not require a global clock, it ~~and~~ therefore~~,~~ it has a potential advantage in terms of design effort.

There have been a number of other asynchronous interconnect proposals [], but~~. However,~~ they ~~se proposals~~ were not well suited to implementations with standard CAD tools. In contrast, our asynchronous interconnect design can be implemented using a standard ASIC design flow. **In Chapter 3 (Accordingly OR To that end, in Chapter 3 - *transition between sentences*)** we build on the two-stage concentrator network developed in Chapter 2. We target the second stage of the network that is responsible for aggregating the signals from all parts of the die and **also**~~.~~ ~~We~~ compare our proposal to a standard pipelined synchronous implementation. The results

demonstrate that there is a region of the design space where the asynchronous implementation provides an advantage over the synchronous interconnect by removing the need for clocked inter-block pipeline stages**, while maintaining (and simultaneously maintains)** high throughput. We then demonstrate a CAD tool enhancement that would significantly increase the space where the asynchronous design has an advantage over a synchronous implementation. In addition, wr ~~W~~e ~~also~~ provide a detailed comparison of the power, area and latency of the two strategies over a wide range of IC implementation scenarios. This **work (tool enhancement OR comparison)** is significant to our implementation of a post-silicon debug infrastructure since it simplifies a challenging step in the insertion of our debug logic. It is also significant to the general problem of integrated circuit design ~~since~~ because it demonstrates that it is possible to create effective asynchronous circuits using standard CAD tools, even though **they (refers to the CAD tools)** have been developed primarily for synchronous design styles.

1.2.4 Enhanced Programmable Logic Core Architectures for High-Speed On-Chip Interfaces

The fourth major contribution of this thesis is the development of enhancements to the architecture of embedded programmable logic cores (PLCs) in order to enable the implementation of high-speed, rate-adaptive interfaces circuits. The architecture of our debug infrastructure requires interfacing high-speed fixed function logic (the circuits under debug) to circuits implemented in a programmable logic core (the debug circuits). 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 []. Without careful consideration the programmable logic may affect the performance of the overall debug infrastructure. We address this problem by proposing changes to the structure of the PLC itself; these architectural enhancements enable circuit implementations with high performance interfaces.

INSERTED PARAGRAPH BREAK In Chapter 5, we demonstrate PLC architectural changes that target both *system bus interfaces* and *direct synchronous interfaces*. These changes in the PLC architecture maintain all the key attributes of a general purpose PLC []; ~~. T~~the standard FPGA CAD tools for placement, routing and static timing still work with only slight modification []. Our results demonstrate a significant improvement in PLC interface timing~~;~~,, **ensuring (which ensures)** that interaction with full-speed fixed-function logic is possible. We

are able to show that we can do this without compromising the basic structure or **routiblity (route-ability)** of the programmable fabric. In addition we show that these new structures are very area efficient. The area impact on circuit designs that do not need to make use of our new enhancements is less than 1%. These results are significant to our post-silicon debug infrastructure because~~,~~ using these PLC enhancements~~, we are~~ enables us to create debug circuits that are able to interact directly with the full-speed logic in the circuit under debug. ~~These results are also significant to the architecture of programmable logic cores and integrated circuits in general, since~~ As the results ~~y~~ demonstrate that circuits implemented in programmable logic can interface directly to fixed- function circuits, they are also significant to the architecture of programmable logic cores and integrated circuits in general.~~.~~