Using Embedded Infrastructure IP for SOC Post-Silicon Verification

Yu Huang
Mentor Graphics Co.
1601 Trapelo Rd.,
Waltham, MA, 02451
Yu_huang@mentor.com

Wu-Tung Cheng
Mentor Graphics Co.
8005 S.W. Boeckman Rd.,
Wilsonville, OR 97070
Wu-tung_cheng@mentor.com

ABSTRACT
This paper presents a method to embed an FPGA core in a SOC as an infrastructure IP that can exploit transaction-based verification methodology to verify and debug the first silicon. The primary objective for post-silicon verification is to reduce the time taken for validating the first silicon. Additionally, verifying silicon at chip-level is expected to speedup the silicon debugging and thus reduce the time to market. Experimental results presented in this paper demonstrate that the proposed method can be implemented with small overhead.

Categories and Subject Descriptors

General Terms
Algorithms, Design, Reliability, Verification.

Keywords
Infrastructure IP (I-IP), FPGA Core, Post-Silicon Verification, Transaction-Based Verification.

1. INTRODUCTION
The developments of modern semiconductor process technologies and EDA tools continuously allow the realization of a more complex system on a single die with higher clock frequency and larger system density. Reusing embedded IP cores in System-On-Chip (SOC) has played a key role to improve the frequency and larger system density. Reusing embedded IP cores a more complex system on a single die with higher clock technologies and EDA tools continuously allow the realization of 1. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2003, June 2-6, 2003, Anaheim, California, USA.
Copyright 2003 ACM 1-58113-688-9/03/0006…$5.00.

pre-layout verification methods such as simulation based technologies, static timing verification and emulation systems quite inaccurate. Although some post-layout physical verification methodologies are available in the industry, the verification problem cannot be completely solved at pre-silicon phase because the above-mentioned problem is further exacerbated by signal integrity (SI) and design integrity (DI) issues with the evolution to DSM. SI issues include crosstalk, IR drop, power and ground bounce, etc. DI issues include electron migration, hot electron, wire self-heating, etc [2]. In this scenario, it is quite likely that the first silicon is designed with errors that escape the pre-silicon verification. Hence the designers are confronted with two problems:

In case that the design fails verification, we need a method to debug the silicon effectively and efficiently.

In this paper, we propose a novel idea called “Post-Silicon Verification.” The basic idea of our method is to embed an Infrastructure IP into a SOC to monitor the transactions among different cores in verification mode and flag an error in case an illegal transaction is detected. In debug mode, it will pinpoint which steps of the transaction failed the verification and provide users the status of the failed step for further analysis.

The paper is organized in the following manner. In Section 2 we give a brief review of related work. In Section 3 the proposed Post-Silicon Verification architecture is described with an application example. Experimental results are presented in Section 4, followed by the conclusion section.

2. REVIEW OF RELATED WORK
2.1 Infrastructure IP
In SOC designs, the design process involves an IC that is often made up of large pre-designed and pre-verified reusable semiconductor intellectual-property (IP) blocks, which are called embedded IP cores. Most of the known IP cores perform a certain function, e.g., an embedded CPU, an embedded DSP, an embedded RAM or an embedded FPGA core, etc. In contrast, Infrastructure IP (I-IP) is not functional, i.e., doesn’t contribute to the normal functionality of a given IC. Rather, I-IP is embedded in an IC solely to ensure its manufacturability and lifetime reliability [3]. Various I-IP blocks are used for different purposes such as test [4], diagnosis [5], silicon debugging [6], fault tolerance [7] and yield optimization [8], to name a few.
2.2 Transaction-Based Verification

Transaction-Based Verification is one of simulation-based verification methods currently used as pre-layout verification. A transaction is a single transfer of a package of data and/or control. It can be as simple as a memory read/write or as complex as the transfer of an entire structured data packet (e.g., an ATM cell) through a communication channel [9]. The transactor serves as an abstraction layer between the test program and the design. It is constructed as a collection of tasks, each of which executes a particular kind of transaction. A task might also be composed of a sequence of lower level tasks, each of which executes a particular kind of lower-level transaction. The tasks at the lowest level will generate/expect a given set of waveforms on a set of inputs/outputs of the design under test. Therefore we can consider a transactor as a library of functions (tasks) organized in a hierarchical manner. A test program will generate a sequence of function calls, which will invoke the corresponding functions in the hierarchical library to perform the verification. The invoked function will also invoke its sub-functions hierarchically until reach the lowest-level function to generate the entire set of waveforms. Hence the complexity of a detailed protocol at signal-level is encapsulated within a transaction. Raising the level of abstraction from signals to transactions facilitates the creation of tests, speedups the verification and improves the productivity.

The proposed Post-Silicon Verification looks like a reversed procedure of the above-mentioned transaction-based verification. The basic idea is to embed an I-IP in SOC to monitor the transactions between cores and generate a signature (explained in Section 3) based on the detected signal sequences. The generated signature is used to identify which transaction was completed. It will be compared with the expected signature of the applied transaction to indicate whether this transaction is correctly performed on-chip. This method will provide the following benefits:

(1) As we already know, transactions are highly abstracted from a sequence of signals, comparing signatures of transactions is much more cost-efficient than comparing signal sequences.
(2) Testbenches designed at pre-silicon transaction-based verification can be reused at post-silicon verification, which saves the effort of regenerating testbenches.
(3) The function-oriented nature of the transaction-based verification makes the functional coverage analysis much easier, which improves the design confidence.
(4) Silicon debug becomes more efficient since the functional scenario for an existing bug is easily identified. However, the proposed method is suitable for silicon debug at chip-level. The previously proposed silicon debug inside a core is still necessary.

3. PROPOSED VERIFICATION METHOD

3.1 Atomic Transactions

The most important part of designing post-verification I-IP is to compute the signature for a given transaction.

Definition 1: The signature for a transaction is a hardware-generated codeword after the on-chip compression for a sequence of signals detected during this transaction.

Before we introduce how to compute the signature for a transaction, we have to explain atomic transaction.

Definition 2: An atomic transaction is the smallest unit for a meaningful functional operation. The transaction either happens completely or not at all, and if it happens, conceptually it happens in a single indivisible action.

A transaction with meaningful functionality, no matter how complex it is, can always be divided into a sequence of atomic transactions. Theoretically, as long as all atomic transactions can be extracted by monitoring the signal sequences, a transaction can always be identified by tracing a FSM. Now, the original problem of transaction identification on-chip is boiled down to two sub-problems:

(1) Is it possible to identify all atomic transactions?
(2) Is it possible to trace a FSM to identify arbitrary transactions?

The first sub-problem can be answered by considering an example. Our method is more suitable to be applied on the bus-based SOC architecture. Hence, we use ARM core based Bluetooth SOC design [10] as an example to illustrate its application scenario. It uses AMBA bus architecture [11] to integrate cores including ARM7TDMI processor core, Ericsson Bluetooth Baseband controller core, RAM, ROM/Flash, TIC interface core, etc. We add into the chip the proposed Post-Silicon Verification I-IP core, which is hooked to AMBA bus and connected with JTAG interface. In bus-based SOC architecture, typically an arbiter will decide which core is the bus master, and ensure one bus master initiates data transfer at a time. Since the proposed I-IP is only used to monitor the transactions on AMBA bus, it is only used as a slave. By monitoring the AMBA bus, it is easy to identify master. To detect which slave is involving the current transaction, one can monitor a group of control signals called “HSELx” because each slave “x” has its own “select signal” [11]. Verification of SOC requires scrutiny of all bus activity at the system functional level such as caches hit/miss, memory latency, collision monitoring, request/response matching, etc. However the number of atomic transactions between a master and a slave is actually quite small. For example, a CPU core and a RAM core have only two atomic transactions in between, i.e., Read and Write. As another example, only two atomic transactions exist between a master and an arbiter, which are Request and Grant. Assuming the Bluetooth example has 7 pairs of cores that have transactions in between (broadcast mode is not considered for simplicity) and each pair of cores has only 2 atomic transactions, the total number of atomic transaction is 7*2=14, which can be encoded with a 4-bit codeword. Therefore enumerating a complete set of atomic transactions is realistic because the total number of atomic transactions in a system is small.

The second sub-problem is intractable. Suppose a total of K atomic transaction and a transaction can be divided into D atomic transactions, a K-tree with depth of D has to be searched to
identify this transaction. \(D\) is large for a complex transaction, which makes the exact searching time and memory consuming. Therefore, we apply a signature generation method explained in the following sub-section.

## 3.2 Transaction Signature Generation

Figure 1 shows logic block structure of the proposed I-IP. The atomic transaction detector is used to identify \(K\) different atomic transactions by monitoring the bus signals. After an atomic transaction is detected, it is encoded to \(k = \lceil \log_2 K \rceil\) bits signature and is saved into a \(k\)-bit register through a De-Multiplexer. The DeMUX is controlled by a self-reset counter, which is incremental by 1 each time an atomic transaction is detected. At the start of a verification the counter is initialized to 0, and it resets back to 0 each time it counts to \(N\). The internal state of the counter will control the DeMUX to determine which register is used to store the signature of the currently detected atomic transaction. Note that there are a total of \(N\) \(k\)-bit registers to store the signatures, where \(N\) is a variable that pre-determined by considering the tradeoff between area overhead and the possibility of error masking (explained below). At the end of each verification the error flag is turned on when a mismatch is identified by comparing the currently generated signature with the expected signature stored on chip by shifting data in from JTAG interface before the verification starts.

If more than one transactions have the same \(k\times N\) bits signature, it is possible that the hardware-generated signature is same as the pre-stored signature, but in fact they represent two different transactions. Such an alias produces an error masking, which leads to lower verification quality. The error masking possibility for the proposed signature computation asymptotically approaches \((1/K^k)\).

**Proof:** For a transaction system with \(K\) atomic transactions, each transaction can be represented by one node on a \(K\)-tree. Assuming that the most complex transaction (the one has the longest transaction sequence) is composed of \(D\) sequential atomic transactions, the depth of the \(K\)-tree is \(D\). Now, if we only use \(N\) \((N-D)\) registers to store the signatures for \(N\) consecutive atomic transactions, there are total of \(K+K^2+K^3+...+K^{D-N}\) aliases for any one transaction. And the total number of transactions is \(K+K^2+K^3+...+K^D\). Therefore the possibility of an error masking for one transaction is \(P_e=(K+K^2+K^3+...+K^{D-N})/(K+K^2+K^3+...+K^D)\). Now, if all transactions are equally probable, which might not usually a realistic assumption, then \(P_e\) can be considered to be the probability of error masking for the system. Note that in order to obtain more accurate results, we need to know the realistic distribution of the transactions among a \(K\)-tree with depth of \(D\), which needs careful analysis of the transactions that possibly used in one system and hence leads to much more computational cost. When \(N\) is big enough, the previously obtained \(P_e\) asymptotically approaches \((1/K^k)\). Q.E.D.

For the given example, \(K=14\), if we set \(N=16\), then \(P_e \approx 7.143\times10^{-18}\), which is good enough in practice.

The 2-bit control signals \((C_{1}C_{0})\) come from the JTAG interface. The detailed connection between I-IP and JTAG is skipped in this paper because it is a well-known technique. These 2-bit signals are used to set I-IP to stand-by mode, verification mode or debug mode.

## 3.3 Debug Methodology

The proposed verification procedure is applied on chip for each transaction specified by user. If no error occurs during this procedure and the functional coverage reaches to a certain level, the design can be released and the FPGA core could be re-programmed to serve other purposes. However, as we already pointed out in Section one, this “lucky” scenario may not always happen due to so many uncertainties coming along with the DSM technologies. As a result, silicon debug is necessary.

Let us assume that the error flag is asserted during the verification of a transaction composed of 100 atomic transactions. We need to use a binary search to find where the fail happens, i.e., the first failing atomic transaction. The algorithm works by making a new transaction that is increasingly composed of a smaller (by half) atomic transaction subsets until the failing atomic transaction is identified. In the given example, we will first make a transaction composed of the first 50 atomic transactions of the original one and verify the new transaction on chip. If it passes the verification, then we will try another transaction that composed of the first 75 atomic transactions of the original one and so on. When we find the failing atomic transaction, we set \(C_{1}C_{0}\) to enter debug mode so that we can dump out the signals on the buses to analysis what happens at the failing point. If there is some silicon debug hardware embedded inside the cores that involved the failing transaction, it is better to observe the internal states of the core as well during the analysis.

## 4. EXPERIMENTAL RESULTS

The proposed I-IP has not been applied to a real silicon yet at the time being. However we did some experiments on designing such an I-IP and evaluating its cost. The experiments are conducted by using the example Bluetooth mentioned in Section 3.1. Note that the design we used for experiment is not a complete Bluetooth design, but is good for our evaluation purpose. We assume that there are 7 pairs of cores have transactions between, which are (ARM--Arbiter), (ARM--RAM), (ARM--ROM), (ARM--TIC), (ARM--Bluetooth Controller IP), (TIC--Arbiter) and (Bluetooth Controller IP--Arbiter). We also assume that each pair of cores could have only 2 atomic transaction such as Read/Write transactions between ARM core and the other cores (not include Arbiter) and Request/Grant transactions between Arbiter and the other cores. Therefore, a total of 14 atomic transactions need to be detected by the detector. The detected atomic transactions will be encoded into a 4-bit codeword saved to a register selected by the counter as shown in Figure 1. The total number of signature registers, \(N\), is set to 8, 16, 32 one at a time to evaluate the area overhead. The widths of data and address busses are set to 32. The control signals are compatible with AMBA-AHB standard [11]. The I-IP design shown in Figure 1 is written in VHDL and synthesized to a small size FPGA in XLINX 4000 family (4008PQ160) by using Leonardo-Spectrum™.

Table 1 shows the experimental results to evaluate the area overhead and the probability of error masking. Columns 2 to 4 give the utilization of IOs, FG Function Generators and CLB Flip Flop in XLINX4008PQ160 respectively. In the last column the possibility of error masking are presented. We assume the largest transaction is composed of 100 atomic transactions, i.e. \(D = 100\), and we calculate the exact \(P_e\) rather than using its asymptotic equation. It can be seen from the experimental results that the
The proposed method can be implemented with small area overhead. Therefore, it can be downloaded into a small size FPGA core embedded in SOC. One critical requirement is that the IO resource on the FPGA core should be enough to interface with data/address bus signals and control signals. This is especially true when wider data bus protocols (e.g., 64 or 128 bits) are used. An interesting observation is that increasing \( N \) does not lead to much functional area overhead. However, the Flip-Flops will be consumed proportionally. As for the Possibility of Error Masking, \( N=16 \) is good enough for a transaction composed up to 100 atomic transactions. This ensures the verification quality with low cost.

**Table 1. Experimental Results**

<table>
<thead>
<tr>
<th>( N )</th>
<th>IO Utilization</th>
<th>FG Function Utilization</th>
<th>CLB Flip Flop Utilization</th>
<th>Possibility of Error Masking (( P_e ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>91.67%</td>
<td>39.38%</td>
<td>8.65%</td>
<td>6.776e-10</td>
</tr>
<tr>
<td>16</td>
<td>91.67%</td>
<td>42.75%</td>
<td>16.56%</td>
<td>4.591e-19</td>
</tr>
<tr>
<td>32</td>
<td>91.67%</td>
<td>47.66%</td>
<td>27.14%</td>
<td>2.108e-37</td>
</tr>
</tbody>
</table>

As we pointed out earlier, this example is not a complete Bluetooth design. Hence, area overhead would be increased when applying the proposed I-IP to a complete SOC design. Also, if a SOC has multiple buses, it might be beneficial to embed more than one I-IP cores, one core for each bus. This is because in DSM designs, it is better to monitor bus signals locally than routing them to a centralized I-IP. Remember that the primary objective for the post-silicon verification is to identify those SI and DI issues unpredictable at pre-silicon phase.

**5. CONCLUSIONS**

A method was proposed to embed an FPGA core in a SOC as an infrastructure IP that can exploit transaction-based verification methodology to verify and debug the first silicon. The idea was illustrated from the system model down to the block-level hardware design. The objective of the proposed method was to aid post-silicon verification, validation and debug and hence reduce the time to market. Experimental results presented show that the proposed method can be implemented with low area overhead and acceptable possibility of error masking.

**6. REFERENCES**


---

**Figure 1. Block Diagram of Proposed I-IP**