

ENHANCING RELIABILITY AND FLEXIBILITY OF A SYSTEM-ON-CHIP USING RECONFIGURABLE LOGIC

Wei Jiang, Tushti Marwah, Don Bouldin
Electrical and Computer Engineering
University of Tennessee
Knoxville, TN, USA

Abstract

In this paper we present the design of a SoC baseline platform with a Leon2 CPU. An Advanced Encryption Standard (AES) module and a reconfigurable core form the IP blocks that are attached to the SoC through AMBA bus. The reconfigurable core is inserted into the design using tools developed by DAFCA, Inc. (Design Automation for Flexible Chip Architectures) for post-silicon debugging and verification. Hence, a re-spin may be avoided and the time-to-market will be reduced.

I. Introduction:

SoC is a major revolution in IC design where the whole functionality of a system is placed on a single chip. Its advantages include high performance, shorter design cycle time and space efficiency, whereas the challenges include deep sub micron design complexities, verification and integration. Presently, SoC's can have as many as several tens of million gates, multiple IP cores, and complex on-chip buses and protocols. The integration of all the components into a system and the verification of such a big design have become a very challenging job.

Most of today's virtual components (i.e. IP cores) don't have well-defined contents and interfaces; they are often fuzzy and more like "patches in a quilt, which have to be carefully stitched together" [1], hence, integrating existing IP blocks to form a larger system is not a simple task. Moreover, with the rapid shrinking of the feature size, more physical errors are bound to occur due to timing, crosstalk, noise, temperature, and process variation. At the same time, the designers are losing visibility into the design as the size of the design increases [2]. The internal pins are buried inside the chip and this makes it even harder to find errors in the design after the chip is fabricated. The whole debugging process requires multiple re-spins and delays the product's time-to-market by several months.

In this paper we present the design of a platform-based SoC with added reconfigurable logic blocks as IP blocks and in-silicon debugging logic to enhance the reliability and the

flexibility of the design. With the help of the debugging logic IP blocks, the designer can regain the visibility and the controllability of the complex SoC design. Thus, the number of re-spins and the product's time-to-market will be reduced.

II. Background

Platform-based design is becoming the method of choice for designing SoCs for embedded systems [3]. It has extensive planned design reuse, which enables designers to create a succession of derivative designs. In this approach, the main focus areas for the designer are interface standardization, virtual system design, and designing the system architecture and interface between the blocks. The basic idea behind this is to re-use significant portions of previous designs to reduce the time-to-market, which generally results in greater revenue for the product. Under this concept, the first goal is to develop a complete SoC that is central to its product line. Usually there is a processor, a real-time operating system, peripheral IP blocks, some memory and a bus structure. Once the baseline platform is fully functional, a derivative design in which only a few virtual components are added or dropped can be accomplished easily [4].

In this paper, we will discuss the verification and debugging of an SoC using reconfigurable logic blocks. Adding reconfigurable logic to the SoC also provides flexibility for changing functional parameters or protocols after fabrication. Thus, functions not envisioned in the original design can be implemented. The merits of adding reconfigurability to an ASIC chip are evident: traditional ASICs achieve their performance advantages with direct hardware implementation, however this advantage is sometime overshadowed by the fact that the silicon design is most often fixed. Traditional ASIC design is rapidly changing as increasing ASIC/SoC development costs are forcing designers to make silicon more flexible.[5] Designers have long used FPGAs to implement software algorithms in hardware to increase performance, as we can see so many systems have some kind of FPGA or programmable devices at the board-level of the design.

Although the concept of adding reconfigurable logic into traditional ASIC/SoC has been around for a while, reconfigurable systems have had only a minor impact to date. One of the reasons is that it needs software support and clear investment return to make it attractive. In our design we have explored a new trend in the SoC debugging and verification: using reconfigurable logic for in-silicon debugging and design verification. In our design, reconfigurable logic was inserted into the baseline design using tools recently developed by DAFCA, Inc. (Design Automation for Flexible Chip Architectures) for post-silicon debugging and verification. This logic enables the SoC to be verified after fabrication, and in some cases, errors can be fixed using the reconfigurable structures. Hence, a re-spin may be avoided and the time-to-market will be reduced.

III. Technical Approach

Our baseline platform design consists of a Leon2 processor[6] as the CPU, AMBA on-chip bus, an Advanced Encryption Standard (AES) module and a Reconfigurable module attached to the bus Leon2 as user IP blocks. The block diagram of our design is shown in Figure 1.

Since the main key components (e.g. CPU, Bus, AES) are open source and can be obtained at no charge and databases of commercial and non-commercial IP blocks are now online,[7][8] SoC developers can identify reusable virtual components to be integrated into their platforms. Although business and legal issues must be pursued individually for each component, the development of a derivative SoC design is certainly facilitated

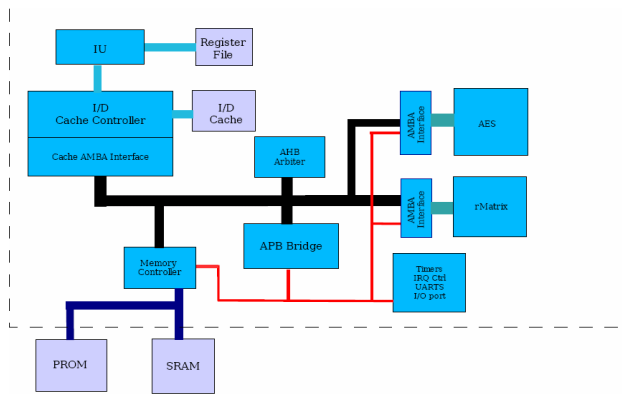


Figure 1. SoC Platform Block Diagram

A. SoC Design Components

1) Leon2 CPU

The Leon2 CPU is a 32-bit SPARC-V8 CPU that was developed by the European Space Agency. The source code of the CPU is written in VHDL and can be obtained on-line at no charge. The processor is highly configurable and particularly suitable for SoC Designs [6]. It is designed for embedded applications with the following features on-chip: separate instruction and data caches, hardware multiplier and divider, interrupt controller, debug support unit with

trace buffer, two 24-bit timers, two UARTs, power-down function, watchdog, 16-bit I/O port and a flexible memory controller.

2) AMBA bus

The Advanced Microprocessor Bus Architecture (AMBA) on-chip bus is used in our design. AMBA AHB (Advanced High-performance Bus) and APB (Advanced Peripheral Bus) are used for communication between the processor, memory and IP Blocks, which are attached to both buses. In our design they are implemented as masters on the AHB bus but slaves on the APB bus. The AHB arbiter decides which master will get control of the bus. The only master on the APB bus is the APB Bridge that converts the system bus transfers into APB transfers. It latches the address, decodes it and generates the peripheral select signal. We have used the APB bus for control signals of the IP blocks and the AHB bus for data transfer. The test bench that is used in our design performs a boot of the CPU and transfers data and instructions to and from the user IP blocks via the AMBA bus.

3) AES module

A 128-bit AES decryption module is used in our design as a user IP block attached to AMBA bus. The module was developed at the University of Tennessee as part of our cryptographic key protection research project. The design reads in 128 bits of encrypted or cipher text as well as the encryption key, performs decryption and writes out 128 bits of plain text. The module has been verified with simulation and tested on a Virtex II Pro FPGA.

4) Reconfigurable Block

In our design, a small reconfigurable block is also attached to the AMBA bus serving as a second IP block. Since one goal of our project is to debug the design and the AES module has been thoroughly verified, it is less likely that there will be a design error in our AES IP block once the chip is fabricated. We inserted a small embedded-FPGA using the DAFCA pre-silicon tools in order to give us the flexibility to load different designs into our platform. Given that the block is reconfigurable, we can load a design with intentional errors to test the debugging method.

5) Debugging Logic

Reconfigurable instrumentation is inserted into our design using the pre-silicon DAFCA tools; this will enable users to isolate and repair bugs, as well as accelerate verification. It provides at-speed access to internal signals on the chip, delivers instrumentation for trigger and capture events, and also aids in in-situ repairs and signal generation. The DAFCA solution has both software and in-silicon components. In silicon, DAFCA provides a set of instrumentation capabilities that users tailor to their specific debug requirements. The software environment enables the user to specify where the instrumentation is placed on the chip (pre-silicon) and provides the debug, configuration and analysis capabilities (post-silicon). DAFCA on-chip instruments use JTAG to connect to the debug software to eliminate the need for additional pin support. DAFCA

Reconfigurable Debug Infrastructure (ReDI) comes in two forms: customizable logic generated by the user with DAFCA tools and hardware library elements.

The customizable logic provides both wrapping and tapping capabilities. A wrapped port/signal can be observed or controlled by the debug instrument. The debug instrument introduces a mux delay in the signal path, while a tapped port/signal can only be observed with the debug instrument. The customizable logic consists of:

- rWRAP: A one-dimensional reconfigurable logic block array used for wrapping ports/signals.
- rMATRIX: A two-dimensional reconfigurable logic block array used for wrapping ports/signals.
- r1500: Reconfigurable wrapper cells compliant with IEEE Standard 1500.
- CMUX: A highly configurable MUX used for tapping ports/signals.
- CMUXB: A configurable single stage 2:1 MUX array.

The hardware library elements are comprised of:

- Primary Controller (PCON): It provides the interface between the JTAG TAP and the rest of the debug infrastructure.
- Serial Access Node (SAN): It provides an interface between the end-point and the instrumentation access channel.
- Monitor: It is a programmable “controller” and is used to manage assertions, triggers and tracer activity within the Debug module.
- Tracer: It also forms a part of the Debug module and is used for storage of state information during logic analysis.

B. SoC Design with DAFCA Design Flow

The design flow of our SoC is shown in Figure 2. The reconfigurable IP blocks are inserted at the Register Transfer Level (RTL), followed by logic synthesis and physical place and route.

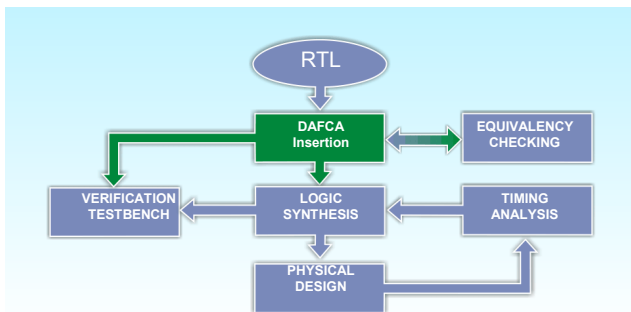


Figure 2. DAFCA Design Flow

1) RTL coding

Our SoC is designed at the Register Transfer Level. Most of the design is coded in VHDL. The Leon2 CPU is configured and slightly modified to suit our purpose. The instruction and data cache sizes in the system are minimized to save space in the chip. In order to integrate different IP blocks into the design, an AMBA bus interface is developed for each IP block such that it enables the cores to act as AHB bus masters and APB bus slaves. We have predefined the style of communication among the cores so that same interface can be used for other cores with minor modification. At this point, pre-layout simulation is done in ModelSim to verify the logic functionality of the design.

2) ReDI instrumentation

As described above, the ReDI instrumentation is inserted with DAFCA pre-silicon tools. We have wrapped most of the critical control circuits such as the control signals on the AMBA bus, as well as signals of the AMBA interface for the IP blocks because of the lack of test coverage. The user IP blocks are untouched since they are usually thoroughly verified, so there is less probability of error there. Besides the control signals, the data buses are tapped to give the designer the ability to observe internal data transfer after the design is fabricated.

A Monitor and a Tracer memory block are also inserted to perform more complex debugging tasks and to store the state information that can be viewed in the post-silicon debugging environment. All of the tapped/wrapped signals are connected to the Monitor/Tracer module through a daisy chain of configurable multiplexers (CMUX). A PCON and several SANs are used to form the interface to JTAG TAP and form the serial access channel, which is used to access and configure the rest of the debug infrastructure.

After the RTL instrumentation is done, simulation is also performed to make sure that the functionality of the design is not affected.

3) Logic Synthesis and Place and Route.

The design is then synthesized with the Synopsis Design Compiler targeting a TSMC 180-nm process. Artisan RAMs are used to implement the cache and register file in the Leon2. The synthesized net-list is placed and routed with the Cadence SoC Encounter tool. Post-layout simulation is done after the layout is generated using ModelSim.

4) Post-silicon Debugging

With the DAFCA instrumentation in place, we can perform at-speed in-system debug. The user can configure the instrumentation by identifying the signals to be monitored, and set internal traps or triggers. The ‘Personality Editor’ package of the tool enables us to program the reconfigurable wrappers to realize assertions, logic modifications and fixes. We can then run at-speed patterns through the system by executing system software. The internal state recorded by the Tracer in the debug module is then available for examination through the debug environment provided by NOVAS Debussy [9].

The user selects a net or set of nets to be observed, and the DAFCA tools automate routing the nets through an interstitial PAN network, programming triggers, and starting the tracer block. The result is that the SoC is no longer a black box. The user has the ability to debug the silicon at speed, in the system, using the real logic and regains visibility to the signals that had become inaccessible.

C. Area and Performance Overhead

One of the most important considerations in using reconfigurable logic is the overhead. The flexibility of the design does not come for free. Instead, it imposes overhead in the SoC design in terms of area and delay. The amount of reconfigurable logic introduced in a SoC design is always about the tradeoff between the flexibility and the area/delay of the design. The impacts in delay and area for adding different DAFCA IP blocks are shown in Table 1. The area overhead for each ReDI IP core depends on the complexity and functionality of the block. As in timing, during normal operation, the additional delay for a wrapper is the delay of the MUX and the extra wiring delay caused by the increase of the circuit size. It should be noted that tapping a signal does not introduce any MUX delay.

Table 1. Area and Performance Overhead

ReDI Block	rMUX	r1500	rWRAP	rMATRIX	rMonitor
Capability	Observe	Observe, Control	Observe, Control, Modify	Observe, Control, Modify	Observe, Control, Analyze
Mux Delay	No	Yes	Yes	Yes	No
Size (gates)	10-100 / signal	20-100 / signal	100-500 / Signal	500-2000 / signal	2K/chip

IV. Experiments and Results:

The RTL design has been successfully instrumented and synthesized targeting a TSMC 180-nm process. The SoC platform, which has several components and reconfigurable blocks, has about 1.3M transistors.

We have reduced the size of the original design to the minimum to save design space. In the meanwhile, we have been very generous adding in ReDI instrumentation to explore the strength of the reconfigurable logic in silicon debugging. The transistor count for the original design was 850K, which is about 2/3 of the final design size.

The physical layout was obtained with Cadence place and route tools. Several floorplans were tried in order to minimize the area and meet the timing constraints. The final floorplan is shown in Figure 3.

Simulations are done after each stage: RTL development, DAFCA instrumentation insertion, synthesis, and physical

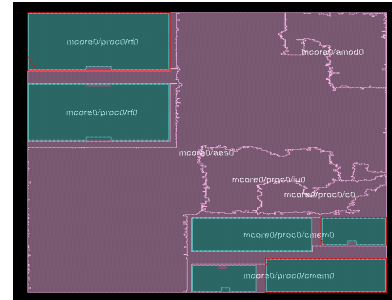


Figure 3. SoC Design Floorplan

place and route. DAFCA post-silicon tools are used to generate the ReDI block configurations and to simulate the functionality of the reconfigurable logic. The configuration bits are generated and loaded into the design through a JTAG interface. Then the instrumented design is simulated in NCSIM by Cadence. We plan to send the design for fabrication on a TSMC 180-nm process available via MOSIS. A testing board is being designed in order to test the chip after the design is fabricated.

V. Conclusion

In this paper, we discussed a new approach for modern SoC development. Compared to the traditional SoC design, we have proposed to use platform-based SoC with reconfigurable debugging logic. The DAFCA design flow was discussed to increase the flexibility of the chip and to achieve in-silicon, at-speed debugging. We believe that this approach will greatly help the SoC designers debug their designs and hence improve design reliability by reducing the number of re-spins and the product's time-to-market.

REFERENCES

- [1] Don Bouldin, "Platform-Based System-on-Chip Design", *Proceedings of 2003 Microelectronic Systems Education Conference (MSE)*, Anaheim, CA, pp. 48-49, June 1-2, 2003.
- [2] DAFCA Inc., "In-Silicon Solutions for Silicon Debug", DAFCA whitepaper.
- [3] David Fritz, "Why platform-based design works better than a discrete IP approach", <http://www.us.design-reuse.com/>
- [4] Srivastava, R., "Development of An Open Core System-on-Chip Platform", M.S. Thesis, University of Tennessee, August 2004.
- [5] Cary Synder, Steve Leibson, "Point/Counterpoint: Configurable Vs. Reconfigurable", <http://www.semiview.com/>
- [6] <http://www.gaisler.com/>
- [7] <http://www.opencores.org/>
- [8] <http://www.design-reuse.com/>
- [9] <http://www.novas.com/>