ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Multi-Step Verification Environment for a Chip Design using SoC platform

Je-Hoon Lee and Duk-Gyu Lim

Div of Electronics, Infnormation and Communication Eng., Kangwon National University, Samcheok Campus, 1 Joongang-ro, Samcheok, Gangwon-do, 245-711, Rep. Of Korea

limdg@kangwon.ac.kr

Abstract

This paper presented an efficient verification strategy for the platform based design. A goal of the verification task is to detect all design faults and provide with full verification coverage at the earlier design. The proposed verification strategy employed iterative verification stages. For a case study, this strategy was used in a verification of a modem chip design complying with IEEE 802.11a standard. It was successfully verified the entire design functionality and its interface with 100% coverage in shorter design cycles.

Keyword : SoC (system on chip), verification, platform-based SoC.

Introduction

Benefits of SoC solutions are reduced size, low cost, lower power consumption and increased performance. However, design complexity is drastically increased. Recent technological advancements yield an integrating numerous functions into a single chip [1-2]. The number of IPs in a single chip continuously increases making complexity a major design problem.

The platform based design methodology is widely adopted for designers to over-come this design complexity. Thus, the designers use predefined architectures and IPs to reduce design time and complexity [1-3]. It, however, requires a proper simulation and verification environment. Designers usually develop a custom IP first. In order to make a full system, they add the custom IP into a pre-defined architecture with some standard IPs. In this case, designers must verify the custom IP at first, and then the IP must be verified within the entire system. This verification task must consider several issues such as verification coverage, time-to-market and so on. Different verification coverage metrics are defined to assess the design adequately such as function cover-age, statement coverage, branch coverage, interface coverage. None are sufficient to prove a design works, but all are helpful in pointing out areas of the HDL not yet tested.

Nowadays time-to-market issue presses a product developing time. The verifica-tion task, however, occupies 30-70% of whole product developing time [4-7]. It is very important to carry out the verification task in shorter design cycles.

This paper deals with these issues and presents an efficient verification strategy for the platform based SoC design and illustrates an experimental design complying with IEEE 802.11a WLAN standard that was verified by this strategy. The paper organized as follows. Chapter 2 gives a short description of the experimental design of IEEE 802.11a baseband processor. Chapter 3 introduces proposed verification strategy in detail. Chapter 4 illustrates verification results and discussions. Finally, Chapter 5 gives a conclusion.

Verification Strategy

The verification task is to detect and eliminate all design faults as earlier as possible. In order to detect all design faults, the verification task should give full coverage. It includes function coverage, statement coverage, branch coverage and interface coverage. The function coverage checks every function. The statement coverage checks each line of HDL. The branch and interface coverage confirm each direction of every branch and interfacing the blocks in the entire system. The verification tools need a perfect verification environment. It takes all possibilities that must be verified. In addition, the verification environment concerns the verification time. The verification time is categorized by two sub tasks fault detection and fault elimination.

This paper presents an efficient verification strategy for the platform based SoC design and illustrates an experimental design complying with IEEE 802.11a WLAN standard.

ISSN: 2277-9655
Impact Factor: 1.852

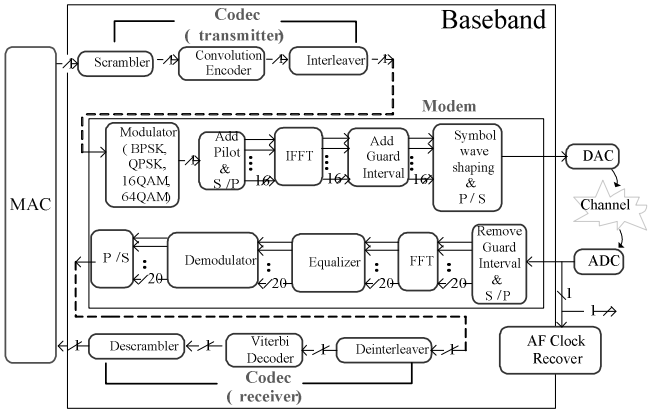

Figure 1. IEEE 802.11a WLAN baseband processor

The IEEE 802.11a is a standard for the baseband processor of a WLAN communication system at 5GHz [8]. It employs an orthogonal frequency-division multiplexing (OFDM) with 52-sub carriers that gives the maximum data rate of 54 Mbit/s. The data rate depends on a radio channel such as 48, 36, 24, 18, 12, 9 then 6 Mbit/s. The basic principle of OFDM is to divide a high-rate data streams to multiple sub carriers so that they transmit data in parallel over multiplexed orthogonal sub carriers.

The experimental baseband processor complying with 802.11a standard has major blocks of transmitter, receiver's codec, modem, interfaces with Medium Access Control (MAC) layer and the front-end RF circuit as shown in Figure 1. The transmitter comprises a codec, a modulator and IFFT blocks. The receiver includes a synchronization block, a FFT, an equalizer, a demodulator and a Viterbi decoder in the codec. We apply the HW/SW co-design technique to design the baseband processor. The MAC functions is implemented with C/C+. The target SoC chip includes the baseband IP, an ARM processor for the MAC and an AMBA bus system.

The proposed verification strategy is shown in Figure 2. We verify a design in 3 steps simulation, emulation and firmware level implementation using a SoC platform. At the first step, we verify a circuit at simulation level with test benches that confirms system behavior [4]. It takes much time to detect the faults even though it is more suitable to eliminate detected faults. At the second step, the circuit is verified on a HW emulator. The HW emulator is suitable for fault detection because of its high speed and flexible test benches with C/C+. At the third level, we employ a platform based verification based on a HW/SW co-emulator.

In the proposed verification flow, the circuit level simulation at step 1 eliminates major faults. A HDL simulator checks simulation waveform of the design as shown in Figure 2a. We use an FPGA based emulator at the step 2. The verification is performed on the emulator through various flexible test benches in C/C++. The emulator dumps data to sample signals and to store them in the internal memory of the emulator as shown in Figure 2b. The fault is detected by checking the output of the emulator or waveforms dumped in. The data in the memory are sent to the host computer after the emulation is done. It provides electronic waveform for a VCD format. In the verification system, the dumping step is optional that is only activated when it is requested.
For example, the dumping process is selected if the emulation output is not enough to detect a fault. This flexible solution provides high verification productivity. However, these two steps detect and fix bugs of a function block. It is not sufficient to detect bugs of an interface and the whole system. Step 3 detects faults associated with a hardware/software interface and a system integration on a SoC platform as shown in Figure 2c.

The baseband processor is synthesized and mapped into a HW emulator iProve from Dynalith [9]. Figure 3 shows the verification environment. The MAC and testbench are implemented in C/C++ on a host PC. The host PC and the HW emulator is connected through a PCI bus. A specific API (Application Programming Interface) layer in the iProve makes transactions between a C/C++ model and a Modelsim model for the baseband processor. Such configuration provides HW/SW co-development and co-verification including a source-level debugging through a C/C++ debugger and a waveform-based hardware debugging through a FPGA dumping process.

The platform based verification makes possible to verify the hardware/software interface and system integration on a platform. It removes a remaining fault before the system integration. Figure 4 shows the platform based verification environment. The platform consists of the Probase [10] and the iProve. The Probase contains an ARM core module, an AMBA bus, memories, and some peripherals, while the iProve accommodates the baseband processor. The HW/SW co-development and co-verification are possible because it includes a source-level debugging with In Circuit Emulation (ICE) and waveform-based hardware debugging through the iProve.

ISSN: 2277-9655
Impact Factor: 1.852

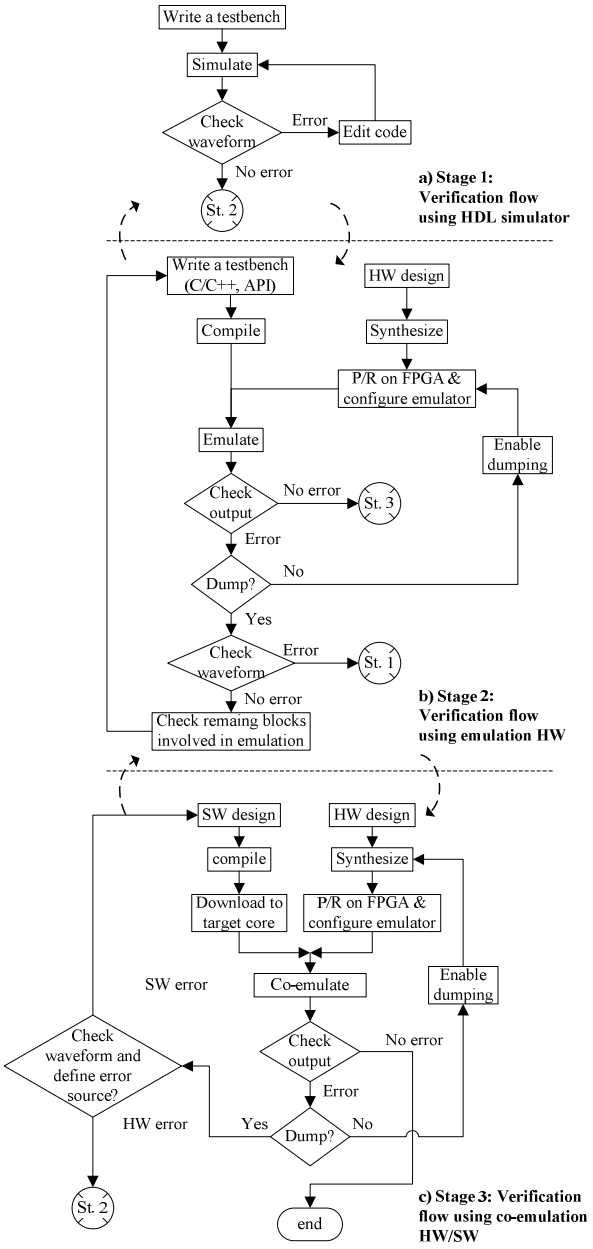

Figure 1. Three staged verification strategy

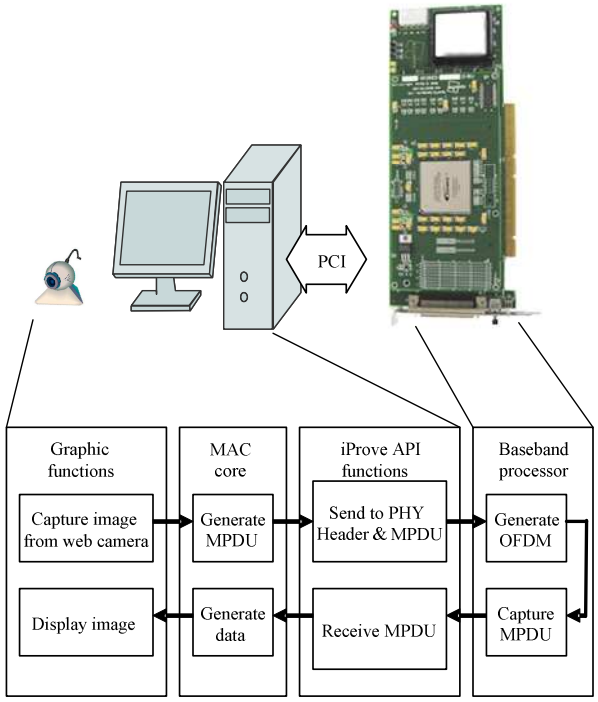

Figure 3. Emulation based verification

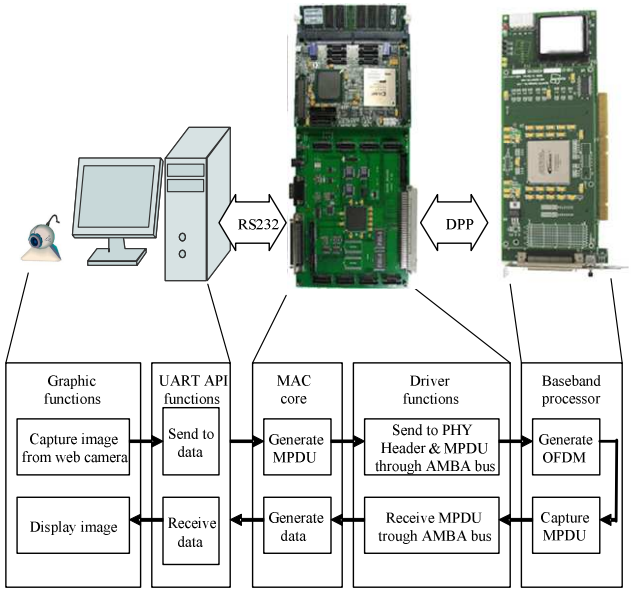

Figure 4. Platform based verification

Verification Results

This chapter summarizes the verification results. Figure 5 shows the comparison result of different verification approaches used in the proposed verification environment in term of verification speed. The simulation based verification (SBV) shows the lowest speed. The emulation based verification (EBV) has the highest speed. The platform based verification (PBV)

ISSN: 2277-9655
Impact Factor: 1.852

shows lower verification performance than EBV because PBV has two kinds of the SoC platform. It is a prototype of a SoC including an ARM processor, an AMBA bus and the baseband processor. The transmission speed is limited by the physical connection between these boards.
Figure 6 shows verification coverage of each verification environment. The SBV has about 50% coverage accounting function and statement coverage. Note that we assume that each verification metrics has 25% of the full verification coverage, and then the SBV's verification coverage would be 50% of the full verification coverage. The EBV is able to fulfill the function coverage, statement coverage and branch coverage because its simulation speed is very high. The PBV shows reasonable emulation speed then it can report all metrics including interface coverage.

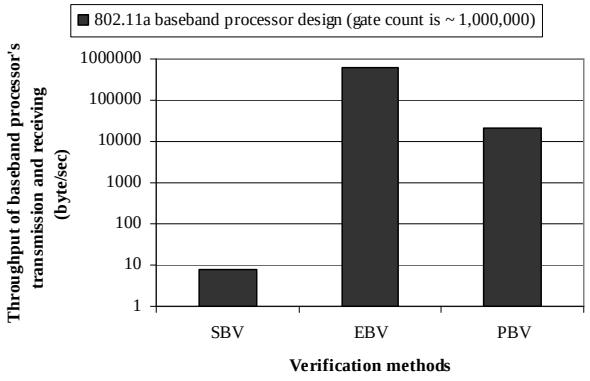

Figure 5. Throughput of a baseband processor's transmitter and receiver in different verification approaches

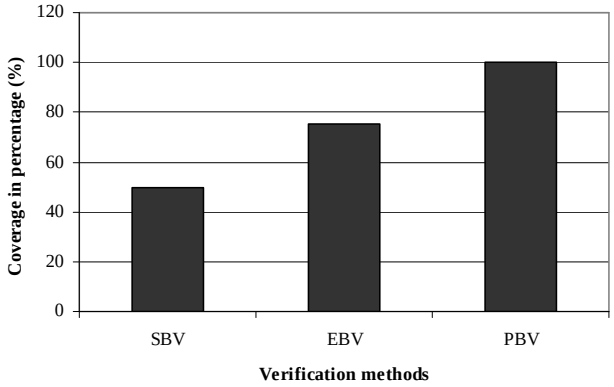

Figure 6. Verification coverage of each verification method

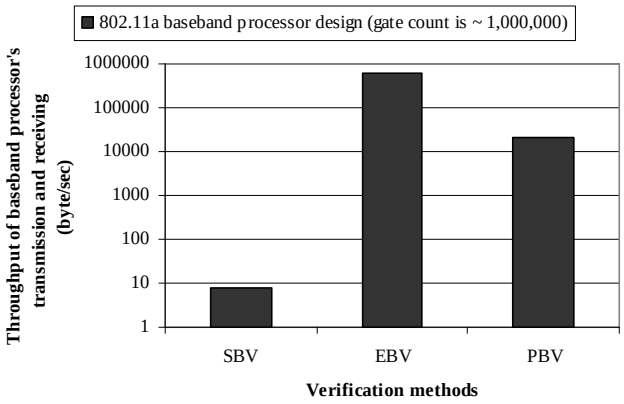

Figure 7. Setup time to start verification

Figure 7 shows the setup time to start verification for each verification environment. The SBV needs 5 minutes to initiate new verification. But EBV, PBV take much longer time to initiate a new verification because it starts from synthesis of HDL source codes to FPGA download.

We assume that the verification time can be classified by two sub tasks fault detection and fault elimination. Figure 8 shows the time breakdown of the fault detection and elimination of the verification task. In the proposed verification strategy, the fault elimination time is the sum of all simulations. The fault detection is from all the verification steps. But most fault detection is achieved by the EBV. According to this verification approach, the total verification time reduces.

ISSN: 2277-9655
Impact Factor: 1.852

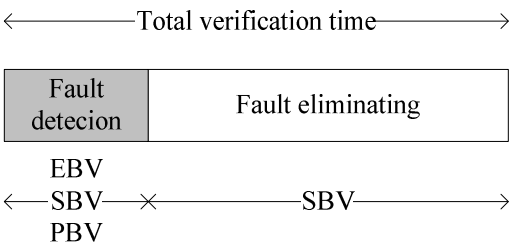

Figure 8. Total verification time

Conclusion

This paper presents an efficient verification environment for the platform based SoC design. The goal of the verification task is to detect all design faults and to eliminate with full verification coverage as earlier design. The proposed verification strategy employs several iterative steps verifying a design. For a case study, we designed an IEEE 802.11a baseband processor as a SoC with an ARM and an AMBA platform. The design verification covers whole functionality and its interface. The proposed environment has an iProve is used as an emulator hardware carrying 6 million gates FPGA and a SoC platform Probase has an ARM9 processor and a million gates of FPGA. The simulation results show 100,000 times higher verification speed compared to a conventional RTL level simulation for emulating an IEEE 802.11a baseband processor.

References

[1] *S. Sarkar, S. Chandar and S. Shinde, "Effective IP reuse for high quality SoC design," Proc. of ISOCC 2005, pp. 217-224, Sept. 2005.*

[2] *A. S. Vincentelli, G. Martin, "Platform-Based Design and Software Design Methodology for Embedded Systems," IEEE Design & Test of Computers, pp. 23-33, Dec. 2001*

[3] *M. Keating and P. Bricaud, "Reuse Methodology Manual," Kluwer Academic Publishers, 1998*

[4] *K. Wakabayashi, T. Okamoto, "C-based SoC design flow and EDA tools: an ASIC and system vendor perspective," IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 19, No. 12, pp. 1507-1522, Dec. 2000.*

[5] *R. Jindal, K. Jain, "Verification of Transaction-Level SystemC models using RTL Testbenches," Proc. of MEMOCODE 2003, pp. 199-203, Jun. 2003*

[6] *C. Y. Wang, S. W. Tung, and J. Y. Jou, "An Automorphic Approach to Verification Pattern Generation for SoC Design Verification Using Port-Order Fault Model," IEEE Transactions On Computer-Aided Design Of Integrated Circuits And System, Vol. 21, No. 10, pp. 1225-1232, Oct. 2002*

[7] *M. Cupak, F. Catthoor, and H. J. De Man, "Efficient System-Level Functional Verification Methodology for Multimedia Applications," IEEE Design & Test of Computers, Vol. 20, No. 2, pp. 55-64, Apr. 2003*

[8] *ISO/IEC, Wireless LAN MAC and PHY Specifications —High-Speed Physical Layer in the 5 GHz Band, ISO/IEC 8802-11:1999(E)/Amd 1:2000(E), New York IEEE, 2000.*

[9] *http://www.dynalith.com/ipr, Jan. 200*

[10] *C. M. Kyung, "ARM and FPGA based Co-Emulator for SoC Design and Verification," Center for SoC Design Technology at KAIST, Apr. 2005*