Flexible Multi-IP Verification Methodology Based on an FPGA Platform

Jin Woo Song[1] and Ki-Seok Chung[2]
Information & Communication Engineering, Hanyang University ,Korea
IT/BT #715, 17, HaengDang-Dong, SeongDong-Gu
133-791, Seoul, Korea(Republic of)
E-Mail: [1]jinwoo02@gmail.com, [2]kchung@hanyang.ac.kr

Abstract: It is well-known that in ASIC designs, verification is more difficult and time consuming than design itself. As the number of IPs in an SoC design increases, verifying multiple IPs together is really important to reduce time-to-market. In this paper, we propose a novel FPGA platform based verification methodology which tests multiple IPs together using a single testbench. We've found that commercially available FPGA platforms such as Altera Cyclone, Xilinx Spartan provide excellent environment in verifying the functionalities of mutually interactive multiple IPs. In our methodology, an FPGA device is used mainly for verification purposes. We program the soft core CPU, the bus architecture and other peripherals into the FPGA, which will execute C-based testbench and mutually interactive IPs are also programmed into the FPGA device. We implement a set of tools which consists of a communication interface and a wrapper generator which will automatically connect the bus architecture and the IP module together. Using this platform, we have verified up to 5 IPs together successfully, but we can verify more IPs together easily. Time and effort to verify complex IPs have been significantly reduced using this methodology.

1. Introduction

Nowadays, IP-based design methodology which integrates pre-built SW and HW IPs to build a system is widely accepted because the complexity of the system is growing very fast. Using this methodology, the design time has been significantly reduced, but the verification becomes the real bottleneck. It is well-known that the cost of verification typically accounts for 50%-80% of the total design cost. For fast verification process, hardware accelerated simulations or emulation techniques are commonly used rather than full simulation which is often unacceptably slow. Therefore, co-simulation and co-verification techniques gained more attention to verify complex IPs. The advantages of co-simulation will be ease of use, powerful debugging capability, flexibility and fast speed. However, many of these approaches require having very expensive devices and tools. And also in-circuit emulators are often connected to the development host using slow external busses, so that real time communication and monitoring between development host and target board is not suitable. Furthermore, it is almost impossible for in-circuit emulators to test such IPs which have the sizes of multi-million-gates.

In this paper, we propose an efficient verification methodology based on an FPGA platform which overcomes the aforementioned weaknesses of existing approaches. Our methodology has several advantages over the existing methods. First, it does not require any expensive equipment for verification. Only commercially available FPGA devices such as Altera Cyclone or Xilinx Spartan with some software tools are needed.[1] Second, it provides a uniform environment for co-design and co-verification. Third, it is a standard C-based test methodology which eliminates the need to learn new languages or new tools. Lastly it enables us to verify mutually interactive multiple IPs together under various execution scenarios. Also, automatically generated wrapper and testbench are readable and modifiable so that the designers can modify the generated wrapper and the testbench to suit their further needs.

For an automated verification process, we implemented a set of tools. [2] First, to formalize communication between an embedded processor core and the DUT (Device Under Test), the communication interfaces based on Wishbone Rev B.3 protocols are implemented. [3] Second, a wrapper generator which helps the DUT communicate through the on-chip bus is designed. The wrapper generator accepts input and output specifications from IP designers using a GUI (Graphical User Interface). The specification of IPs is converted into a verilog wrapper which enables communication between the processor core running C testbench and the DUT.

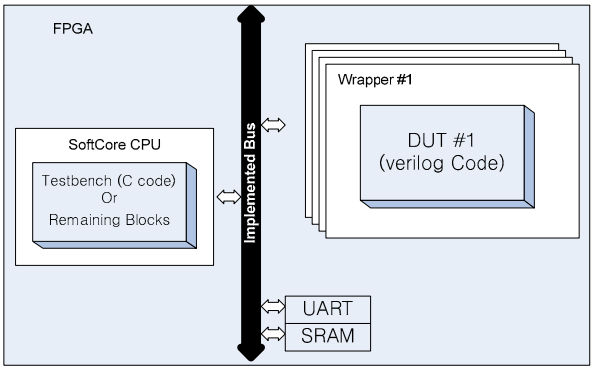

Fig.1 Proposed platform structure for Multi IP Verification

2. Motivation

Most of existing verification approaches have tried to reduce time and effort for verification. Our tool is similar to the existing works in spirit, but we claim that our methodology is much simpler and far more cost-effective. Our methodology does not require any expensive test/emulation board or tools. Every component except development board is publicly available for free. Also our method is a simple and [4] fast transaction-level verification technique using a standard C testbench. Therefore, it is very easy to learn and use our verification system. Basically, what we want to do are to synthesize a prototype of the DUT with our pre-implemented embedded processor core and to program into an FPGA device. The DUT is verified by a C testbench which is running on the processor core. Since the architecture has a processor core on an FPGA device with key interfaces for design and verification, we do not need any other expensive device for verification. Also using OpenRISC core as a processor core for verification has an additional advantage since we have the full RTL source of processor architecture which can be modified to suit our needs. So we can say the architecture is flexible. Therefore, it is very easy to verify multiple IPs together under various scenarios like multi-core based platform. Since we use a C testbench, it is very easy to write testbenches, and easy to conduct testing with many different parameters.

3. Flexible Multi-IP verification platform

3.1 Our approach

Figure 1 shows our verification platform which consists of a processor core and other peripherals programmed on FPGA. This is a typical structure of an SoC platform. To minimize the cost of implementing our verification environment, we employ an open source processor core and bus architecture such as OpenRISC and Wishbone Bus. This core and the bus architecture will be programmed into FPGA, and the IPs under test will be attached to the bus automatically by our tools. Other software IPs including testbenches will be executed by the processor core that we programmed in FPGA. Our SoC platform based verification provides a very flexible testing environment since the boundary between the SW part and HW part is not fixed. Hence, almost the same testing environment can be used regardless of whether we test a single IP or multiple IPs. Only a more complicated test scenario written in C code would be used when we test multiple IPs together. This is very useful when we start from (initially) a 100% SW implementation and gradually move some of the performance-critical blocks into HW implementations. The communication interface and the wrapper will be automatically generated if the designer provides the input/output specification of the designs under test.

Since our proposed platform only suggests that a single process core will be used to test any other components programmed in FPGA together including any other core(s), we can use our platform to test multi-IPs including multi-core designs. Our methodology can be used with any of commonly used SoC platforms as long as the on-chip bus functional model is properly implemented in the tool. In this paper, we implement our tools to automate the testing process for an [5] OpenRISC SoC platform. The platform employs OpenRISC 1200 core and the WishBone bus architecture Rev. B.3 which are all modifiable. Since they are all modifiable designs we can flexibly transform our platform in several ways. In the following subsections, we will explain how we implement our tools and how we verify multiple IPs using the Xilinx Spartan Platform

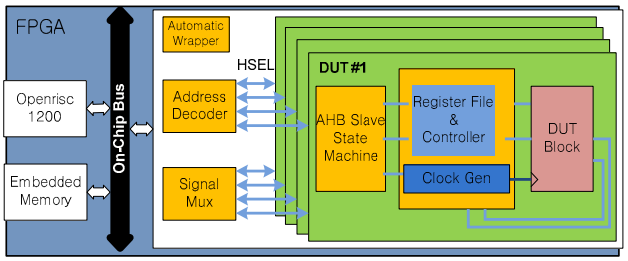

Fig. 2 Block Diagram for Multi-IP Verification

3.2 Architecture for Multi-IP Verification

Our proposed verification platform does not require any expensive hardware equipment other than an FPGA device itself. How we use the device to verify multiple IPs is shown in Figure 2. Multiple IPs programmed in FPGA will be surrounded by automatically generated wrappers, and will be communicated through an on-chip bus to the embedded processor core. The processor core will execute a C testbench and the core will optionally execute the SW implementation if some part of the system is implemented in SW. The on-chip bus is in general much faster that the off-chip bus used to connect an external separate emulation board. Therefore, a very fast and low cost test can be achieved. Our proposed architecture is based on the OpenRISC core and the Wishbone bus structure. Thus, we have designed a protocol interface which enables the DUT connected to the Wishbone bus to communicate with the embedded processor core.

3.3 Formalization of Bus Functional Model

The OpenRISC that we use is fully compatible with Wishbone Rev B.3 bus architecture. And we also have designed a bridge that translates the Wishbone bus signals to widely used AMBA AHB bus signals. So the designs under test which have already been designed compatible with the Wishbone or AMBA bus interface can be tested without any wrapper. If not, our tool will generate a wrapper which will help general IPs communicate with the bus interface [3]

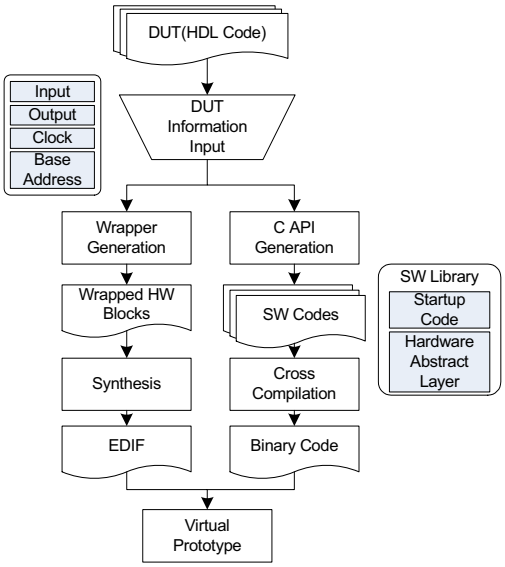

Fig.3 Wrapper Generation and Verification Flow

3.4 Implementation of Wrapper Generator

[3] We have implemented a tool that generates the wrapper automatically to test multiple IPs on an SoC platform. Figure 3 shows our wrapper generation flow. Our wrapper generation tool is written in Visual C++, and the information of the external interface of the DUT is provided by the IP designer using a GUI. The information of the external interface of the DUT will be (i) the number of blocks, (ii) input/output specification of each block (iii) the information on the clock signal, and (iv) the initial address of each block.

From this information, a wrapper is automatically generated by our tool. The wrapper will make the DUT(s) communicate with the testbench through the on-chip bus architecture. We currently have the maximum number of five DUT blocks to be tested simultaneously but we can test more IPs by extending our tool in a straightforward manner. A group of C APIs is generated and it makes testbench to access the DUT block. There are two different versions of C API groups. One is for firmware level and one is for device driver level under Linux OS kernel. When we test IPs under a Linux OS environment, applications can access physical address of IPs using “mmap()” function.

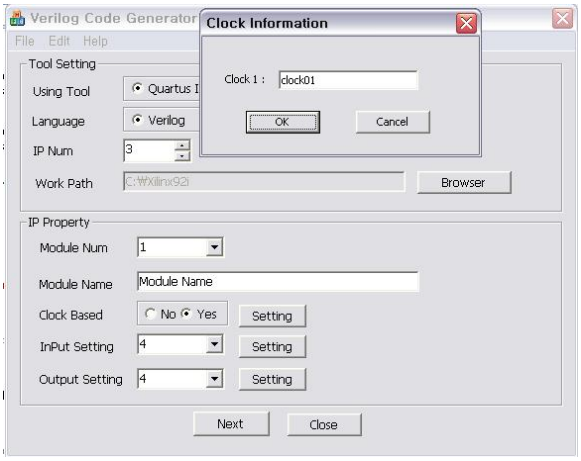

Fig. 4 User Interface of Wrapper Generation Tool

4. Case Study

4.1 Verification Environment

The platform for verification that we propose consists of on open core processor called OpenRISC and some peripherals such as UART, SRAM, PLL, and JTAG. We use [6] Spartan-3 FPGA (XC3S4000) with 4M gate and we let the OpenRISC core running at clock rate of 50Mhz. [7] For cross-compilation, the GNU toolchain for OpenRISC-32bit is used.

4.2 Test environment

To verify that our platform and tools can effectively test the functionality of various IPs, we have conducted testing of a 32-point FFT(Fast Fourier Transform), a 8*8 JPEG Encoder and a 8*8 JPEG Decoder.

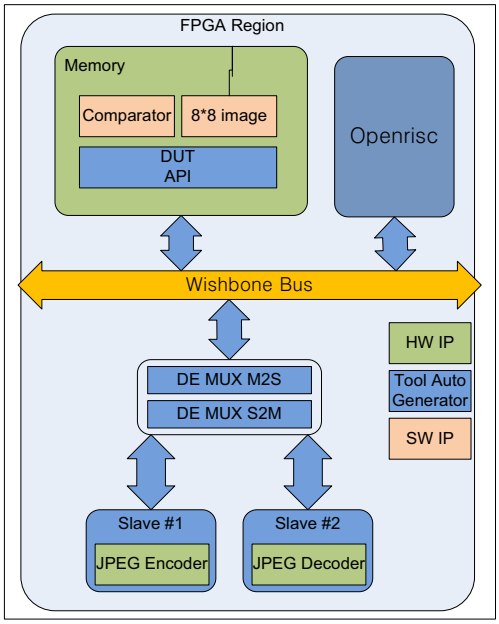

Fig. 5 Block Diagram of Verification Environment

Figure 5 shows a block diagram of the overall system that we implement. An 8*8 JPEG Encoder and an 8*8 JPEG Decoder are written in Verilog. A comparator is written in C as a SW IP. The verilog codes of hardware IPs and a wrapper by using wrapper generation tool are compiled and programmed in FPGA. Then we carry out a test which consists of the following steps (1) an 8*8 image goes into the JPEG encoder and output is generated. (2) the output goes into the JPEG decoder to get the corresponding decoded 8*8 image. (3) The comparator compares the original image and the decoded one. This is a typical scenario of testing an encoder and a decoder together.[8]

4.3 Experimental result

We have measured the speedup of writing of C testbenches and our automatic wrapper generation when compared with writing HDL testbenches and writing a wrapper manually. [9] After we explain the detailed functionalities of our DUTs, we ask four designers to write testbenches and wrapper codes manually. And then the same set of testbench and a wrapper file have been generated by our tool. It took for experienced designers to write testbench and wrapper files in 6.3 hours on average. But our tool generates the files within two minutes. Also, we could modify testbench file easily to test the DUT under various different scenarios. Adding and removing certain DUT can be done easily and various probing and monitoring have been done easily as well. All of these are possible since the testbench is written in the standard C language.

Table 1 :
Comparison of Verification Speed Efficiency of Wrapper Generation

Module	Design Time(hour)		Verification Time	
	Manual	Auto	HDL simulation	Proposed platform
32-point FFT	5	0.1	105 Sec	5 Sec
JPEG Encoder	7	0.1	120 sec	2 Sec
JPEG Decoder	6	0.1	130Sec	6 Sec
JPEG Encoder& Decoder (Multi-IP)	7.5	0.1	170 Sec	9 Sec

Table 1 shows the comparison results on the simulation time. We have tested various IPs by either ModelSim(HDL simulator) or our method.

Table 2 : Comparison of Verification Time

Module	Verification Total Time (Sec)					
	HDL Simulator			Proposed Platform		
	V_{Setup}	V_{Ing}	V_{Result}	V_{Setup}	V_{Ing}	V_{Result}
32-point FFT	181	4	912	165	1	37
JPEG Encoder& Decoder (Multi IP)	320	47	1558	160	9	129

In all tests that we carried out, the testing time of our method is much shorter than conventional HDL simulations. As you can see from the table, the more complicated the IP becomes, the bigger speedup we can achieve. Table 2 shows that the verification using our platform is at least 5.5 times faster that the HDL simulation This is not only because our method is an emulation based

verification, but also because using C testbench is much easier for writing and testing than using HDL testbench. Also, the correctness of the output results can be more easily checked by using the reference model in the fixed point format. To compare the verification time quantitatively, the following terms are defined.

$$V_{Total} = V_{Setup} + V_{Ing} + V_{Result}$$

where V_{Total} is the total verification time and V_{Setup} is a setup time which includes testbench writing and test environment setting. V_{Ing} is the actual simulation (or emulation) time, and V_{Result} is the time that takes to get the results from the testbench. [3]

5. Conclusion

In this paper we propose a low cost, yet very effective IP verification methodology. Our method is very effective since we only use a commercially available FPGA platform without having to add any expensive hardware. A couple of SW tools are written to generate the communication interface and a wrapper automatically. Our method relies on a soft core processor, standard C testbenches and a wrapper in Verilog. Therefore, the designers can modify the test structure, the testbench, and wrappers easily if the automatically generated or given files are not sufficiently satisfactory. Compared with other existing approaches, we claim that our methodology enables extremely low cost, fast verification with ample flexibility. Since the testbench and the DUT are communicating through a fast on-chip bus (Wishbone bus), real time monitoring and testing are also possible. In the near future, using our methodology, we are planning to test a very complicated IP like a multi-core design which contains multiple open core processors.

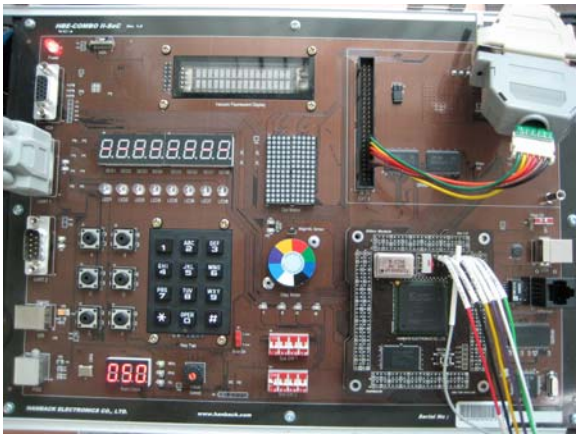

Fig 6 Evaluation Board for our Verification Platform

Acknowledgements

This work was supported by "System IC 2010" project of Korea Ministry of Knowledge Economy. This research was also supported by “Seoul R&D Program” in 2008". This work was also partially supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government(MOST) (No. R01-2007-000-20891-0) .

References

[1] B. Black and J shen, “Calibration of Microprocessor Performance Models” , Computer, V31, N5, pp.59-65, May 1998.

[2] Richard Herveille, OpenCores Organization， *Specification for the:WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores*
Revision: B.3, Released: September 7, 2002

[3]Joo-Yul Park, Ki-Seok Chung, Multi IP Verification Methodology Using SoC Platform : November 9, 2007

[4] Xu S., et al. “A TLM platform for system-on-chip simulation

[5] OpenRISC 1000 Architecture Manual1 November 4, 2005

[6] http://xilinx.com

[7] Using the GNU Compiler Collection
by Richard M. Stallman and the GCC Developer Community
Last updated 23 May 2004

[8] F.Balarin, et al. “Hardware-Software codesign methodology for DSP applications”, Design & Test of Computers, pp. 16-28, 1993.

[9] W.Wolf “A decade of hardware/software codesign”, Computer Volume 36,pp. 38-43, 2003.

[10] Rui Wang, “Reuse issues in SoC verification platform”, Computer Supported Cooperative Work in Design Conference, pp.685-688, 2004.
and verification”, VLSI-TSA-DAT symposium pp.220-221, 2005.

[11] Sang-Heon Lee, et al. “SoC Design Environment with Automated Configurable Bus Generation for Rapid Prototyping” ASIC/ASICCON Conference, pp.41-45, 2005.

[12] http://www.altera.com

[13] http://www.opencores.org

[14] http://mentor.com

[15] http://www.cadence.com

[16] http://www.synopsys.com