I. J. Communications, Network and System Sciences, 2008, 2, 105-206
Published Online May 2008 in SciRes (http://www.SRPublishing.org/journal/ijcns/).

A Co-verification Method Based on TWCNP-OS for Two-way Cable Network SOC

Chong LI[1], Xiaotong ZHANG, Yadong WAN, Qin WANG
Department of Computer Science and Technology
University of Science and Technology Beijing, Beijing, P.R.China
E-mail: [1]lichong0564@163.com

Abstract

Co-verification is the key step of software and hardware codesign on SOC. This paper presents a hw/sw co-verification methodology based on TWCNP-OS, a Linux-based operating system designed for FPGA-based platform of two-way cable network (TWCNP) SOC. By implementing HAL (hardware Abstraction level) specially, which is the communications interface between hardware and software, we offer a homogeneous Linux interface for both software and hardware processes. Hardware processes inherit the same level of service from kernel, as typical Linux software processes by HAL. The familiar and language independent Linux kernel interface facilitates easy design reuse and rapid application development. The hw/sw Architecture of TWCNP and design flow of TWCNP-OS are presented on detail. A software and hardware co-verification method using TWCNP-OS is proposed, through the integrated using of Godson-I test board and TWCNP, which realizes the combination of design and verification. It is not a replacement of the co-verification with generic RTOS modeling, but is complementary to them. Performance analysis of our current implementation and our experience with developing this system based on TWCNP-OS will be presented. Most importantly, since the introduction of TWCNP-OS to our FPGA-based platform, we have observed increased productivity among high-level application developers who have little experience in FPGA application design.

Keywords: CM, SOPC, MIPS, Co-verification, FPGA, HAL

1. Introduction

Now CATV mainly transmits via analog signals in some regions and countries. In order to realize A/D transmission, two-way network chip based on HFC (Hybrid Fiber-Coax) is becoming the core technology and key equipment in interactive digital television and other integrated digital services. SOC enables the realization of all these functions in one chip and has reusable technology of IP module to develop the SOC which has the independent intellectual property has important meaning.

The hardware and software codesign is commonly used in SOC design. It is different from traditional design methods that simulation and verification is used to find and rectify mistakes in time and optimized the system in the end. With the complexity increasing in SOC design, scientific design method and high efficient verification are growing more important [1]. Hw/sw co-verification does not just occur at the system integration point but rather throughout the design process. A high efficient verification platform should provide simple and scientific verification environment to ensure the performance of whole system. Co-verification includes hardware verification, software verification and software-hardware interface verification. Using concurrent design and co-verification to shorten project period and combine hardware behavior model with software running environment, construct a high efficient co-verification platform has been the research hotspot recently.

While traditional hw/sw codesign researches have produced encouraging results in the area of hw/sw partitioning, cosimulate, cosynthesis, and co-verification, most of them rely on self-contained design environments that are based on their specific input languages or library API's [2]. As a result, migrating existing software

designs to two-way cable network platform using these traditional hw/sw codesign methodologies would have incurred major re-engineering efforts, including learning a new language and API, getting familiar with a new design environment and reimplementing existing designs in the new language environment.

So, an easy to use hw/sw interface that allows rapid application development and migration should be (1) familiar and intuitive to both software and hardware engineers; and (2) language independent. We achieve this goal by setting hw/sw boundary at the embedded operating system kernel level.

Embedded operating systems (EOS) have become one of the most important components of embedded systems due to the growing complexity of the system functionalities as well as the increasing time-to-market pressures. With this trend, a demand for fast cosimulation of hardware and embedded software including an EOS is also becoming stronger in order to validate the functionality of the overall systems. With this trend, a demand for fast cosimulation of hardware and embedded software including an EOS is also becoming stronger in order to validate the functionality of the overall systems.

In this paper, we present TWCNP-OS, an operating system designed specifically for FPGA-based platform of two-way cable network (TWCNP) SOC, which is key device based on DOCSIS (Data-Over-Cable Service Interface Specifications) on HFC network whose CPU is Godson-I. Under TWCNP-OS, hardware and software share the same familiar Linux interface and the same level of support from the OS kernel. We use the concept of hardware process [3], which is the same as a normal Linux process except its "program" is an FPGA hardware design instead of software program. HAL is implemented, and communications between hardware and software are accomplished through it, which uses conventional Linux inter-process communication (IPC) mechanisms, such as shared file, pipe, shared memory, signal, and message passing. Hardware processes have access to system resources as their software counterparts, such as the general file system, standard input, standard output.

By building a cross GNU/GCC compiler for Godson-I, we can develop software in host machine (PC).Software designs can be developed in C/C++ language development environment a designer is familiar with. For hardware designs to communicate with the kernel, TWCNP-OS defines a standard message passing network that resembles the software system call interface by maintaining the hw/sw interface-HAL at the kernel level. This standardized network allows hardware designs be developed in any hardware language environment of choice.

The remainder of this paper is organized as follows. Section 2 surveys related work on hw/sw cosimulate/co-verification with RTOS supports, and analyses the difference all of them including our TWCNP. Section 3 introduces the system architecture, particularly the software and hardware architecture of CM platform in tow-way CATV. Section 4 elaborates on the co-verification approach based on TWCNP-OS, and its detailed design process is presented. Section 5, we give the system testing and performance comparison of the platform designed by this method.

2. Related Work

Hw/sw co-verification has been studied around the world for more than a decade. Simulator is an usual method of hw/sw co-verification, which must join with other parts of the system verification problem to create a complete verification solution. A number of commercial cosimulators have come onto the market. The cosimulators are currently one of indispensable CAD tools in the design of embedded systems and systems-on-chip.

There is no single hardware-software co-simulation problem and as a result there is no single tool or technology that can successfully solve all the problems associated with co-verification of hardware and software. Each developer must trade off the desired performance against the level of accuracy. Each developer must trade off the desired performance against the level of accuracy. For hardware designers the trade-off is principally between software simulation and the relatively faster technologies of hardware acceleration, emulation or rapid prototyping. For software developers of embedded system application software the primary need is simulation speed and the trade-off is among a variety of simulation approaches which achieve greater degrees of speed at the cost of diminishing the accuracy of the modeling of the hardware.

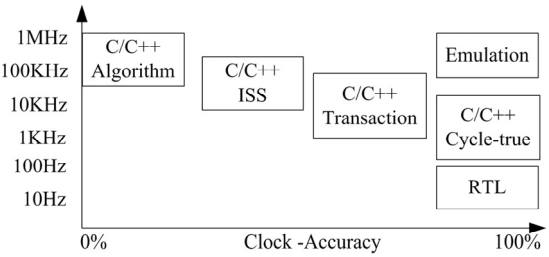


Figure 1. Trade-off between simulation accuracy and speed

HW emulation is a very important alternative to overcome the SW speed problems [4]. A co-verification platform using C++ simulator and FGPA (Field-Programmable Gate Arrays) emulator is presented in [5], gets a good simulation speed, and provides an accuracy/efficiency tradeoff through various abstraction levels (Figure 1). When the C/C++ abstraction is at the algorithm level, the simulation speed can reach about 1 MHz, but the simulation results have no cycle accuracy.

The speed of hardware emulator is typically up to 1 MHz, and they preserve cycle accuracy.

However, since the cosimulators do not feature explicit supports for RTOSs, a designer needs to use an ISS in many cases to run embedded software including an RTOS. Due to the slow execution speed, extensive simulation of large software is impossible.

In the recent years, several research efforts have been made to model RTOSs for system-level design and cosimulation. SoCOS presented in [6] is a system-level design environment where the OSAPI library provides generic RTOS system calls to application software. OSAPI is a virtual RTOS to enable native execution of embedded software. After simulation-based validation, the OSAPI calls are replaced with the system calls of the actual RTOS used in the final implementation to obtain the final software code. In [7], a similar approach is presented. A main difference is that the RTOS model in [7] is build upon an existing system-level design language, i.e., SpecC [8], so that existing CAD tools such as simulators can be used. Techniques presented in [9] also use the SpecC language to model the preemptive behavior. In [10], an OS model is proposed for fast and time-accurate cosimulation. The model focuses on accurately modeling the RTOS overhead during task execution as well as the preemptive behavior. In [11], a method which automatically generates RTOS-dependent software from SystemC description is presented. The method replaces SystemC's constructs for concurrency and communication with corresponding RTOS service calls. In this sense, it can be considered that SystemC involves a simple RTOS model in itself. With the development of Embedded OS, [12] gives us a RTOS-centric hw/sw co-simulator, which features co-simulation with functional simulation models of hardware written in C/C++ and co-simulation with HDL simulators, supports a complete simulation model of an RTOS based on uITRON [13].

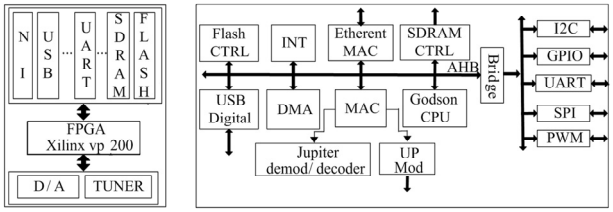
A common weakness of these OS models is that they support only a limited set of RTOS services in order to make the models generic and independent of specific RTOSs. for example, [12] have more than 80 service calls but only for μITRON-based RTOSs platform. However, the RTOS model in [7] supports only 16 service calls. These services may need to be fully utilized in order to write high quality software. Therefore, it is easily imagined that the quality of software automatically generated by these previous methods is lower than that of hand-crafted RTOS-dependent software. Instead, we have developed fully supports RTOS services by porting the standard MIPS/Linux kernel to our platform, such high-quality software can be designed and simulated. But there are a weakness for hardware-dependent, and have to put a Virtual hardware level in HAL when some hardware component haven't finished.

TWCNP shares a similar design philosophy as BORPH [3] in providing a unifying coarse-grain hardware (FPGA) and software component interface. They are not a complete system to perform typical hw/sw codesign tasks such as partitioning, cosynthesis, cosimulate, or verification. Instead, by providing basic Linux OS services, it acts as a platform on which these tasks can be carried out. In fact, our TWCNP is not a replacement of the cosimulators with generic RTOS modeling, but is complementary to them.

The main contribution of TWCNP-OS is that by leveraging conventional Linux semantics to FPGA-based platform of two-way cable network (TWCNP) SOC, it provides a unique, unified environment for both FPGA and software application designers. The Linux semantics is familiar to developers across many research domains, thus lowering the barrier-to-entry into FPGA-based SOC. Furthermore, since TWCNP-OS is implemented as an extended Linux kernel, a TWCNP-OS managed system may leverage all commodity Linux software applications for developing, testing, benchmarking, and deploying FPGA applications.

3. Two-way CM Platform Architecture

The CM (cable modem) platform is based on HFC network, which can increase the transmission quality efficiently, and integrate the CATV and internet digital network. Using the CM can expediently develop many services based on Internet protocol, such as VoIP, VOD, HDTV and high-speed web browsing. The design of CM includes three parts: evaluation board design, embedded software design and SOC design. Verification is used through the whole process of SOC development. The software and hardware co-verification is stressed in this paper.



(a) FPGA evaluation board (b) TWCNP SOC hardware frame

Figure 2. Overall frame of FPGA emulation board

The hardware architecture of CM is described in Figure 2. It includes three modules:

1) EuroDOCSIS protocol processor: It includes independent developed physical layer modem (Jupiter), EuroDOCSIS MAC protocol processor and radio frequency interface circuit.
2) BUS and peripheral equipment module: Interconnection through AMBA bus and peripheral equipment includes IIC, GPIO, UART, PWM and Ethernet MAC.

3) Godson-I CPU: Godson-I IP core, Based on MIPS instruction, can apply to both the universal and embedded system, has a good compatibility.

The software architecture which is described in Figure 3, includes three main parts:

1) Operating System: based on MIPS/Linux configuration, and making Linux run on CM based on Godson-I, including Bootloader and OS kernel, It supports memory management, file management, task schedule, and modularization.
2) Hardware driver: it ensures hardware working high-efficiently and properly. The driver such as GPIO, IIC, network adapter can either be compiled into OS kernel or be loaded dynamically.
3) EuroDocsis protocol stack and CM application: realizing central functions defined in EuroDocsis, it provides software guarantee to bridge CM and CMTS (cable modem terminal system). Application software defines a set of user interface and ensures function maintenance and management to CM.

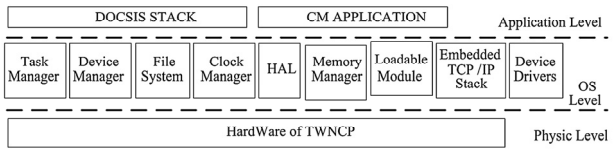


Figure 3. Software frame of two-way cable network CM

OS is the cornerstone of the software design platform, which manages all software and hardware directly, provides interface to application and system call. It is also used to validate hardware and software function, guide the hardware design to be more reasonable.

4. Co-verification Based on TWCNP-OS

There are two common verification methods on soc design.

1) The first is software verification which simulates the system behavior through establishing emulation model. Through this method, system's detailed status should be observed but the run time is much longer. ISS (Instruction set simulation) is an example of it.
2) The second is the hardware verification. Now the prevalent one is the SOPC based on FPGA, which can online program and has accurate clock cycle. This way has a higher emulation speed, but the system's status cannot be observed. It includes three aspects: software verification, hardware verification, and software-hardware interface verification.

Combining the merits of these two kinds of method, a co-verification method based on TWCNP-OS is proposed in this section, which is supposed that the hw/sw partitioning and hardware cosynthesis have finished. Firstly, we introduce the co-verification platform, give a general idea of porting the MIPS/Linux to our SOC. Secondly, Software and Hardware co-verification based on TWCNP-OS will be introduced.

The co-verification platform (Figure 4) is composed of host, two self-existent target machines and equipments which connects them each other, such as UART, USB and network. Host is a common PC, which provides cross-compilation environment, console and HDL simulator. Godson-I test board and FPGA development board (TWCNP) constitute target machine. Godson-I test board adopts Godson-I CPU and it8172G chip sets, MIPS instruction set and Linux OS can be run in it. This one is mainly used in the development of DOCSIS protocol stack. Being composed of FPGA vp200, peripherals and interfaces, FPGA board is the hardware verification platform of the whole SOC system, which is mainly used in the verification of EuroDOCSIS protocol processor and related hardware logic.

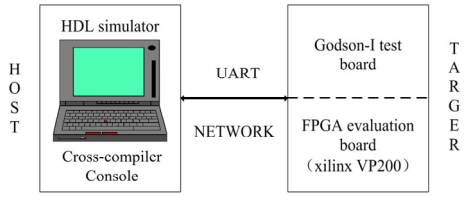


Figure 4. Co-verification platform of the TWCNP

4.1. Realization of TWCNP-OS

TWCNP-OS is an operating system designed for TWCNP based on MIPS processor. It extends a standard Linux kernel to include support for FPGA's in our platform. Treating FPGA's as both coprocessors and processor, TWCNP-OS treats FPGA's as hardware process in the system as normal computational resources. The interface between hardware and software is HAL. All processes can therefore be either software programs, or hardware designs running on FPGA's. Therefore, to the rest of the system, communicating with a hardware process is no different from communicating with a normal Linux process. This homogeneous handling of hardware and software in the kernel forms the foundation of coarse grain hw/sw codesign boundary. Figure 5 depicts this conceptual block diagram.

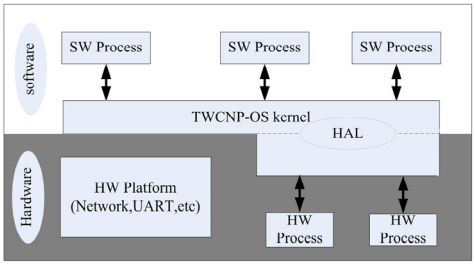


Figure 5. TWCNP-OS extends a traditional Linux system with hardware process support. HW/SW processes share the same I/O interface and communicate by HAL

Like embedded OS, TWCNP-OS includes cross-compiler, Bootloader and OS Kernel too. Cross-compiler is necessary to design an embedded OS, which is a compiler capable of creating executable code for a platform other than the one on which the compiler is run. It is a tool that one must use for a platform where it is inconvenient or impossible to compile on that platform, like microcontrollers that run with a minimal amount of memory for their own purpose. Whole process of building a cross-compiler can find in [15].

Bootloader is used to boot the machine, such as initializing basic CPU registers, UART, SDRAM controller, and copying OS kernel from Flash to memory. Its design is derived from PMON [14], which is a freeware ROM-monitor developed for early LSI Logic MIPS R3000 evaluation boards. Since its creation, PMON has become a very common firmware for MIPS evaluation boards and development systems. It contains support for several boards, R4000-style exceptions, and uses the GNU make system. So the major revision is the address of hardware components. There are three important parts will be down:

1) CPU register in Godson-I have to be set up the normal value, such as interrupt, timer, cache, memory management;
2) The drivers of hardware components should be taken effect to use UART, serials and console etc. We can debug and interact with target machine.
3) Only after moving the kernel from Flash kernel space to memory, the kernel can boot and manage whole system.

Linux is one of the common embedded OS. For MIPS architecture of Godson-I, OS mainly adopts MIPS/Linux schema, using ANSI C, MIPS assembly language and cross-compiler-MIPS-GCC 3.23 as SDK. TWCNP-OS is divided into two parts. The hardware independent part is realized by MIPS assembly language. The primary design flow is described in figure 6. We will introduce the design of TWCNP-OS on the basis of MIPS/Linux kernel source code in the following part. Overall design steps as following:

1) Hello World! — Get board setup, serial porting working, and print out "Hello, world!" through the serial port.
2) Get early printk working — Make the first TWCNP-OS image and see the printk output from kernel.
3) Serial driver and serial console — Get the real printk() function working with the serial console.
4) KGDB — KGDB can be enormously helpful in our development.
5) CPU support — Because Godson-I CPU is not currently supported, we need to add new code that supports it in arch directory of Linux source codes.
6) Board specific support — Create your board-specific directory. Setup interrupts routing/handling and kernel timer services.
7) HAL — It makes the upper layer software be independent of the bottom hardware and manages or simulates a large number of the bottom hardware.
8) Ethernet drivers — We should already have the serial port working before attempting this. With ethernet driver working, we can set up a NFS (network file system) root file system which gives you a fully working Linux user space.
9) ROMFS root file system — Alternatively you can create a user space file system as a ROMFS image stored in a ramdisk, or Flash root file system.

Here we will emphatically present HAL [19] (Figure 6(b)), which is the interface between Hardware and software, which includes device driver, configuration manager, control manager and data manager and VHL (virtual hardware level). It includes all the functions that manage and schedule the hardware (hardware process) uniformly for the SOC, and can support a many of different hardware devices or virtual hardware devices (VHL). So we can send messages to hardware using standard system call just as to a software process by HALIF (HAL interface). Overall control module is the key of HAL implementation. Configuring manager takes charge of configuration channel. Control manager supplies control channel. Data manager takes charge of two-way data channel. Driver module regards all hardware devices which are managed by HAL as one device, and its run driver is regard as a hardware process.

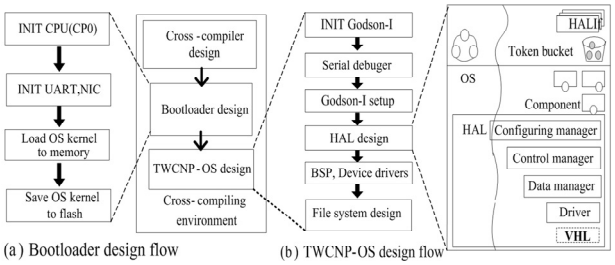


Figure 6. Design flow of the TWCNP-OS, comparing to generic Linux kernel, we give a HAL on kernel

4.2. Software Verification Based on TWCNP-OS

Software is an important part of the system, verification of software is necessary. Software verification mainly uses ISS, though it has accurate clock cycle, the simulation speed is comparatively slow. The software verification proposed in this paper simulates hardware on target machine. Host and Godson-I test board to validate the validity of software directly on function level. Software verification is mainly used to DOCSIS protocol stack module and the relative CM application program. According to DOCSIS specification, DOCSIS protocol stack module is divided into hardware-dependent part and hardware-independent part. The hardware-independent software realizes user level

function such as VOD etc.

In the traditional software and hardware concurrent design, the design of hardware-dependent software can not progress until the accomplishment of the hardware design. VHL can efficiently resolve this problem. The VHL established on OS can be one or a group of virtual hardware, which simulate hardware to carry out function verification. VHL is extension of driver. it can not only apply uniform interface to application, drive hardware, but also response according to hardware's actual characteristic. If the hardware design has been accomplished, we only need to simply change the VHL to drivers and the software need not to be changed. Thus, hardware-dependent software design and verification can be processed without hardware environment which has great meaning.

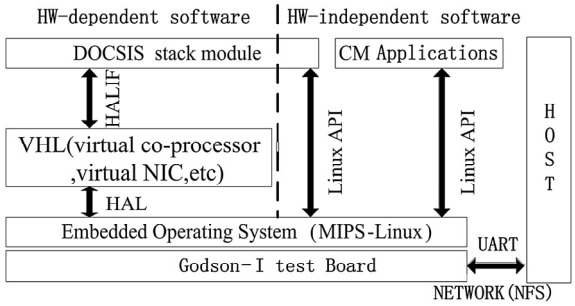


Figure 7. The sketch map of software co-verification, the left part of dotted line denotes the hw-dependent software, the right one is hw-independent software. Each uses a different verification method

TWCNP-OS make it possible to use VHL in software verification. The DOCSIS protocol stack and CM application program validated on development board can drive hardware on TWCNP-OS. Godson-I test board is mainly used in software verification. Target host is the auxiliary equipment which provides convenience for the verification. The software verification flow is described in Figure 7, the broken line divided software into hardware-dependent part and hardware-independent part. The hardware-independent part is validated directly on development board and run on target FPGA board. The hardware-dependent part is mainly in DOCSIS protocol stack module which needs the support of MAC co-processor. Using modularize mechanism to establish virtual MAC co-processor on VHL can simulate MAC co-processor to realize software-hardware interaction. Thus the software is independent on or weak dependent on hardware. Virtual MAC co-processor construct standard data, notify the hardware-independent part in DOCSIS module through interrupt to process data analysis. The verification of CM application program use C/S mode, the server responses to the client by simulating CMTS.

4.3. Hardware Verification Based on TWCNP-OS

Hardware is the support platform of embedded system. Hardware verification validates through simulating each unit's behavior and mainly has two modes. One mode is pre-simulation which simulate all hardware modules to validate if the hardware accord with the design requirement. The common simulation tool is ModelSim [17], NC-verilog and NC-VHDL [18]. The other mode is integrated post-simulation which can put composite latency files into integrated simulation model to estimate the effect brought out by gateway delay. The hardware verification introduced in the following base on the accomplishment of pre-simulation and is in the domain of FPGA-based post-simulation.

Any verification should carry out according to some standards. We adopt TWCNP-OS and DOCSIS protocol stack module as test program to validate hardware. TWCNP-OS which bases on Linux structure has high reliability and stability through long time verification. DOCSIS protocol stack module has passed the software verification. We can validate the MAC co-processor as long as modify the virtual MAC co-processor.

The TWCNP-OS-based hardware verification has high stability and simulation speed. At the time of the accomplishment of verification of hardware and software, the hw/sw interface has been validated. This verification method is on the basis of passing pre-simulation for each hardware module and validates the function of hardware system. For the complexity of the system, the hardware system is divided into two parts: MAC co-processor and other hardware logics. The hardware verification has three steps:

1) The first step is the verification of MAC co-processor. We first connect FPGA with the bus of Godson-I test board, download the hardware logic of MAC coprocessor to FPGA, regard MAC co-processor as a peripheral. Then modify the virtual MAC co-processor, delete its function of simulation hardware, make it be a driver to control hardware. At last, validate it through DOCSIS protocol stack. Thus we can easily find out errors in it. Furthermore, the concurrent process of hardware verification can improve the efficiency.
2) The second step is validating other hardware logic and software-hardware interface using TWCNP-OS and FPGA board. The hardware is validated respectively by calling the drivers compiled to drive hardware.
3) The third step is the system verification on FPGA board by using TWCNP-OS and software module after integrating the hardware logic. The function and performance test is processed in this step. The detailed process is introduced in next section.

5. System Testing and Performance Comparison

The whole system on the actual running environment for

testing is the last step of verification. Therefore, we will give a topology of cable network (Figure 8) and construct a test environment (Figure 9) accordingly.

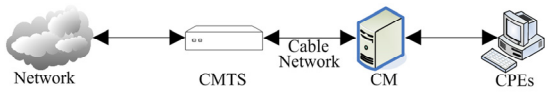


Figure 8. The topology of cable network

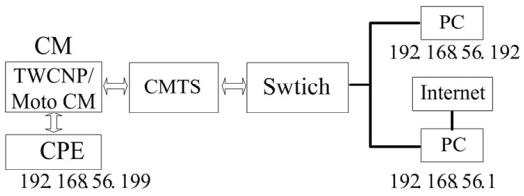


Figure 9. Performance test environment of CM

Stability, time delay and network congestion are the performance target of CM. Stability is a basic target for any device. Here it means data can be transferred to CMTS safely without losing data. Time delay is another target to estimate performance, if it can't meet the some special requirement of the system, it will be no-good. We will test the response time between CPE (customer premises equipment) and PC. Network congestion is an important problem, it is good rule to evaluate whether the device is excellent or not. Because some equipment can work well without network congestion, but it will induce packages losing when it happens network congestion. So it is a complemental test to stability.

In the following parts in this section, we will give some tests to get the above performance targets respectively in TWCNP and SM5100. SM5100 is a mainstream CM produced by Motorola, and its performance can delegate a standard of CM in a certain sense. Special declaration, test results will be influenced by real environment.

Test 1

To get the response time of CM, We send 100000 IP data packages by ping command from CPE to PC via TWCNP and SB5100 CM respectively, each package is 74bytes, the results as table 1.

Table1. Functional test and comparison of CM emulation platform

Test platform	T_{ave}	T_{ans}	T_{ans} (max)	p
TWCNP	12ms	10ms	17ms	0
SM5100	9ms	6ms	15ms	0
Parameters	N_{sum}=100000，*packet size*=74 bytes			

The following is the formula of calculate different response time. Here, T_{ans} is response time, p is rate of loss package, T_{ave} is average response time, T_{sent} is time to reach destination, T_{resv} is response data package cost time, N_{lost} is number of loss package, N_{sum} is total number of package.

$$T_{ans} = T_{sent} + T_{resv} \quad (1)$$

$$P = \frac{N_{lost}}{N_{sum}} \times 100\% \quad (2)$$

$$T_{ave} = \frac{\sum_{i=1}^{N_{sum}} \left(T_{sent}[i] + T_{resv}[i]\right)}{N_{sum} - N_{lost}} \times (1-P) \quad (3)$$

From testing results of Table 1, we can get that the average network response performance is similar between TWCNP and SM5100.

Test 2

To get network congestion and stability results, we fabricate a mass of normal at the speed of 1.4 MB/s and congestion data packets at a speed beyond upper line by CommView on CPE, and send them to PC via TWCNP an SM5100 about 24 hours. At the same time, we will capture the throughput on PC.

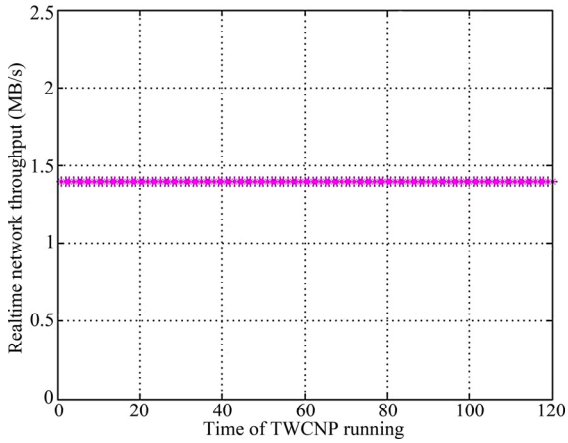


Figure 10. The stability test of TWCNP

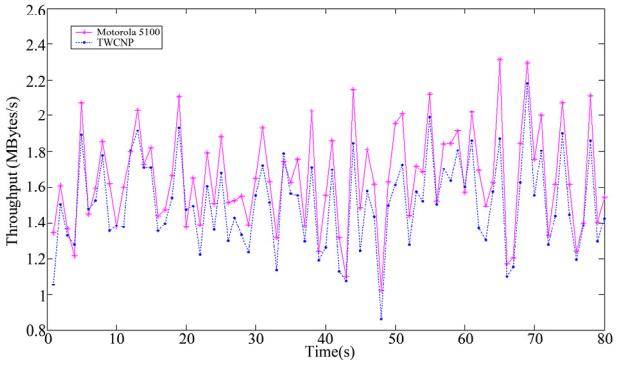


Figure 11. Data throughput analyses of TWCNP and Motor 5100 CM

Sampling 120 minutes normal packets (Figure 10)

and Sampling 90s' data on congestion time (Figure 11), we get that throughput of TWCNP is still up to 1.52M B/s, MOTOROLA 5100s' is 1.67 M B/s under network congestion. TWCNP can transfer all packets to PC at same speed. The result of this case study demonstrates that it gets a high stable throughput.

From the above tests, we can find that the performance of our TWCNP is equivalent or a little low to SM5100. The main reason is that the hardware of them is not on the same level, Godson-I CPU (only 26M Hz) frequency is far lower comparing to SM5100'processor, so throughput should be lower. But it still exerts a good performance.

6. Conclusions

This paper presents a hw/sw co-verification methodology based on TWCNP-OS, a Linux-based operating system designed for FPGA-based platform of two-way cable network (TWCNP) SOC, and then particularly expatiates on the software and hardware co-verification method. Through the implementation of TWCNP-OS on TWCNP and development on host/two targets, we realize the combination of design and verification. It is not a replacement of the co-verification with generic RTOS modeling, but is complementary to them.

The experiment makes it clear that the verification platform is simple-built and cost-effective. The scientific verification method shortens the development cycle and the validated system is stable and can be applied in the verification of other embedded system. Most importantly, since the introduction of TWCNP-OS to our FPGA-based platform, we have observed increased productivity among high-level application developers who have little experience in FPGA application design.

7. References

[1] N. Ohba and K. Takano, "An SoC design methodology using FPGAs and embedded microprocessors," Proceedings of the 41st annual conference on Design automation, San Diego, CA, USA, June 07–11, 2004.

[2] Habibi and S. Tahar, "Design and verification of SystemC transaction-level models," IEEE Trans. VLSI Syst., 14(1): pp. 57–68, January 2006.

[3] H. So, A. Tkachenko, and R. Brodersen, "A Unified Hardware/Software Runtime Environment for FPGA-Based Reconfigurable Computers using BORPH," CODES+ISSS, 2006.

[4] D. Atienza, P.G. Del Valle, G. Paci, et al., "A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip," Proceedings of the 43rd annual conference on Design automation, San Francisco, CA, USA, July 24–28, 2006.

[5] Y. Nakamura, K. Hosokawa, I. Kuroda, K. et al., "A fast hardware/software co-verification method for system-on-a-chip by using a C/C++ simulator and FPGA emulator with shared register communication," Proceedings of the 41st Design Automation Conference (DAC'04), San Diego, Calif., USA, pp. 299–304, June 2004.

[6] D. Desmet, D. Verkest, and H. De Man, "Operating system based software generation for systems-on-chip," Proceedings of Design Automation Conference (DAC), 2000.

[7] A. Gerstlauer, H. Yu, and D. Gajski, "RTOS modeling for system level design," Proceedings of Design Automation and Test in Europe (DATE), Embedded Software Forum, 2003.

[8] SpecC Technology Open Consortium, http://www.specc.org/.

[9] H. Tomiyama, Y. Cao, and K. Murakami, "Modeling fixedpriority preemptive multi-task systems in SpecC," Proceedings of Workshop on Synthesis and System Integration of Mixed Technologies (SASIMI), 2001.

[10] Y. Yi, D. Kim, and S. Ha, "Virtual synchronization technique with OS modeling for fast and time-accurate cosimulation," Proceedings of International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2003.

[11] F. Herrera, H. Posadas, P. Sanchez, and E. Villar, "Systematic embedded software generation from SystemC," Proceedings of Design Automation and Test in Europe (DATE), Embedded Software Forum, 2003.

[12] S. Honda, T. Wakabayashi, H. Tomiyama, et al., "RTOS-centric hardware/software cosimulator for embedded system design," Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, Stockholm, Sweden, September 08–10, 2004.

[13] ITRON, http://www.assoc.tron.org/itron/.

[14] http://www.linux-mips.org/wiki/PMON.

[15] http://cross-lfs.org/view/svn/mips/index.html.

[16] http://linux.junsun.net/porting-howto/.

[17] Mentor Graphics Corporation, http://www.mentor.com.

[18] C. Wang, X.G. Xue, et al., "FPGA/CPLD Design TOOL-Specification for Xilinx ISE 5.X," Posts and Telecom Press, 2003.

[19] S. Yoo, I. Bacivarov, A. Bouchima, et al., "Building fast and accurate SW simulation models based on hardware abstraction layer and simulation environment abstraction layer," Proceedings of the Design, Automation and Test in Europe (DATE'03), Munich, Germany, pp. 500–506, March 3–7, 2003.

