Verifying a Virtual Development Environment for Embedded Software

Febiansyah Hidayat, Hadipurnawan Satria, Jin B. Kwon
Department of Computer Science and Engineering, Sun Moon University
e-mail: havban@gmail.com, hadi198@yahoo.com, jbkwon@sunmoon.ac.kr

임베디드소프트웨어 가상 개발환경에 대한 검증

페비안시아 히다얏, 하디푸르나완 싸트리아, 권진백
선문대학교 컴퓨터공학과

Abstract

Increasing use of embedded systems has made many improvements on hardware development for specific purpose. Hardware changes are more expensive and harder to implement rather than software changes. Developers need tools to do design and testing of new hardware. Many simulation tools have been made to mimic the hardware and allow developer to test programs on top of new hardware. Virtual Development Environment for Embedded Software (VDEES) is one of the alternatives available. It provides an open source based platform and an Integrated Development Environment (IDE) that can be used to build and testing newly made component, faster and at low-cost.

I. Introduction

VDEES has been developed since 2005, based on open source platforms. VDEES is build on top of Eclipse platform and uses SID as the virtual environment. SID is component based. It means, to build a new kind of hardware is to build a new component in the SID framework.

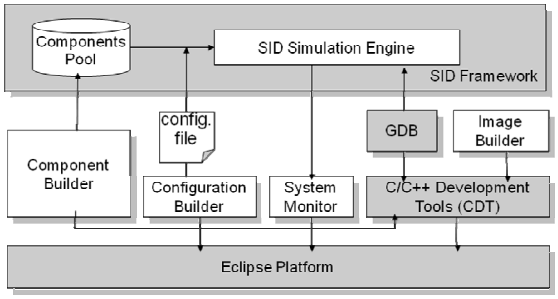

Fig 1. VDEES Architecture

This paper will explain verification steps we have done using VDEES. There are five target program used to test. Description about the physical environment and virtual environment will be presented on section 2 and 3. The development and testing of components and binary image building will be included in section 4 to 6.

II. Physical environment

We used an MBA-2410 board as target board that consists of ARM920T with 32 MB of RAM. It has some modes to run, using Smart Card or using memory. In this project, the program is running initially by using memory, addressed at 0x30000000 because the physical RAM is located at that address.

ARM920T has the same instruction set as the ARM7TDMI processor that is supported by the SID framework. The difference is only on the availability of Memory Management Unit (MMU) component that is needed when an Operating System (OS) is running on the board. In our case, the MMU is not needed since only one single program, non OS-based is running. In that case, existing ARM7TDMI virtual component is sufficient.

The IDE used to develop the program is Code Warrior (CW). It supports many kinds of board. Debugging feature is also available using AXD debugger. CW is recommended and distributed with the manual compliant, along with the board. Another component used is Spider debugger, to do debugging.

III. Virtual Environment

We are using VDEES that has SID framework inside of it. The VDEES run on Eclipse with C/C++ Development Toolkit (CDT) plug in support. SID framework should be installed too in the system, for more information, you can access this link [7].

GNU ARM compiler is needed, it is a separated element from eclipse, but already included in the VDEES installation package. Different from regular gcc, the arm-elf-gcc (gcc of GNU ARM) will build the binary to comply with the target machine, that may have different instruction set than the host machine. The language itself is C, that we will explain later on the source code adaptation, some differences between CW and GNU ARM.

`arm-elf-gdb` (gdb of GNU ARM) is used to do monitoring and debugging of the running application. The two applications communicate through socket with defined port. Then, the value will be displayed in the Eclipse window. VDEES can do monitoring to the level of virtual component variable status. It should give an in-depth view of current running program on the virtual environment.

Fedora Linux 4 and UBuntu Linux 8.4 is the OS

"This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) Support program supervised by the NIPA(National IT industry Promotion Agency)" (NIPA-2009-C1090-0902-0020)

platform we used in this development.

IV. Pre Development

At this stage, we analyzed the pre-existing components of SID, for example, the ARM CPU, Loader, LCD, Timer, Interrupt component. From that list, we can compare it to the real board components especially on the behavior and the existence of registers.

We listed the required components of the program by analyzing the source code. We did not want to implement non existing component in the SID framework while it is not needed in the program. We just need to make sure the program will run well on the virtual environment, by getting the list of registers that will experience value changes on runtime. The registers implementation is important, since the MBA-2410 is accessing components through memory addresses. This gives many conveniences to the developers as they only need to refer a component using regular address. Which register address is accessed shows which component is being used.

The list of components we needed to be built are: Analog to Digital Converter (ADC - Handle conversion data from analog to digital data) , *Touch Screen Interface* (using mouse click as event trigger), *LCD Controller* (Process the data from the video buffer and update the the LCD screen), *Time*r (periodically trigger interrupt), *Clock and Power management* (manage the clock speed given to variative components), and *Interrupt Controller* (Provide interface to the interrupt queue in the processor). Some of those components are extension of pre-existing one.

–

V. Development

V.1. Custom Component

VDEES provides wizard to create a custom component, the wizard will generate basic template files required to build the component. Those template files can be modified to our needs.

C++ editor and builder are available in VDEES as the extension of CDT plugin from eclipse. For Tcl/Tk code editing, we use external editor, *gedit* or *vim*.

The first component implemented was ADC Controller and Touch Screen Interface using C++ and Tk. The last one implemented is Interrupt Controller, since at first we could actually bypass the code that uses the interrupt component.

V.2. Image Binary Building

As we mentioned before, the physical environment uses CW to build the binary image of program. CW uses almost similar syntax of C and Assembly compared to the GNU ARM compiler. Some keywords are different, so we modified it referring to the GNU ARM code style.

The basic change is modifying the names to be all lower case, since Linux is case sensitive. In the C source code, we changed the interrupt keyword of Interrupt Request Queue (IRQ) from "__irq" to "__attribute__((interrupt("IRQ"))))". And most of changes was done in the Assembly code where we modified it refer to [6].

We commented codes that are accessing MMU component, because we do not want to implement it yet, the current program has no OS running and will only use regular memory address mapping.

VI. Performance evaluation

We tested the component using print out debugging. There were some errors on arm-elf-gdb, we could not access current runtime variables, only able to start and stop the execution.

From five target programs, only one program fails to run well on the simulation. However, in the overall performance of the system, all programs ran very slow. The scheduling factor in SID should be investigated. Between printout of `stdio` debugging and executing current instruction set, there is so much difference elapsed time.

The monitoring feature of VDEES is not running well, only "run", "stop" and memory view feature is available. The in-depth component variable cannot be accessed; it may be problem in the configuration file or in the component implementation.

VII. Conclusion

We have developed custom components and verified how the VDEES run. It helps developer to do faster design and testing of newly hardware without waiting for real implementation of hardware available. Debugging feature is still not functioning well, and it will be our future work to investigate. But overall, using VDEES will benefit developer to do simulation before implementing the real hardware.

References

[1] Seal, David, A*RM Architecture Reference Manual*, Addison-Wesley, 2001.

[2] H. Satria, B. Wibowo, J.B. Kwon, J.B. Lee, Y.S. Hwang, *A Virtual Development Environment for Embedded Software using Open Source Software*, IEEE Trans. on Consumer Electronics, May 2009.

[3] MBA-2410 User manual, [24 November 2003]

[4] VDEES Homepage, http://cslab.sunmoon.ac.kr/vdees/, 2009

[5] Ronetix, *ARM cross development with GNU Toolchain and Eclipse version 1.1*, May 2007

[6] ARM Architecture Reference Manual , 2005

[7] SID reference, http://sourceware.org/sid, 2009