Virtual ARM Platform for Embedded System Developers

Alex Heunhe Han, Young-Si Hwang, Young-Ho An, So-Jin Lee, Ki-Seok Chung
Dept. of Electronics and Computer & Communications Engineering, Hanyang University
alexhan@hanyang.ac.kr

Abstract

More and more embedded system developers and system-on-chip designers reply on microprocessor-based design methodology to reduce time-to market. ARM processor has been a major player in embedded system industry over the last 10 years. However, there are many restrictions on developing embedded software using ARM processor in the early design stage. For those who are not familiar with embedded software equipment, testing their software on a real ARM hardware platform is a challenging job. To overcome such a problem, we have designed Virtual ARM Platform, which offers easier testing and debugging environment to ARM based embedded system developers. Major benefits that can be achieved by utilizing a Virtual ARM platform are (1) reducing development cost, (2) lowering the entrance barrier for embedded system novices, and (3) making it easier to test and debug embedded software designs. Unlike many other purely software-oriented ARM simulators which are independent of real hardware platforms, our proposed Virtual ARM Platform is specifically targeted on SYS-Lab 5000 ARM hardware platform, (designed by Libertron, Inc.,) which means that Virtual ARM Platform imitates behaviors of embedded software as if the software is running on the target embedded hardware as closely as possible. This paper shows how Virtual ARM Platform is designed and how it can be used to reduce design time and cost.

1. Introduction

An embedded system is a special-purpose computer system designed to perform a small set of dedicated functions, sometimes with real-time computing constraints. It is usually embedded as part of a complete device which includes hardware and mechanical parts. Embedded systems span all aspects of modern life and examples of their use are numerous. Not only consumer electronics including personal digital assistants (PDAs), mp3 players, mobile phones but also telecommunications systems, transportation systems, and medical equipments employ numerous embedded systems such as mobile network system, anti-lock breaking system (ABS), GPS, electronic stethoscope. [1]

One of the most important hardware components of an embedded system is microprocessor. Microprocessor has played a main role in development of IT industry, especially in popularization of personal computer (PC) and internet. Each microprocessor has its own characteristics as it is used in various categories. [2] A series of ARM processors have RISC-type architectures, and they have been widely used in a number of embedded designs. Because of not only their high performance and low cost but also power saving features, ARM variants are dominant in all corners of consumer electronics, from portable devices (PDAs, mobile phones, media players, handheld gaming units, and calculators) to computer peripherals (hard drives, desktop routers).

Virtual ARM Platform, which we are going to propose in this paper, is an ARM simulator designed considering target hardware. In contrast to other virtual ARM machines that have been designed only in software model, our Virtual ARM Platform enables most similar operations to target embedded system. By “similar operations”, we mean that our Virtual ARM Platform allows embedded system developers to develop and test their embedded application as if they do on a real H/W platform. To implement a target-specific virtual ARM platform, we have chosen SYS-Lab5000 ARM hardware platform, (designed by Libertron, Inc.,) as our target H/W platform. Since a target-specific virtual platform allows the target platform specific details to be tested without the real target platform, design time and cost can be greatly reduced. Especially, for educational institutions where a sufficient number of embedded equipments are not deployed, this type of target specific virtual platform will be greatly helpful to teach students how to design a target specific embedded system while minimally requiring the time to use the real hardware. Our Virtual ARM Platform implementation is based on ARM simulations by using SimIt-ARM simulator [3] and Graphic User Interface by using QT library [4]. Also to extend the target specific capability with ease, we have designed an emulator which automatically produces plat-form-specific environment settings from the given platform-specific information for easy maintenance and adjustment. Furthermore, we have implemented a timer interrupt handler for simple O.S simulation.

The rest of this paper is organized as follows. In Section 2, we discuss how it is different to utilize Virtual ARM Platform from traditional ways of developing embedded system. In Section 3, we summarize some of closely related existing studies. Section 4 presents how our proposed Virtual ARM Platform is organized and how it works. In Section 5, we present experimental results. Section 6 will conclude this paper.

2. Needs for virtual ARM platform

2.1. Development of embedded system S/W

It is widely known that there is a significant difference between developing software for general computing platform such as PC and developing software for embedded system. Therefore, software developers for general platforms tend to have many difficulties in developing embedded software because embedded system software development additionally requires the generated executable program to be downloaded on the target embedded system to examine its execution. This means that the software development environment (typically PCs) and the software execution environment (the target embedded platform) of embedded system are not same, while general software has the same development and execution environment (both PCs). Figure 1 shows the general development and test procedure of embedded system software.

While PC software only needs PC in every stage of its development, embedded software essentially needs to execute on real embedded system hardware. For those who are not familiar with embedded software equipment, testing their software on a real hardware platform is a challenging job. They should learn about overall embedded system, and must be adept in using

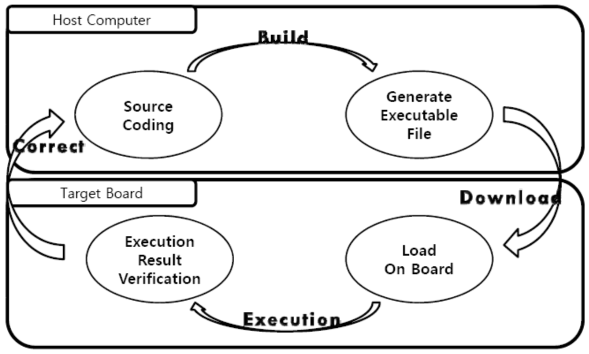


Figure 1. Development of embedded system software

hardware and downloading programs on hardware. Also, if they cannot afford to have enough (often highly expensive) embedded equipment, their opportunity to test their software on the real target platform will be severely limited. To overcome such a problem, we have designed Virtual ARM Platform, which offers easier testing and debugging environment to ARM based embedded system developers.

2.2. Usage of virtual ARM platform

Virtual ARM Platform enables to observe the execution result of embedded software as if they are downloaded and executed on a real hardware ARM Platform. Developers can write program codes, build executable files, and verify their programs by using Virtual ARM Platform in the development host (PC). Since it doesn't need any hardware but PC, there is no downloading stage in developing procedure. Figure 2 shows the procedure of embedded system development when using Virtual ARM Platform.

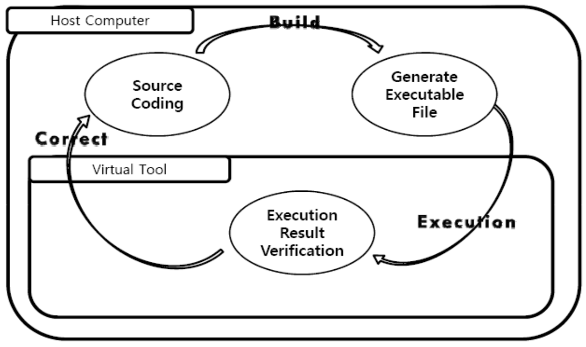


Figure 2. Development of Embedded System Software Using Virtual ARM Platform

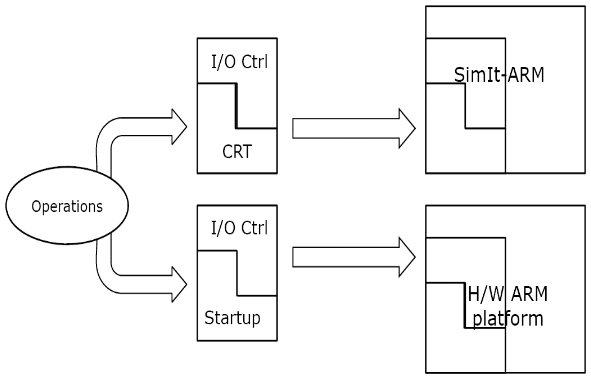


Figure 3. Execution of Firmware Level Code in SimIT-ARM and H/W ARM Platform

3. Related work

Recently, many different approaches have studied Virtual Tool for verification of embedded software in host PC. In this section, we briefly summarize some of them.

As one of the most well-known simulators for ARM, ARMulator [5] can be used to provide virtual prototyping environment for embedded system development. By virtually implementing hardware IP and porting an OS such as µC/OS-II [6] on ARMulator, it is possible to carry out simulation without H/W platform.

SimIt-ARM [3] is an instruction-set simulator that runs both system-level and user-level ARM programs. Also, SimIt-ARM supports two popular simulation styles: interpretation and dynamic compiled simulation. But as shown in Figure 3, SimIt-ARM cannot build firmware-level executable program without linking I/O control codes with C Runtime Library (CRT), while hardware platform links I/O control codes with its startup code. Since initialization codes are different for each target platform, developers can't use the same executable program in a virtual tool and a target H/W ARM platform.

All these approaches have tried to offer developers to write a source code, build an executable program, and execute it in the host PC. But unfortunately, these tools can only simulate the executable file for a specific target microprocessor. Since it cannot simulate the execution for the whole target hardware platform, the control of peripheral devices is impossible, while imposing considerable limits in code simulations.

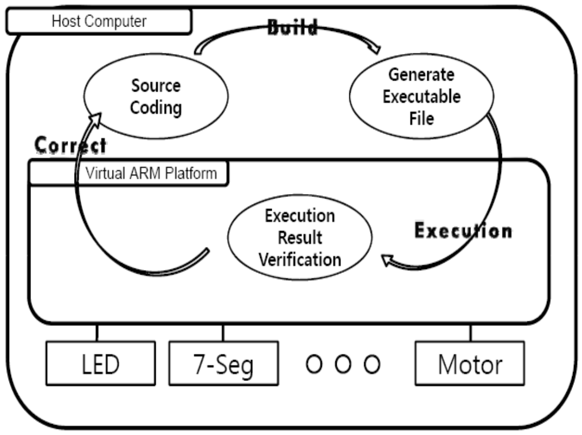


Figure 4. Virtual ARM Platform Offering Control for Peripheral Devices

4. Implementation of virtual ARM platform

We propose Virtual ARM Platform to offer extended control for peripheral devices which conventional ARM simulators cannot offer. By adding control codes for peripheral devices to existing ARM simulators such as SimIT-ARM or SimpleScalar, Virtual ARM Platform can control peripheral devices such as LED, 7-segment, step motor, etc. Figure 4 shows the procedure of embedded system development using Virtual ARM Platform that offers "target-like" control for peripheral devices.

Virtual ARM Platform consists of Virtual ARM Simulator, Graphic User Interface, Input Event Handler, Timer, and I/O Device Models. As Figure 5 shows, when Virtual ARM Platform is started, it initializes Graphic User Interface environment and displays the main GUI window on screen. Then Virtual ARM Simulator is executed, processing the instructions in the executable file and sending the results to Virtual ARM Platform. When Virtual ARM Platform gets the simulation results from Virtual ARM Simulator, it updates its GUI in cooperation with the execution report from Virtual ARM Simulator. When Input Event Handler detects any input events, it analyzes the event and terminates the program if the event is the termination event. If the event is not the termination event, the handler sends the processed input event to Virtual ARM Simulator, which simulates the instructions with the transferred input data.

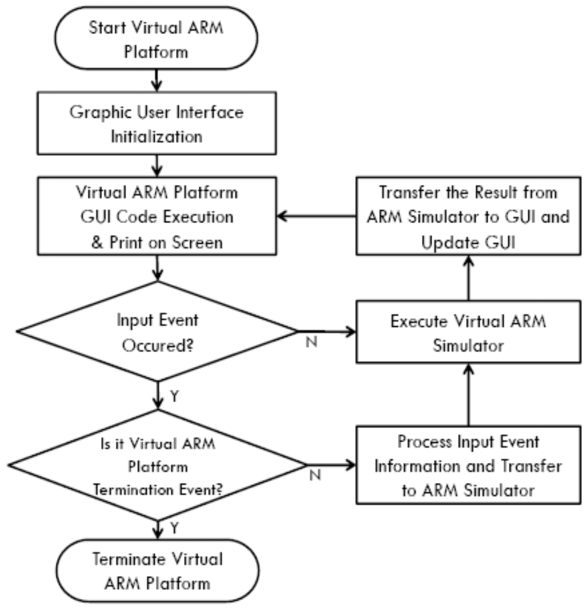

Start Virtual ARM Platform
Graphic User Interface Initialization
Virtual ARM Platform GUI Code Execution & Print on Screen
Transfer the Result from ARM Simulator to GUI and Update GUI
Input Event Occured?
N
Execute Virtual ARM Simulator
Y
Is it Virtual ARM Platform Termination Event?
N
Process Input Event Information and Transfer to ARM Simulator
Y
Terminate Virtual ARM Platform

Figure 5. Execution Procedure of Virtual ARM Platform

4.1. Virtual ARM simulator

Virtual ARM Simulator not only decodes and executes the given ARM instructions just as a real ARM processor does, but also handles the transferred events from Virtual ARM Platform. As Figure 6 shows, when Virtual ARM Simulator decodes an ARM instruction, it checks whether the decoded instruction includes control address of peripheral devices. If it doesn't, Virtual ARM Simulator simply executes the ARM instruction and sends the results to Virtual ARM Platform. However, if it does, Virtual ARM Simulator has to do additional work. Since Virtual ARM Platform employs memory mapped I/O in controlling peripheral devices, a platform-specific data file contains device control register access addresses and corresponding device control addresses. An example of platform specific data file can be found in Table 1. By looking up H/W platform specific data, Virtual ARM Simulator finds which device it should control, and then finds how to control the corresponding devices. Finally, Virtual ARM Simulator processes the device control by sending the saved device control address and data to Virtual ARM Platform.

By using the platform-specific data file, developers are allowed to use the same code to control the peripheral devices in both Virtual ARM Platform and the real H/W platform.

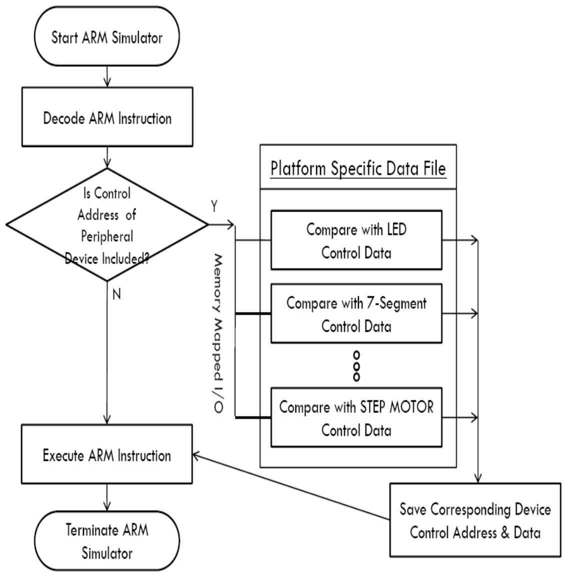

Start ARM Simulator
Decode ARM Instruction
Is Control Address of Peripheral Device Included?
Y
N
Memory Mapped I/O
Platform Specific Data File
Compare with LED Control Data
Compare with 7-Segment Control Data
Compare with STEP MOTOR Control Data
Execute ARM Instruction
Save Corresponding Device Control Address & Data
Terminate ARM Simulator

Figure 6. Execution procedure of Virtual ARM Simulator

Table 1. Sample of the platform specific data file targeted on SYS-Lab 5000 ARM H/W Platform

Access Address	Device Control Register	R/W	Size(Bit)
0x0a00_0004	LCR(LED Control Register)	W	24
0x0a00_0008	SGCR(Segment Control Register)	W	14
0x0a00_000c	DMLCR(Dot-Matrix LED Control Register)	W	4
0x0a00_0014	TLICR(Text-LCD Instruction Control Register)	W	8
0x0a00_0018	TLDCR(Text-LCD Data Control Register)	W	8
0x0a00_001c	KMCR(Key-Matrix Control Register)	R	4
0x0a00_0028	SMCR(Step Motor Control Register)		

STEP_CNT	STEP MOTOR SPEED
0x3d08f	100Hz
0x1e847	200Hz
0x4ffff	400Hz
0xa2c1	600Hz
0x7a11	1kHz

4.2. Graphic user interface

Graphic User Interface is used to handle input events, synchronize the program, and print the simulation results on screen. See Figure 7 for the overall structure of our GUI.

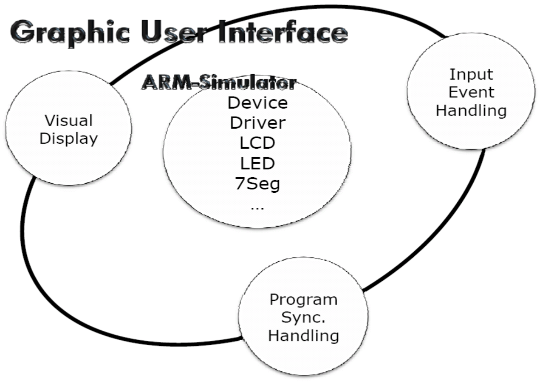
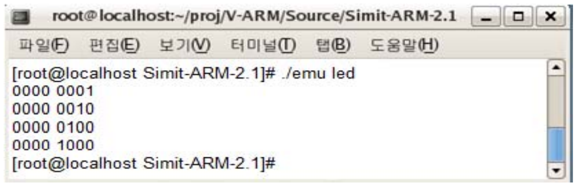


Figure 7. Graphic User Interface and Virtual ARM Simulator


root@localhost:~/proj/V-ARM/Source/Simit-ARM-2.1
파일(F) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)

```
[root@localhost Simit-ARM-2.1]# ./emu led
0000 0001
0000 0010
0000 0100
0000 1000
[root@localhost Simit-ARM-2.1]#
```

Figure 8.1. LED Control in Virtual ARM Platform without GUI

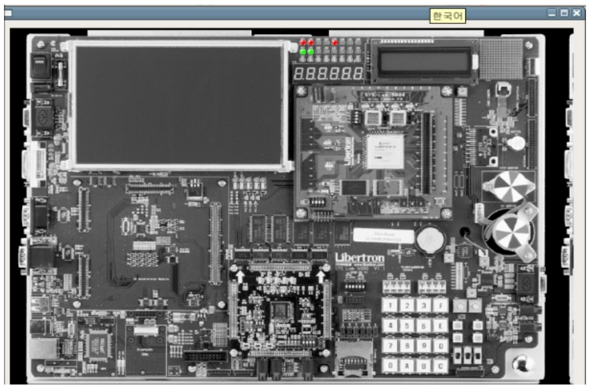


Figure 8.2. LED Control in Virtual ARM Platform with GUI Targeted On SYS-Lab 5000 Platform

GUI looks just like the real target H/W platform. GUI shows the result of peripheral device control by printing the changes in the device visually. Also, Virtual ARM Platform detects external inputs when the developer clicks the external input devices in GUI such as keypad or switch. Figure 8.2 shows an example of implemented GUI, while Figure 8.1 shows an example of Virtual ARM Platform without GUI, when the developer is controlling the peripheral devices.

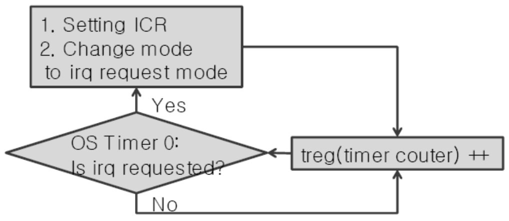

1. Setting ICR
2. Change mode to irq request mode
Yes
OS Timer 0: Is irq requested?
treg(timer couter) ++
No

Figure 9. How OS Timer Works in Virtual ARM Platform

4.3. Timer and I/O devices

We have also implemented a timer called “Timer” inside of Virtual ARM Platform. Timer is initialized when the Virtual ARM Platform starts. By using Timer, developers can test watchdog timer interrupt, context switching and simple OS simulation such as μC/OS-II [6]. Figure 9 shows Timer used as the OS Timer.

Virtual SDRAM enables Virtual ARM Simulator to use it just like an SDRAM module. By using Virtual SDRAM as virtual memory, Virtual ARM Simulator can simulate firmware level code by linking I/O control code section to the startup code just as a real H/W ARM Platform does. Figure 10 shows that the same firmware level code is executed in both our Virtual ARM Platform and the real H/W ARM Platform.

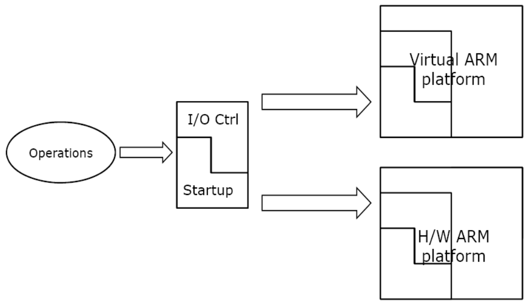


Figure 10. Execution of firmware level code in Virtual ARM Platform and H/W ARM Platform

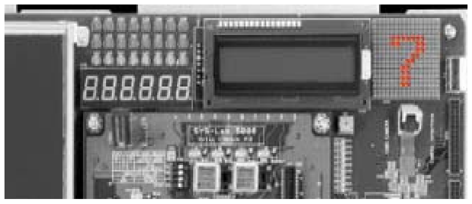


Figure 11.1. Implementation of Dot-matrix

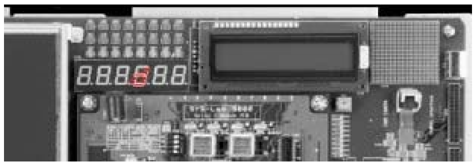


Figure 11.2. Implementation of 7-Segment

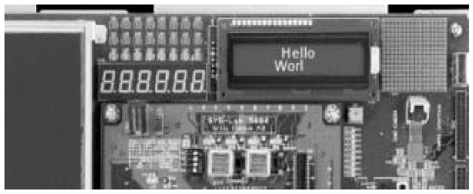


Figure 11.3. Implementation of Text-LCD

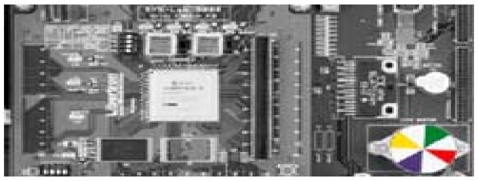


Figure 11.4. Implementation of Motor

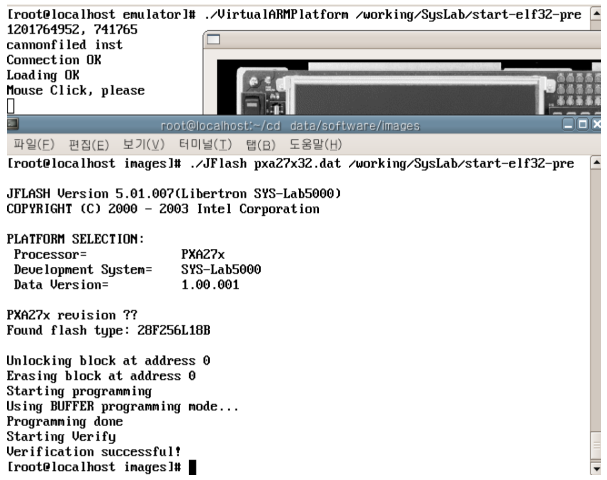
Also we have implemented LED, Dot-Matrix, 7-Segment, Text-LCD, Step Motor, DC Motor, and Keypad module. LED (Figure 8.2), Dot-matrix (Figure 11.1), 7-Segment (Figure 11.2) are implemented to convert given control data to graphical output synchronized with CPU clock time. Text-LCD (Figure 11.3) is implemented to print the given text data on the LCD screen step by step under the control of given text control data. Step Motor and DC Motor (Figure 11.4) are implemented in such a way that they visually spin without being synchronized with the CPU clock but under the control of given speed control data. Keypad is implemented to detect the mouse click per each key button, sending corresponding input data to Virtual ARM Platform.

5. Experimental results

We have carried out experiments to test the performance of our proposed Virtual ARM Platform. We made test application described in Figure 12, which requires the complete control of every device on our embedded board. By using the test application (Figure 13), we could see both the target real H/W platform and our Virtual ARM Platform act exactly same, as Figure 14.1 and Figure 14.2 show the identical execution results.

```
1:  Initialize Text-LCD
2:  Print “Virtual ARM” on Text-LCD
3:  Activate STEP MOTOR to spin
4:  Sequentially Turn ON&OFF LED Lamps by Rows
5:  Get Keypad Input
6:  Shows input data from Keypad through 7-Segment
7:  Shows input data from Keypad through Dot-Matrix
8:  Repeat step 5 to step 7
9:  OS controls task1(step 4) and task2(step 8)
```

Figure 12. Test Application


```
[root@localhost emulator]# ./VirtualARMPlatform /working/SysLab/start-elf32-pre
1201764952, 741765
cannonfiled inst
Connection OK
Loading OK
Mouse Click, please
```
root@localhost:~/cd data/software/images
파일(F) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
```
[root@localhost images]# ./JFlash pxa27x32.dat /working/SysLab/start-elf32-pre

JFLASH Version 5.01.007(Libertron SYS-Lab5000)
COPYRIGHT (C) 2000 - 2003 Intel Corporation

PLATFORM SELECTION:
 Processor=             PXA27x
 Development System=    SYS-Lab5000
 Data Version=          1.00.001

PXA27x revision ??
Found flash type: 28F256L18B

Unlocking block at address 0
Erasing block at address 0
Starting programming
Using BUFFER programming mode...
Programming done
Starting Verify
Verification successful!
[root@localhost images]# █
```

Figure 13. Cosimulation of Test Application in Virtual ARM Platform and SYS-Lab5000

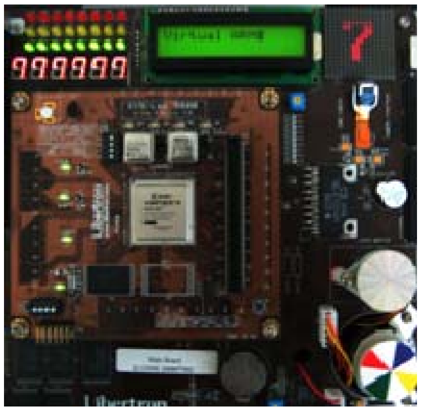


Figure 14.1. Execution Result of Test Application on the real board called SYSLab-5000

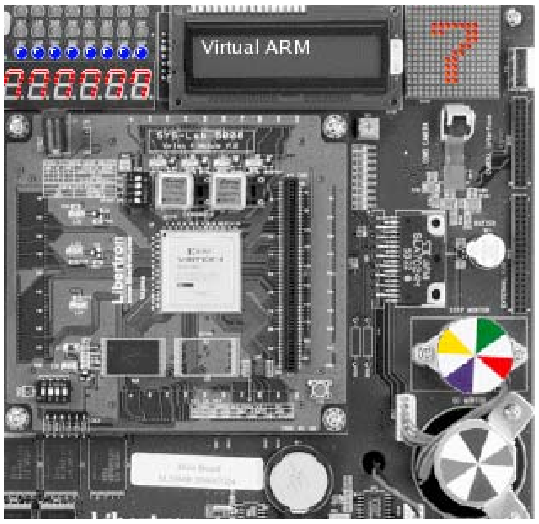


Figure 14.2. Execution Result of Test Application on Virtual ARM Platform targeting SYSLab-5000

6. Conclusion

In this paper, we have proposed Virtual ARM Platform that offers developers an excellent test environment which is very similar to a real H/W ARM platform. By using our proposed Virtual ARM Platform, developers will be able to develop ARM based embedded system software without buying a target H/W platform. Since developers can test their program immediately in host PC without downloading executable files on the real platform, the overall development procedure will be faster and more comfortable. Also, a GUI interface which resembles the target H/W ARM platform is user-friendly, and therefore, will lower the entrance barrier for embedded system novices. Especially, Virtual ARM Platform will be ideal as a college-level education tool for the students who need to practice their knowledge in embedded system software without any time, spatial and financial limits.

7. Acknowledgement

This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute for Information Technology Advancement)" (IITA-2008-C1090-0801-0045)

8. References

[1] Y. S. Li and S. Malik, "Performance analysis of Real-Time Embedded Software," *Kluwer Academic Publishers,* 1999.

[2] A. Krishnaswarmy and R. Gupta, "Profile Guided Selection of ARM and Thumb Instructions," *CM SIGPLAN Joint Conference on Languages Compilers and Tools for Embedded Systems & Software and Compilers for Embedded Systems* (LCTES/SCOPES), Berlin, Germany, June 2002.

[3] http://simit-arm.sourceforge.net

[4] http://trolltech.com/products/qt

[5] Gon Kim, Sang-Young Cho, and Jungbae Lee, "Virtual Prototyping Environment on ARMulator", *Korea Computer Congress 2004 Vol. 2.*, 2004, pp. 592-594.

[6] Jean J. Labrosse, MicroC/OS-II Real Time Kernel 2/E, R&D Technical Books, 2002.

[7] P. Schaumont, D. Ching, I. Verbauwhede, "An interactive codesign environment for domain-specific coprocessors," *ACM Transactions on Design Automation for Embedded Systems*, January 2006.

[8] W. Qin, S. Malik. Flexible and Formal Modeling of Microprocessors with Application to Retargetable Simulation, *Proceedings of 2003 Design Automation and Test in Europe Conference (DATE 03)*, Mar, 2003, pp.556-561.

[9] W. Qin, S. Rajagopalan, S. Malik, A Formal Concurrency Model Based Architecture Description Language for Synthesis of Software Development Tools, *ACM 2004 Conference on Languages, Compilers, and Tools for Embedded Systems*, June 2004, pp. 47-56.

[10] Rajesh Kumar Gupta, "Co-synthesis of Hardware and Software for Digital Embedded systems," PhD thesis, Stanford University, April 1991.