Implementation of a Simple Emulator Platform with Limited Resources

Myoung-Keun You and Gi-Yong Song
School of Electrical and Computer Engineering
Chungbuk National University, Cheongju, 361-763, Korea

***Abstract* - The implementation of a simple hardware emulator platform using limited resources is presented in this paper. The emulator platform consists of an evaluation board based around ARM processor core and a starter kit loaded with an FPGA chip, and uses tools from open-source software. The software part of the emulator is executed by ARM processor after being loaded into SRAM on evaluation board, and hardware part of it is implemented on FPGA chip on starter kit. The data transfer between ARM processor and hardware module on FPGA chip is carried out through the RS-232 serial port of each board. The evaluation board and host PC are connected by serial cable for debugging using arm-elf-gdb and on-board debugger of evaluation board. Starter kit and host PC are connected by JTAG cable for bitstream of hardware module to be programmed into FPGA chip. Although the emulator platform is constructed with limited budget combining elementary evaluation board, starter kit, and tools readily available from open source, it could be adopted to emulate various kinds of moderate size tasks.**

I. Introduction

There is no formal definition of an embedded system, but it is generally considered to be processor core with software and dedicated hardware module designed to solve a specific problem or task. This is in contrast to a general-purpose computer. Embedded systems use a microprocessor, combined with other hardware and software, to solve a specific problem. Memory is also required to store the software. Flash memory is commonly used to hold the software and allow for field upgrades. In addition to processor and memory, embedded systems have hardware functions such as interrupt controllers and UARTs. Embedded system software refers to the OS and application software. The OS provides services to application software, and application software implements the tasks. The steps of the process of embedded system design are generally as follows; the requirements and product specification, system architecture determination, microprocessor selection, design and integration of hardware and software. For the execution of hardware design, logic simulation, simulation acceleration, emulation and hardware prototyping have been used. Logic simulation refers to event–driven simulation and uses HDL to represent the hardware module. Simulation acceleration refers to the mapping of a particular hardware module into hardware platform. Emulation refers to the mapping of an entire design into hardware platform. Hardware prototyping refers to the construction of a custom hardware [1].

The implementation of a simple emulator platform consisting of two boards with limited resources is demonstrated in this paper. The software part of the design is executed by ARM processor on one board, and hardware part is implemented on an FPGA chip on the other board. An evaluation board with ARM processor on it, AT91EB40A [2-3] from Atmel, for embedded system design and the Spartan-3 starter kit [4] on which an FPGA chip, Spartan-3, is mounted are used in this paper. The evaluation board communicates with starter kit through RS-232 serial ports. As a target application for construction and verification of an emulator platform, a 4-stage FIR filter is implemented on FPGA. The operations involving all components such as ARM processor, hardware modules on FPGA, memory and UART are checked to see whether the emulator works correctly.

This paper is organized as follows; section 2 briefly introduces the Atmel AT91EB40A evaluation board, Xilinx Spartan-3 starter kit and UART, section 3 demonstrates the implementation of an emulator platform, section 4 shows the results from experiment to verify the functionality of the emulator, and section 5 discusses the conclusion.

II. Preliminary

A. AT91EB40A

The AT91EB40A evaluation board [2-3] from Atmel loads AT91R40008 microcontroller in which ARM core is mounted. The microcontroller based on von Neumann architecture provides with single address space according to memory-mapped I/O method for memory and all I/O ports.

The ARM needs to have valid boot code at 0x00000000 on power-up, and hence this location needs to be mapped to ROM, at least initially. Almost all ARM variants either have a memory-management unit that can be used to remap RAM to logical address 0x00000000, or some internal chip selection logic that can be used as select lines sometime after initial boot.

The flash memory is divided by the jumper JP1 into two sections; the lower section contains the Atmel bootloader and Angel debugger software and it is write-protected, and the upper section is available for user programs. JP1 is used to boot standard or user programs.

Table 1 shows memory layout of the board at power-on and after being remapped by Angel debugger, and Fig. 1 shows the block diagram of the AT91EB40A evaluation board.

Table 1. Memory layout before and after remap

	Before remap	After remap
SRAM 256K	0x0030_0000 – 0x0033_FFFF	0x0000_0000 – 0x0003_FFFF
Flash 2M	0x0000_0000 – 0x001F_FFFF	0x0100_0000 – 0x011F_FFFF
Peripherals	0xFFC0_0000 – 0xFFFF_FFFF	0xFFC0_0000 – 0xFFFF_FFFF

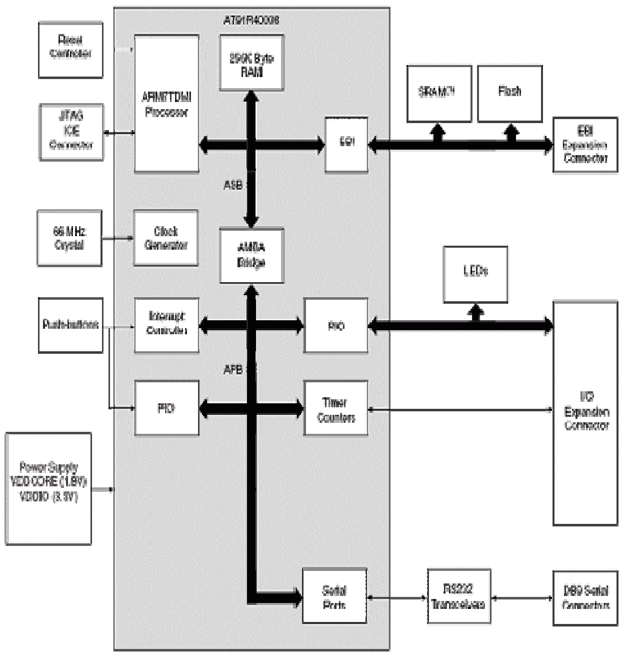

Figure 1. Block diagram of AT91EB40A board
Reprinted from AT91EB40A evaluation board user guide

B. *Spartan-3 Starter Kit Board*

The Spartan-3 starter kit [4] provides a self-contained development platform for designs targeting the Spartan-3 FPGA from Xilinx. It features a 200K gate Spartan-3, on-board I/O devices, and two memory chips, making it the platform to experiment with any new design, from a simple logic circuit to an embedded processor core. Fig. 2 shows the block diagram of Spartan-3 starter kit.

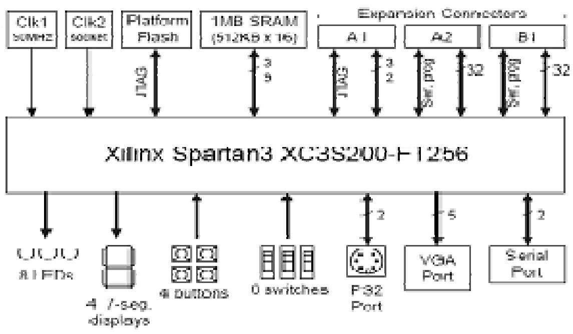

Figure 2. Block diagram of Spartan-3 starter kit
Reprinted from Spartan-3 starter kit board user guide

C. *UART*

The UART(universal asynchronous receiver/transmitter) [4-5] is the key component of the serial communication subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.

When a word is given to the UART for asynchronous transmissions, a bit called start bit is added to the beginning of each word that is to be transmitted. The start bit is used to alert the receiver to which a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. After the start bit, the individual bits of the word of data are sent, with the LSB(least significant bit) being sent first. When the entire data word has been sent, the transmitter may add a parity bit that the transmitter generates. The parity bit may be used by the receiver to perform simple error checking. Then at least one stop bit is sent by the transmitter.

When the receiver has received all of the bits in the data word, it may check for the parity bits, and then the receiver looks for a stop bit. If the stop bit does not appear when it is supposed to, the UART considers the entire word to be garbled and will report a framing error to the host processor when the data word is read.

A UART usually contains the following components;

- a clock generator, usually a multiple of the bit rate to improve sampling in the middle of a bit period
- input and output shift registers
- transmit/receive control
- read/write control logic

III. Implementation of an Emulator Platform

The block diagram of the emulator platform is shown in Fig. 3, and the function of each component of the block diagram is explained below.

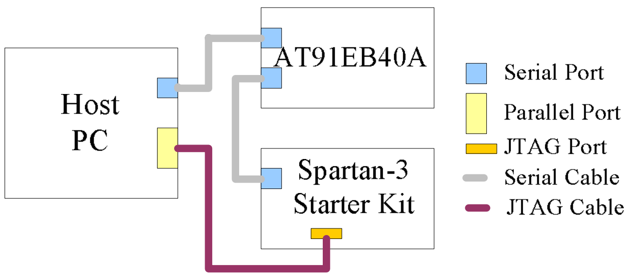

Figure 3. Block diagram of the emulator platform

A. *Host PC*

The host PC is connected to AT91EB40A evaluation board [2-3] through serial ports and to Spartan-3 starter kit [4] through parallel ports. The application program which ARM processor will execute is written in C and then cross-compiled on host PC. While cross-compiling the application program with Makefile, the linker, arm-elf-ld, determines the address layout of the final executable image using linker script file which contains the information on where to load the application program in SRAM on evaluation board. The executable image is transferred to AT91EB40A evaluation board through serial ports by debugging tool, arm-elf-gdb. In addition, the bitstream file is created using Xilinx's ISE after hardware module is designed with HDL, and then programmed into FPGA chip on Spartan-3 starter kit through JTAG cable. The host PC debugger, arm-elf-gdb, communicates with the Angel debugger of AT91EB40A evaluation board through serial ports. The executable image produced from cross-compilation is loaded into SRAM on AT91EB40A evaluation board and executed by arm-elf-gdb commands shown in Fig. 4.

```
arm-elf-gdb Transposed_FIR.elf
set remotebaud 9600
target rdi com1
load
continue
```

Figure 4. arm-elf-gdb commands

B. *AT91EB40A*

For emulation, the Angel debugger on AT91EB40A evaluation board loads the executable image transferred from host PC into SRAM and executes it. Then the results are checked to see whether the emulation is doing correctly. The commands for program execution are transferred from host PC debugger, arm-elf-gdb. During emulation, part of the executable image is transmitted to Spartan-3 starter kit through serial port attached to AMBA(advanced microcontroller bus architecture) APB(advanced peripheral bus) of AT91R40008 microcontroller to drive the hardware module on FPGA. The executable image should be moved to on-board flash memory using flash-loader program if it is to be executed again in the future.

C. *Spartan-3 Starter Kit Board*

Top view of the hardware module implementation on Spartan-3 FPGA chip is shown in Fig. 5.

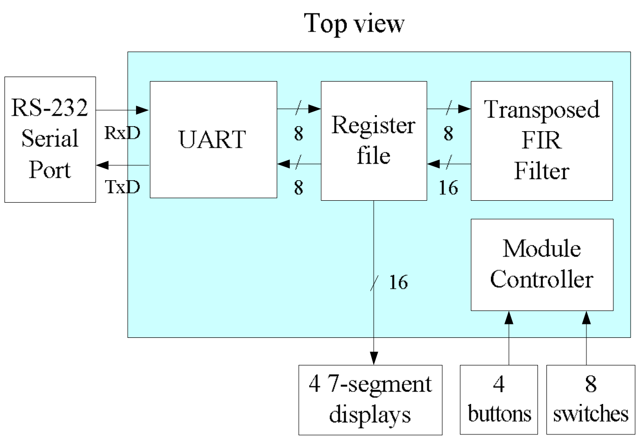

Figure 5. Top view of the hardware module

Top view includes several components; UART, register file, transposed FIR filter and module controller. The FIR filter which processes convolution sum is a typical operation being involved in various applications regarding DSP, so it is selected as a task performed by the hardware module on the emulator platform. When implementing FIR filter, the transposed structure is preferred as the FIR filter with transposed structure does not need either an extra shift register for input sequence or an extra pipeline stage for adder of the products [6]. The block diagram of the FIR filter in transposed structure is shown in Fig. 6.

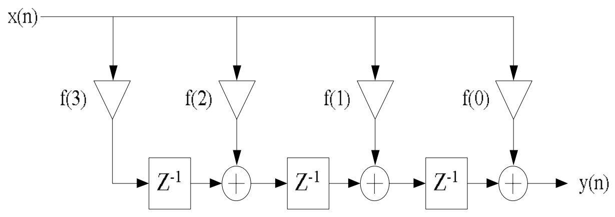

Figure 6. Transposed FIR filter

The register file is for storing input/output sequence of an FIR filter, and accessed in a FIFO manner. The bitstream file is created using Xilinx's ISE [7] after all components of top view are designed with HDL. The bitstream file should refer to the information on number of pins of FPGA chip which are connected to I/O devices of Spartan-3 starter kit such as buttons, switches, 7-segment displays and serial ports. The bitstream file is programmed into Spartan-3 FPGA chip using Xilinx's iMPACT [7] and JTAG cable. The hardware module stores the data transferred from AT91EB40A evaluation board into register file, transfers data from register file to transposed FIR filter, or outputs data stored in register file to four 7-segment displays according to module controller. The module controller which is an FSM(finite state machine) controls data flow inside top view. The state of buttons and switches are fed to module controller as shown in Table 2 where btn and sw are abbreviations of button and switch, respectively.

Table 2. The operation of the module controller

Input of controller	The operation of controller
btn_0	Store the 8-bit output of UART into register 0
btn_1	Store the 8-bit output of UART into register 1
btn_2	Store the 8-bit output of UART into register 2
btn_3	Store the 8-bit output of UART into register 3
sw_7	Reset
sw_6	Start convolution operation
sw_5	Enable 4 7-segment displays
sw_4&sw_3&sw_2	Transfer the value of corresponding register to 7-segment displays

A picture of the emulator platform is shown in Fig. 7.

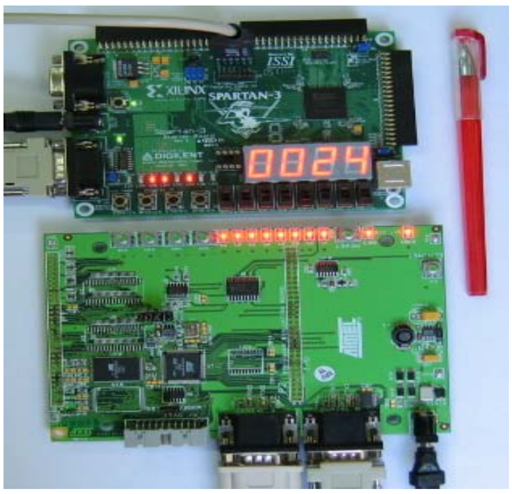

Figure 7. Implementation of the emulator platform

IV. Experiment

A 4-stage transposed FIR filter with 8-bit coefficients 0x01, 0x02, 0x04 and 0x08 is implemented on FPGA chip. When AT91EB40A evaluation board transfers input sequence of four 8-bit data 0x01, 0x02, 0x03 and 0x04 to Spartan-3 starter kit through serial ports, 7-segment display shows output sequence of 0x0024 at 5th and 0x0028 at 6th clock after starting convolution operation. Table 3 shows the implementation result of hardware module on Spartan-3 FPGA. Max delay is from the net with reset.

Table 3. Implementation result of hardware module

Number of External IOBs	35 / 173
Number of Slices	237 / 1920
Max Delay (ns)	4.854

V. Conclusions

A simple emulator platform which uses software tools from open source is implemented with an evaluation board based around ARM processor core and starter kit mounting an FPGA chip. The software part of the emulator is executed by ARM core on the evaluation board and hardware part is implemented on FPGA chip on the starter kit. The connection between host PC and evaluation board, and between evaluation board and starter kit are built through serial ports for each, and the connection between starter kit and host PC is made through parallel ports. Emulation with a test target which includes hardware module for transposed FIR filter was done on the emulator platform, and the results were checked to be successful.

An emulator platform which consists of an inexpensive evaluation board and simple starter kit, and uses tools from open-source software could easily be built using readily available resources with minimum cost. This emulator platform constructed with a limited budget could be adopted to emulate tasks of moderate size.

Acknowledgments

The authors would like to express thanks to XUP(Xilinx University Program) which donated the Spartan-3 starter kit and tools, and open-source software from which the complete sourcecode for software tools are readily available without fees.

References

[1] Jason R. Andrews, *Co-Verification of Hardware and Software for ARM SoC Design*, Elsevier Inc., pp. 3, 14-16, 29-34, 2005

[2] Lewin A.R.W. Edwards, *Embedded System Design on a Shoestring : Achieving High Performance with a Limited Budget*, Elsevier Science, pp. 114-117, 153-157, 2003

[3] Atmel, *AT91EB40A Evaluation Board User Guide*, http://www.atmel.com

[4] Digilent, *Spartan-3 Starter Kit Board User Guide*, http://www.digilentinc.com

[5] Wikipedia, *Universal Asynchronous Receiver/Transmitter*, http://en.wikipedia.org/wiki/UART

[6] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, pp. 79-81, 2001

[7] Xilinx, Inc., http://www.xilinx.com