Research, Development and Application on Information and Communication Technology

CoMoSy: a Flexible System-on-Chip Platform for Embedded Applications

Van-Huan Tran, Xuan-Tu Tran
SIS Laboratory, University of Engineering and Technology, VNU Hanoi
144 Xuan Thuy road, Hanoi 10000, Vietnam. Email: {huantv, tutx}@vnu.edu.vn

Abstract—**Thanks to the rapid evolution of semiconductor technology, System-on-Chip (SoC) paradigm has become one of the most common design methodologies for quickly developing embedded systems to meet the high demands of embedded applications. In this paper, we present the design and implementation of a SoC platform targeted to controlling and monitoring applications. This proposed platform is composed of a 32-bit processor and a dozen of common hardware interfaces, providing the programmability and connectivity to peripheral devices such as memories, LAN network, monitor, keyboard, ADC, DAC, or other I/O devices. In addition, to increase the flexibility of the system and to rapidly develop end-user applications, we also deploy an application-specific software framework with the robustness of a lightweight kernel and real-time applet management. The system model has finally been validated through the realization of a remote control camera system.**

Keywords—**System-on-Chip (SoC), embedded systems, FPGA-based design, microprocessor systems, embedded applications.**

I. Introduction

Nowadays, System-on-Chip (SoC) paradigm has become very common in the design of embedded systems, allowing full software/hardware to be integrated into a single chip. In such embedded applications, SoCs should provide high functional flexibility as well as processing capability [1]. Thanks to the evolution of semiconductor technology, system designers can integrate more and more intellecture properties (IPs) into a system to meet the needs of applications. However, the growth of integration scale also leads to many new challenges in SoC design such as performance, on-chip communication, power consumption, and design cycles. These factors definitely depend on the design principles, design and implementation technologies.

To overcome these challenges, several ASIC platforms of SoCs have been previously proposed in the literatures as well as commercial products. However, these platforms were developed for application-specific systems and they have limitations on flexibility and configurability. Even if we intend to improve those factors, it will make the system more complex. In order to increase the flexibility and reduce the complexity of embedded systems, FPGA technology is obviously selected thanks to its advantages such as scalability, reconfigurability, rapid prototyping, short time-to-market, and low NRE (non-recurrent engineering) cost [2].

In fact, FPGAs have recently become very popular in implementing logic circuits. They can be used for many types of applications such as rapid prototyping platform [3], telecommunications [4], digital signal processing [5]. The flexibility and scalability of FPGAs have made them suitable for implementing embedded SoCs, where a complete system can be implemented on a single programmable chip. In this case, the soft-core processor provided by manufacturers can be reused for developing embedded systems. This soft-core processor is a microprocessor which is fully described in software, usually in an HDL (Hardware Description Language) and can be synthesized and implemented on FPGAs. One of the most advantages of the soft-core processors is that they can be easily customized to adapt the needs of a specific target application.

For those reasons, we have developed an FPGA-based SoC platform, named as CoMoSy (**Co**ntrolling and **Mo**nitoring **Sy**stem), using Xilinx MicroBlaze soft-core processor. This platform is intently designed to be used for a large range of embedded systems in controlling and monitoring applications. With this platform, application mapping processes will be much easier and the production time will be significantly reduced. The proposed platform and its design methodology will be fully described in this paper, including both hardware and software issues.

The remaining part of this paper is organized as follows. Section II presents in detail the hardware architecture model and implementation of the proposed SoC platform. Section III describes the application-specific software framework which is specially developed for the target hardware system. Section IV provides some main experiments and obtained results. Finally, further discussions and conclusions are given in section V.

II. Proposed Hardware Architecture – design and implementation

A. CoMoSy architecture design

As mentioned above, thanks to the flexibility and configurability of FPGA technologies system designers have many choices in architecture design for their embedded systems. However, before mapping an application into a targeted FPGA technology, the first thing should be considered is the partitioning of software and hardware parts. The software part can be built and executed on one or more processor cores if needed while the hardware part is modularized into particular functional blocks. These funcational blocks can be developed from the existing IP (Intellectual Property) cores provided by FPGA suppliers or fully developed by designers. In addition, an application platform may have more than one system configuration file (i.e. FPGA bitstream file and software execution file), allocated in different memory sections (using a ROM or a flash memory). They can be selected to configure/reconfigure the FPGA device at power-on/run-time for specific purposes. A better hardware/software partitioning will produce a higher efficiency for the targeted embedded system.

The second thing is to select a proper FPGA device in order to fit the design. It means that the selected FPGA device does not only provide enough logic cells for the design but also has to meet the other requirements of the design such as performance and processing capability. For example, some Xilinx FPGA series can provide either MicroBlaze soft-core processor or PowerPC hard-core processor [6]. In this case, a high-performance application might consider using PowerPC processor rather than MicroBlaze processor because the hard-core processor is obviously better in performance than the soft-core processor. Otherwise, MicroBlaze soft-core processor is more suitable for the applications in which a higher level of flexibility and configurability is needed.

To develop embedded systems for controlling and monitoring applications using Xilinx FPGA, we have proposed a general architecture as described in Figure 1. As requested by targeted applications, this platform is composed of three kinds of functional units: processing units, data acquisition units, and data visualization units. The data acquisition units get data from either digital devices or analog devices (e.g. a set of sensors) by providing standard hardware input interfaces to have connections with those devices. The MicroBlaze processor manipulates and processes the acquired data, gives out decisions to control the periperal devices. In addition, some processed results are transferred to the data visualization units, where the data can be represented in many ways (text, graphic, light or sound) to build friendly human-machine interfaces.

In general, CoMoSy is a 32-bit bus-based platform using a MicroBlaze soft-core processor [7] as the central processing unit. MicroBlaze is a RISC (Reduced Instruc-

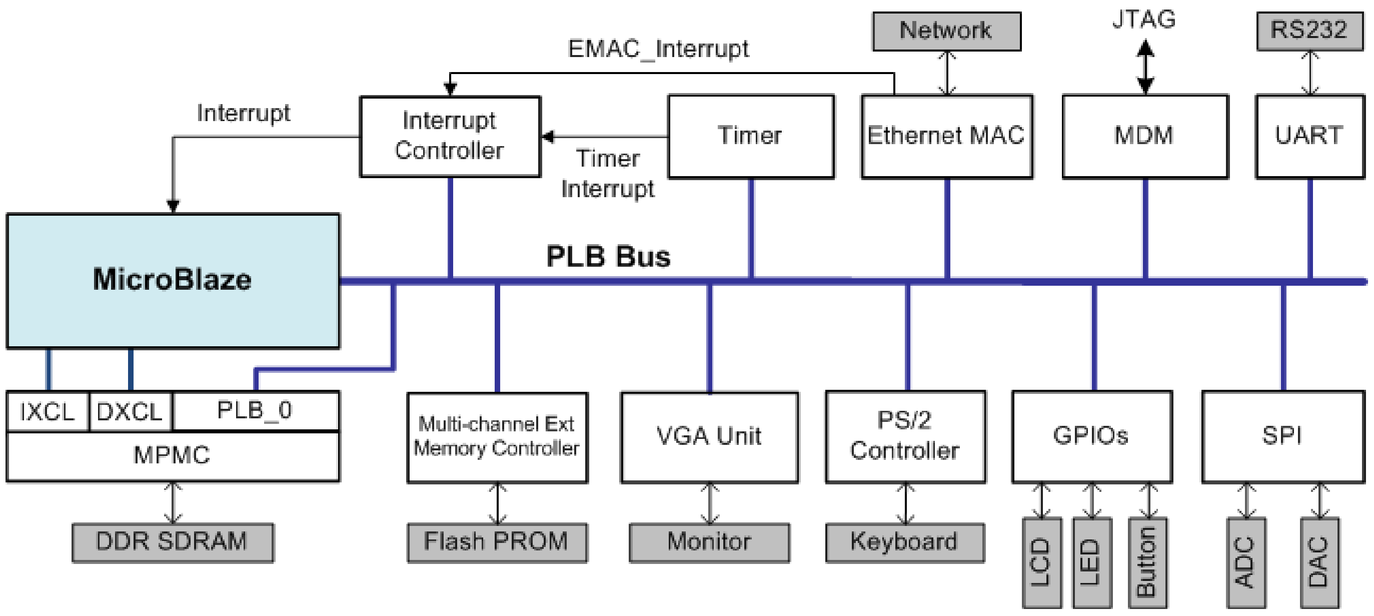


Figure 1. General Architecture of CoMoSy platform.

tion Set Computer) processor that modelled in HDL to be implemented on Xilinx FPGAs. Some optional function blocks integrated in this core are enabled to accelerate the processing ability such as Floating Point Unit (FPU), integer divider, integer multiplier, barrel shifter. This processor core interacts with other IP cores through IBM CoreConnect PLB bus system [8]. In addition to the processor, we have developed a dozen of IP cores. Some Xilinx IP cores have been reused to reduce design time and the others have been completely developed at the laboratory. The rest modules of the platform can be described as follows.

- MPMC core is a multi-port memory controller used to interface with DDR SDRAM. The MPMC unit provides three channels for independently accessing data on DDR SDRAM, where two of them serve as data cache channel and instruction cache channel of MicroBlaze processor, denoted as DXCL and IXCL respectively. The remaining channel (PLB_0) provides a fast channel to transfer program data and instructions when executing. MPMC unit usually operates at a higher frequency than the processor core.
- Multi-channel external memory controller unit is another memory controller used to connect the system with a flash memory. This memory is intended to store the FPGA bitstream files and/or software execution files. It sometimes can be used as a file system device.
- MDM (MicroBlaze Debug Module) unit is a hardware debug module used to debug the operations of MicroBlaze processor by using JTAG (Joint Test Action Group) interface.
- Timer unit is a 32-bit programmable interval timer which is required by almost software kernels for establishing software timers and task schedulers.
- Interrupt controller core handles the interrupt signals arising from peripheral blocks and notifies the processor core by driving processor's interrupt signal. In the proposed platform, there are two interrupt sources, timer interrupt and Ethernet MAC interrupt.
- Ethernet MAC unit is a $10/100Mbps$ Ethernet Media Access Controller. This controller is compatible

with IEEE Std. 802.3 specification.

- UART, SPI, PS/2 units provide serial data links to connect with several devices such as RS-232 ports, analog-to-digital converters (ADCs), digital-to-analog converters (DACs), PS/2 keyboard, PS/2 mouse, etc.
- GPIO units provide bi-directional digital ports for discretely connecting to usual devices such as LEDs, LCDs, switches, buttons, etc. The port-width of each GPIO unit can be customized by a parameter of the core.
- VGA unit is a special core which is completely designed at our laboratory in order to provide a friendly interface to VGA-compatible monitors. Although this unit is individually developed at the laboratory but it can be used for next FPGA-based system developments thanks to its compatibility to Xilinx FPGA design flow. The design of this unit will be described in detail in the next sub-section as a case study.

In the proposed platform, a master core (e.g. processor) accesses to slave cores using memory-mapped I/O methods. The address bus is shared for both memory controllers and the other slave devices. Thus, each unit in the system has its own address space determined by two parameters: base address and high address. The base addresses are automatically generated by Xilinx design tool while the high addresses are calculated from the memory/address size of IP cores and their corresponding base addresses.

As a simple architecture, the hardware model uses only one bus channel for interconnecting processing components (i.e. IPs) on the system. A new component can be easily integrated into the system thanks to the help of the design tools. Of course, the new component has to be compatible with PLB bus interface. However, the growth of the hardware architecture may affect the response ability of the bus as well as the reliability of the system. In this case, system designers should consider creating several bus channels, for example, one channel for low speed components and another for high-speed components. Hence, the system is really flexible and scalable on the change of the hardware architecture model.

B. VGA unit

To build a friendly human-machine interface for the platform, in addition to simple text-based LCDs we have developed a VGA monitor controller unit (called VGA unit) to increase the efficiency in displaying text as well as graphics. The architecture of this VGA unit is designed as described in Figure 2 and modelled using VHDL language. Then, it is implemented using Xilinx design flow [9] in order to make the design compatible to Xilinx design methodology, and therefore it can be used for the next designs. The VGA unit will display an image on the screen by driving the color signals and synchronization signals of the VGA interface according to the pixel data of the image.

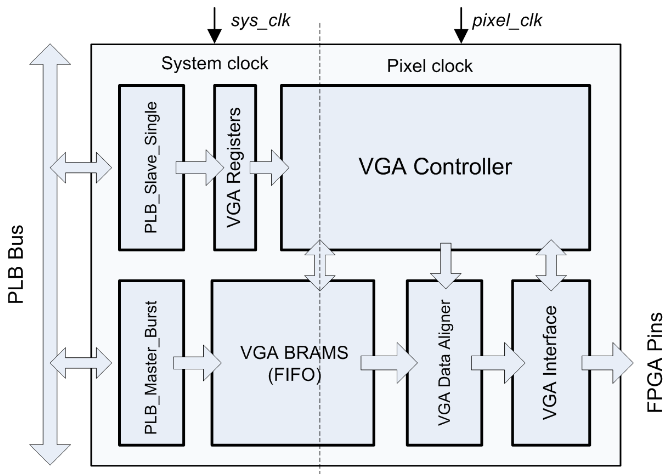


Figure 2. Proposed architecture of VGA unit.

The size of an image is usually larger than memory resources available on FPGAs, for example, a 640×480 pixels at $24bpp$ raw image has a size of $912.6Kbytes$ while the Spartan-3E family has a limit of $81Kbytes$ block RAM [10], the image therefore cannot be entirely stored inside the VGA unit. The image should be held on an off-chip memory (e.g. SDRAM) and transferred into this VGA unit by small data blocks during the display time.

There are two solutions that could be addressed to handle the data transfer operations. The first solution is to attach a bus master interface to the VGA unit to be able to access data on the external memory. The second one is to use a Direct Memory Access (DMA) core to co-operate with the VGA unit. In the second case, VGA unit does not need to include any bus master interface, it receives the data transmitted from DMA core. However, the processor has to initiate the operation of the DMA core whenever the VGA unit wants to refresh the screen. For that reason, we prefer the first solution to gain the flexibility and independency of the VGA unit.

One of the most challenges in designing VGA unit is to determine the depth of memory buffer within the core and the buffering strategy because these parameters will affect the cost-efficiency and display quality of the system on a specific display resolution. In this work, we have taken into account these constraints for VGA standard. In consequence, the architecture of VGA unit (presented in Figure 2) is composed of:

- PLB Master interface with burst transfer mode is usually used for transferring data from the external memory to FIFO block.
- PLB Slave interface serves the reading/writing operations between the processor and VGA unit.
- Three 32-bit registers contain the command data and the properties of the images such as width, height, base address.
- VGA BRAMS/FIFO is a $256 bytes$ dual-port memory buffer for storing a bulk of data.
- VGA Data Aligner synchronizes data between FIFO block and VGA Interface block.
- VGA Interface drives the synchronization signals of the monitor.
- VGA Controller is reponsible for VGA timing and the operations of the whole VGA unit.

There are two clock domains in this design: system clock (*sys_clk*) and pixel clock (*pix_clk*). The pixel clock absolutely depends on the display resolution and the refresh rate of the screen specified by standards (see Table I). Even though two clock signals are independent, the system clock must be higher than the pixel clock to ensure that the appearance is smooth (without flickering) and the overhead on bus traffic is eliminated.

Table I
DISPLAY STANDARD SPECIFICATION

Display standard	**Pixel frequency** (MHz)
VGA 640×480 @$60Hz$	25.175
SVGA 800×600 @$60Hz$	40.000
XGA 1024×768 @$60Hz$	65.000
WXGA 1280×800 @$60Hz$	83.460

The VGA unit is then modelled in VHDL language and simulated by using ModelSim simulator (Mentor Graphics). The synthesis results on Spartan-3E device show that the VGA unit can operate at a speed up to $160MHz$. The unit is thus possible to provide higher resolutions than VGA standard such as XGA or WXGA.

C. Implementation

In this section, we present the implementation of the proposed platform on Xilinx Spartan-3E device to evaluate some typical metrics of the system. From the system specifications described above, the design and implementation can be completely managed by Xilinx Platform Studio environment [9]. All required peripheral hardware blocks are integrated into the system and they are configured to a proper operation mode. Then, we let the tool automatically perform synthesizing and implementing processes with provided input constraints. Finally, the design is verified again in order to check whether it meets the system specifications, timing and power consumption issues before being loaded into FPGA devices.

Table II shows the resource utilization on Spartan-3E XC3S500E device, distributed on the logic cells (including slides and look-up-tables); Input/Output pads (I/Os); primary RAM blocks (BRAMs); Digital Clock Manager blocks (DCMs), and hard-macro multipliers (MULTs). Obviously, the whole design is fit on Spartan-3E XC3S500E, most logic elements and BRAMs are used (more than 80% of available resources). The design also meets the timing constraints to operate at $50MHz$.

Table II
RESOURCE UTILIZATION (SPARTAN-3E-XC3S500E)

Name	**Available**	**Used**	**Utilization**
Logic	9312	7616	82%
I/Os	232	86	37%
BRAMs	20	19	95%
DCMs	4	2	50%
MULTs	20	7	35%

In addition to hardware overhead, power consumption is another important parameter in embedded system design. The power consumption of our system is estimated using XPower Analyzer tool. Table III summarizes two metrics of power consumption, total quiescent power and total dynamic power. Because we implement our design using FPGA technologies, the only concerned parameter is the dynamic power of the design. Most of the quiescent power depends on the target technologies and we cannot change.

The distribution of power consumption of the design is also shown in Figure 3. The power consumed on I/Os is the most considerable (80% of the whole system power consumption) because the I/Os usually require higher current than the other entities.

Table III
POWER CONSUMPTION

Type	**Power** (mV)
Total quiescent power	105
Total dynamic power	643
Total consumed power	748

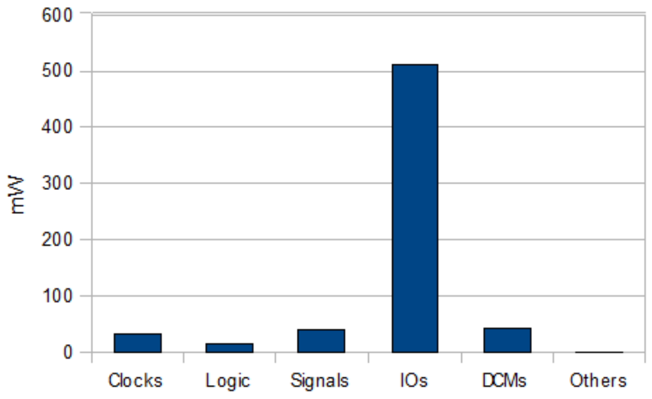


Figure 3. Power consumption distribution.

III. SOFTWARE SYSTEM

In order to adapt to the change of the hardware architecture, we develop a scalable and reliable software framework for the proposed hardware. The main purpose of this framework is to provide mechanisms for managing hardware resources and user applets. To do this, we has adopted several open software libraries, modified and integrated them to build the software framework as presented in Figure 4. In this software framework, the most important part is the kernel based on Xilkernel [11]. This kernel can support the most common POSIX APIs (Portable Operating System Interface – Application Programming Interface) [12] such as thread management, thread synchronization services, and thread communication services.

In this software framework, the lowest layer is driver layer. This layer is composed of Xilinx IP drivers (used for Xilinx IP cores) and VGA driver (used for our hardware design of VGA unit). Upon the drivers is the

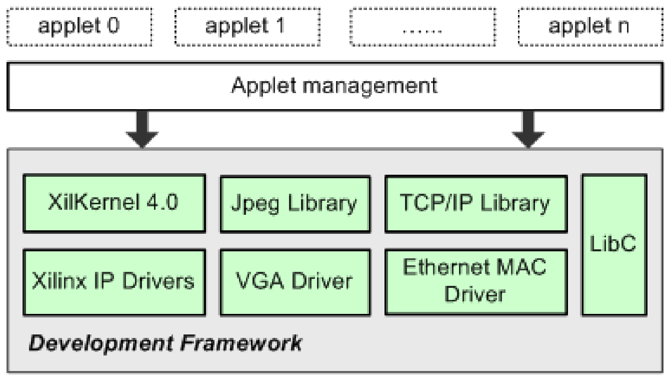


Figure 4. Software development framework.

kernel and library layer. In this layer, beside the kernel we integrate several popular libraries such as standard C library, JPEG decoder library [13], TCP/IP protocol stack library [14] to help software designers to quickly build their applications. In higher layers, we develop an applets management layer to handle all user applets. Each applet is a small application performing a specific function, described by the following object:

```
 typedef struct s_req_handler {
   /* applet/request ID */
   unsigned int m_req_id;
   /* callback function */
   void (*m_callfunc)(void);
} req_handler_t;
```

In addition, to help the users to easily interact with the system, the applets management layer is designed to be able to create a command-line interface where users can type a command to execute the corresponding applets. The applets management layer has been built with three threads: GPIO handler thread, keyboard handler thread, and request handler thread (see Figure 5). The GPIO handler thread performs monitoring states of digital input ports and maps each state to a specific request ID (identification). Similarly, the keyboard handler thread receives user commands from keyboard and maps each of them to a specific request ID. These request IDs are then sent to the request handler thread to specify which applet would be invoked. Thanks to this software model, software designers are able to quickly develop end-user software applications and embed them to the rest of system without considering the reliability of the software system.

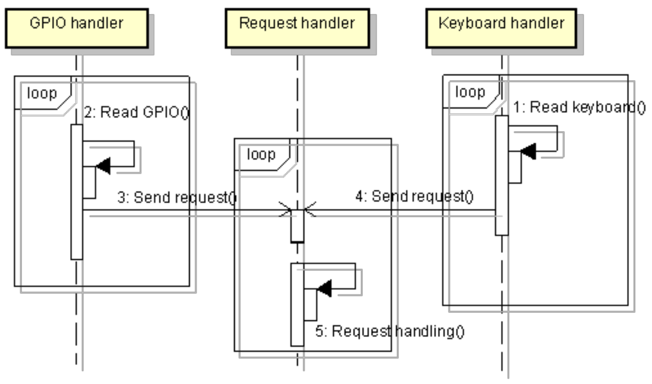


Figure 5. Sequence diagram of the applet management.

IV. Experiments

Verification and validation are two key processes in system design flow and take much time in the design cycle. The using existing resources (including IP cores and software libraries) provided by EDA tools and open source community not only helps system designers save time but also improves the stability and reliability of targeted systems. At a higher level, we present an experiment to verify hardware/software models and their corporations in the platform by developing a remote camera application. In this context, we implement the entire CoMoSy platform on Spartan-3E Starter Kit [15]. This board is based on Spartan-3E FPGA devices with all necessary peripherals for the application such as $512 Mbits$ SDRAM, $128 Mbits$ PROM, Ethernet PHY controller, LCD, LEDs, buttons, etc. PS/2 port and VGA port are also connected to a PS/2 keyboard and monitor as standard input/output devices.

A desktop PC running Linux operating system is used to establish communication with the FPGA board. A camera is connected to this PC via USB interfaces.

Transferring data between PC and the testing board is done on a LAN network thanks to the integrated Ethernet MAC controller as mentionned above. Figure 6 presents the sequence diagram for the remote camera application.

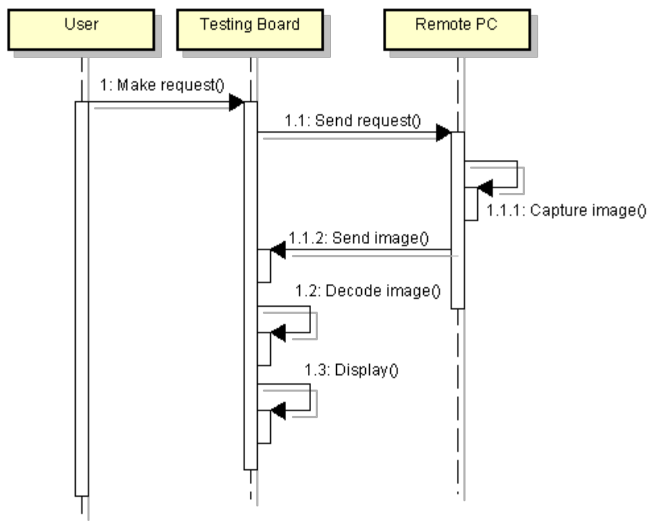
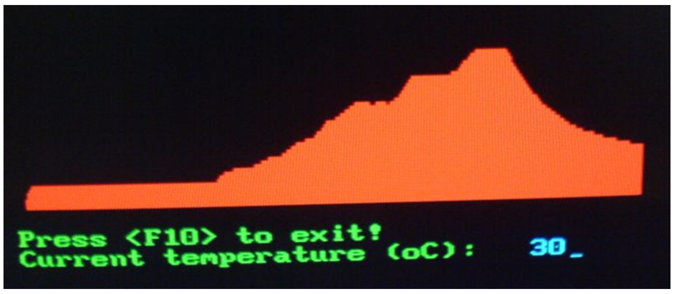


Figure 6. Sequence diagram for the remote camera application.

The scenario is that the users first make a capturing request on the testing board, and then the testing board sends this request to the remote PC through LAN network. Whenever PC receives a proper request, it captures an image from camera, encodes the image and then sends it back to the testing board. After completely receiving the image, the testing board will decode and display the decoded image on the screen. The system provides several ways for the users to invoke the capturing request, either typing a console command or pressing a key binding or hitting a button on the board. So that, it makes the system more flexible in interacting with the users.

Such system requires two application softwares to handle all actions on PC and on the testing board. The embedded application running on the testing board is built with the APIs provided by the software system. The PC application software is written in C++ language using the built-in classes of Qt Designer [16] and Video2Linux library [17]. Inter-process communication between two applications is based on socket mechanism. Figure 7 presents the results of our experiment, the captured image is shown on both PC application and the testing board (in 24-bit color mode) with Xilinx Virtex-4 FPGA development kit.

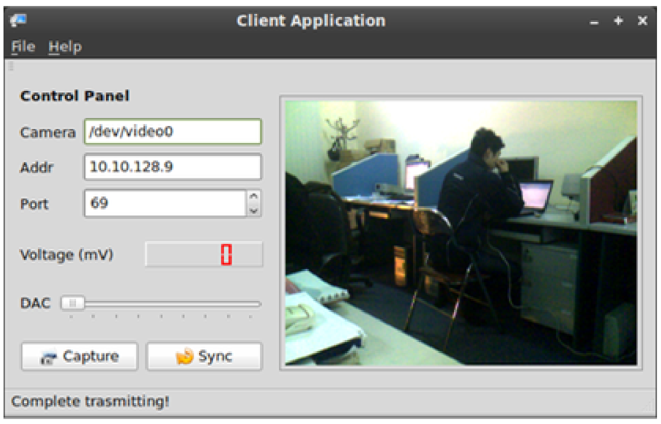
On the same testing model, we also make other experiments to validate the operation of analog-to-digital converter (ADC), digital-to-analog converter (DAC) via a shared SPI interface. One of these applications is the temperature monitoring application. In this application, a temperature sensor is connected to an ADC. This ADC is interfaced with the system by using SPI unit. The temperature variation is displayed on the testing board as presented in Figure 8.



```
Press <F10> to exit!
Current temperature (oC):   30_
```

Figure 8. Temperature monitoring application.

Due to the logical resources of Spartan-3E device, the performance of CoMoSy achieved on this board is limited, it may take a few seconds to completely decode a JPEG image with the frame size of 320 × 240 pixels. By implementing on high-performance devices or higher integration density devices (Virtex-II or Virtex-4 for example), the system performance can be significantly improved through optimization processes and the reconfigurability of MicroBlaze processor. Nevertheless, these experiments have approved the functionalities as well as the applicability of CoMoSy platform for controlling and monitoring applications.



Client Application

File Help

Control Panel

Camera /dev/video0

Addr 10.10.128.9

Port 69

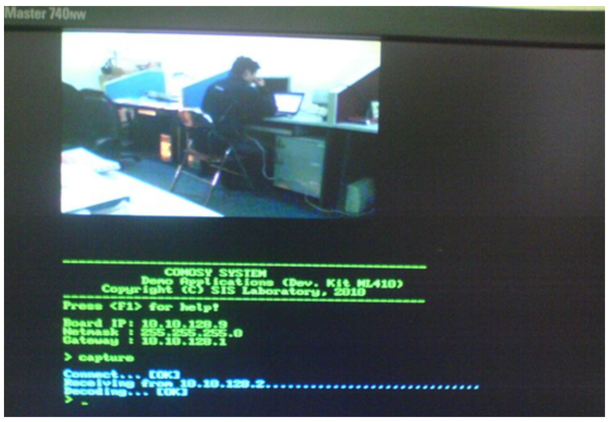
Voltage (mV) 0

DAC

Capture Sync

Complete trasmitting!

(a)



```
                    COMOSY SYSTEM
          Demo Applications (Dev. Kit ML410)
      Copyright (C) SIS Laboratory, 2010
Press <F1> for help!
Board IP: 10.10.128.9
Netmask : 255.255.255.0
Gateway : 10.10.128.1
> capture
Connect... [OK]
Receiving from 10.10.128.2..............................
Decoding... [OK]
> _
```

(b)

Figure 7. Validation results: (a) Client application view on PC side with captured image and other parameters; (b) Embedded application view on CoMoSy side with decoded image.

V. Conclusion

In this paper, we presented the design and implementation of a flexible System-on-Chip platform for controlling and monitoring applications using Xilinx FPGA technology, both software and hardware issues. The proposed platform uses a 32-bit MicroBlaze soft-core processor and PLB bus as the base system. We have integrated several hardware uinits/interfaces to provide functional capabilities for a wide range of embedded applications, including GPIOs, SPIs, UART, PS/2, DDR SDRAM, Ethernet MAC, and VGA unit. In particular, the VGA unit is an in-house design but it is compatible with Xilinx design flow and therefore can be used for other designs. This VGA unit supports multi-resolution displays. The software system is intently developed from existing resources, including a lightweight kernel, IP core drivers, TCP/IP protocol stack library, JPEG decoder library to let users build their software applications quickly at high abstraction level.

Acknowledgment

This work is partly supported by Vietnam National University, Hanoi (VNU) under the research project No. PUF.08.06.

References

[1] Yen-Kuang Chen and Sun-Yuan Kung. Trend and Challenge on System-on-a-Chip Designs. *Journal of Signal Processing Systems*, 53(1-2):217–229, 2008.

[2] Greg Martin. Platform ASICs versus FPGA. In *LSI logic Corporation*, September 2005.

[3] Dominik Langen, Jorg-christian Niemann, Mario Porrmann, Heiko Kalte, and Ulrich Ruckert. Implementation of a RISC Processor Core for SoC Designs - FPGA Prototype vs. ASIC Implementation. In *Proceedings of the IEEE-Workshop: Heterogeneous reconfigurable Systems on Chip (SoC)*, Hamburg, Germany, 2002.

[4] P. Murphy, F. Lou, A. Sabharwal, and J. Patrick Frantz. An FPGA Based Rapid Prototyping Platform for MIMO Systems. In *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*, pages 900–904, Pacific Grove, CA, November 2003.

[5] Chi-jui Chou, Satish Mohanakrishnan, and Joseph B. Evans. FPGA Implementation of Digital Filters. In *Proceedings of the 4th International Conference on Signal Processing Applications and Technology*, pages 80–88, 1993.

[6] Xilinx Corporation. Virtex-4 Family Overview: Data Sheet. *User Guide*, 2007.

[7] Xilinx Corporation. MicroBlaze Processor Reference Guide. *User Guide*, August 2008.

[8] Xilinx Corporation. PLBV46 Interface Simplifications (v1.0). *User Guide*, October 2007.

[9] Xilinx Corporation. Embedded System Tools: Reference Guide. *User Guide*, June 2009.

[10] Xilinx Corporation. Spartan-3E FPGA Family: Data Sheet. *User Guide*, August 2009.

[11] Xilinx Corporation. OS and libraries document and collection. *User Guide*, June 2009.
[12] IEEE. POSIX: Portable Operating System Interface. *Standard Specification*, 2006.
[13] Luc Saillard. Tiny Jpeg decoder. *Technical report*, 2007.
[14] Adam Dunkels. Design and implementation of the lwIP TCP/IP stack. *Technical report*, February 2001.
[15] Xilinx Corporation. Spartan-3E Start Kit Board User Guide. *User Guide*, 2009.
[16] Nokia. Qt – Cross-platform application and UI framework. *Technical report*, 2010.
[17] Gerd Knorr. Video4Linux. *Technical report*.
[18] SECONS Ltd. VGA Signal Timing. *Technical report*, 2008.

Van-Huan Tran was born in Vietnam, in 1985. He received the Bachelor of Science in Electronics and Telecommunication from University of Engineering and Technology, Vietnam National University, Hanoi in 2007. He is currently working as a researcher of the key laboratory for Smart Integrated Systems – VLSI Systems Design group, University of Engineering and Technology, VNU Hanoi.

His research interests include System-on-Chip design and verification, FPGA-based design, embedded systems, VLSI systems/circuits design for multimedia application.

Xuan-Tu Tran was born in Nghe An, Vietnam, in 1977. He received a B.Sc. degree in 1999 from Hanoi University of Science and a M.Sc. degree in 2003 from Vietnam National University, Hanoi, all in Electronics Engineering and Communications; and a Ph.D. degree in 2008 from the CEA-LETI, MINATEC (in collaboration with Grenoble INP), France in Micro Nano Electronics.

Xuan-Tu Tran has worked as a lecturer at Vietnam National University, Hanoi (1999-2003), as a research engineer at the CEA-LETI, MINATEC, France (2003-2008), and then as an assistant professor at the University of Engineering and Technology (UET), VNU Hanoi (2008-recent). He was a visiting/invited professor at the University Paris-Sud 11, France (2009, 2010), visiting professor at Grenoble INP in 2011. He is currently deputy director of the key laboratory for Smart Integrated Systems and head of VLSI Systems Design group. He is in charge for CoMoSy, VENGME projects for embedded systems and multimedia applications. His research interests include design and test of systems-on-chips, networks-on-chips, design-for-testability, asynchronous/synchronous VLSI design, and hardware architecture for multimedia applications.

He is a member of the IEEE, IEEE CAS, and the Executive Board of the Radio Electronics Association of Vietnam (REV).