Rapid-Prototyping Emulation System Co-emulation Modelling Interface for SystemC Real-Time Emulation

David Carroll
School of Informatics and Engineering,
Institute of Technology Blanchardstown, Dublin 15.
IRELAND

Abstract: - This paper describes the Communications Interface Protocol that was implemented successfully as a co-emulation modelling interface between SystemC model and a reconfigurable hardware platform. The information presented represents part of research into the suitability of using SystemC, in conjunction with a suitable reconfigurable hardware system platform, to provide a real-time emulation environment allowing SystemC simulations to be directly translated into real-time solutions for rapid prototyping of embedded systems.

Keywords: – emulation, systemc, reconfigurable, prototyping, co-verification, real-time.

1 Introduction

The development of architectures and platforms suitable for the emulation of systems described with system level description languages, such as SystemC[1][2][4], is an active research area. To date research is largely targeted towards co-verification platforms, focused verification of a design rather than real-time emulation. Nevertheless there is a significant correspondence in the research required to develop both fast verification systems and real-time systems[7][8][12][13].

This paper describes the SystemC communications interface protocol that was implemented successfully in a Digital Camera focus application. A co-emulation modelling interface (known as a Transactor), was developed as part of research[14] into the suitability of using SystemC, in conjunction with a suitable reconfigurable hardware system platform, to provide a real-time emulation environment. Thus allowing SystemC simulations to be directly translated into real-time solutions for rapid prototyping of embedded systems[5].

The consequent Rapid Prototyping Co Emulation System Platform (PESP) is suitable for direct real-time implementation of SystemC based models, when the SystemC based design is interfaced with a reconfigurable hardware system platform, using the Transactors described in this paper. The reconfigurable hardware platform is based on an FPGA development platform oriented towards consumer level multimedia applications and provides a reconfigurable hardware real-time emulation environment for prototyping real-time application designs. The Hardware subsystem is designed around a Xilinx MicroBlaze and Multimedia Development Board (MMDB)[10], incorporating the IBM On-chip Peripheral Bus (OPB) bus architecture implementation, on a Xilinx Virtex II FPGA[9] based platform. Hardware functional blocks developed as part of the system are written as VHDL modules and are implemented in the FPGA fabric using Xilinx ISE and EDK software tools.

This paper presents an overview description of the PESP platform, followed by a brief description of the SystemC controller model. The Transactors that make up the co-emulation modelling interface is then described in detail.
The Transactors exist in both the SystemC portion of PESP and in the Hardware platform. As such the Transactor description is divided into five sub-sections, starting in the SystemC section of PESP and ending on the hardware platform, implemented through the Transactor Communications Manager. The final two sections of the paper include a brief statement regarding the Example Application used to demonstrate PESP and the paper conclusion.

2 PESP Platform Overview

The PESP system, shown in Fig.1, consists of three main elements, the SystemC Controller model, Hardware Components and a Transactor interface. The SystemC Controller model, is implemented in the Software Environment and controls the activities of the system. The Hardware Components are implemented in the Hardware Environment and implement the system functions. The third element in Fig.1, the Transactor, communicates with SystemC Controller Model at the transaction level and communicates with the Hardware Components at the Bit-signal level. The Transactor is employed to facilitate communications between the SystemC Controller Model and the Hardware Components.

![Fig.1: PESP Platform main elements](image)

The hardware aspects of PESP, shown in the block diagram in Fig.2, are implemented on a FPGA based hardware platform to provide greater flexibility during the design process, by facilitating the implementation of different hardware configurations. The Xilinx MMDB was chosen to provide the reconfigurable hardware, as it provides a range of multimedia oriented features (such as video and audio codecs) combined with the facility for implementing additional hardware functionality on the Virtex-II XC2V2000 FPGA device fabric.

![Fig.2: PESP block diagram](image)

3 PESP SYSTEMC Controller Model

The PESP SystemC Controller is a SystemC model designed using a modular design approach which facilitates the division of overall functionality into manageable pieces, aiding the verification testing, sub-model debug, simplifying movement of functionality during design exploration and simplifying addition/removal of sub-models to/from the overall system. During SystemC simulations, the PESP Controller model is responsible for the user interface to the PESP system, and for the generation and control of commands, which are then implemented in the hardware components, in real-time, through the Transactors. The SystemC Controller model consists of several sub-models (shown in Fig.3), all implemented using THREAD processes[11]. A single inter-model interface type (SC_FIFO<> ) was used through the Controller model for all inter-model connections.
4 PESP Transactors

The multi-layered architecture of the SystemC Controller model is shown in Fig.3. User commands and system function control and monitoring is carried out by means of communication with the Driver Layer. The Driver Layer consists of the SystemC Message Port models, the Message Channels and Transactors. The Transactors interface directly with the Hardware Components and are implemented using a combination of hardware and software, namely software functionality running on the MicroBlaze processor with the hardware functionality built on top of the OPB.

The communications interface between the SystemC model and the re-configurable hardware is modelled on the Standard Co-Emulation API: Modelling Interface (SCE-MI)[3] infrastructure and presented in [14]. The SCE-MI specification describes a modelling interface based on a multi-channel abstract bridge, providing multiple communication channels that allow software models describing system behaviour to connect to structural models implemented on hardware emulation systems.

4.1 Transactor Message Timing

The Transactor is a form of abstract gasket, which communicates at the transactional level (e.g. READ and WRITE instructions) with the SystemC Controller model, decomposing untimed messages into a series of cycle-accurate clocked signals. These clocked signals form the communication interface between the Transactor and the hardware, on the hardware side of the system model. Similarly the hardware communicates with the Transactor at the signal level, where the cycle accurate signals are recomposed into transactional level messages, for transfer to the SystemC Controller model interface.

Once a message has been sent to the Transactor, the SystemC Controller sub-model processes are all suspended, although simulation time continues to run, until status information is received back to the Message Port model from the Transactor (Fig.3). Process suspension is achieved by using blocking read requests to the STATUS ports of the sub-models, which are connected by FIFO channel interface types. A blocking read request to an empty port suspends the THREAD process requesting the read, until
such time as there is data in the interface channel registers connected to the that port. Thus the SystemC Controller can request only one function for execution at a time. After the request, the SystemC Controller processes are suspended until execution status information is returned from the hardware platform, through the Transactor, after which the SystemC Controller can request execution of another function.

4.2 Transactor Implementation

The Transactors are implemented across a combination of SystemC models (on the host PC) and using a combination of hardware and software on the MMBD hardware platform. The Transactor implementation overview block diagram in Fig.4 shows the hardware implementation across the two platforms, while Fig.5 shows the Transactor MicroBlaze platform implementation. In the SystemC Controller model, the Port Interface and the Port models form part of the Transactor that is resident on the host PC. The Port model is responsible for the sending and receiving messages to and from the hardware platform through the host PC interface port.

![Fig.4: Transactor implementation overview block diagram](image)

4.3 Transactor Communications Flow

The communications flow begins when one of the SystemC Message Port models requests the execution of a function by the hardware platform. The request is made by writing the request character to its CMD output port. The request message is passed to the Port Interface model via the interface channel between the two models. Meanwhile the Message Port process gets suspended, pending a status return message from the Port Interface model. The Port Interface model passes the request message to the Port model and gets suspended, while the Port model passes the request message to the MMBD through the host PC communications port and it too get suspended.

The message is received by the Interface Transceiver device on the MMBD and is passed to the MicroBlaze based system implemented in the FPGA (Fig.5). The request message enters the MicroBlaze system through the UART Control module where the message is stored in input FIFO buffers. The UART Control Module asserts the interrupt request signal connected to the Interrupt Controller module, which in turn generates the CPU interrupt request. Once the interrupt has been granted, a software Interrupt Service Routine (ISR) is executed. The ISR contains the Transactor Communications Manager (TCM), which reads the incoming message from the UART Control module, interprets the message and calls the appropriate software function to execute the request. Depending on how the function has been implemented on the hardware platform, the called function will execute the requested function in software, hardware, or a combination of hardware and software. For functions implemented in hardware, the called function will load the appropriate Hardware Component bus interface register(s) (see Fig.5) with command and data information to implement the requested function in hardware. This completes the conversion of the transaction level request from the SystemC Message Port model to the signal level implementation. For functions implemented in software, the called function executes the functions directly in the MicroBlaze software environment.

Once the requested function execution has completed the hardware or software (depending on implementation) will notify the calling function, which will notify the TCM. The TCM will write the appropriate response message into the UART Control
Module output FIFO buffer and return control to the ISR. The ISR returns control to the CPU, while the UART Control module sends the reply message to the SystemC Message Port model STATUS port via the Transceiver communications interface chip, the Host PC port, the SystemC Port model and the Port Interface model. The SystemC model processes that are suspended will resume execution once they receive the reply message that the function has successfully completed.

4.4 Transactor Hardware

The Transactor hardware implemented on the MMBD consists of part of the MicroBlaze based system implemented in the FPGA fabric and the Transceiver interface chip. The part of the MicroBlaze based system related to the Transactor hardware, shown in Fig.5, is based on the OPB, which is part of IBM’s CoreConnect architecture. The Transactor hardware implementation consists of several modules connected together by the OPB interface bus and the MicroBlaze CPU.

The Hardware Components are connected to the OPB via three three types of OPB Interface modules, the Control, Status and Data, designated by C, S and D in Fig.5, each consisting of a VHDL module core, based on a modified OPB General Purpose Input/Output core.

4.5 Transactor Communication Manager (TCM)

The PESP software used to implement the Transactor in the MicroBlaze based system (implemented in the FPGA) is written using ANSI C. The software consists of two main programs as shown in Fig.6. The ISR contains the TCM program, which is responsible for interpreting and auctioning incoming requests from SystemC Controller model.

The MAIN, ISR and TCM program software flows, shown in Fig.6, begins with the MAIN function program flow executing when the MicroBlaze is initialised. Before entering a continuous loop, the MAIN program enables the interrupt capability of the MicroBlaze, Interrupt Controller, UART Control modules and configures MMDB based devices. When a request to the hardware is received via the Transactor, the UART Control module generates an interrupt request and the Interrupt Controller module generates an interrupt request to the CPU, Step 1 in Fig.6. The CPU changes the program flow to the UART ISR (Step 2). Next the UART ISR disables the interrupt controller from accepting any further interrupts and actions the TCM. The TCM verifies the incoming request and calls the appropriate function to execute the request (Step 3). This will change the program flow to the called function, which will directly instruct the hardware to carry out the request via the OPB Interface modules. Once the...
function has been completed (Step 4), the program flow will return to the TCM and status will be returned to the SystemC Controller model. Program flow is then returned to the UART ISR where the Interrupt Controller module is re-enabled and terminate the UART ISR, returning program flow to the continuous loop in the MAIN function, as shown in Step 5.

5 Example Application
A digital camera application was chosen to functional test and verify the operation of the Transactors implemented in the PESP system. The digital camera system was chosen as it is an example of a widely available digital system which is sufficiently complex to provide scope for different architecture and technology configurations. The digital camera function was successfully implemented and demonstrated the usefulness of the Transactor as a co-emulation modelling interface for use with SystemC for direct real-time implementation of SystemC based models. The focus application will be described in detail in a later paper.

6 Conclusion
This paper presented the hardware and software implementation details and architecture of a communication interface protocol, known as Transactors, which was successfully employed in research into the suitability of the underlying architecture of SystemC for direct real-time implementation of SystemC based models.

The higher level SystemC models used in PESP are currently implemented on a PC; however models could be transferred from the PC platform to the MicroBlaze itself, thus implementing the entire system on the MMDB.

References:
[1] OSCI; www.systemc.org
[7] Wolfgang Klingauf, Robert Gunzel, Rapid Prototyping with SystemC and Transaction Level Modelling, Technical University, Braunschweig, Germany, DAC 2005

ISBN: 978-960-6766-83-1

ISSN: 1790-2769
using FPGA-based Rapid Prototyping Boards, RSP’04