The ITB Journal

Volume 7 | Issue 1 Article 5

2006

# Rapid-Prototyping Emulation System for Embedded System Hardware Verification, using a SystemC Control System Environment and Reconfigurable Multimedia Hardware Development Platform

Dave Carroll

Richard Gallery

## Recommended Citation

Carroll, Dave and Gallery, Richard (2006) "Rapid-Prototyping Emulation System for Embedded System Hardware Verification, using a SystemC Control System Environment and Reconfigurable Multimedia Hardware Development Platform," *The ITB Journal*: Vol. 7: Iss. 1, Article 5.
doi:10.21427/D7JT8G
Available at: https://arrow.tudublin.ie/itbj/vol7/iss1/5

# Rapid-Prototyping Emulation System for Embedded System Hardware Verification, using a SystemC Control System Environment and Reconfigurable Multimedia Hardware Development Platform

**Dave Carroll, Richard Gallery**
dave.carroll@itb.ie , richard.gallery@itb.ie
School of Informatics and Engineering,
Institute of Technology Blanchardstown, Dublin 15, Ireland

## Abstract

*This paper describes research into the suitability of using SystemC for rapid prototyping of embedded systems. SystemC[1][2][3] communication interface protocols [4][5] are interfaced with a reconfigurable hardware system platform to provide a real-time emulation environment, allowing SystemC simulations to be directly translated into real-time solutions. The consequent Rapid Prototyping Emulation System Platform[1], suitable for the implementation of consumer level multimedia systems, is described, including the system architecture, SystemC Controller model, the FPGA configured MicroBlaze CPU system and additional logic devices implemented on the Multimedia development board used for the hardware in the PESP, illustrated in the context of a typical application.*

## Introduction

The paper describes research into the development for SystemC of an emulation environment in which SystemC based specifications can be translated directly into real-time hardware/software implementations. This would allow SystemC based designs to be implemented directly as real-time solutions, and would provide a rapid prototyping route (improved time-to-market) for complex hardware/software systems.

This paper describes a prototyping platform, incorporating hardware acceleration under software control, suitable for the real-time implementation of SystemC designs,. The development of

this system includes research into the architecture required to enable SystemC to be used directly for real-time systems, and the implementation of a prototyping system, with an application focus on consumer level multimedia systems, to investigate the viability of the approach.

The SystemC section of the prototyping system provides a tool for the specification, implementation and evaluation of different embedded system architectures and facilitates selection and evaluation of different communication interface approaches between the embedded SystemC functional models within the design.

The hardware acceleration platform, oriented towards consumer level multimedia applications, is based on an FPGA multimedia development platform, which provides a reconfigurable hardware real-time emulation environment, for prototyping real-time application designs, in conjunction with the SystemC system model. The Hardware subsystem is designed around a Xilinx MicroBaze and Multimedia Development Board[11][2], incorporating a MicroBlaze[3] processor[16] architecture implementation, on an Xilinx Virtex II[9] FPGA based platform. Hardware functional blocks developed as part of the system are written as VHDL[4] models[5] [18][19][20] in the FPGA.

## Real-time implementation of SystemC based designs

Research questions with regard to the practicality of developing real-time emulation systems for SystemC designs include:

- Whether the underlying architecture of SystemC is suitable for direct real-time implementation of a SystemC based model
  - ⇨ If not, is there some subset of, or enhancement to, the SystemC approach, which is suitable for real-time implementation.
  - ⇨ Which modelling approaches, when deployed in SystemC based designs, are suitable for (real-time) emulation.
- The interaction of SystemC with real-time OS. This is an active development area within SystemC, and also in hardware-software co-verification systems (e.g. [6][7][8]).
- What hardware/software architecture(s) for emulation platforms are required to allow real time implementation of SystemC designs?
- What hardware components are required within such real-time platforms to facilitate SystemC designs?
- What interfaces are required between the software and hardware aspects of such systems?

This paper discusses the PESP and in doing so presents a suitable SystemC based design, that when interfaced with a reconfigurable hardware system platform, using a suitable communication interface protocol, demonstrates that the underlying architecture of SystemC is suitable for direct real-time implementation of a SystemC based model. The paper discusses a modular modelling approach, which when deployed in the PESP SystemC based design, is suitable for (real-time) emulation. The author presents an hardware/software architecture for

[1] Here after referred to as PESP
[2] Here after referred to as MMDB
[3] Harvard architecture, 32bit CPU, defined and supplied by Xilinx
[4] VLSI Hardware Descriptive Language
[5] Refers to a functional block configured within the FPGA, written in VHDL or Verilog.

an emulation platform that facilitates the real time implementation of the SystemC design and defines the hardware components used in that design.

## A Co-Verification Platform

There is already active research into the development of architectures and platforms suitable for the emulation of systems described with system level description languages (such as SystemC). To date the research is largely targeted towards co-verification platforms[6], where fast verification of a design is required, rather than real-time emulation [6]. Nevertheless there is a significant correspondence in the research required to develop both fast verification systems and real-time systems.

For example in [4] the functional requirements for a co-emulation modelling interface are discussed and an architecture and API is provided (SCE-MI) to achieve these requirements. SCE-MI has been developed to meet growing industry demands for verification platforms for Systems-on-Chip (SoC) designs. It provides a mechanism via which the software aspects of an SoC design may be simulated on a workstation, the hardware aspects may be simulated on a hardware verification platform, and an efficient interface mechanism (hardware-software transactors) between the two, represented in Figure 12. SCE-MI facilitates the use of a system design language such as SystemC, as it offers the potential for fast verification (as the hardware aspects of the system may be emulated using real hardware, as opposed to using a workstation based software simulator).

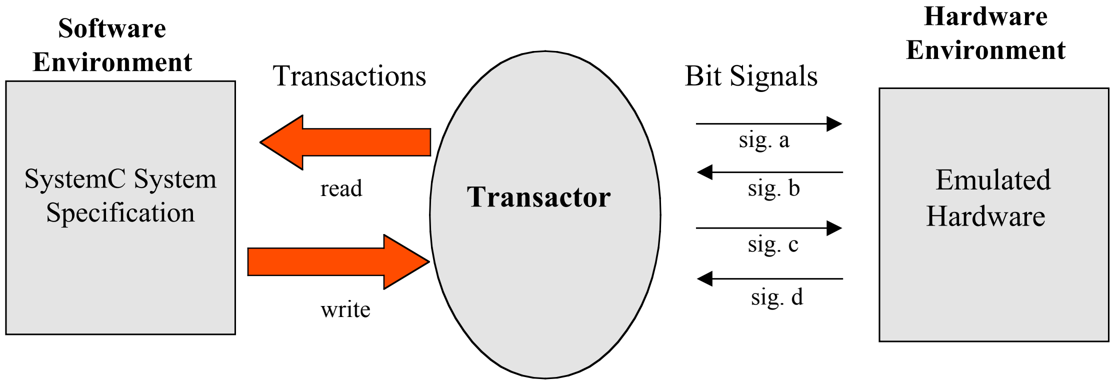

*Figure 12: co-emulation modelling interface*

In a SystemC design it would be normal to initially produce a high-level abstract model of the entire system, with the process of refinement then being used to redesign the system, such that

[6] Sometimes termed hardware/software co-emulation or co-verification platforms. In general the software aspects of the design under test (DUT) are run on a workstation, the hardware aspects are run on the hardware emulation platform, with a mechanism provided for communication and synchronisation between the two.

software components of the system modelled at a high level of abstraction, and hardware components modelled at an appropriate lower level of abstraction. Once this process of refinement has taken place, the hardware components must be simulated and verified. The Zebu system[21] provides a software/hardware co-verification platform using an implementation of the SCE-MI protocol. Using Zebu the software and hardware components of the design can be verified together, with the hardware emulation system allowing much more rapid verification time than would be the case using a software simulation of the hardware components.

The hardware-software transactor is a form of abstract gasket, which forms part of the SCE-MI infrastructure. The transactor communicates at the transactional level (e.g. Read and Write commands) with the software side of the system 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 software interface. Figure 13 shows an example of a message being translated from the software side of a system model, through the transactor, to the hardware signal side of the transactor.

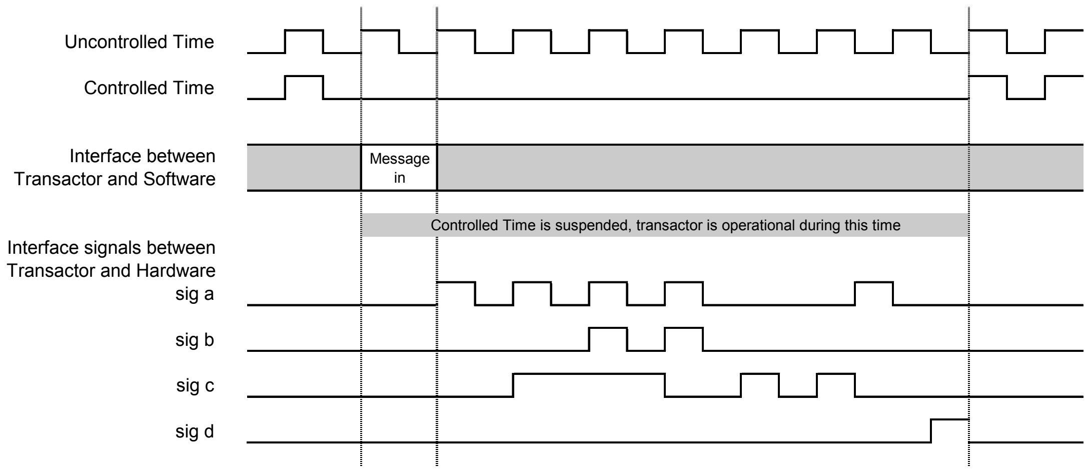
Uncontrolled Time
Controlled Time
Interface between Transactor and Software
Message in
Controlled Time is suspended, transactor is operational during this time
Interface signals between Transactor and Hardware
sig a
sig b
sig c
sig d

**Figure 13: message timing through transactor**

During the period from when the message is received by the transactor, controlled time will be suspended within the software environment via handshaking between the transactor and the SCE-MI infrastructure. The transactor will decompose the incoming message and generate required cycle accurate bit level signals to drive the hardware. Once the transactor actions have been completed, controlled time will be resumed via the transactor handshaking and the SCE-MI.

## PESP Description

The PESP has been developed with a focus on investigating the application of SystemC based rapid prototyping in consumer level multimedia applications. Its is based around the concept of a SystemC based control system, supported by application specific hardware accelerators and other components, with communication between the software and hardware managed by transactors based upon the SCE-MI protocol.

Currently this system is based on a SystemC control system using set-up on a host PC[7] and a MMDB[11] hardware development platform, containing a Xilinx, MicroBlaze softcore processor[16] implemented on a XilinxVirtex II FPGA fabric, along with other ASIC devices. The SystemC control system and the MMDB communicate through transactors, utilising a serial or other interface (such as an Ethernet link) at the physical layer. It should be apparent that the use of a PC in the development system is for convenience of development purposes only, and that, outside of the research environment, the System C control system would naturally be implemented, for example, on the MicroBlaze softcore processor.

### *Platform Architecture*

The architecture of PESP can be described using a multi-layered model to describe the overall system in an abstract form, de-coupling the system architecture from the implementation details. The multi-layered architecture consists of an Application Layer (AL), Presentation Layer (PL) and Driver Layer (DL), as shown in Figure 14.

The PL is responsible for the management of the data-flow, which implements the various functions of the platform and is controlled by the AL. The PL converts information received from the AL into a defined order and communicates with the DL, sending requests to and receiving status from the DL. The DL controls each of the hardware components (e.g. in the case of a Digital camera application, a task may consist of image capture) directly, passing required parameters and enabling the hardware components. The DL will receive status information from the hardware components and pass appropriate status response to the PL. Where several tasks are required to complete one single user command (e.g. capture an image and save it to memory), the PL will define the sequence of execution and control the execution, through the DL.

[7] Running a Microsoft Windows operating system

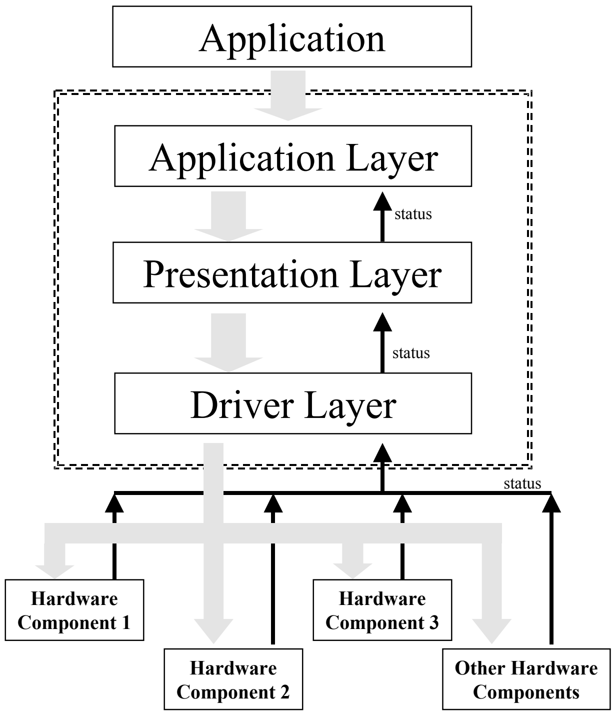

***Figure 14: Multi-layered Architecture Model***

The DL is the only layer that interfaces directly with the Hardware Components of the system. Thus if any changes are required to the hardware, only the DL will be required to be modified to accommodate the change. The Hardware Component blocks are under the control of the DL and may be implemented directly in hardware, software, or a combination of both hardware and software. For example an image compression feature may be implemented in hardware, in software or in a combination of both, depending on the optimal implementation for the particular system, which will be determined by the design engineer based on system design goals and constraints. For example, hardware implementation of the compression feature might result in a faster execution of this feature than the software implementation of the same feature, however this may result in an unacceptable increase in hardware costs.

## *SystemC and MMDB Communications Interface*

The overall platform architecture described above is now mapped to an implementation incorporating SystemC, a re-configurable hardware board. The communications interface between the SystemC model and the re-configurable hardware is modelled on the Standard Co-Emulation API: Modelling Interface[4]. This specification describes a modelling interface based on a multi-channel abstract bridge, providing multiple communication channels that allow software models describing system behaviour, such as the PESP SystemC controller models, to connect to structural models describing implementation of a hardware emulation system.

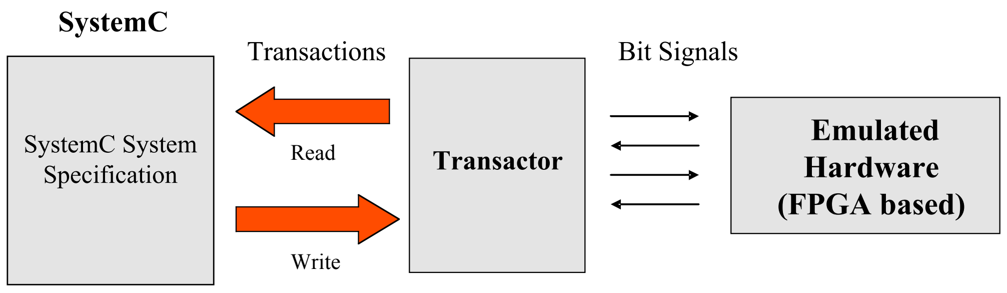

**Figure 15: Interface between the SystemC controller and MMDB platform**

Figure 15 shows a high level view of the communications interface between the SystemC controller system and the MMDB based hardware emulation system. The SystemC system is on the left of the figure and consists of several SystemC models describing the controller functions, the Xilinx MMDB platform is on the right of the figure. The Transactor performs the interface mechanism linking the high level SystemC controller description and the hardware implementation of functions within the FPGA MMDB development platform. The Transactor implementation consists of several layers of software and hardware, from the SystemC model down to the PC hardware drivers, across the physical serial interface to the MMDB development platform and onto the MicroBlaze system implemented on the FPGA fabric.

## SystemC Controller System Description

The PESP SystemC is constructed using a modular approach to provide partitioning between the different functional elements of the overall controller. Some of the benefits obtained from using this approach include:

1. The ability to break the complex system to smaller more manageable pieces, which proved useful when defining the overall SystemC system. A list of sub-functions required by the system was initially generated and a SystemC model created for each sub-function. Each model was then tested individually to ensure functionality. When the overall system was built, errors were diagnosed quickly by focusing on the interconnections between the system sub-elements.

2. The movement of functionality between different models, which proved useful during the Control model design. For example, moving all of the message display functions to the message display model, which has previously been contained in the system control

model. The modular nature of the system made the movement of the functionality a relatively straightforward process, with only two SystemC models affected.

3. The simplification of addition and/or removal of models from the system. This was particularly useful when moving functionality from the SystemC Controller System to hardware, as the hardware platform was being built. Additional SystemC models were required in order to implement the SCE-MI protocol. It was simply a matter of defining the functionality and inserting the models into the existing SystemC system. The ability to change or refine the communication interfacing between the different models is relatively straightforward when the system is designed with a modular construction. This proved very useful when deciding which interface channel type to use to connect the SystemC models that make up the overall SystemC Controller system. Different interface channel and port types can affect the flow of data from one SystemC model to another, communication refinement is described in detail in [2][3] and [5]. Primitive channels (sc_fifo) were chosen for communications between SystemC models in this system. The sc_fifo primitive channel provides the ability to communicate between SystemC models using a blocking and non-blocking read/write instruction; which ensures that data does not get lost between data transfers. Primitive channels also suspend the operation of a Thread model if the channel is full or empty, depending on the requested command. Figure 16 shows the SystemC models designed for this system and the interconnections used.

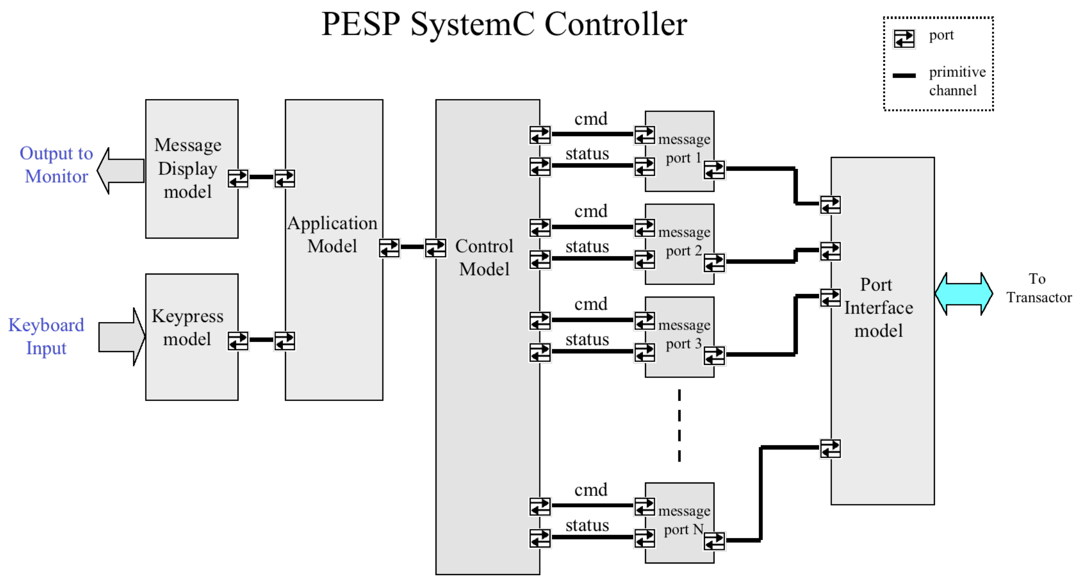

**Figure 16: Block diagram for PESP SystemC Controller**

The controller system architecture can be represented as a multi-layered architecture model as shown in Figure 17, where the AL of the system consists of the SystemC application model. The PL function of the controller system architecture is defined in the control model. The PL puts commands received from the AL in a defined order and communicates with the DL, sending requests and receiving status. The PL defines the order of execution and controls the hardware component activities through the DL. The PL will interpret commands and ensure that only predefined and acceptable user commands will be responded to and passed on to the DL. The PL will also communicate command execution status to the user, via the AL. In this system the DL is spread across the Port Interface model (Figure 16) and on to the MMDB platform. The DL contains the transactors, resident on the MMDB, and is responsible for all communications with the hardware.

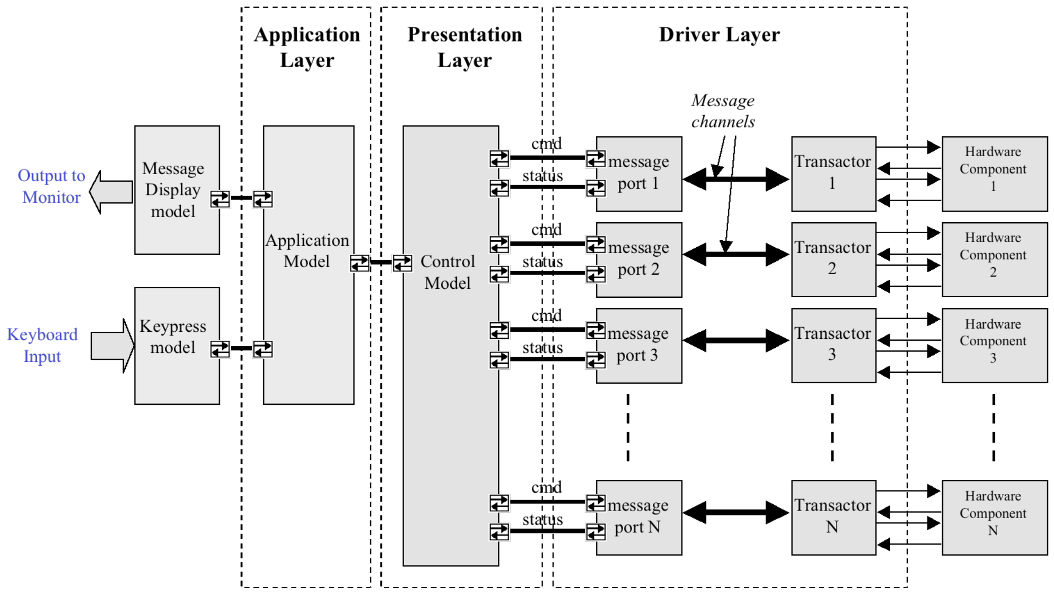

***Figure 17: multi-layered architecture of SystemC Controller model***

## Reconfigurable Hardware

The MMDB was chosen to provide the reconfigurable hardware as it provided a range of multimedia oriented features (such as video and audio codecs), combined with an FPGA to allow additional hardware functionality to be developed. This system is built around the Xilinx MicroBlaze 32-bit RISC Soft processor, which, as currently configured, acts as part of the transactor, interpreting commands from the SystemC controller, controlling hardware components on the MMDB, and managing dataflow between those hardware components themselves, and the higher level SystemC modules. The transactor is implemented through software (written in C programming language[17]) and hardware, which instructs the MicroBlaze hardware to initialise the image grabber hardware using clock cycle accurate bit signals through the MicroBlaze Peripheral Bus interface registers Figure 18.

As has been eluded to earlier, the higher level SystemC models are currently implemented on a PC, but it would be the intention, in a later stage of development, to transfer all software functionality to the MicroBlaze itself, in which case it would implement the entire system architecture described in Figure 14.

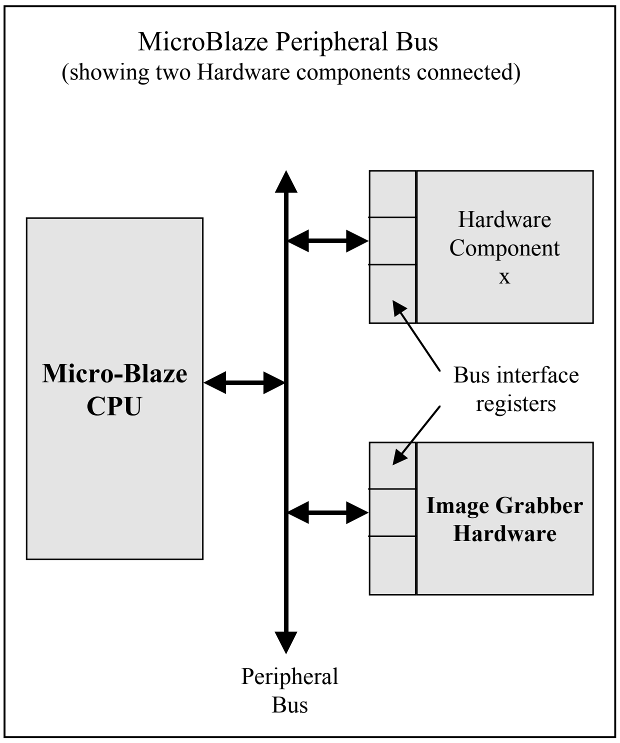

*Figure 18: simplified MicroBlaze Peripheral Bus connection*

Figure 19 shows the hardware components that were developed as part of the research and are controlled by the transactor. The Display Engine Module, Image Grabber, Workarea Control Module and Resync Module are all VHDL modules, designed, constructed and tested using the Xilinx ISE design environment [12]. The peripherals are attached to the MicroBlaze CPU via the OPB[8] Interface Bus, which is a 32Bits data bus and is configured to run at 81 MHz. All devices are accessed and controlled by software running on the MicroBlaze processor. Video images are decoder external to the FPGA and enter the FPGA via the Video Resync module[9] in 10-bit YCbCr PAL data format [13][14][15]. Here the data is synchronised with the output from a 27 MHz-clock generator model, contained in the FPGA (not shown in the figure). Data exiting the Video Resync model is passed directly to the Video Encoder, a single chip, outside the FPGA boundary, on the development board.

[8] MicroBlaze On-Chip Peripheral Bus

[9] This is a VHDL model, which is used to synchronise the input data with the 27 MHz input clock. The model implements a data FIFO function, using a Dual Port Block RAM to temporarily store a portion of a line of video data.

Image display switching between the live video input data stream and the Display Engine output (containing a saved image) is controlled by the Display Engine under the control of the software running on the MicroBlaze.

The Image Grabber VHDL model is used to capture and store images, and is under the control of the MicroBlaze software. Communications to and from, the software takes place through Control, Status and Data models connected to the OPB interface bus. Images are stored in the Image Memory[10] device, located outside the FPGA. The SRAM is controlled by the Image Grabber VHDL model.

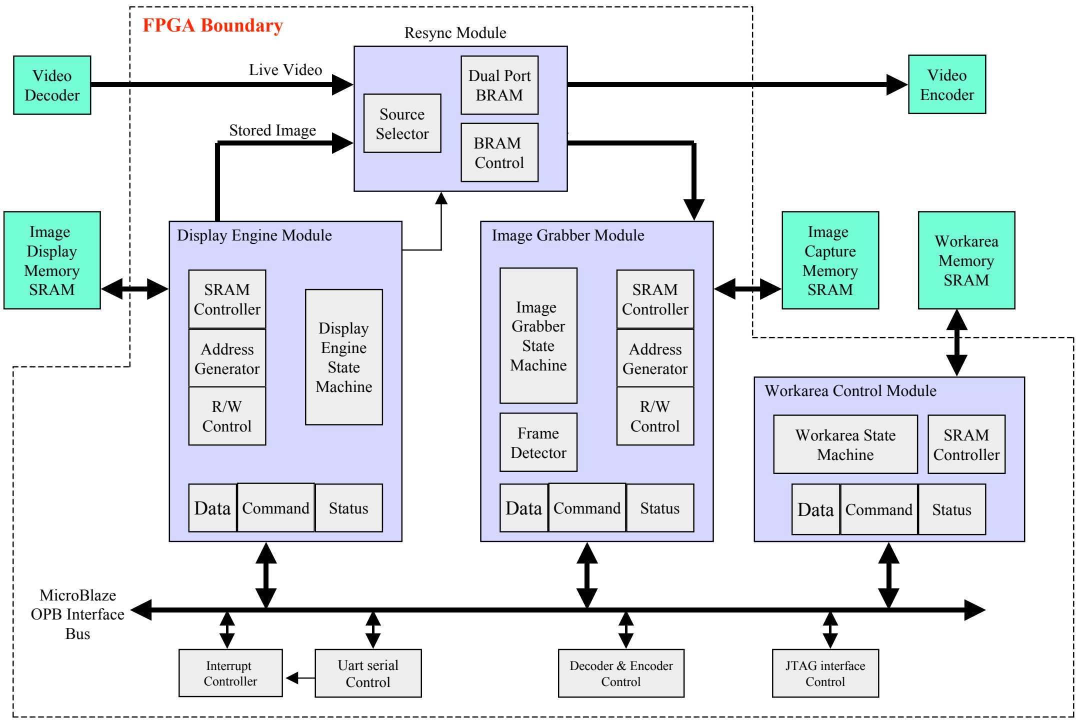

**Figure 19: FPGA based Video System Hardware Architecture**

The Display Engine model is responsible for the display of captured images. It provides a continuous stream of 10-bit video data to the Video Resync module, synchronised with the system 27 MHz clock. The Display Engine also controls the contents of the Image Display Memory device, which is located on the MMDB, external to the FPGA. The Display Engine is under the control of the MicroBlaze via a Control, Status and Data models connected to the OPB interface bus.

The Interrupt Controller is used to interrupt the CPU upon request from the Uart Serial Control model, which is connected to the RS232 interface and the OPB interface bus.

[10] A 512K x32 SRAM

Commands to and from the SystemC Controller are sent through the Uart model. The JTAG Interface Control model is used for debug interface access for the EDK system.

The Workspace Memory consists of a single SRAM device on the MMDB and is controlled by the Workarea Controller connected to the OPB bus. This storage facility is used for temporary storage of images as they are being manipulated.

### *Sample PESP Application Implementation*

A digital camera was used as the focus application for functional testing of the PESP platform. The digital camera application 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 functions also suited the application devices, which are integrated into the Xilinx MMDB hardware development board (however PESP is not board specific). A digital camera is an example of a system that uses several data processing functions, which are relatively self-contained and as such simplify the implementation of functions using a modular approach. For example the image grab function is not dependent on image display function and as such both functions can be implemented in separate hardware components.

## Conclusion & Future work

This paper discussed the PESP, which consists of an integration of both a SystemC controller and a reconfigurable multi-media hardware development platform.

This paper presented a suitable SystemC based design, that when interfaced with a reconfigurable hardware system platform, using a suitable communication interface protocol, demonstrates that the underlying architecture of SystemC is suitable for direct real-time implementation of a SystemC based model.

The paper discussed a modular modelling approach, which when deployed in the PESP SystemC based design, is suitable for (real-time) emulation. The paper also discussed an hardware/software architecture for an emulation platform that facilitates the real time implementation of the SystemC design and defines the hardware components used in that design.

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, which would implement the entire system architecture described in Figure 14 on the MMDB. The interaction of SystemC with real-time OS was not considered as part of this paper and could be an area of focus for future work. This is an active development area within SystemC, and also in hardware-software co-verification systems (e.g. [6][7][8]).

## References

[1] OSCI; www.systemc.org
[2] OSCI; "Functional Specification for SystemC 2.0", available at www.systemc.org
[3] T. Grotker, S Liao, G. Martin, S Swan; "System Design with SystemC"; Kluer Academic Publishers, ISBN 1-4020-7072-1
[4] Standard Co-Emulation Modelling Interface (SCE-MI) Reference Manual, Version 1.0, Published by Accellera, available from www.eda.org/itc
[5] J. Stickley, Duaine Pryor; "Functional Requirements Specification: Standard Co-Emulation Modeling Interface (SCE-MI)"; www.systemc.org
[6] Emulation and Verification Engineering; "Universal Design Verification Platform for IP, FPGA, ASIC Design and Embedded Software Developers"; www.eve-team.com
[7] Jon Connell, Bruce Johnson , Early Hardware/Software Integration Using SystemC 2.0, ESC San Francisco 2002.
[8] L. Benini et al., SystemC Cosimulation and Emulation of Multiprocessor SoC Designs, IEEE Computer, April 2003, pp. 53-59.
[9] Virtex II data sheet (Xilinx doc. no. DS031-1), www.xilinx.com
[10] Embedded Systems Tools Guide (EDK 3.2.2), www.xilinx.com
[11] Xilinx Virtex II MicroBlaze and Multimedia Development Board, www.xilinx.com
[12] ISE user guide, www.xilinx.com
[13] A Technical Introduction to Digital Video, Charles Poynton, John Wiley & Sons, ISBN 0-471-12253-X
[14] Video & Television Engineering, J. Whitaker and B. Benson, McGraw Hill, ISBN 0-07-069627-6
[15] ADV7185 Video NTSC/PAL Decoder data sheet, www.analog.com
[16] MicroBlaze RISC 32-Bit Soft Processor data sheet, www.xilinx.com
[17] The C Programming Language, Kernighan & Ritchie, Prentice Hall, ISBN 0-13-110362-8
[18] VHDL made Easy, D. Pellerin & D. Taylor, Prentice Hall, ISBN 0-13-650763-8
[19] VHDL for Designers, S. Sjoholm & L. Lindh, Prentice Hall, ISBN 0-13-473414-9
[20] Digital System Design with VHDL, M.Zwolinski, Prentice Hall. ISBN 0-201-36063-2
[21] Co-Simulation Between SystemC and a New Generation Emulator, C. Alquier, S. Guerinneau, L.Rizzatti, L. Burgun, DesignCon 2003