SYSTEM LEVEL MODELING METHODOLOGY OF APPLICATION SPECIFIC INSTRUCTION SET PROCESSOR (ASIP) USING SYSTEMC

Rana Mukherji[1] and Manishita Das[2]
[1]Faculty of Science and Technology, The ICFAI University, Dehradun – 248197, Uttarakhand, India and [2] Institute of Science and Technology for Advanced Studies and Research (ISTAR), Vallabh Vidyanagar, Anand – 388120, Gujarat, India

[1]E-mail: rana.mukherji@gmail.com

Abstract: *In recent years, the development of application specific instruction set processors (ASIP) is the exclusive domain of the semiconductor houses and core vendors. This is due to the fact that constructing such architecture is a difficult assignment that needs skilled knowledge in distinct domains: application software development tools, processor hardware implementation, and system integration and verification. To specify the design and implementation of such systems and incorporate the functionality implemented in both hardware and software forms, we are compelled to move on from traditional Hardware Description Languages (HDLs). Since C and C++ are dominant languages used by chip architects, system engineers and software engineers today, we believe that a C++ based approach to hardware modeling is necessary. This will enable codesign, providing a more natural solution to partitioning fuctionality between hardware and software. In this paper, we discuss a design approach of SystemC (a C++ class library) for ASIP at the system-level which provides necessary features for modeling design hierarchy, concurrency and reactivity in hardware. To exemplify and validate the method we employed it to the design of a 32-bit ASIP for Hindi Text-to-Speech Synthesis developed by CEERI, Pilani (INDIA).*

Keywords : *ASIP , System and, System Level Design*

1. Introduction

Due to the ever-decreasing feature size of today's semiconductor processes, the cost of a mask set has already crossed the one-million-dollar line. To pay off this investment, a design must be applicable for multiple purposes. The flexibility needed to achieve this is commonly provided by programmable elements. A unique opportunity to trade off the flexibility of general-purpose processor cores against the performance of hard-wired logic is offered by application-specific instruction-set processors (ASIPs). The instruction set of an ASIP is dedicated towards a particular class of applications by compound instructions that speed up critical parts of the applications without compromising the flexibility of the processor in its application domain [1, 12, 13,19].

Due to the diversity of the application domains that ASIPs are specialized in, it demands greater attention during synthesis tool development [2, 3, 11, 14]. It is the function of the synthesis tool to offer abstraction from the low level details of the hardware, in order to make the implementation of algorithms a tractable problem for the human programmer. It merely shifts the burden from the application developer to the synthesis tool designer. Generally, there are two types of approaches by the commercially

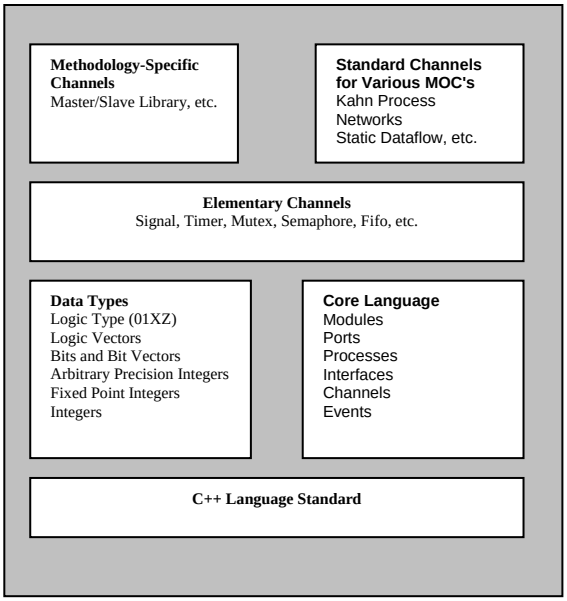

Fig. 1: SystemC Language Architecture [16]

Date of submission : 13. 11. 2010 Date of acceptance : 20. 08. 2011

supported flows: the tool oriented design flow or a language oriented design flow. Here we are only discussing the language oriented design flow.

The goal of the language-oriented approach is to create an environment where implementations can be realized in a 'stress-free' programming language; it makes some sense to assume such a language from the outset, and to implement it in a top-down fashion. it is summarized as follows:

1. Design or choose a programming language which provides behavioral semantics for the constraints of the algorithms
2. Generate transformation schemes for each of the language's behavioral constructs[15].

SystemC one of the languages which is capable of providing all the requirements for the ASIP architecture exploration. It is an emerging standard modeling platform based on C++ which it allows describing a fully functional model that incorporates design constraints and has a simulation environment for an integrated validation against a set of test vectors. More of its feature is discussed in section 2. The work presented in this paper refers to an optimized system level framework for implementation of an ASIP using SystemC platform. In Section 3, we describe the architecture design overview of ASIP of Hindi text-to-speech conversion (developed by IC Design Group, CEERI, Pilani, INDIA). In section 4, we illustrate the SystemC implementation methodology of the said processor. In Section 5, we show the testing environments and results. Finally, Section 6 draws the conclusion.

2. SystemC LANGUAGE

SystemC has been initiated by the Open SystemC Initiative (OSCI). OSCI is a non-profit association that has been found by several industrial, academical and individual partners. The aim of OSCI is the standardization of SystemC as an open source standard for system level design. Since the SystemC library is open source, various kinds of modifications and extension libraries are publicly available, too [15,6,5,8,20]. Fig. 1 summarizes the SystemC language architecture and Fig. 2 describes the design environment of SystemC .

The system description language SystemC provides hardware constructs, implemented in a C++ class library. The hardware models specified using SystemC can be compiled on a large number of supported architectures using a standard C++ compiler [16, 9]. The compiled executables can be cycle accurate simulations as well as untimed algorithmic descriptions of the given design. The executable specifications can be used for evaluation, debugging and refinement purposes without the usage of a commercial simulator. Depending on the abstraction level the simulation speed can be a multiple of a functional equivalent HDL model. Because of its unrestricted C++ conformance each SystemC model can be combined with other

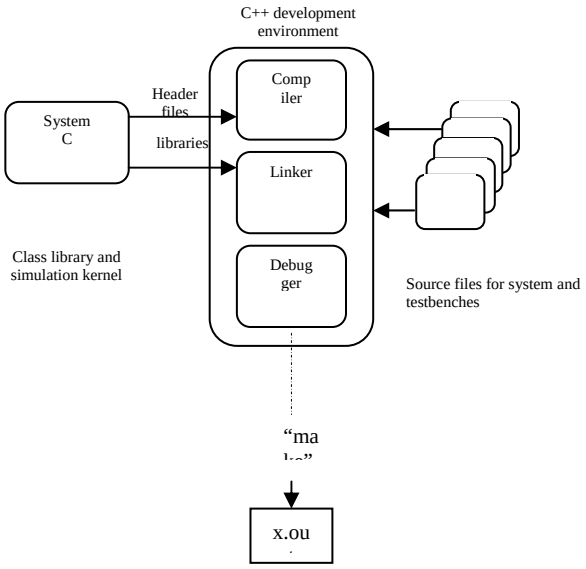

Fig. 2: SystemC in a C++ development

software libraries. This allows system engineers to take advantage of HW/SW Co-Design and to refine their SoC designs with a high level of flexibility. Another benefit of SystemC, coming with its C++ conformance, is a wide range of abstraction levels that can be used to simplify huge system designs [21]. Complex communication protocols and control logic can easily be separated from functional parts of the specification. For this reason SystemC offers techniques that can raise or lower the level of abstraction. The TLM library implements such a technique to support SystemC's efficient refinement methodology [7].

SystemC combines HDL typical features, like concurrency as it appears in hardware, with software paradigms, like object orientation. These features distinguish SystemC from VHDL, Verilog and SystemVerilog and enable system description capabilities. SystemC allows real polymorphism which includes the application of arbitrary memory access using pointers and dynamic memory allocation. Even the concept of virtual functions that binds overloaded class members to function pointers, is applicable in system descriptions. Special benefits, like channels, make SystemC ideal for describing complex communication protocols and their easy reuse.

3. Architecture Design of the Asip

In this section, we describe the design of the ASIP (Fig. 3) developed by the CEERI, Pilani, INDIA [4, 18]. It can serve as an efficient platform for embedded systems running the parametric speech synthesizer in portable/mobile applications. An application-friendly instruction set and a supporting micro-architecture have been created that permit execution of the parametric speech synthesizer in real-time using a relatively small gate count and memory size and potentially low power consumption following the design philosophy of [4]. It had an execution unit following with a number of application specific dedicated functional blocks-some with combinational architectures and others with their own optimized sequential architectures and associated controllers. These functional units and the integer and floating-point memory blocks of required size along with the necessary temporary registers were connected through a single bus for transfer of data among them. This approach provided a mechanism for interpreting an application-specific, user-friendly instruction set onto sufficiently high-level functional units by moving data to them over the bus and triggering them.

The 32-bit instruction-set of the processor was designed to provide several application-specific, user-friendly, 'high-level' instructions which implemented frequently-repeating, logically meaningful, computational patterns specific to the application-formant-based parametric speech synthesis. These include instructions like:

1) resonator: which computes the function of as second-order resonator.
2) setabc: which computes the three resonator coeffi cients — given the frequency of resonance and the resonance bandwidth.
3) rand12: which generates a sequence of 12 pseudorandom numbers.
4) exp: which computes the value of the exponential function for its operand.
5) sin-cos: which computes the values of the sine and cosine functions for its operand.

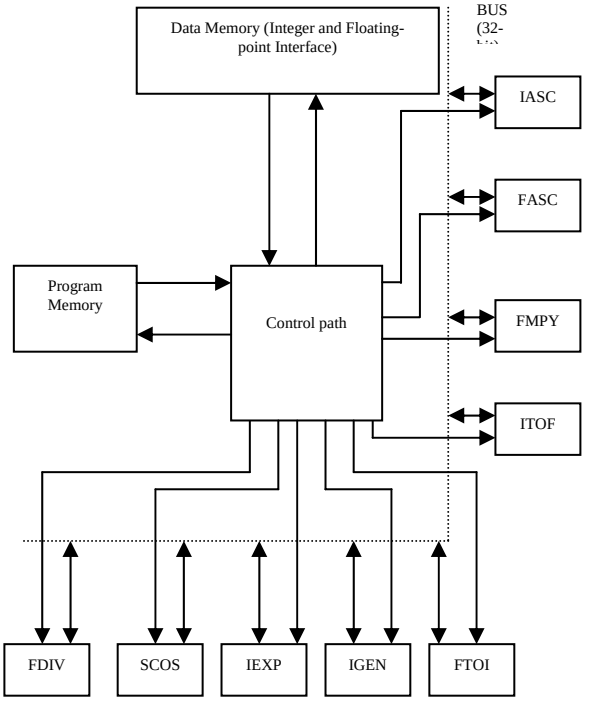

Fig. 3: CEERI ASIP [18]

Besides these instructions, a number of low-level, general-purpose instructions are also included for addition, subtraction, multiplication, division, data-moves, and type conversion of operands. The program control instructions include various conditional branch instructions (based on results of relational operators on variables), instructions for supporting loop constructs, and for subroutine calls and returns. The processor's instruction set has a set of 44 instructions

4. Model Implementation

Fig. 4 illustrates the basic structure of framework of the methodology. we start with the traditional methods used to capture the customer requirements, a Product Requirements Document (PRD). From the PRD, a ASIP-SAM (ASIP System Architecture Model) is developed. The ASIP-SAM development effort may cause changes or refinement to the PRD. In an algorithmic intensive system, the ASIP-SAM will be used to refine the system algorithms.

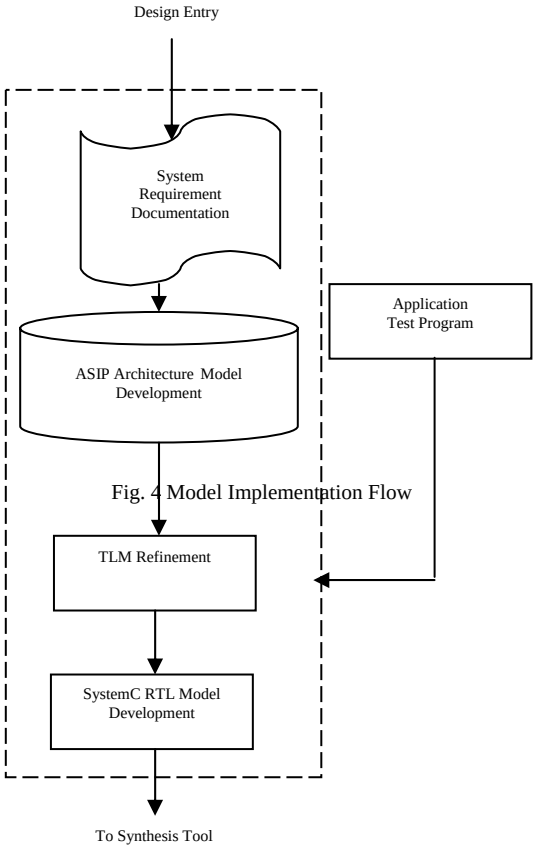

Fig. 4: Basic Structure of Framework of Methodology

The ASIP-SAM development consists of breaking the processor's functionality into a set of instructions. A component's functionality represents all possible behaviors that the components can assume, with behavior meaning the set of actions that the component performs during the execution of an application. The objective of ASIP-SAM development is to capture the specification of the system in terms of design behavior with the least amount of design work. This first step hides the complexity of the processor's internal implementation behind the simple interface offered by the instruction set. There is a tradeoff in selecting the right set of instructions: having many fine-grained instructions can lead to greater accuracy, but it requires a longer simulation time than having fewer coarse-grained instructions.

The next stage TLM Refinement involves simulating the application program and extracting a trace file for the processor. A trace is the sequence of instructions/data items a component executes during its simulation. The aim is to estimate the component's switching activity.

The last step (SystemC RTL Model Development) consists of mapping the instructions requested by the various tasks performed by the component into abstract functional units that are used to estimate complexity-that is, gate count and timing delay. Given switching activity and complexity, the framework can also compute the component's power per instruction and execution time per instruction.

To Solve the equation 3*e^2 - 4.5*e^6 + 7.2

11000000000001000000000100000000 // EXP #4, #4

00101100000001000000000000000000 // MULTFF #4, #0

11000000000001010000000101000000 // EXP #5, #5

00101100000001010000000001000000 // MULTFF #4, #0

00001000000001000000000101000000 ADD FF #4,#5

00001100000001000000000010000000 // ADDFC #4, #2

Fig. 5 Object Code with its Assembly Pneumonics

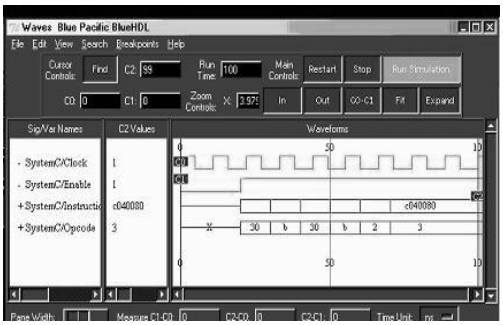

Fig. 6: Waveform Result

5. Results

The design is implemented using WindowsXP as platform and Microsoft Visual Studio 2005 as simulator for SystemC whereas the Waveforms are seen through BlueHDL VCD Viewer. In this framework, we simulated ten different application programs and validated the functionality. One of the test program and its result are shown in the Fig. 5 and Fig. 6.

6. Conclusion

In this paper, we presented the SystemC processor design platform – a framework for the design of application specific integrated processors. The platform supports the architecture designer in different domains: architecture exploration, implementation, application software design and system integration/verification. An ASIP developed by CEERI , Pilani, was completely realized using this novel design methodology – from specification to implementation.

Acknowledgment

The work reported in this paper is a byproduct of many, vibrant, insightful discussions with many creative individuals. The authors of this paper would like to thank all of them especially stressing contributions of: Dr. R S Shekhawat, Dr. G.P. Srivastava, Dr. R.C. Ramola, Graziano Pravadelli (Department of Computer Science at the University of Verona, Italy), Md. Rashid Ansari, R.K. Chaurasia , Ranjan Mishra and Dr. Umesh Gupta.

References

[1] A.Alomary, T.Nakata, Y.Honma, M.Imai, N. Hikichi, “An ASIP instruction set optimization algorithm with functional module sharing constraint.”, Proc. ICCAD-93, 1993, pp. 526-532.

[2] Andreas Hoffmann, Oliver Schliebusch, Achim Nohl, Gunnar Braun, Oliver Wahlen and Heinrich Meyr, “A Methodology for the Design of Application Specific Instruction Set Processors (ASIP) Using the Machine Description Language LISA”, Proceeding International Conference on Computer Aided Design (ICCAD’01), 2001 pp. 625-630.

[3] A. Hoffmann, F. Fiedler, A. Nohl, and S. Parupalli.A methodology and tooling enabling application specific processor design. In VLSID ’05, Washington, DC, USA, 2005, pp 399–404.

[4] Chandra Shekhar, Raj Singh, A. S. Mandal, S. C. Bose, Ravi Saini and Pramod Tanwar, "Application Specific Instruction Set Processors : Redefining Hardware-Software Boundary," vlsid, , 17th International Conference on VLSI Design, 2004, pp. 915.

[5] Christian Genz, Rolf Drechsler, “System Exploration of SystemC Designs”, Proceedings of Emerging VLSI Technologies and Architectures (ISVLSI’06), pp 522-528, 2006

[6] Claudio Talarico, Min-sung Koh, Esteban Rodriguez-Marek, “System Level Performance Assessment of SOC Processors with SystemC “,Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07) , pp. 523-530, 2007.

[7] Frank Ghenassia. Transaction Level Modeling with SystemC TLM:Concepts and Applications for Embedded Systems, Springer, 2005.

[8] IEEE Std. 1666-2005, Standard SystemC Language Reference Manual, IEEE, 2006.

[9] J. Bhasker; “A SystemC Primer:” Star Galaxy Publication; edition February 2003.

[10] Jérôme Chevalier, Maxime de Nanclas, Luc Filion, Olivier Benny, Mathieu Rondonneau, and Guy Bois, El Mostapha Aboulhamid, A SystemC Refinement Methodology for Embedded Software, Proceedings of IEEE Design & Test of Computers , 2006, pp 148-158.

[11] J.V. Praet, G.Goossens, D.Lanneer , H. De Man: “Instruction set definition and instruction selection for ASIPs.”, Proc. Int. Symp. on High-Level Synthesis, 1994, pp. 11-16,.

[12] Li Zhang, Shuangfei Li , Zan Yin and Wenyuan Zhao , “A Research on an ASIP Processing Element Architecture Suitable for FPGA Implementation”, International Conference on Computer Science and Software Engineering, 2008, pp 441-445.

[13] Manoj Kumar Jain, M. Balakrishnan, Anshul Kumar, “ Integrated On-Chip Storage Evaluation in ASIP Synthesis” , Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design, 2005, pp 274 - 279 .

[14] M. Goudarzi, S. Hessabi, A. Mycroft, "Object-Oriented Embedded System Development Based on Synthesis and Reuse of OO-ASIPs", Journal of Universal Computer Science, Vol. 10, No. 9 , 2006, pp. 123-135.

[15] Rana Mukherji, “SystemC-based Design Approach for Modeling Reconfigurable Computing Systems”, The Icfai University Journal of Science and Technology, Vol 4, No.3, 2008, pp 30-40.

[16] Rana Mukherji, “Behavior Modeling of the Application Specific Instruction Set Processor (ASIP) For Text-to-Speech Synthesis Using SystemC”, .M.Tech. Dissertation, Panjab University, Chandigarh, INDIA, 2004.

[17] Rana Mukherji, T R Choudhary , Amit Chatterjee, "An Approach to Implement AMBA APB Timer IP Using SystemC" , Journal of Computer Science and Mathematics, Vol. 1, No. 3, 2010, pp 310-318.

[18] Ravi Saini, Pramod Tanwar, A. S. Mandal, S. C. Bose, Raj Singh, Chandra Shekhar, "Design of an Application Specific Instruction Set Processor for Parametric Speech Synthesis," vlsid, , 17th International Conference on VLSI Design, 2004, pp.773.

[19] S.Saponara, L.Fanucci, S Marsi, G.Ramponi, D.Kammler and E.M. Witte, "Application-Specific Instruction- Set Processor for Retinex-Like Image and Video Processing," IEEE Transactions on Circuits and Systems II: Express Briefs, Vol.54, No.7, 2007, pp 596–600.

[20] S. Swan "SystemC Transaction Level Models and RTL Verification," Proc. 43th Design Automation Conf. (DAC 2006), IEEE Press, 2006, pp. 90-92.

[21] Thorsten Grötker, Stan Liao, Grant Martin, Stuart Swan, System Design with SystemCTM, Kluwer Academic Publishers, 2002.

Rana Mukherji received Masters of Science in Electronics and Masters of Technology in Instrumentation Engineering from Panjab University, Chandigarh in 2002 and 2004 respectively. He is currently working as Faculty Member in Faculty of Science and Technology, The Icfai University, Dehradun, INDIA. He had published various papers in reputed national and international conferences and journals. He is actively associated with many technical bodies VSI (VLSI Society of India), ISI (Instruments Society of India, IISc, Bangaluru, INDIA), IMS (Indian Microelectronics Society, Chandigarh, INDIA), OSCI (Open SystemC Initiative) and Educational Consultants of India Ltd. (A Govt. of India Enterprise). His current research areas include VLSI, Reconfigurable Architectures, SystemC and Robotics.

Manishita Das has submitted her PhD thesis in the field of Environmental Sciences from Sardar Patel University, Vallabh Vidyanagar, Gujarat, India. Presently, she is working as Faculty Member at Amity Institute of Biotechnology (AIB), Amity University Rajasthan, Jaipur, INDIA. She has a total of eight (08) publications in various journals of national and international repute. Her areas of interest comprise Wetland Pollution Mitigation and its biodiversity study, Bioremediation and ASIP development for Environmental Issues.