HW-SW Co-Synthesis: The Present and The Future

Sri Parameswaran
Department of Computer Science and Electrical Engineering
The University of Queensland
Queensland 4072
Australia
sridevan@elec.uq.edu.au

Abstract

As we move towards several million transistors per chip it is desirable to move to higher levels of abstraction for the purposes of automated design of systems. Increasing performance of microprocessors in the marketplace is moving the balance between software and hardware. In this environment, it is necessary to adapt our tools to create systems, which encompass these fast microprocessors rather than compete with them. It is important to adapt other peripheral components such as sensors and RF circuits into our design methodology.

I. INTRODUCTION

How do we design with several hundred million transistors effectively and quickly? Hardware Software Co-design is said to provide the answer for designing such large systems. HW/SW Co-design [2,14,21,20,17,19,73,75] is a wide area of research consisting of simulation, validation, synthesis etc. In this seminar we will look at the synthesis aspect of co-design.

In the particular research area of synthesis, three distinct methods have emerged. The first method is the Co-processor method, which usually consists of a central off the shelf processor, and some application specific integrated circuits in order to speed up those parts of the program which are slow in the processor. The second method is the Application specific instruction processor method, in which a unique processor is designed and created for a particular application. This processor will have a unique instruction set and architecture, which is specifically tuned for the application. A third approach is to partition the task at hand into different components, and allocate each component to a separate processor.

Each of these three methods start with a variety of input specifications[15,4,5], such as software languages (C), Hardware Description languages (VHDL) [25] and specialized languages (Spec Charts). These languages are at times modified to include specific characteristics such as timing information and parallelism [29,30,6,41,42,63].

In discussing the three major methodologies the profiling techniques will be have to be critically considered. Further the quality of results from present estimation techniques will have to be examined. We will also need to look at the speedups achieved in Hardware Software Co-synthesis projects so far and compare the speedups with other traditional methods to see whether these Co-synthesis methods are really useful. We need to explore techniques to improve current speedups. We have to examine application programs and the data associated with them in order to improve the final result.

We also have to look at architectures that are presently being used and their weaknesses. Further, we need to explore architectures that could be used in the future with the current trends in microprocessor design etc. We have to analyse effects of emerging standardization such as Virtual Systems Interconnect (VSI).

II. LANGUAGES

Standard languages such as C and VHDL are very popular input languages. These have the disadvantage of either being software oriented or hardware oriented. Other specifications methods such as graphical or semi graphical methods are becoming popular.
C++, FORTRAN etc are also used in specifying synthesis systems. These suffer because of the single thread nature and because no timing constructs exist. The advantages of these languages are that these are widely used and thus have good support and several analysis tools. Several extensions are available for these languages such as PVM, P4, HPF, and MPI.
There are several in house languages such as Jade and SAM which are probably more suitable but have little or no support. Languages which are state charts [22] based such as SPECHARTS are useful in describing systems are becoming popular, but are not widely available yet.

III. PARTITIONING

As we rapidly move towards a time of low cost, high-speed microprocessors and cores [1,13,26,50,51,52,53,67], the line between software and hardware is changes from month to month. What someone would have said with certainty should have been implemented in hardware just a year ago, is probably implementable in software for a fraction of the cost without any sacrifice in performance today. In light of this transience in the state of the art we have to critically look at what are the alternative we can research with certainty over the next few years. A research project started today will often take a few years to complete, and can be totally irrelevant to the market place, leaving the researcher feeling ineffective.

There are several types of architectural models, which use both processors and ASICs. Models included a single processor and a single ASIC, single processor with several ASICs, several processors and several processors with several ASICs[54]. All systems, which automatically synthesize circuits based on these models, include an estimation system and a partitioning system. The estimation system allows the quick evaluation of alternative partitioning solutions in the design space. Partitioning solution allows the total task to be optimally shared by processors and ASICs, according to a given set of criteria, be it speed, cost or low power.

Partitioning algorithms described are usually very effective and fast. However these tools depend on estimation tools and profiling tools for their final partition which then makes it quite unreliable.

IV. PROFILING TOOLS

Profiling tools [74] are a necessity to get information on how long a particular segment of code takes to execute and how many times a loop (with indeterminate loop counts) is executed. Tools such as GPROF are not accurate and give widely varying results. Execution graphs have been used which have also been shown to be not accurate.

The major reason behind these inaccuracies is the fact that the architecture greatly influences the final result. Since the architecture is not known at the beginning, the profiling will inevitably be wrong.

Simulation [18, 47] and emulation tools can be used to accurately predict performance. However, these tools are slow, and are only useful at the latter stages of the design cycle.

V. ESTIMATION TOOLS

Estimation tools have been notoriously ineffective in the past. Three of the most used estimation tools have been: profiling tools to estimate the time taken in software for a given length of code (see last section); area estimation tools to assess the probable size of the ASIC when finally implemented; and the execution time estimation tools for ASICs which, estimate the execution time of an ASIC.

The tools used to estimate the size of area can be extremely error prone, particularly since it is difficult to estimate the interconnection area. The error in estimate can be very costly since the cost of chips is stepped rather than a linear function. For example a chip 2.01x2sq.mm chip can cost a great deal more than a 2x2sq.mm chip. This situation can possibly improve as more and more pre-fabricated cores are used in design, which would then reduce the total amount of unknown interconnection area.

The time taken for a particular ASIC to execute is another estimate, which is difficult without the final layout, since the clockwidth cannot be predicted. Often predictions are based on the number of cycles, but this is almost useless without the clockwidth information.

The other important consideration is the power estimation. This estimate at the present time is very sketchy, though there are some experts who given a particular process, number of gates, clockspeed etc can estimate the power to within 10% of its real value.

VI. FUTURE OF COSYNTHESIS

There are several areas where real progress has to be made. These are accurate but quick profiling and estimation techniques, better specification languages, improved communication methodologies, automated architecture synthesis and testing of such systems [3,8,31,34-37,28,54-56,62]. Heterogeneous systems, which are automatically synthesized, will be a major research area in the near future [10,12,57,59,66,70-72]. Compiler optimization for simple systems is now available [7,11,16,23,24,32,33,38-40,43-46,48,49,52,58,60,61,64], but will have to be modified and improved for heterogeneous systems.

There are other areas where little or no work has been carried out. For instance there is little work in automatically integrating sensors to digital systems, or interfacing RF circuits to digital systems, or integrating analog drivers to embedded systems.

References

[1] Advanced RISC Machines Ltd. ARM. web pages. http://www.arm.com/, 1995.
[2] H. Alomary, T. Nakata, Y. Honma, M. Imai, and N. Hikichi. *An ASIP instruction set optimization algorithm with functional module sharing constraint.* Int. Conf. on Computer-Aided Design (ICCAD), pp. 526-532, 1993.
[3] U. Bieker and P. Marwedel. *Retargetable self-test programgeneration using constraint logic programming.* 32nd Design Automation Conference, 1995.
[4] F. C. Belz, , and D.C Luckham, *A new language-based approach to the rapid construction of hardware/software system prototypes*. In Proc. Third International Software for Strategic Systems Conference (Feb. 1990), pp. 8-9.

[5] G. Berry, and G. Gonthier, *The Esterel synchronous programming language: Design, semantics, implementation.* Science Of Computer Programming 19, 2 (1992), 87-152.

[6] G. Boriello. *Software scheduling in the cosynthesis of reactive realtime systems*. Proceedings of the 31th Design Automation Conference, pp. 1-4, 1994.

[7] D. G. Bradlee, S. J. Eggers, and R. R. Henry. *Integrating register al location and instruction scheduling for RISCs.* Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 122-131, 1991.

[8] D. Brahme and J. A. Abraham. *Functional Testing of microprocessors.* IEEE Trans. On Computers, pp. 475-485, 1994.

[9] S. Brown et. al. *Experience in Designing a LargeScale Multiprocessor using FieldProgrammable Devices and Advanced CAD Tools.* In Design Automation Conference .pp. 427-432. 1996

[10] J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: *A Platform Heterogeneous Simulation and Prototyping.* In European Simulation Conference .1991.

[11] M. Cornero, F. Thoen, G. Goossens, and F. Curatelli. *Software synthesis for realtime information processing systems.* in: P. Marwedel, G. Goossens (ed.): Code Generation for Embedded Processors, Kluwer, 1995.

[12] R. C. Covington, S. Madala, V. Mehta, J. R. Jump, J. B. Sinclair, *The Rice Parallel Processing Testbed.* In Proceedings of the 1988 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (1988), pp. 4-11.

[13] EE Times. web pages. http://techweb.cmp.com/techweb/eet/embedded/embedded.html, 1995.

[14] R. Ernst, J. Henkel, and T. Benner. *HardwareSoftware Cosynthesis for Microcontrollers.* IEEE Design & Test of Computers , 64-75. 1993

[15] European Computer Research Center ECRC. *ECLIPSE 3.4 user manual, ECRC common logic programming system.* ECRC GmbH, Munich, 1994.

[16] Fauth and A. Knoll. *Automated generation of DSP program development tools using a machine description formalism*. Int. Conf. on Audio, Speech and Signal Processing, 1993.

[17] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong, *Specification and Design of Embedded Systems.* Prentice Hall, 1994.

[18] S. Goldschmidt, *Simulation of Multiprocessors: Accuracy and Performance.* Ph.D. Thesis, Stanford University, June 1993.

[19] R. K. Gupta, and G. D. Micheli. *HardwareSoftware Cosynthesis for Digital Systems.* IEEE Design & Test of Computers (Sept. 1993), 29-41.

[20] R. Gupta. *Cosynthesis of Hardware and Software for Embedded Systems.* Kluwer Academic Publishers, 1995.

[21] R. Gupta. *Hardware software codesign of embedded systems.* Tutorial at the VLSI Design Conference, Bangalore, 1996.

[22] D. Harel, *Statecharts: a visual formalism for complex systems.* Science of Computer Programming 8 pp. 231-274. 1987

[23] Hartmann. *Combined scheduling and data routing for programmable ASIC systems.* EDAC, pp. 486-490, 1992.

[24] I..J. Huang and A. Despain. *Generating instruction sets and microarchitectures from applications.* Int. Conf. on CAD (ICCAD), pp. 391-396, 1994.

[25] IEEE. *IEEE Standard VHDL Language Reference Manual, Std 1076.* IEEE Press, New York, 1987.

[26] Intel Corp. web pages. http://www.intel.com/product/techbriefs/index.html, 1996.

[27] Kalavade, A., and Lee, E. A. *A global criticality/local phase driven algorithm for constrained hardware/software partitioning problem.* In International Workshop on HardwareSoftware Codesign (Sept. 1994).

[28] G. Kruger. *A tool for hierarchical test generation.* IEEE Trans. on CAD, Vol. 10, pp. 519-524, 1991.

[29] D. Ku and G. De Micheli. *High Level Synthesis Under Timing and Synchroniszation Constraints.* Kluwer Academic Publishers, 1992.

[30] D. Ku, and G. D. Micheli. *HardwareC -A Language for Hardware Design (version 2.0).* CSL Technical Report CSL- TR-90-419, Stanford University, Apr. 1990.

[31] R. P. Kurshan. *Reducibility in analysis of coordination.* LNCS 103, pp. 19-39., 1987

[32] D. Lanneer, J. Van Praet, A. Kifli, K. Schoofs, W. Geurts, F. Thoen, and G. Goossens. *CHESS: retargetable code generation for embedded DSP processors.* in: P. Marwedel, G. Goossens (ed.): Code Generation for Embedded Processors, Kluwer Academic Publishers, 1995.

[33] E. Lee. *Programmable DSP architectures, parts i and ii.* IEEE ASSP Magazine, Oct. 1988 & Jan. 1989, 1988.

[34] J. Lee and J.H. Patel. *An architectural level test generator for data path faults and control faults.* Proc. of the Intern. Test Conference, pp. 729-738, 1991.

[35] J. Lee and J.H. Patel. *Hierarchical test generation under intensive global functional constraints.* Proc. 29th Design Automation Conf., pp. 261-266, 1992.

[36] J. Lee and J.H. Patel. *An instruction sequence assembling methodology for testing microprocessors.* Proc. of the Intern. Test Conference, pp. 49-58, 1992.

[37] J. Lee and J.H. Patel. *Architectural level test generation for microprocessors.* IEEE Trans. on ComputerAided Design, pp. 1288-1300 , 1994.

[38] R. Leupers. *Algorithms for address assigment in DSP code generation.* ICCAD, 1996.

[39] R. Leupers and P. Marwedel. *A BDDbased frontend for retargetable compilers.* European Design & Test Conference, 1995.

[40] R. Leupers and P. Marwedel. *Timeconstrained code compaction for DSPs.* Int. Symp. on System Synthesis (ISSS), 1995.

[41] Y.T. S. Li, S. Malik, and A. Wolfe. *Performance estimation of embedded software with instruction cache modeling.* Int. Conf. on Computer-Aided Design (ICCAD), pp. 380-387, 1995.

[42] Li, J., and Gupta, R. K. *HDL Optimizaiton using Timed Descision Tables.* In Proceedings of the 33 rd Design Automation Conference (June 1996).

[43] S. Liao, S. Devadas, K. Keutzer, and S. Tijang. *Code optimization techniques for embedded DSP microprocessors.* 32nd Design Automation Conference, pp. 599-604, 1995.

[44] S. Liao, S. Devadas, K. Keutzer, S. Tijang, and A. Wang. *Storage assignment to decrease code size.* Programming Language Design and Implementation (PLDI), 1995.

[45] Liem and P. Paulin. *FlexWare - A flexible firmware development environment.* Proc. European Design & Test Conference (EDAC-ETC - EUROASIC), pp. 31-37, 1994.

[46] Liem, P. Paulin, and A. Jerraya. *Address calculation for retargetable compilation and exploration of instruction set architectures.* to appear: 33rd Design Automation Conference, 1996.

[47] D. C. Luckham, J. Vera, D. Bryan, and L. *Augustin, Partial Ordering of Event Sets and Their Application to Prototyping Concurrent Timed Systems.* Journal of Systems and Software (July 1993).

[48] P. Marwedel. Introduction. in: P. Marwedel, G. Goossens (ed.): *Code Generation for Embedded Processors*. Kluwer, 1995.

[49] S. Novack, A. Nicolau, and N. Dutt. A unified code generation approach using mutation scheduling. in: P. Marwedel, G. Goossens (ed.): Code Generation for Embedded Processors, Kluwer Academic Publishers, 1995.

[50] P. Paulin. DSP design tool requirements for the nineties: An industrial perspective. HighLevel Synthesis Workshop, Dana Point, Cal., 1992.

[51] P. Paulin, M. Cornero, C. Liem, F. Na abal, C. Donawa, S. Sutarwala, and C. Valderrama. *Trends in embedded systems technology: An industrial perspective.* in: M.G. Sami, G. De Micheli: Hardware/Software Codesign, Kluwer Academic Publishers, 1996.

[52] P. Paulin and C. Liem. *Embedded systems: Trends and tools.* Tutorial at the European Design & Test Conference, 1996.

[53] P. Paulin, C. Liem, T. May, and S. Sutarwala. *DSP design tool requirements for embedded systems: A telecommunications industrial perspective.* Journal of VLSI Signal Processing, pp. 23-47, 1995.

[54] J. V. Praet, G. Goossens, D. Lanneer, and H. De Man. *Instruction set definition and instruction selection for ASIPs.* 7.th Int. Symposium on HighLevel Synthesis, pp. 11-16, 1994.

[55] J. V. Praet, D. Lanneer, G. Goossens, W. Geurts, and H. De Man. *A graph based processor model for retargetable code generation.* European Design & Test Conference, 1996.

[56] J. Rajski and J. Tyszer. *Multiplicative window generators of pseudo random test vectors.* European Design & Test Conference (ED&TC), 1996.

[57] Reinhardt, S. K., Hill, M. D., Larus, J. R., Lebeck, A. R., Lewis, J. C., and Wood, D. A. *The wisconsin wind tunnel: Virtual prototyping of parallel computers.* In ACM SIGMETRICS (1993).

[58] Rimey and Hilfinger. *Lazy data routing and greedy scheduling for application-specific processors.* 21st Annual Workshop on Microprogramming (MICRO-21), pp. 111-115, 1988.

[59] J. P. Singh, W. D. Weber, and A. Gupta,. *SPLASH: Stanford parallel applications for shared memory.* Technical report CSL-TR-91-469, Computer Systems Laboratory, Stanford University, Stanford University, CA 94305, April 1991. Available from ftp://mojave.stanford.edu/pub/splash/report/splash.ps.

[60] M. Strik, J. van Meerbergen, A. Timmer, and J. Jess. *Efficient code generation for in-house DSP cores.* European Design & Test Conference, pp. 244-249, 1995.

[61] Sudarsanam and S. Malik. *Memory bank and register allocation in software synthesis for ASIPs.* Intern. Conf. on Computer-Aided Design (ICCAD), pp. 388-392, 1995.

[62] S.M. Thatte and J.A. Abraham. *Test generation for microprocessors.* IEEE Trans. on Computers, pp. 429-441, 1980.

[63] D. Thomas, and P. Moorby. *The Verilog Hardware Description Language.* Kluwer Academic Publishers, 1996.

[64] Timmer. *Conflict modelling and instruction scheduling in code generation for in-house DSP cores.* 32th Design Automation Conference, 1995

[65] C.A. Valderrama and et al. *A unified model for co-simulation and co-synthesis of mixed hardware/software systems.* European Design & Test Conference, 1995.

[66] J. E. Veenstra, and R. J. Fowler. *MINT: A front end for efficient simulation of shared-memory multiprocessors.* In Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) (Jan. 1994), pp. 201-207.

[67] L. T. Wang., N. E. Hoover, E. H. Porter, and J. J. *Zasio Ssim: A software levelized compiled-code simulator.* In Proceedings of the Design Automation Conference (1987), pp. 2-8.

[68] R. Woudsma. *EPICS, a flexible approach to embedded DSP cores.* Int. Conf. on Signal Processing and Applications and Technology, 1994.

[69] V. Zivojnovic and et al. *DSPstone: A DSP-oriented benchmarking methodology.* Proc. of the Intern. Conf. on Signal Processing and Technology, 1994.

[70] Hui Guo and Sri Parameswaran. *System level pipelining.* In 1996 APCHDL Conference Proceedings, pages 28-33, 1996.

[71] Hui Guo and Sri Parameswaran. *Unrolling loops with indeterminate count in system level pipelines.* In 14th Australia Microelectronics Conference(MICRO'97) Conference proceedings, 1997.

[72] Sri Parameswaran and Hui Guo. *Partitioning of system level pipelines.* In 14th Australia Microelectronics Conference(MICRO'97) Conference proceedings, 1997.

[73] Matthew F. Parkinson, Paul M. Taylor, and Sri Parameswaran. *An automated hardware/software codesign (hsc) using vhdl.* Proceedings of First Asian Pacific Conference on Hardware Description Languages, Standards and Applications, 1993.

[74] Matthew F. Parkinson, Paul M. Taylor, and Sri Parameswaran. *A profiler for automated translation of signal proceeding algorithms into high speed hardware/software hybrid architectures.* Proceedings of 11th Australian Microelectronic Conference, 81-84, Oct. 1993.

[75] W. Wolf. *Guest editor's introduction, Hardware-software codesign.* IEEE Design and Test of Computer, 10:5, 1993.