Introduction to Embedded Systems: A Reading List [*]

Sumit Gupta
Center for Embedded Computer Systems
University of California at Irvine
http://www.cecs.uci.edu/∼sumitg
E-mail: `sumitg@cecs.uci.edu`

1 System Specification: Abstract Modeling and Languages

The design of a system begins with specifying its functionality which includes its behavior over time, i.e., its temporal properties. To help us specify, understand and organize exactly what the functionality of the system should be, we can use a variety of conceptual models. Furthermore, this functionality can be specified at various levels of abstraction such as algorithmic behavior, structural connection of functional blocks or at a logic level as a netlist of gates [1]. These concepts are well covered in books such as [2, 3, 4].

Embedded systems used in real-time applications can be modeled as reactive systems. Specification of reactive systems is discussed in [5, 6]. [7, 8] have a good introduction on how synchronous programming languages such as *Esterel* and *Argos* are used for modeling reactive systems.

Designers use various abstract models for representing the system specification. These abstract models are designed to capture the control and data flow and the timing behavior of the design. Earlier work on using state-oriented models such as *StateCharts* and *SpecCharts* added concurrency and hierarchy into finite state machine models to enhance their ability to capture all aspects of the design of a large system [9, 10, 11, 12].

There are several other modeling styles such as communicating sequential processes, discrete event systems et cetera [13, 14] and a good comparison of the various models is provided in [15].

More recently, several C/C++ like languages have been proposed to specify and model systems [16, 17]. These languages have the ability to capture the design at every level of abstraction and are usually *executable*. Since these languages are similar to C and C++, they come with software tools which compile them and run them on standard UNIX and Windows platforms. This executable specification, coupled with the ability to use the same language at every level of abstraction, greatly aids in the gradual refinement of a design from its behavioral description down to its logic level description. SpecC, SystemC, CynApps and Ocapi are some of the offerings in this area [18, 19, 20, 17]. These system description languages are supported with tools which allow easy integration of IP (Intellectual Property) blocks with a *core* processor to design entire *systems-on-a-chip* [21, 22].

2 Hardware-Software Codesign

Increasing complexity has led system architects to *co-design* hardware and software in embedded systems by automatically synthesizing the hardware and the software from the initial system specification.

[*]Published at the VLSI Design and Test Workshop, Delhi, India, 2000

The evolution of techniques in this field can be followed in [23, 24, 25, 26, 27, 28, 29, 30]. This research work has led to the development of several codesign tools, some of which are now being marketed commercially [31, 32, 33, 34, 35].

Timing estimation, task partitioning and scheduling, hardware synthesis and software synthesis comprise some of the most important and hardest problems in the codesign of embedded systems. Timing estimation involves estimating the timing budgets or constraints for each task in the system [36, 37, 38]. This timing information can then be used for partitioning the tasks into blocks that will be synthesized into hardware and tasks that will be run as software on programmable processors [39, 40, 41, 42, 43, 44]. The process of timing estimation and task partitioning is aided by scheduling techniques, which ensure that the timing constraints of the system specification will be met by the final design [45, 46, 47, 48].

3 Hardware Synthesis

Once the tasks are partitioned into hardware and software, the hardware blocks can be synthesized directly from the behavioral description specified in the system specification; this is known as *high level synthesis*. High level synthesis maps a behavioral description by scheduling the operations on the resources allocated within the given timing constraints and produces a structural register transfer level (RTL) design [49, 50, 51, 52, 53, 54, 55, 56].

The RTL design can then be synthesized using logic synthesis tools that map components such as adders, multipliers et cetera to gates, perform boolean optimization, state minimization and finally, generate the netlist of the final design [2, 57, 58, 59, 60, 61, 4].

4 Software Synthesis

As with hardware synthesis, code is generated for the tasks that are mapped to software after partitioning. These software tasks are mapped to specific programmable processors or *cores* which are usually off-the-shelf DSPs or micro-controllers [62, 63]. Software synthesis algorithms have to adhere to physical memory size constraints while satisfying the timing constraints. Since the system specification is inherently concurrent, and the software generated for many target architectures has to be sequential, the software generation process requires linearization or scheduling of operations [64, 65, 66].

Another crucial component of software synthesis is code optimization. Due to strict timing constraints imposed on embedded systems by real-time concerns, the code optimization problem is more complex than for general-purpose systems [67, 68, 69, 70, 71, 72, 73].

5 Hardware-Software Co-simulation

Systems containing hardware and software components present special problems for simulation. Durng co-simulation, events occuring in both the diverse computation domains need to be co-ordinated without comprimising the speed of the simulation [74, 24, 75, 76, 77].

6 Resources on the Internet

There are a number of sites which maintain links to research groups, researchers, companies, projects et cetera related to embedded systems, computer-aided design and real-time systems [78, 79, 80, 81, 82].

Several efforts have been made in the past to collect benchmarks for high level and logic synthesis of hardware designs. These, along with some newer benchmarks and open source processor cores are also available on the internet [83, 84, 85, 86, 87, 88]

References

[1] D. D. Gajski and R.H. Kuhn. Guest editor's introduction: New VLSI tools. *IEEE Computer*, December 1983.

[2] G. De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

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

[4] S. Gupta and R. K. Gupta. *The VLSI Handbook*, chapter ASIC Design. CRC Press and IEEE Press, 2000. Chapter 64.

[5] A. Burns and A. Wellings. *Real-Time Systems and Their Programming Languages*. Addison Wesley, 1990.

[6] H. Kopetz. *Real-Time Systems*. Kluwer Academic, Boston, MA, USA, 1997.

[7] G. Berry. *The Foundations of Esterel*. MIT Press, 2000. Editors: G. Plotkin, C. Stirling and M. Tofte.

[8] N. Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic, 1993.

[9] D. Harel. StateCharts: a visual formalism for complex systems. *Science of Programming*, 8, 1987.

[10] M. von der Beeck. A comparison of StateCharts variants. In *Proc. of Formal Techniques in Real Time and Fault Tolerant Systems*, pages 128–148, Berlin, 1994. Springer-Verlag.

[11] F. Vahid, S. Narayan, and D. D. Gajski. SpecCharts: A VHDL front-end for embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(6):694–706, June 1995.

[12] A. Girault, B. Lee, and E. A. Lee. Hierarchical finite state machines with multiple concurrency models. Technical Report UCB/ERL M97/57, Electronics Research Laboratory, Berkeley, CA 94720, October 1998.

[13] C. A. R. Hoare. Communicating sequential processes. *Comm. of the ACM*, 21(8), 1978.

[14] C. Cassandras. *Discrete Event Systems, Modeling and Performance Analysis*. Irwin, Homewood IL, 1993.

[15] E.A. Lee and A. Sangiovanni-Vincenteli. Comparing models of computation. In *Proceedings of the International Conference on Computer-Aided Design*, November 1996.

[16] R. K. Gupta and S. Y. Liao. Using a programming language for digital system design. *IEEE Design and Test of Computers*, April 1997.

[17] S. Vernalde, P. Schaumont, and I. Bolsens. An object oriented programming approach for hardware design. In *IEEE Computer Society Workshop on VLSI*, Orlando, April 1999.

[18] D. D. Gajski, J. Zhu, R. Domer, A. Gerstlauer, and S. Zhao. *SpecC: Specification Language and Methodology*. Kluwer Academic Publishers, January 2000.

[19] Synopsys Inc., http://www.systemc.org. *SystemC Reference Manual 1.0*.

[20] CynApps Inc., http://www.cynlib.com. *Cynlib Reference Manual*.

[21] R. Domer and D. Gajski. Reuse and protection of intellectual property in the SpecC system. In *Proceedings of the Asia and South Pacific Design Automation Conference*, Yokohama, Japan, January 2000.

[22] Y. Zorian and R. K. Gupta. Introduction to core-based design. *IEEE Design and Test of Computers*, October 1997.

[23] R.K. Gupta and G. De Michelli. Hardware-software cosynthesis for digital systems. *IEEE Design and Test of Computers*, September 1993.

[24] R.K. Gupta, C.N. Coelho Jr., and G. De Michelli. Synthesis and simulation of digital systems containing interacting hardware and software components. In *Proceedings of the 29th Design Automation Conference*, June 1992.

[25] R. Ernst, J. Henkel, and T. Benner. Hardware-software cosynthesis for microcontrollers. *IEEE Design and Test*, 12, 1993.

[26] M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, and A. Sangiovanni-Vincentelli. A formal specification model for hardware/software codesign. In *Proceedings of International Workshop on Hardware-Software Codesign*, October 1993.

[27] P. Chou, R. Ortega, and G. Borriello. The Chinook hardware/software co-synthesis system. In *International Symposium on System Synthesis*, Cannes, France, September 1995.

[28] A. Kalavade and E. Lee. A hardware/software codesign methodology for DSP applications. *IEEE Design and Test*, September 1993.

[29] D.E. Thomas, J.K. Adams, and H. Schmitt. A model and methodology for hardware-software codesign. *IEEE Design and Test of Computers*, 10(3):6–15, 1993.

[30] T. Ismail, M. Abid, and A. Jerraya. COSMOS: A codesign approach for communicating systems. In *Proceedings of the International Workshop on Hardware-Software Codesign.* IEEE CS Press, 1994.

[31] F. Balarin, P. Giusto, A. Jurecska, C. Passerone, E. Sentovich, B. Tabbara, M. Chiodo, H. Hsieh, L. Lavagno, A. Sangiovanni-Vincentelli, and K. Suzuki. *Hardware-Software Co-Design of Embedded Systems, The POLIS Approach.* Kluwer Academic Publishers, April 1997.

[32] J. Henkel and R. Ernst. A hardware-software partitioner using a dynamically determined granularity. In *Proceedings of the Design Automation Conference*, 1997.

[33] K. Rompaey, D. Verkest, I. Bolsens, and H. De Man. CoWare – a design environment for heterogeneous hardware/software systems. In *Proceedings of the European Design Automation Conference*, 1996.

[34] C. Valderrama, M. Romdhani, J. Daveau, G. Marchioro, A. Changuel, and A. Jerraya. Cosmos: A transformational co-design tool for multiprocessor architectures. In J. Staunstrup and W. Wolf, editors, *Hardware/Software Co-Design: Principles and Practice.* Kluwer Academic Publishers, 1997.

[35] SpecC system home page. http://www.cecs.uci.edu/~specc.

[36] A. Mathur, A. Dasdan, and R. K. Gupta. Rate analysis of embedded systems. *ACM Trans. on Design Automation of Electronic Systems*, 3(3):408–36, July 1998.

[37] A. Dasdan, D. Ramanathan, and R. K. Gupta. A timing-driven design and validation methodology for embedded real-time systems. *ACM Trans. on Design Automation of Electronic Systems*, 3(4):533–53, October 1998.

[38] S. Malik, M. Martonosi, and Y.-T. S. Li. Static timing analysis of embedded software. In *Proc. 34st Design Automation Conf.*, pages 147–52. ACM/IEEE, 1997.

[39] R. K. Gupta. Special issue on partitioning methods for embedded systems. *Design Automation for Embedded Systems*, 2(2):123–261, March 1997.

[40] J. Madsen, J. Grode, and P. Knudsen. Hardware/software partitioning using the LYCOS system. In J. Staunstrup and W. Wolf, editors, *Hardware/Software Co-Design: Principles and Practice.* Kluwer Academic Publishers, 1997.

[41] K. Olokutun, R. Helaihel, J. Levitt, and R. Ramirez. A software-hardware cosynthesis approach to digital system simulation. *IEEE Micro*, 14(4):48–58, August 1994.

[42] A. Kalavade and E.A. Lee. A global criticality/local phase driven algorithm for the constrained hardware/software partitioning problem. In *Proceedings of the International Workshop on Hardware-Software Codesign*, 1994.

[43] F. Vahid and D.D. Gajski. Specification partitioning for system design. In *Proceedings of the 29th Design Automation Conference*, June 1992.

[44] R. Ernst and J. Henkel. Hardware-software codesign of embedded controllers based on hardware extraction. In *Proceedings of the International Workshop on Hardware-Software Codesign*, 1992.

[45] J. Xu and D. L. Parnas. On satisfying timing constraints in hard real-time systems. *IEEE Trans. Software Eng.*, 19(1):70–84, January 1993.

[46] P. Chou, E.A. Walkup, and G. Borriello. Scheduling for reactive real-time systems. *IEEE Micro*, 14(4):37–47, August 1994.

[47] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[48] M. H. Klein, T. Ralya, B. Pollak, and R. Obenza. *A Practitioner's Handbook for Real-Time Analysis : Guide to Rate Monotonic Analysis for Real-Time Systems.* Kluwer Academic Publ., Boston, MA, USA, 1993.

[49] D. D. Gajski, N. D. Dutt, Allen C-H. Wu, and Steve Y-L. Lin. *High-Level Synthesis: Introduction to Chip and System Design.* Kluwer Academic, 1992.

[50] R. Camposano and W. Wolf. *High Level VLSI Synthesis.* Kluwer Academic, 1991.

[51] A. A. Jerraya, H. Ding, P. Kission, and M. Rahmouni. *Behavioral Synthesis and Component Reuse with VHDL.* Kluwer Academic Publishers, 1997.

[52] D. D. Gajski and L. Ramachandran. Introduction to High-level synthesis. *IEEE Design and Test of Computers*, Winter 1994.

[53] G. De Micheli, D. C. Ku, F. Mailhot, and T. Truong. The Olympus synthesis system for digital design. *IEEE Design and Test of Computers*, pages 37–53, October 1990.

[54] R. Potasman, J. Lis, A. Nicolau, and D. Gajski. Percolation based synthesis. In *Design Automation Conference*, 1990.

[55] C.P. Ravikumar, S. Gupta, and A. Jajoo. Synthesis of testable RTL designs using adaptive simulated annealing algorithm. In *Eleventh International Conference on VLSI Design, India*, 1998.

[56] L.C.V. dos Santos and J.A.G. Jess. A reordering technique for efficient code motion. In *Design Automation Conference*, 1999.

[57] G. D. Hachtel and F. Somenzi. *Logic Synthesis and Verification Algorithms.* Kluwer Academic, 1996.

[58] R. H. Katz. *Contemporary Logic Design.* Benjamin/Cummings Publishing, 1994.

[59] E. J. McCluskey. *Logic Design Principles.* Prentice Hall, 1996.

[60] S. Devadas and A. Ghosh an K. Keutzer. *Logic Synthesis.* McGraw-Hill, 1994.

[61] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. MIS: A multiple-level logic optimization system. *IEEE Transactions on CAD/ICAS, CAD-6*, November 1987.

[62] Texas Instruments. *TI TMS320C6x User's Guide.*

[63] Philips Semiconductor. *Trimedia TM1000 Programmable media processor databook.*

[64] P. Chou and G. Borriello. Software scheduling in the co-synthesis of reactive real-time systems. In *Proceedings of the Design Automation Conference*, 1994.

[65] G. Goosens, K. Rabaey, J. Vanderwalle, and H. De Man. An efficient microcode compiler for custom DSP-processors. In *Proceedings of the International Conference on Computer-Aided Design*, 1987.

[66] R.K. Gupta, C. Coelho, and G. De Michelli. Program implementation schemes for hardware-software systems. *IEEE Computer*, January 1994.

[67] P. Murthy, S. Bhattacharyya, and E.Lee. APGAN and RPMC: Complementary heuristics for translating DSP block diagrams into efficient software implementations. *Design Automation of Embedded Systems*, 1997.

[68] W. Sung, J. Kim, and S. Ha. Memory efficient software synthesis from dataflow graphs. In *International Symposium on System Synthesis*, March 1996.

[69] H.Meyr S.Ritz, S.Pankert. High level software synthesis for signal proc. systems. In *International Conference on Acoustics, Speech and Signal Proc*, 1990.

[70] R. Gupta. Operation serializability and software generation for embedded systems. In *European Design and Test Conferece*, March 1996.

[71] D. Powell, E. Lee, and W. Newman. Direct synthesis of optimized DSP assembly code from signal flow block diagrams. In *International Conference on Acoustics, Speech and Signal Proc*, March 1990.

[72] M. Chiodo and et al. Synthesis of software programs for embedded control applications. In *Proceedings of the Design Automation Conference*, 1995.

[73] S. Gupta, M. Miranda, F. Catthoor, and R. Gupta. Analysis of high-level address code transformations for programmable processors. In *Design, Automation and Test in Europe*, 2000.

[74] J. Buck, S. Ha, E.A. Lee, and D.G. Messerschmitt. Ptolemy: a framework for simulating and prototyping heterogeneous systems. *International Journal of Computer Simulation*, special issue on Simulation Software Development(3), January 1990.

[75] S. Lee and J.M. Rabaey. A hardware/software co-simulation environment. In *Proc. of the Int. Workshop on Hardware-Software Codesign*, October 1993.

[76] J. Rowson. Hardware/software co-simulation. In *Proceedings of the 31st Design Automation Conference*, June 1994.

[77] G. Kuch, U. Kebschull, and W. Rosenstiel. A prototyping architecture for hardware/software codesign in the COBRA project. In *Proceedings of the International Workshop on Hardware-Software Codesign*, 1994.

[78] S. Gupta. Links to embedded system codesign and CAD research groups, people and projects. http://www.cecs.uci.edu/∼sumitg/CadPages.html.

[79] C. Daly. Embedded systems internet resources. http://www.compapp.dcu.ie/∼cdaly/embed/embedsys.html.

[80] SIGDA. Sigda's links to CAD sites. http://sigda.acm.org/Links/CAD_Sites/.

[81] Design Automation Cafe. Eda community web site. http://www.dacafe.com.

[82] Design and Reuse. System on a chip, IP core reuse web site. http://www.design-reuse.com.

[83] S. Gupta. Links to benchmarks useful for research in design of embedded systems. http://www.cecs.uci.edu/∼sumitg/CadPages.html#Bench.

[84] UC Irvine: Dutt and Panda. 1995 High-Level synthesis design repository. ftp://ftp.cecs.uci.edu/pub/hlsynth/HLSynth95, 1995.

[85] MCNC. Benchmarks for the fifth international workshop on High-Level synthesis. Available via anonymous FTP at mcnc.mcnc.org, 1991.

[86] CMU Low Power Group. Benchmarking an open source DSP core. http://www.ece.cmu.edu:80/∼lowpower/benchmarks.html.

[87] Open Cores. Creating a set of open source IP cores. http://www.opencores.org.

[88] European Space Agency. Leon: Open source SPARC compatible processor. http://www.estec.esa.nl/wsmwww/leon.