Toward Fast and Accurate Architecture Exploration in a Hardware/Software Codesign Flow

DIRK STROOBANDT
Electronics and Information Systems (ELIS) Department
Ghent University
Sint-Pietersnieuwstraat 41, B-9000 Gent
BELGIUM
dstr@elis.rug.ac.be http://www.elis.rug.ac.be/˜dstr/

Abstract: – Embedded systems design combines software implementations running on an on-chip processor and dedicated hardware components. It also introduces IP-components (Intellectual Property) to be reused and integrated in Systems-on-a-Chip (SoCs). This means a tremendous paradigm shift from the traditional system design. This paper introduces an embedded systems design flow in which the major challenge is the exploration of the design space for optimal architecture configurations. We show that automation of this architecture exploration phase heavily relies on fast and relatively accurate performance estimates for both hardware and software implementations simultaneously. For performance estimation of hardware, we advocate the introduction of a priori interconnect estimations in architecture exploration tools and show how such estimates can be used beneficially.

Key-Words: – Embedded Systems Design, Hardware/Software codesign, A Priori Interconnect Estimation.

1 Introduction

Today, the technological world is no longer dominated by microprocessors. Many systems around us (cellular phones, cars, airplanes, washing machines, etc.) contain and are controlled by electronic chips consisting of both processor blocks to execute software programs and dedicated hardware blocks for the fast evaluation of specific functions. Such systems are called *embedded systems*. In fact, embedded systems already largely outnumber computer chips. With the progress in technological capabilities (driven by Moore's law, stating that the number of transistors on a chip doubles every 18 months) it is currently possible to combine several components of embedded systems on a single chip. This has become known as System-on-a-Chip (SoC) design.

The design of embedded systems and of SoCs introduces many design challenges. Analogue and RF components are starting to be integrated together with the rest of the system and their design can no longer be seen separate from the overall system design. Another major challenge is the increased freedom to choose the system architecture and tailor it to the application at hand. The number of processing cores available for embedded systems is increasing rapidly and also predesigned hardware blocks (Intellectual Property - IP - blocks) are becoming widely available. Combine this with the choice between either a software implementation on a processor and a hardware approach on a dedicated hardware part (or IP block). The conclusion: an exponential increase in the number of design choices. Within this vast range of choices, the embedded system designer has to optimize for power, timing, area, yield, cost, or other performance criteria.

Evaluating the performance criteria in detail for all possible designs and then picking the best one, is simply impossible. The evaluation of the different architectures has to be based on preliminary, very fast, but reasonably accurate performance estimates. In this paper, we focus on existing techniques for estimating performance parameters of hardware IP blocks based on a priori interconnect prediction. We break a lance for further research in the domain of a priori performance estimates of hardware and software implementations in the embedded systems context. Based on such estimates, automating the exploration of architecture design options becomes feasible and an easier and more automated design flow for embedded systems comes within reach.

Section 2 presents an overview of the embedded systems design flow and shows why very fast performance estimates are crucial in such a flow. Current research on performance estimates in hardware is introduced in section 3.

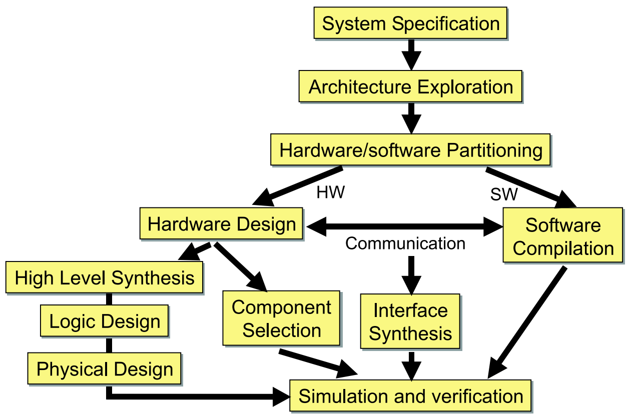

Fig. 1. The embedded systems design flow.

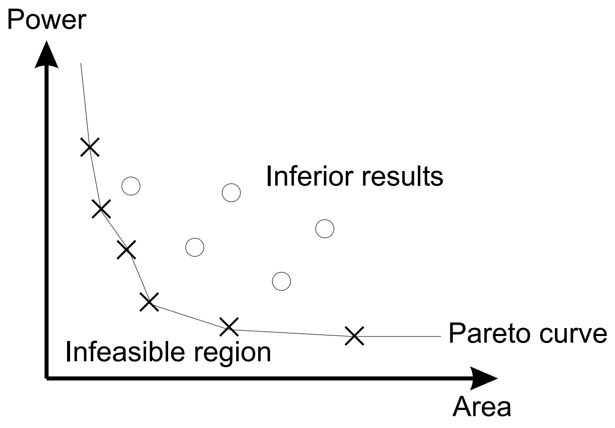

Fig. 2. The Pareto curve of Pareto-optimal results separates the infeasible region from the region where inferior results are found.

2 Architecture Exploration

The design of embedded systems is a relatively new research topic in that it combines hardware and software design issues. The embedded design flow is depicted in figure 1.

The design starts with the correct *specification* of the system, its functionality and performance requirements (data throughput, power dissipation, cost, etc.). The specification should describe the system without already distinguishing software from hardware parts to leave all options open. A unified language to describe software and hardware should be found that simplifies both the design of the system (or parts of it) in hardware and in software. It should also enable easy verification at all design steps. Standard software languages (such as C, C++, or JAVA) lack some of the basic constructs to describe hardware. They have difficulty in describing timing issues, concurrency, structural hierarchy and state transitions. Standard hardware description languages (such as VHDL and Verilog) do not have explicit elements for state transitions and have problems describing communication. They also make a transition to software languages almost impossible. There have been several initiatives to augment these standard languages with new constructs to enable embedded system design. JAVA was used in the JavaTime approach at the University of California at Berkeley [15]. The design of a new description language based on VHDL was the goal of an industrial consortium in 1996 and was supposed to lead to a System-Level Design Language (SLDL). However, the most promising approaches took the software language C as the basis and augmented it with elements to describe concurrency, state transitions, structural and behavioural hierarchy, exception handling, timing, communication and synchronisation. SpecC [12] originated from the University of California at Irvine but the language that is most probable to become the new standard for embedded system design is SystemC [13]. It is endorsed by a consortium of leading design companies and gets a lot of attention at various conferences and symposia. It is bound to become an IEEE standard soon.

The next step in the embedded system design process of figure 1 is *architecture exploration*. This is the most important step since it is at this stage that the most important design decisions are taken. Architecture exploration mainly consists of searching the vast design space for possible architectures (combinations of design entities) that meet the target requirements as good as possible. Principle requirements are a low power dissipation and a low design cost. Searching the design space requires extensive design libraries to find the right components, IP-blocks, or software descriptions (together with the right processor to run it on). Even more important is the ability to predict the performance of each of the individual library entities within the configuration currently under investigation. Indeed, one has to validate choices made in this design step by obtaining performance estimates of processing speed, memory allocation, silicon area, power dissipation, etc. upfront. It is clear that such a validation can not be done by actually designing all possible design configurations all the way through (that would take ages). In the design exploration phase, very fast and reasonably accurate performance estimates need to be available. We will elaborate on this issue in the following section.

The use of fast performance estimates for design options results in so-called Pareto-curves (figure 2). Each design solution corresponds to a performance

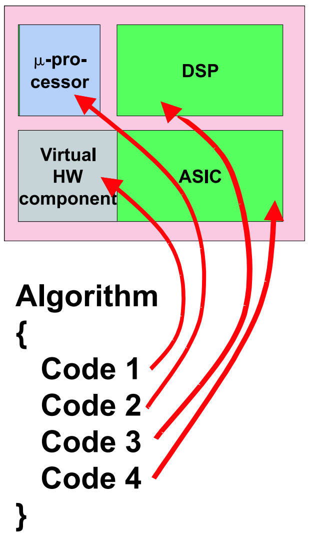

Fig. 3. Hardware/software partitioning in an embedded system design flow.

estimate for a certain optimization criterion, such as power and area. Generally, it is not possible to have one solution that is optimal for both criteria at the same time. This results in an entire set of solutions that have an optimal combination of performance results. These are called Pareto-optimal. For such solutions, there does not exist another solution that is better in one criterion without being worse in the other criterion. All Pareto-optimal solutions are on a *Pareto front* or *Pareto curve*. Solutions above the curve or to the right of it are inferior since they are dominated by a Pareto-optimal solution that either consumes less area or less power.[1] Solutions to the left or below the Pareto front are infeasible (otherwise a solution on the Pareto front would be dominated by this solution and, by definition, that solution would no longer be on the front). The goal thus is to find the Pareto-optimal solutions and only use those for further exploration (depending on specific requirements on some of the performance criteria). All other solutions can be discarded. Because we are only interested in Pareto-optimal solutions, we can use very fast estimates as a first indication of whether or not a proposed solution is far from or near to the Pareto front. If it is far away, it can be discarded immediately. If it is close, we can gradually improve the accuracy of the estimates (of course taking more time to do this) to assess the actual quality of the proposed solution.

[1]Note that the definition of inferior regions and infeasible regions is presented here for optimization criteria that are minimized. For maximized criteria (such as speed), the Pareto front location will be different.

The third step in the embedded system design flow, *hardware/software partitioning*, has been explored extensively in the recent years. The main task here is to distribute the various tasks the system has to perform to specific hardware blocks and specific software instances to be executed on one or more processors (see figure 3). A first requirement for this is to be able to decide whether a task should be performed in hardware or in software. For this choice, similar performance estimates as in the architecture exploration step are needed, only this time for a fixed architecture (with known processors and known hardware resources). Since the number of possibilities is lower, a more time-consuming estimation method may be used that provides more accurate performance estimates. However, in real-time systems, the scheduling of tasks has to be done in real time and fast performance estimates are again needed.

Hardware/software partitioning is much better understood today than architecture exploration (which is still mainly a manual operation). Several tools have been introduced that perform the partitioning (with different levels of expert designer guidance possible). Some examples of such tools are GPP (General Purpose Partitioner) from UC Riverside [14], Cosyma (U. Braunschweig [6]), Lycos (T.U. Denmark [19]), POLIS (UC Berkeley [21]), Chinook (U. Washington [3]) and CoWare (U. Leuven, Belgium, now a spin-off company).

If the software branch of the design flow (figure 1) is chosen, this software has to be compiled to a specific processor available on the embedded system platform. Currently there exist many application specific processors (ASIP's) that have been optimized for specific tasks. Optimal software compilation is a separate research domain and one of the main challenges is to find good and fast performance estimates that mimic the behaviour of optimized compilers without actually compiling code.

Hardware design will more and more evolve into the reuse of Intellectual Property (IP) blocks since it becomes infeasible to design every billion-transistor design from scratch. In that case the design effort is reduced to the selection of the right IP-component. New initiatives such as the Virtual Socket Interface Alliance (VSIA, [33]) and the extensive IP-reuse libraries offered by, e.g., Design and Reuse [7] already provide such services. However, parts of designs will still need to be tailored to the application domain and the design of new application specific hardware IP will still be needed. The design of such blocks goes through a conventional hardware design flow of high level synthesis, logic design and physical design. Going back to the architecture exploration step, we observe an additional huge challenge in estimating the performance of a digital design block before any of the implementation details are known.

A major aspect of embedded systems design is the *communication* between the processor-based software parts and the dedicated hardware blocks. More than ever before, the design of communication protocols is coming in the picture. Busses have to be synthesized between processors, memory and IP-blocks. Since IP-blocks come with their own communication protocols, translators will need to be synthesized as well. And all this is now moved from board-level design to on-chip design since complete embedded systems are now integrated in a single chip (System-on-a-Chip). The separation of computations and communication on chip has become essential for supporting "plug-and-play" for IP-blocks.

Finally, all the design steps in the embedded system design flow (figure 1) have to be verified and extensive simulations have to be run on the system to verify that it performs as specified.

From the design flow, it is clear the most crucial step in embedded system design is the architecture exploration. In that step, the designer still has all the freedom to choose an implementation that solves a design problem in an optimal way. Choices made in this design step have a significant impact on the overall design outcome. Hence, each design choice should be carefully selected and verified. We are still far from automating this design step and it is still mainly a manual job for an experienced designer. However, the complexity is huge and the design space enormous so that even expert designers can never fully cover all possibilities. A manual architecture selection will at best only explore part of the Pareto-optimal solutions or might even stick to sub-optimal solutions that are not on the Pareto-front at all.

Although an automatic architecture exploration is still unachievable with current tools, recent years have brought new tools that facilitate the search through the design space by providing extensive libraries and performing the comparison of design solutions automatically. However, the performance figures for each individual component must be introduced in such systems (by hand or included in the component library) and seldom take the surrounding architectural environment into account, let alone the algorithmic context in which the component (either hardware or software) is supposed to operate. Fast automatic performance estimates are the key for a new generation of architecture exploration tools that are able to provide a set of Pareto-optimal architectures consisting of software and hardware components from a library with a correct (accurate enough) weighting of the performance criteria.

Early evaluation of processor architecture decisions are found in, e.g., [2], [11]. Many people are currently working on the analysis of programming code and its impact on performance. Others investigate different processor architectures (e.g., memory management [20]) or perform power estimates for processor architectures based on simple notions of the type of programs that will run on them, e.g., [10]. These are only some examples of the research directions toward software performance estimates that could be used for architecture exploration. In the next section, we describe some current research directions in hardware performance estimation in somewhat more detail.

3 Hardware Performance Estimates

With Moore's law in place (stating that the number of transistors on a chip doubles every 18 months), the performance of a chip is becoming more and more dependent on the interconnections. Relative to the functional transistors, interconnects take more area on the chip, consume more power, are responsible for most of the signal delay, and contribute most to the total cost of the chip. Today, interconnects are the limiting factor for both performance and density, i.e., the value and the cost of a VLSI system.

Where chip design used to be focused on the optimization of the functional blocks, these days one has to account for the wiring as well. Today, a focal point for improved interconnect modelling, more cost-effective system architectures, and more productive design technology centers on new methods and models for *a priori system-level interconnect prediction.* Although basic works in this area are almost thirty years old, no cohesive research community for interconnect prediction was established until the first international workshop on System-Level Interconnect Prediction (SLIP) [25] in April 1999. An overview of the basic concepts of and recent research work on a priori interconnect prediction is presented in a book completely devoted to this field [24] and a collection of recent research work can be found in two special issues of IEEE Transactions on VLSI Systems [30], [31].

3.1 Interconnect Prediction Applications

In hardware design, the physical design step transforms a structural description of a design to a real chip layout. It consists of the consecutive steps of floorplanning (roughly deciding where functional blocks will be placed on the chip layout), placement (a detailed placement of the gates on the chip layout) and routing (assigning routes to the interconnects between gates). The Computer-Aided Design (CAD) tools for placement optimize for small interconnection lengths between gates or, alternatively, for small delays in the critical wires.[2] This requires knowledge of the interconnect routing. Routing, on the other hand, can only be done after the place of the gates is known. Hence, during the layout of computer chips, several iterations between placement and routing are needed. To reduce or even eliminate the number of placement/routing iterations, a priori interconnection length estimations are very helpful because they allow an evaluation of placements without a routing step, leading to a better initial placement and a better initial routing result. CAD tools for layout generation therefore especially benefit from a priori (i.e., pre-layout) wire length estimation techniques [23]. The same techniques can also be used even earlier in the design flow for evaluating architecture exploration solutions.

Current applications of a priori interconnect estimation are found in technology extrapolation [1]. For estimations of the performance of future designs, very little is known about the design and a priori techniques are essential. The same applies to the evaluation of new chip architectures. A priori estimates immediately provide a solid ground for drawing preliminary conclusions about the benefits of new chip architectures and for comparing different architectures.

3.2 Interconnect Prediction and Rent's Rule

A priori interconnect estimation typically requires three models (figure 4): (i) a circuit model (ii) a model for the physical chip architecture in which the circuit will be placed, and (iii) a model for the layout generation (placement and routing). Current practice models the circuit as a collection of logic gates connected to each other through interconnections, models the chip architecture as a (two-dimensional) Manhattan grid, and assumes a "good" placement (i.e., one that successfully minimizes wire lengths) and enough space available to route all interconnects along the shortest (Manhattan) path. However, these simple models do not suffice to make powerful estimations about the resulting layout. For this, one needs to have a notion of (i) the complexity of the interconnection topology and (ii) the quality of the placement. This information is provided by the so-called Rent's rule.

In 1971 Landman and Russo [18] described a relationship between the average number of terminals T of a part of the circuit (a *module*) and the average number of logic gates (basic logic blocks B) inside the module (basically a relation between interconnect and logic). This relation is given by

$$T = tB^p \tag{1}$$

and is called *Rent's rule.* The parameter t is the average number of terminals per logic gate and the exponent p is the *Rent exponent.* Its value depends on the complexity of the interconnect topology and on the quality of the placement.[3] Rent's rule proves to be valid for most designs and it is recently shown that it applies to any homogeneous design [5].

3.3 Donath's Model

Rent's rule has been used in wire length estimation for the first time by Donath in 1979 [9]. The idea is simple: the circuit and the Manhattan grid are both partitioned into equally large parts and each circuit part is mapped to a grid part. This partitioning process is repeated recursively until all logic gates are assigned to a single grid cell in the Manhattan grid. The average number of interconnections between parts at a certain hierarchical level is estimated from Rent's rule (details can be found in [9]) and the average length of a connection at each hierarchical level is estimated by making some simple assumptions.

Despite the simplicity of Donath's model, it is able to predict the scaling of the average wire length as a function of circuit size quite well. However, Donath found that his quantitative average wire length predictions were approximately a factor of 2 off from measured values for real circuits.

In [27], [28] Stroobandt et al. improved Donath's model. Independently, Davis et al. [8] proposed a non-hierarchical method for wire length estimation that leads to very similar results. A more detailed analysis of wire length models can be found in [5], [24].

[2]Critical wires are those wires that are on a long chain of interconnections that have to be traversed by a signal in a single clock cycle.

[3]Both a more complex interconnect topology and a less optimized placement are reflected in a higher Rent exponent. A discussion of these different aspects of the Rent exponent can be found in [32].

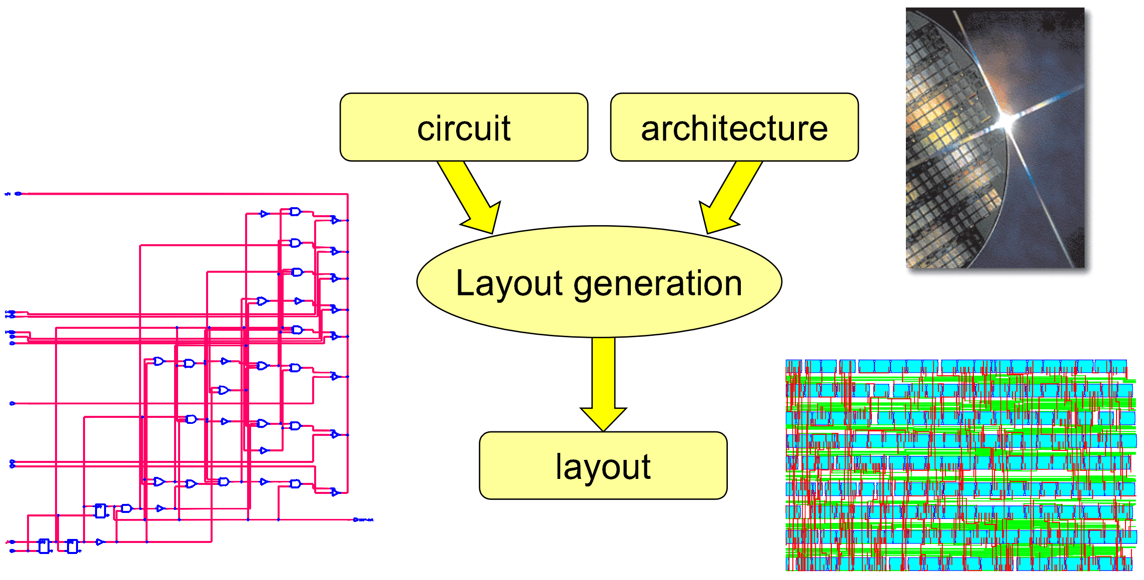

Fig. 4. The three components of models for physical design: the circuit, the chip architecture and the layout generation. The combination of these models results in the (model for the) layout.

3.4 Other Model Extensions

The introduction of three-dimensional chip architectures has induced the need to extend the estimation models to three-dimensional grids [26]. Other extensions include taking into account external interconnections [29] and multi-terminal nets [22] and estimating the global wire length distribution (for heterogeneous systems-on-a-chip) separately [34].

Current research uses these interconnect estimation models for the prediction of routing resources (area and number of layers) [16], [17] and chip yield prediction [4]. Future research is looking at delay characterization and the prediction of power dissipation in interconnects. With these extensions, fast and relatively accurate prediction of hardware performance is coming near. Introducing these techniques in architecture exploration tools promises to dramatically improve the ease of designing embedded systems.

4 Conclusion

The challenges imposed by the combination of software and hardware in the design of embedded systems, together with the increasing complexity as a result of Moore's law, crystallize in the most important step of embedded system design: architecture exploration. To be able to automate that step, fast and efficient performance estimates are needed for both software and hardware implementations of system parts. With these estimates, an efficient search of the entire design space becomes feasible and the designer can limit the final implementation possibilities to those that are Pareto-optimal.

In this paper, we have presented an overview of recent evolutions in hardware performance estimation based on a priori interconnect prediction techniques. It took thirty years for research on system-level interconnect prediction to mature but significant progress has been made in the last couple of years, mainly because the interconnect problems did not become apparent until recently. We have introduced the field and the models that are the basis for wire length prediction: Rent's rule and Donath's wire length estimation model. Recent advances have been highlighted and we have indicated how these techniques could benefit embedded systems architecture exploration.

References:

[1] A. Caldwell, Y. Cao, A. Kahng, F. Koushanfar, H. Lu, I. Markov, M. Oliver, D. Stroobandt, and D. Sylvester. GTX: The MARCO GSRC technology extrapolation system. In *IEEE/ACM Design Automation Conf.*, pages 693–698, June 2000.

[2] S. Chai, T. Taha, D. Wills, and J. Meindl. Heterogeneous architecture models for interconnect-motivated system design. *IEEE Trans. on VLSI Systems, Special Issue on System-Level Interconnect Prediction*, vol. 8 (no. 6): pages 660–670, 2000.

[3] The Chinook Project, A hardware-software co-synthesis CAD tool for real-time embedded systems. University of Washington, CSE Department. www.cs.washington.edu/research/projects/lis/www/chinook/

[4] P. Christie and J. Pineda de Gyvez. Pre-layout prediction of interconnect manufacturability. In *Intl. Workshop on System-Level Interconnect Prediction (SLIP 2001)*, pages 167–173, March 2001.

[5] P. Christie and D. Stroobandt. The interpretation and application of Rent's rule. *IEEE Trans. on VLSI Systems, Special Issue on System-Level Interconnect Prediction*, vol. 8 (no. 6): pages 639–648, December 2000.

[6] COSYMA, COSYnthesis for eMbedded Architectures. www.ida.ing.tu-bs.de/research/projects/cosyma/overview/

[7] D&R, Design and reuse http://www.us.design-reuse.com/

[8] J. A. Davis, V. K. De, and J. D. Meindl. A stochastic wire-length distribution for gigascale integration (GSI) – PART I: Derivation and validation. *IEEE Trans. on Electron Devices*, 45(3):580–589, March 1998.

[9] W. E. Donath. Placement and average interconnection lengths of computer logic. *IEEE Trans. Circuits & Syst.*, CAS–26:272–277, 1979.

[10] L. Eeckhout and K. De Bosschere. Early design phase power/performance modeling through statistical simulation. In P. Bose, editor, *Proc. 2001 IEEE Intl. Symp. on Performance Analysis of Systems and Software*, pages 10–17, Tucson, AZ, USA, November 2001. IEEE.

[11] L. Eeckhout and K. De Bosschere. Hybrid analytical-statistical modeling for efficiently exploring architecture and workload design spaces. In *Proc. 2001 Intl. Conf. on Parallel Architectures and Compilation Techniques (PACT-2001)*, pages 25–34, September 2001.

[12] D. D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, and S. Zhao. *SpecC: Specification Language and Design Methodology*. Kluwer Academic Publishers, March 2000.

[13] T. Grötker, S. Liao, G. Martin, and S. Swan. *System Design with SystemC*. Kluwer Academic Publishers, March 2002.

[14] E. Hwang, F. Vahid, and Y.C. Hsu. FSMD Functional Partitioning for Low Power. *Design Automation and Test in Europe Conference*, pp. 22-28, March 1999.

[15] JavaTime. University of California at Berkeley. http://www.xcf.berkeley.edu/ jmacd/weld/guide.html

[16] A. Kahng, S. Mantik, and D. Stroobandt. Toward accurate models of achievable routing. *IEEE Trans. on Comp.-Aid. Des., special issue on Physical Design*, vol. 20 (no. 5): pages 648–659, May 2001.

[17] A. Kahng and D. Stroobandt. Wiring layer assignments with consistent stage delays. In *Proc. ACM Intl. Workshop on System-Level Interconnect Prediction (SLIP 2000)*, pages 115–122, April 2000.

[18] B. S. Landman and R. L. Russo. On a pin versus block relationship for partitions of logic graphs. *IEEE Trans. on Comput.*, C–20:1469–1479, 1971.

[19] J. Madsen, J. Grode, P. Voigt Knudsen, M. E. Petersen, and A. Haxthausen. Lycos: The lyngby co-synthesis system. *Design Automation of Embedded Systems*, vol. 2 (no. 2): pp. 195 - 236, March 1997.

[20] P.R. Panda, F. Catthoor, N.D. Dutt et al. Data and memory optimization techniques for embedded systems *ACM Trans. Design Automation of Electronic Systems*, vol. 6, no. 2, pp. 149 - 206, April 2001

[21] POLIS, A Framework for Hardware-Software Co-Design of Embedded Systems UC Berkeley. www-cad.eecs.berkeley.edu/Respep/Research/hsc/abstract.html

[22] D. Stroobandt. Multi-terminal nets do change conventional wire length distribution models. In *Intl. Workshop on System-Level Interconnect Prediction (SLIP 2001)*, pages 41–48, March 2001.

[23] D. Stroobandt. A priori system-level interconnect prediction: Rent's rule and wire length distribution models (tutorial). In *Intl. Workshop on System-Level Interconnect Prediction (SLIP 2001)*, pages 3–21, March 2001.

[24] D. Stroobandt. *A priori Wire Length Estimates for Digital Design*. Kluwer Academic Publishers, April 2001. 324 pages. ISBN no. 0 7923 7369 x.

[25] D. Stroobandt and A. Kahng. Workshop notes of the First Intl. Workshop on System-Level Interconnect Prediction, April 1999. Notes available at http://www.elis.rug.ac.be/˜dstr/SLIP.html.

[26] D. Stroobandt and J. Van Campenhout. Estimating interconnection lengths in three-dimensional computer systems. *IEICE Trans. on Inf. & Syst., Special Issue on Synthesis and Verification of Hardware Design*, E80–D(10):1024–1031, October 1997.

[27] D. Stroobandt and J. Van Campenhout. Accurate interconnection length estimations for predictions early in the design cycle. *VLSI Design, Special Issue on Physical Design in Deep Submicron*, vol. 10 (no. 1): pages 1–20, 1999.

[28] D. Stroobandt, H. Van Marck, and J. Van Campenhout. An accurate interconnection length estimation for computer logic. In *Proc. 6th Great Lakes Symp. on VLSI*, pages 50–55. IEEE Computer Society Press, March 1996.

[29] D. Stroobandt, H. Van Marck, and J. Van Campenhout. Estimating logic cell to I/O pad lengths in computer systems. In T. Sasao, editor, *Proc. Workshop on Synthesis and System Integration of Mixed Technologies, SASIMI'97*, pages 192–198. S. Insatsu, Osaka, Japan, December 1997.

[30] *IEEE Trans. on VLSI Systems, Special Issue on System-Level Interconnect Prediction*, vol. 8 (no. 6), December 2000. D. Stroobandt and A.B. Kahng, guest editors.

[31] *IEEE Trans. on VLSI Systems, Special Issue on System-Level Interconnect Prediction*, vol 10 (no 2), April 2002. D. Stroobandt, guest editor.

[32] P. Verplaetse, J. Dambre, D. Stroobandt, and J. Van Campenhout. On partitioning vs. placement Rent properties. In *Intl. Workshop on System-Level Interconnect Prediction (SLIP 2001)*, pages 33–40, March 2001.

[33] VSI Alliance. http://www.vsi.org/

[34] P. Zarkesh-Ha, J. A. Davis, and J. D. Meindl. Prediction of net length distribution for global interconnects in a heterogeneous system-on-a-chip. *IEEE Trans. on VLSI Systems, Special Issue on System-Level Interconnect Prediction*, vol. 8 (no. 6): pages 649–659, 2000.