3 List of Participants

DSP Processor/Compiler Co-Design: A Quantitative Approach

Vojin Zivojnovic
RWTH-Aachen, Germany

In the talk the problem of processor/compiler co-design for digital signal processing and embedded systems is discussed. The main principle we follow is the top-down approach characterized by extensive simulation and quantitative performance evaluation. Although well established in the design of state-of-the-art general purpose processors and compilers this approach is only partly followed by producers of signal and embedded processors. As a consequence, the matching between the processor and compiler is low. The focus of the talk was on three main components of our exploration environment – benchmarking methodology (DSPstone), fast processor simulation (SuperSim), and machine description languages (LISA).

Of three description domains, i.e. behavioral, structural and physical, the structural domain aspects are discussed in the presentation. The architecture concepts at cospecification, system architecture design, HW/SW design, and system qualification phases of codesign process are described by adopting three views at each phase: design, validation, and estimation/analysis. The concepts are concretized by a set of examples taken from the approaches used at VTT Electronics in several research projects. The conclusion is that the structure of the system is presented implicitly or explicitly in all design phases as e.g. models of computation and timing during specification, system architecture templates during system design and target architecture models/implemented architectures during implementation design and system qualification.

State of the Practice of Hardware Software Co-Design at Philips

Kees A. Vissers
Philips Research Laboratories, Eindhoven, The Netherlands

Within Philips there are several divisions, some are geared towards professional equipment like Medical Systems, Industrial Automation, Philips Media and some are geared towards consumer equipment like Semiconductors and Sound and Vision. For embedded systems it is important to make a distinction between these, since the consumer application are high volume and the cost of the final product is important. There are several application domains: control processing, audio and telecom processing and graphics and video processing, each with their own characteristics and methods of design. The challenge in embedded systems is to combine some of these domains.
In system design, models of computation are used that have to be mapped on hardware and software, where there also is a model of the architecture. The results are ICs, programs and a programming environment. The problems to be solved are to design a processor, combined with a compiler, operating system and Application Programmers Interfaces (APIs), dedicated coprocessors with a programming environment, and the application programs. The total process involves specification, design, verification and validation, integration, test and debug, and production. The goal is to get a short time to market and flexible products with predictable design time, development cost and production cost. The current practice is illustrated for an audio application, a telecom application and several video applications.

being able to efficiently solve them. Here, we introduce an ILP model which models resource constraints by a 'flow of a resource' through a graph. Contrary to existing models, the complexity of the ILP model is independent on the latency (deadline) bound, however the complexity is $O(IVI^2)$ binary decision variables where V denotes the set of tasks that has to be scheduled. We analyze the polyhedral structure of the 'flow model' and give procedures to optimize it by eliminating redundant variables and constraints. For task-level problems, the model outperforms existing ILP-formulations.
Finally, we explain the possibility also to model the traveling salesman problem as a scheduling problem on a task graph without any data dependencies in the flow model.

Hardware/Software Partitioning

Don Thomas
ECE Dept., Carnegie Mellon University, Pittsburgh, U.S.A.

A computer-aided design system for partitioning behavioral descriptions among hardware and software components is described. Transformations that raise or lower the system concurrency are defined. These allow us to use what was described as one program and transform it into several concurrently active (and synchronized) processes. Hardware and software size and performance estimates are obtained through behavioral synthesis (SAW) and software compilation. The system then considers how the operations in each of the processes should be implemented: in hardware or software. An example showing a range of 12× in performance as functionality is moved from software to hardware was discussed. More information may be obtained from thomas@ece.cmu.edu.

Aspects of System Architecture in Hardware/Software Codesign

Kari Tiensyrjae
VTT Electronics, Finland

Embedded systems, e.g. in mobile communication, are implemented as heterogeneous systems consisting of interacting parts: real-time interfaces, reactive hard real-time control, DSP flow control, DSP signal processing, and analog and RF processing. Architectures get complex due to 50 % annual growth of silicon capacity enabling implementation of more sophisticated products on a single silicon chip.

with the actual values of the metrics in that point.

Ideas about an Integrated Design Process for Embedded Real-Time Systems

Y. Tanurhan

Within the design of an integrated general tool-environment for real time system development, the first goal is to simplify the handling of requirements. We use hyper techniques to capture the requirements and to map their development to the most suitable tool set.
The system design methodology is based on the functional decomposition of the system. The usage of OO-Methods, the abstraction, information hiding and inheritance allows the user to define and refine the functional blocks with basic blocks towards the transformation function for each block. The next step in the design is to model the execution sequence. In the last step of modeling we use channel refinement and attributes to model signals and their carrier protocols plus latency and service times. For the execution of the transformation we use todays commercial tools, the Statemate, MatrixX and SBS/Workbench. The interfaces to specific HW/SW prototypes environments have also been taken into account via code generation for specified targets in C or VHDL. The approach has been verified by several industrial projects. In the next phase we are going to improve the language called ObjectBlocks which is a synthesis of different specification methods like OMT, Statecharts and Functional Modeling. The early results of this integrated specification language flew into CDIF and VHDL-A standardization.

A Flow-Based Approach to Modeling and Solving Resource-Constrained Scheduling Problems

Juergen Teich
Institute TIK, ETH Zurich, Switzerland

ILP (integer linear program)-based approaches for solving computationally hard problems have become popular throughout the last years. Interestingly, the largest instances of the famous traveling salesman problem are solved using ILP techniques.
Here, we consider ILP models for resource-constrained scheduling problems. Unfortunately, formulating an ILP-model for these problems gives no guarantee for

Semi-Automatic Design Space Exploration for an Embedded Superscalar MIPS Core

Paul Stravers
Philips Research

A decisive stage in the design flow of an embedded system is the assignment of proper values to the design parameters. As an example, consider the design of an embedded super scalar MIPS microcontroller that recently has been done at Philips. First, we write an executable model of the core in ANSI-C. This model contains 62 parameters, including parameters to specify the number of instructions that can be issued per cycle, parameters to fix the depth of the pipeline, set the latency of the functional units, define the number of read and write ports on the register file, fix the size, associativity and replacement policy of the caches, and so on.

We also compile a benchmark set of representative MIPS applications for which we want to find an optimal assignment of the 62 design parameters. We can run the model on the benchmark and obtain a file with metrics that tell us how well the MIPS core performs. Examples of such metrics include the CPI (cycles per instruction), cache miss rates, bus contention and arbitration conflicts, queue overflows, hardware utilization figures, stalls due to data dependencies, etc.

We then use a tool developed by Peter Bingley and Wim van der Linden that helps us explore the huge design space. It runs on top of the Nelsis database to deep track of simulation data and its dependencies. The exploration starts with the selection of a reasonable subset (up to 8 at a time) from the 62 design parameters. For each parameter in the subset we specify the range over which we would like to explore the MIPS model. The amount of potential design is huge: 8 parameters that each can take 10 different values span a sub-space of 100 million design. Since a single run of the benchmark on our MIPS model consumes around 40 minutes, there is no way to try them all.

Therefore the exploration tool runs a regression algorithm to figure out a set of 80 points in the design space from which the remaining 100 million points could be constructed as accurately as possible. These 80 points are composed into a batch and executed on the MIPS model. Parallel execution on a pool of workstations speed up the batch execution.

When the batch finishes, we select a metric, or a function of several metrics that we are interested in. The exploration tool then constructs a mathematical model (e.g. piece-wise linear) of the design space. Finally, the tool offers several ways to visualize and browse the space. If a certain point in the design space appears to be interesting, we can rerun the MIPS model for that particular point, and 40 minutes later we know whether the estimations of the exploration tool agree

Software Modeling and Synthesis through a Virtual Instruction Set

Donatella Sciuto
Dip. Elettronico e Informazione, Politecnico di Milano, Italy

An important issue towards a fully automated system design flow is represented by the software development process. The basic requirements are: to use accurate timing characterization during all stages of the design, to compare alternative architectures and reliable software generation techniques, to ensure the correct functionality of the implementation, to avoid as much as possible the direct user intervention at this level.
The presentation aims at describing a new methodology to address the needs of concurrently implementing the software components of a control-dominated hardware/software system, possibly under real time constraints. An intermediate language (Virtual Instruction Set) for the software implementation has been presented. Such a model is useful for analysis, synthesis and simulation purpose. A static scheduling algorithm defined through the same language allows an efficient sequential execution of the software tasks while satisfying the hard real time deadline constraints imposed by the application. Retargeting onto different assembly languages can be performed through the information contained in technology files describing the translation rules from the Virtual Instructions Set to the assembly language.

Embedded Control using Reprogrammable Hardware

Alexander Sedlmeier
Siemens AG Corporate R & D, Munich

In this presentation some rapid prototyping environments are shown used at Siemens Corporate Research. The presented environments use reprogrammable hardware as basic blocks, are modular and flexible in programming.
The advantages and disadvantages of the presented rapid prototyping environments are worked out and shown. A prospective to future trends concludes the presentation.

and classify these nets into such ones to be implemented by HW, such ones to be implemented by parallel SW and a special class to be implemented by closed-loop controllers.
Here we concentrate on parallel SW. This part is further partitioned/classified into static process and dynamic ones. The latter one has to be supported by a RTOS. For this purpose we developed DReaMS. DReaMS is intended to support the entire cycle from design to the implemented system in a smooth manner. The target system is network of Embedded Control Computers (ECC's). DReaMS is scalable arbitrarily, ranging from 1 processor – single-thread / single-task / single-user to multiprocessor – multi-thread / multi-task / multi-user, where the higher levels are handled by the Peace-OS, originating from GMD-FIRST.
The scaling process is carried out dynamically. The starting point is a minimal kernel, called Zero-DReaMS. Only on demand, additional services are loaded, where we make the distinction between service access and access control. In the static case, additional services can be configured due to static analysis which also is part of DReaMS. By dynamically loading additional services on demand, DReaMS has the tendency to become a complex system. To avoid this, degradation mechanisms have been included into DReaMS as well. Services which are no longer needed and especially redundant sophisticated schedules will be deactivated and after a certain time even removed from the local (soft) cache. DReaMS includes techniques to invoke additional processors so it is self-booting starting from one single host. It even supports porting to various processor types during runtime. The communication system within DReaMS shows an excellent performance.

High-Level Modelling for Embedded Systems

James A. Rowson
Alta Group of Cadence, Sunnyvale, CA

Two modelling paradigms are need for modelling at a high-level independent of Hardware and Software: performance analysis and data flow.
Performance analysis is best for control and communication modeling. Designers use it to enter, analyze architectures, and construct test benches (often more work than the design itself).
Data flow is best for repetitive computationally intensive systems. Entry, algorithm validation, code generation, and test benches are common tasks.
The two paradigms can be combined through co-simulation or even just through using data flow to compute parameters for a performance analysis simulation.

Designing Regular Architectures using the Alpha Language

Patrice Quinton
IRISA, France

The 'Cairn' project aims at specifying and implementing real-time systems on hardware and software components. To do so, systems are modeled as programs mixing two languages: Signal and Alpha.
Signal is a synchronous language in the data flow style, designed for the software implementation of real-time systems. It makes it possible to describe systems where flows of data are combined, taking into account their 'clock', that is to say, the instants of time when they exist. A clock calculus allow properties to be checked at compile time, and help produce an efficient program. Signal has been experimented on many applications.
Alpha, on the other hand, is a functional language based on recurrence equations. Data in Alpha are functions on polyhedron shaped domains. Transformations of Alpha program, such as change of basis and pipelining, provide a technique for refining the description of an algorithm down to an architecture which executes it.
In 'Cairn', we study how Signal and Alpha can be interfaced, in order to specify a system, to co-simulate and codesign it. A first step towards this direction has been to extend the domains of the Alpha language to intersection of polyhedra and integral lattices, in order to interface Alpha programs with the subset of Signal with periodic clocks.

Embedded Control Software Demonstrated using DReaMS, a Distributed, Scalable Real–Time Management System

F. Ramming
HNI, Paderborn University, Germany

A typical application of embedded control is in the automotive area. A modern car is a mobile parallel computer with up to 100 processors. The main problem in this context is to design distributed real-time software, supported by a distributed real-time management system.
In our group we follow an approach based on Extended Hierarchical Timed Predicate/Transition nets. Externally we support various languages, including SEA (external representation of Pr/T-nets) and a combination of Oz with Pictorial Janus. By local transformations, guided by rule-based techniques, we partition

and imprecise information. To ignore the uncertainty will lead to unnecessarily expensive and/or inadequate designs, and high risks of redesign. The presentation proposes a model for the uncertainties and a formulation of the hardware/software partitioning using that model. It will also show the context in which it is placed in the Delft co-synthesis environment.

Trends in Embedded Processor Use, with Case Studies in Consumer Multimedia Applications

Pierre Paulin
SGS-Thomson Microelectronics, France

This presentation addresses trends in embedded processors, with particular focus on demanding applications in:

- wireless communication,
- MPEG 1/2, Videophone,
- 3 D media processors and
- video games.

. We show that application-specific instruction-set processors (ASIPs) are used in the majority of these applications. The main reasons are

- low power and/or cost (GSM)
- performance, parallelism (3 D processors)
- increased flexibility with respect to HW solutions (Videophone, MPEG).

We also show the increasing effort for the embedded SW for the overall design process. In a survey of 25 design teams in a leading telecom system house, it is shown that over 60% of design effect is for embedded SW development. The large majority of the code is written in assembler (90% for DSP, $>$ 75% for microcontrollers). This is due to inefficiencies in C compilers for 8/16/24 bit DSPs and MCUs. These 'low-end' processors account for 95% of total volume. In conclusion, two important R&D directions emerge:

1. High-performance compilers ($<$ 15% overhead) for DSP/MCU.
2. Tools to explore ASIP architecture tradeoffs.

Interface synthesis: The interfaces between modules (components) play a key role in determining the efficiency of a complete system. When moving a bottleneck of a computation from software to hardware (codesign), it is vital that the increased performance of the hardware is not lost in communicating data between the hardware and software. This points to the importance of good models for describing and synthesizing interfaces.
Interface verification: A frequent source of design problems is inconsistent interpretation of an interface, for example use of a different protocol for sending and receiving data. It is, therefore, important to document interfaces carefully. This opens the possibility of formally checking the consistency of interfaces. Although such a verification is only a partial check, it could eliminate a source of subtle design errors.

A Formal Modeling Style for HW/SW Codesign

Giovanni De Micheli
Stanford University, CA, U.S.A

I present a modeling style and synthesis technique for system-level specification that are better described as a set of concurrent descriptions, their synchronization and constraints. This style is applicable to mixed hardware/software systems and in particular to the design of a real-time system that dynamically schedules and synchronizes different modules. Synthesis is divided in two phases. In the first phase, the original specification is translated into an algebraic system, for which complex control-flow constraints and quantifiers of the design are introduced. Next we translate the algebraic formulation into a finite state representation for each concurrent part of the design. Finally, we extract a run-time scheduler, i.e. a controller of the hardware/software system.

Coping with Uncertainties in Embedded System Design

Ralph H.Y.M. Otten
Delft University of Technology, The Netherlands

This presentation touches upon two aspects of embedded system design. One is the fact that some functions will be realized in hardware, while others will be implemented in software. The second aspect is that, since embedded systems will be quite complex, the design trajectory will involve several levels of synthesis. This implies that in the early stages the decisions have to be based on incomplete

prototype or an emulator. It is no longer sufficient to simply provide the ability to probe signals at the gate level. On the other hand, since the specification style of hardware comes close to software, it is desirable to debug hardware in a similar manner as software.
This talk describes an approach to do a synthesis for debuggability of the hardware specification which allows for controlling and observing a running circuit from the source code level. Additionally, first concepts to combine this hardware debugging system with software debugging in order to get an integrated HW/SW-debugging, are discussed.

Sorting out Models of Computation for the Design of Embedded Systems

Edward A. Lee
U.C. Berkeley, CA, USA

I present a preliminary analytical approach, called the tagged signal model, for understanding analyzing and comparing models of computation. Issues treated by the formalism include concurrency, communication, and time. Signals are defined as sets of events, where an event is a tag and a value. The set of tags may be partially ordered or totally ordered; in the latter case, the system is said to be 'timed'. If the set of signals is S, then a 'process' is defined to be a subset of S^{n} for some integer N. A member of this subset is an N-tuple of signals that are said to 'satisfy' the process. Processes compose by intersection. A partition of S^{n} divides the signals into inputs and outputs. A process is said to be functional (or determinate) w.r.t. a partition if it is a partial function mapping of inputs to outputs. I then discuss the question of whether a composition of determinate processes is determinate and connect to known results for timed [Yates] and untimed [Kahn] systems.

Interface Modeling

Jan Madsen and Jorgen Staunstrup
Computer Systems Section, Department of Information Technology, Technical University of Lyngby, Denmark

The design of non-trivial computing systems requires a module concept to structure the design into manageable parts. Modules can be newly created for a specific purpose, but often they are general purpose off-the-shelf components. In both cases it is important that the interface to a module is well defined and fully specified. Our work has stressed two aspects of interfaces.

The Madrid Approach to Codesign with LOTOS

Carlos Delgado Kloos
Dep.Ing. Sistemas Telematico, Univ. Politecnica de Madrid, Spain

We present an approach to co-design based on methodology and a number of tools developed around the formal language LOTOS. LOTOS is a formal description technique based on a process algebraic approach (like CCS and CSP) and on the algebraic ADT language ACT-ONE.
The specification of a system should be initially as abstract as possible, should not imply forced design decisions and should leave enough freedom for algorithmic alternatives. Apart from the functional specification it is also important to capture non functional requirements and timing information. To this end, we have extended LOTOS with the ability to describe non-functional requirements in the same document. LOTOS has also been extended with timing annotations.
Our methodology starts with a specification in LOTOS, where the process part is very simple and all the information is put into the algebraic data type part. From this, several algorithmic versions can be obtained, that refine the original specification improving its efficiency.
Once an algorithmically satisfactory version has been obtained, it has to be partitioned into software and hardware components. This partitioning is made at the level of granularity at the processes. It consists on a first clustering step, which is performed taking into account estimation data and the communication among processes. A group migration between software and hardware components tries to fulfill the required non-functional requirements. For the hardware and software estimations, LOTOS is translated to VHDL and C, using the tools HARPO and TOPO, developed at our department. Estimation at the system (LOTOS) level can be performed using TOPOSIM, which allows to estimate process executions and communications using a statistical input model.
The methodology is illustrated with a couple of case studies, one of which – the Ethernet Bridge – has been prototyped by FZI Karlsruhe using an FPGA board and a Hyperstone processor to run the software. This work was done in the context of the Esprit project COBRA.

Source Level Emulation

Gernot Koch
FZI, Karlsruhe

The raise of the abstraction level of hardware specifications to the algorithmic level has a major impact on the debuggability of a whole system running on a

TIMA/IMAG, Grenoble

Multiprocessor architectures are needed for the design of complex systems made of several hardware and software processors interacting through a communication network. This talk details COSMOS, a codesign environment for multi-processor systems. The design starts with a multithread specification given in SDL. It uses an intermediate form called SOLAR to refine the initial specification into a set of interconnected processors that corresponds to the component of the final architecture. The output of the system is a distributed C, VHDL model where hardware processes are coded in VHDL and software processes are coded in C. The main refinement steps performed by COSMOS are partitioning and communication synthesis. The system was used successfully for the design of a multi-motor robot arm controller. The use of COSMOS allowed to explore several architecture alternatives.

Hardware/Software Partitioning

Asawaree Kalavade
Bell Labs., Holmdel, NJ, U.S.A.

In system-level design, applications are typically represented as task graphs, where tasks have moderate to large granularity and each task has several implementation options differing in area and execution time. The extended partitioning problem involves the joint determination of the mapping (hardware or software), schedule, as well as the implementation option for each task, so that the overall area allocated to nodes in hardware is minimum. This problem is considerably harder than traditional binary partitioning that determines just the best mapping and schedule.

Both extended and binary partitioning are constrained optimization problems and are known to be NP hard. We first present an efficient heuristic called GCLP, to solve the binary partitioning problem. The heuristic reduces the greediness associated with serial-traversal based algorithms by formulating a global criticality (GC) measure. The GC measure also permits an adaptive selection of the objective (optimizing either area or time). The algorithm also takes into account the local characterization of tasks by formulation repeller and attractor measures.

We then present an efficient heuristic for extended partitioning called MIBS, that alternately uses GCLP and an implementation selection procedure. The implementation selection procedure chooses the bin that maximizes the area-reduction gradient. Solutions generated by both heuristics are shown to be reasonably close to the optimal solutions. Extended partitioning generates considerably smaller overall hardware area as compared to binary partitioning.

are taken into account because this overhead may not be neglected. Based on this cost functions HW/SW-partitioning can be performed. The impact of the partitioning on the entire system is estimated during an additional phase before HW–synthesis. This automated HW/SW partitioning is possible for performance driven designs. All tasks are implemented in one codesign tool COD which is part of our design environment PMOSS (Paderborn Modular System for Synthesis and HW/SW-Codesign).

Aspects of Partitioning and Estimation in COSYMA

Dirk Hermann
Technische Universitaet Braunschweig, Germany

The COSYMA system (COSYnthesis of Embedded Architectures) is aimed at the application domain of small embedded systems which are primarily data dominated. As it was presented up to now, COSYMA allows to automatically partition a given C-process onto a target architecture consisting of a single μP-core and an application specific co-processor communicating via a shared memory. Currently COSYMA is being extended: The input description may consist of several C-processes which interact via explicit communication across user defined channels, the target architecture is a user defined set of processors, coprocessors and communication media, and estimation techniques are moved from profile-based to trace-based estimation.
The finest granularity considered in partitioning is basic block level. This causes additional problems like: larger solution space, implicit communication handling, data partitioning and synergetic effects between basic blocks. Nevertheless, a case study performed at the institute of computer engineering showed, that it is necessary to cross process and function boundaries in the partitioning process to achieve good solutions.
Estimation in COSYMA is currently moved towards trace-based estimation techniques, since, especially in data dominated systems, many parts of a code have a single feasible execution path, or at least some paths show to appear more often than others. Since trace data give a quite more detailed picture on how a certain program executes than profile data, runtime estimation, partitioning and scheduling can give better results when based on traces.

Codesign for Multi-Processor Architectures

Ahmed Jerraya

modules along with ASIC hardware can be used to build highly integrated 'single-substrate' systems for dedicated applications. Due to cost and design considerations, efficient design of these systems requires optimization of EC systems to meet a given set of design constraints. An approach to EC system design builds upon compilation and synthesis techniques for software and digital hardware, respectively. In this formulation, opportunities of system optimization are present at various levels of design detail and stages of the design process. In my talk I specifically address three optimizations:

1. presynthesis optimizations on HDL inputs,
2. process-level restructuring and redesign under rate constraint analysis and
3. interface optimization used in partitioning of hardware and software portions.

Presynthesis HDL optimization takes an input behavioral HDL description and produces an (optimized) HDL output. These source-level optimizations are driven by a tabular model of functionality, called Timed Decision Tables (TDTs). I describe this model and show how optimizations related to control-flow and don't cares can be implemented as row and column operation on a pair of matrices derived from TDTs. An HDL input is compiled into a set of hierarchical process graphs. In (2), we describe how rate analysis is done on process graphs and how results of this analysis are used to restructure (and in some cases redesign) processes. Finally in (3) I describe a particular flavor of this high-level partitioning algorithm that is driven by a def-use reaching analysis. We know that this algorithm is a generalization of the well-known KL netlist partitioning algorithm. Results from this partitioning algorithm show a reduction of up to 65% in interface costs over previous approaches.

Automatic HW/SW–Partitioning under Performance Aspects

W. Hardt
University of Paderborn, Germany

Embedded system design faces the HW/SW–partitioning problem. This problem is often solved by explicit designer decisions. We describe a partitioning method based on specification analysis. Static and dynamic aspects influence the overall time behavior and are examined during two analysis phases. Each phase evaluates a cost function and classifies the suitability on the regarded module for HW–implementation. Furthermore costs of the HW/SW–interface and memory access

This talk describes a data-flow analysis of array data structures for data-parallel and/or task-parallel imperative languages. This analysis departs from previous work because it

1. handles simultaneously both parallel programming paradigms, and
2. does not rely on the usual iterative solving process of a set of data flow equations but extends array data flow analyses based on integer linear programming, thus improving the precision of results.

An Object Oriented Petri Net Approach to Co-Design

Robert Esser
Computer Engineering and Network Laboratory, ETH Zurich, Switzerland

CodeSign is a hardware / software co-design environment for real time systems developed at the Institute of Computer Engineering and Communication Networks at the Swiss Federal Institute of Technology (ETH) in Zurich. With CodeSign complex real time systems can be easily modeled. The executable semantics and mathematical analysis of CodeSign models enable many errors to be detected very early in the design process - where they are easily corrected. CodeSign uses Petri nets to describe dynamic behavior and object oriented techniques to not only describe relationships between abstract data types but also relationships between Petri net components thus providing a well defined way of refining models towards implementation. CodeSign's Petri net basis allows all objects to have multiple threads of control enabling complex concurrent systems to be modeled in a formal and well understood way. CodeSign supports multiple formalisms whose semantics including interaction with each other are described in the object oriented Petri net kernel language. Formalisms so described are also object oriented, thus facilitating reuse.

Optimization in Embedded Computing Systems

Rajesh Gupta
University of Illinois at Urbana, U.S.A.

Embedded Computing (EC) refers to use of computing elements in dedicated non-computing applications. EC is driven by recent availability of high-performance inexpensive computer engines such as microprocessors and DSP cores. These

control structures could maximize throughput. The adopted design flow included CAD un-supported phases, like hw/sw trade-off analysis and partitioning. This example shows the use of tool-sets to support users when designing mixed hw/sw systems.

Hierarchical Finite State Machine Models for Modeling and Implementation

Joseph T. Buck
Synopsys, Mountain View, CA, U.S.A.

This talk reviews some of the issues surrounding the semantics of Statecharts specifications, reviewing ground covered by [van der Beeck] but reaching different conclusions. We require a high degree of modularity and the ability to formally verify our designs. Of those Statecharts variants described in the literature, our model most closely resembles Argos and could be considered an extension to it.

HW/SW Codesign of Embedded Systems: The Complete Picture

Raul Composano
Synopsys Inc., Mountain View, CA

Embedded systems display unique characteristics: Real time is usually crucial, correctness is much more important than in general purpose software. They are difficult to maintain/update in the field, and they are increasingly being implemented as mixed HW/SW systems on a chip. This paper reviews the main trends in HW/SW co-simulation and co-synthesis from an industrial ('bottom up') perspective. While co-simulation is being used extensively in practice, co-synthesis is still in an embryonic stage. The presentation ends showing some statistics and trends and speculating about a possible future design paradigm for this kind of systems.

Array Dataflow Analysis for Explicitly Parallel Programs

Jean-Francois Collard and Martin Griebl
U-Versailles, France and U-Passau, Germany

From Bit-Level to Word-Level Decision Diagrams – New Data Structures for Verification and Synthesis

Bernd Becker
Institute of Computer Science, University of Freiburg, Germany

Several types of Decision Diagrams (DDs) have been proposed in the area of Computer Aided Design (CAD), among them being bit-level DDs like OBDDs, OFDDs and OKFDDs. While these types of DDs are often suitable for representing Boolean functions on the bit-level and have proved useful for a lot of applications in CAD, there exists also important functions (e.g. multipliers), that can not be represented efficiently by those DD-types.
Recently, DDs to represent integer-valued functions are ADDs, MTBDDs, EVBDDs, FEVBDDs, (x)BMDs, HDDs (=KBMDs), and K(x)BMDs, attend more and more interest: Using (x)BMDs it was possible to verify multipliers of bit length up to $n = 256$. K(x)BMDs, a generalisation of OKFDDs to the ward-level, can be shown to also 'include' OBDDs, MTBDDs and (x)BMDs. Thus it is possible to represent functions efficiently that have a good bit level or word level structure in one and the same data structure.
We further clarify the computational power of these DD-classes by giving the theoretical background and practical results as well.

Industrial Requirements for Co-Design: A Study in Telecom Domain

Massimo Bombana
Italtel, Italy

In manufacturing sites the design task is approached applying a design methodology supported by integrated and cooperating tool sets. While design flows for hardware development are consolidated, the extension to cope with mixed hardware/software systems still need to be properly defined. In most cases the software development and the hardware development follow different paths, so making the tasks of final integration and simulation very complex. Other requirements come from the telecom market sector. For such applications re-use is a key factor to shorten design time, to cope with complex and changing specifications, and to provide a better time to market.
A case of re-design has been presented: the demodulator/equalizer of a GSM base station has been re-designed subject to stronger timing and performance constraints. The previous solution, based on the use of DSP cores had to be abandoned in favor of ASIP design, where parallel data paths and dedicated

2 Abstracts

HW/SW Partitioning in the Context of the PISH Project

Edna Barros
Dept. Informatics, Recife, Brazil

We have presented a hw/sw codesign approach which is being developed in the context of the PISH Project. This project is ongoing in Brazil and consist of six main activities: partitioning, verification, hw synthesis, sw compilation, prototyping and implementation. This abstract concerns with the partitioning and verification tasks.
The HW/SW partitioning problem can be seen as a mapping of some description (in our case an OCCAM description) into some target architecture. We are considering a pre-defined one composed of one sw component (microprocessor) and some hw components (application specific components). The memory is distributed and communication is done by message-passing through channels.
When partitioning is done, communication must be introduces. An important issue here is to guarantee that the semantics is preserved. The partitioning should also explore the design space by considering hw/sw trade-offs as well as distinct hw implementations.
In order to cope with these problems, we have defined a partitioning based on program transformations: A set of algebraic rules are applied on the original description in order to generate a final OCCAM program which reflects the target architecture.
The set of rules should be finite and for this the partitioning is divided into four main steps:

- splitting: the initial description is split into a set of concurrent processes
- classification: a set of implementation alternatives is associated with each process
- clustering: the processes are grouped in hw and sw clusters
- joining: the processes are grouped by program transformations.

It is important to notice here the separation of concerns: correctness and efficiency in order to make verification feasible. This approach is being implemented and we are starting the analysis of some more complex case studies.

dynamic properties of embedded systems, i.e. communication traffic and execution profiles.

- *Verification, validation and emulation:* Target architecture, functionality, timing, prototyping for hardware and software including FPGA-emulators, co-simulation and co-verification of hardware and software.
- *Synthesis and partitioning:* Regularity, data dependency and coprocessor extraction, knowledge based partitioning, cost functions for hardware, software and communication.
- *Applications:* Software acceleration, coprocessors, instruction set optimization and power consumption.

The organizers would like to thank everyone who has helped to make the seminar a great success.

Edward Lee Giovanni De Micheli Wolfgang Rosenstiel Lothar Thiele

1 Preface

Embedded systems are dedicated computing and control systems embedded into a technical environment. They execute functions in response to specific input stimuli usually within predefined time windows.
A main characteristic of embedded systems is their heterogeneous composition. Embedded systems may be composed of dedicated or weakly programmable hardware components, predesigned programmable components (software), communication networks and analogue interfaces. An embedded system is often tightly integrated with sensors and actuators communicating with the environment.
Currently, most embedded systems are designed manually. However, the growing complexity of these systems makes CAD support for the development process, life-cycle and maintenance indispensable. CAD can potentially reduce the design intervals provided proper methodology is used to enable fast prototyping and to design components in parallel. Finally, in many safety critical applications, the correctness of the realization with respect to specified real-time properties must be proved, for example using verification methods.
The success of computer aided design methods for digital integrated circuits is partly founded on the development of a design methodology based upon abstraction. Here methodologies linking different levels of abstraction have been developed for which synthesis and optimization algorithms provide a well tested path to implementation. Due to their complex heterogenous nature, pragmatic methodologies for embedded systems have not yet been developed. This heterogeneity is reflected in the huge amount of conceptually different models and methods. Despite of the high expectations, many fundamental problems are not solved yet.
This seminar was devoted to models and methods for the design of embedded systems. In particular, the presentations and discussions covered the following topics:

- *Models:* The relationships between the different models of computation, the hardware software dependencies in these models. The search for a sound common model for which useful design methods can be applied.

- *Optimization methods:* Methods for updating an implementation as a result of a change in the specification (redesign) and for hardware-software-codesign (partitioning, optimization, real-time scheduling).

- *Interfaces and communication:* Efficient concepts for linking hardware and software subsystems, formal specification and verification of interfaces between tightly coupled components and the communication between distributed subsystems.

- *Performance simulation and estimation:* Models and methods to estimate

Coping with Uncertainties in Embedded System Design
Ralph H.Y.M. Otten 15
Trends in Embedded Processor Use, with Case Studies in Consumer Multimedia Applications
Pierre Paulin 16
Designing Regular Architectures using the Alpha Language
Patrice Quinton 17
Embedded Control Software Demonstrated using DReaMS, a Distributed, Scalable Real–Time Management System
F. Ramming 17
High-Level Modelling for Embedded Systems
James A. Rowson 18
Software Modeling and Synthesis through a Virtual Instruction Set
Donatella Sciuto 18
Embedded Control using Reprogrammable Hardware
Alexander Sedlmeier 19
Semi-Automatic Design Space Exploration for an Embedded Superscalar MIPS Core
Paul Stravers 19
Ideas about an Integrated Design Process for Embedded Real-Time Systems
Y. Tanurhan 21
A Flow-Based Approach to Modeling and Solving Resource-Constrained Scheduling Problems
Juergen Teich 21
Hardware/Software Partitioning
Don Thomas 22
Aspects of System Architecture in Hardware/Software Codesign
Kari Tiensyrjae 22
State of the Practice of Hardware Software Co-Design at Philips
Kees A. Vissers 23
DSP Processor/Compiler Co-Design: A Quantitative Approach
Vojin Zivojnovic 23

3 List of Participants **25**

Contents

1 Preface **4**

2 Abstracts **6**

HW/SW Partitioning in the Context of the PISH Project
Edna Barros 6

From Bit-Level to Word-Level Decision Diagrams – New Data Structures for Verification and Synthesis
Bernd Becker 6

Industrial Requirements for Co-Design: A Study in Telecom Domain
Massimo Bombana 7

Hierarchical Finite State Machine Models for Modeling and Implementation
Joseph T. Buck 8

HW/SW Codesign of Embedded Systems: The Complete Picture
Raul Composano 8

Array Dataflow Analysis for Explicitly Parallel Programs
Jean-Francois Collard and Martin Griebl 8

An Object Oriented Petri Net Approach to Co-Design
Robert Esser 9

Optimization in Embedded Computing Systems
Rajesh Gupta 9

Automatic HW/SW–Partitioning under Performance Aspects
W. Hardt 10

Aspects of Partitioning and Estimation in COSYMA
Dirk Hermann 11

Codesign for Multi-Processor Architectures
Ahmed Jerraya 11

Hardware/Software Partitioning
Asawaree Kalavade 12

The Madrid Approach to Codesign with LOTOS
Carlos Delgado Kloos 13

Source Level Emulation
Gernot Koch 13

Sorting out Models of Computation for the Design of Embedded Systems
Edward A. Lee 14

Interface Modeling
Jan Madsen and Jorgen Staunstrup 14

A Formal Modeling Style for HW/SW Codesign
Giovanni De Micheli 15

Dagstuhl Workshop
on
Design Automation for Embedded Systems

Organized by

Edward Lee (UC Berkeley)
Giovanni De Micheli (Stanford University),
Wolfgang Rosenstiel (Universität Tübingen),
Lothar Thiele (ETH Zürich)

Schloß Dagstuhl 22.4. – 26.4.1996