Franz J. Rammig, Jorgen Staunstrup,
Gerhard Zimmermann (editors):

Self-Timed Design

Dagstuhl-Seminar-Report; 52
30.11.-04.12.92 (9249)

ISSN 0940-1121

Tel.: +49-6871 - 2458
Fax: +49-6871 - 5942

Das Internationale Begegnungs- und Forschungszentrum für Informatik (IBFI) ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Bezugsadresse: Geschäftsstelle Schloß Dagstuhl
Informatik, Bau 36
Universität des Saarlandes
Postfach 1150
66041 Saarbrücken
Germany
Tel.: +49 -681 - 302 - 4396
Fax: +49 -681 - 302 - 4397

Self-Timed Design

Organizers: F. Rammig, J. Staunstrup, G. Zimmermann

Self-timed circuits have potential performance benefits compared to synchronous circuits. Because of their delay insensitivity and the absence of a clock every part of a circuit can work as fast as possible without violating constraints based on worst case assumptions. Self-timed circuits also have the benefit of potentially very low power consumption and a very wide range of operating temperatures and supply voltages. The disadvantage is the additional chip area necessary and the difficulty of the design. Very little practical experience exists to support the made claims on benefits or disadvantages and synthesis techniques are just evolving.

Self-timed circuits require design methods and CAD tools which are different from the ones used for synchronous circuits. Furthermore, a lot of the accumulated experience with designing synchronous circuits is not appropriate, for example approaches based on a finite state machine controlling the computation. To take advantage of the potential benefits, it is important to find design techniques which make it feasible to design reliable and efficient circuits with reasonable effort.

This seminar therefore brought together participants with an active interest and good background in the field of self-timed circuits with participants from related fields of design. The participants had been asked ahead of time to provide the organizers with a list of up to five topics that they would like to see discussed at the seminar. Participants had been discouraged to present known work or give "sales talks" advocating ones own results.

The answers could be combined to three major topics of general interest and the following descriptions of the topics were sent to the participants together with five examples provided by Jo C. Ebergen.

Synthesis

Synthesis of self-timed circuits from high-level descriptions is an important topic. To stimulate discussion and evaluation of the synthesis techniques a small set of design problems are given to all participants. The organizers want to encourage anybody wishing to give a presentation on synthesis to use one or more of these examples to illustrate their technique. The problems are provided by Jo Ebergen.

Cost/performance

The literature is full of claims about properties, advantages and disadvantages of self-timing. It would be nice to collect documentation for these claims. Presentations are invited from participants who can provide evidence for quantitative properties such as, speed, power consumption, area, etc.

Delay (in-) sensitivity: how much?

It appears that absolutely delay-insensitive circuits are of little or no practical value. Therefore, all published designs are based on some assumptions about delays typically in isochronic forks. This immediately raises the question of how to trade off delay assumptions against other properties. Participants are invited to present contributions that can illuminate this issue.
A detailed description of the examples can be found on page 5 of this report.

The seminar fulfilled the expectations in so far as it was dominated by discussions on the proposed topics that were triggered by a relatively small number of presentations. The exercises turned out to be well accepted and focused the discussion about design methods in an excellent way.

The seminar did not completely answer any of the questions. Surprising for most of the participants was the fact that more real examples of self-timed systems existed than previously assumed and a collection of references was started. Some discussions went back to the fundamental question of the construction of a reliable orbiter which still does not seen to be answered. In general the discussion showed that much progress towards design methods has been made recently but much more effort is still necessary. What could be a better result for research?

Why "Self-Timed" has no Future

Michael Yoeli

To stimulate a discussion on this topic, here are some apparently negative aspects:

* Industry. So far industries (e.g. INTEL, IBM, SIEMENS, ZORAN) have not accepted the DI-approach.
* Also, it seems that so far no "real-life" self-timed / DI chips have been produced.
* Although the correctness of a DI-design is independent of the particular layout as well as of the particular technology (CMOS, BICMOS, GaAs, ECL), the performance (speed, area) can be improved considerably by layout-dependant and technology-dependant design considerations.
* Self-timed circuits might have better average speed than their synchronous counterpart. However, in most applications it is worst-case speed which matters.
* Self-timed / DI circuit , have the advantage of overcoming the "clock-skew" and "glitch" problems of synchronous designs. However, in practice, synchronous circuit designs have well-established ways to handle such problems.
* CAD-Tools. More and more CAD-tools are becoming available for synchronous system design at all levels, in great contrast to the non-availability of cad-tools for asynchronous designs.

Self-Timed Systems

Charles E. Molnar

Self-timed systems are ubiquitous; the only question is how far toward the circuit level should the use of explicit signaling protocols that do not rely on clocks be carried. The reasons for interest in going to the circuit and transistor level include hope for greater performance and efficiency by the elimination of clock distribution overheats, and the ability to perform calculations at "average" rates rather than "worst case" rates. An intellectual gain would be the ability to use uniform and consistent formalisms at all levels of design.

Obstacles include the immaturity of theory and practice in comparison with those for clocked sequential circuit design. This has benefitted greatly from the close correspondence between sequential models for computation and the physical properties of clocked sequential circuits, which has offered a precise mapping between models for computation and models for mechanisms that compute.

What is badly needed in self-timed design is an equally explicit mapping from computational models to physical ones. Attempts to adopt synchronous models to this purpose have not been adequate. Dynamical systems appear to be a good representation for the physics of computing circuits, but the mapping to choose between this level and that of discrete metric-free computational models is only partially understood.

One might hope that solving this "low level" problem might benefit "higher level" design as well, by providing a firm physical basis for the structure of the "higher level" models and reducing the arbitrariness of the choices made in formulating these "higher level" models.

Once More: "Why Self - Timed?"

M. Kishinevski

A brief discussion of "hot points" of self-timed design is presented. We stress an importance of CAD tools in self-timing and give an interpretation for the theoretical results in the arbitration problem.

Design of Self - Timed Systolic Arrays

Simon Jones

Eliminating the clock distribution network is of most immediate value in large processor arrays. The bit-level systolic vector-matrix multiplier is a well established example of such structures. This representation reports on the results of a comparative design study of synchronous and asynchronous implementations of a 7 x 5 bit-level vector-matrix multiplier implemented in a 3μ CMOS gate-array. Details of speed / size and data dependant performance characterizes of both implementations are given

Asymptotic Comparison of Self - Timed and Synchronous Circuits

Mark Greenstreet

Many arguments about the relative merits of self-timed and synchronous designs are based on assumptions of scalability. Asymptotic analysis addresses there assumptions.

It has been shown [1] the self-timed pipelines under a probabilistic model can achieve 'linear speedup' (the rate of operations of a simple processor is independent of the number of processes in the pipeline). In the talk, it was shown how this result can be extended to two dimensional arrays of self-timed processors; however, no closed form solution of the asymptotic performance is currently known. Previous results show that this is not possible for the corresponding synchronous arrays[2, 3].

In addition to providing a basis for comparison, then results suggest a novel approach to distributing a clock signal is a synchronous system by using self-timed network. This is more robust than a buffer tree, and simpler than designs based on phased-locked loops.

References:

1. Greenstreet & Steiglitz: "Bubbles can make self-timed pipelines fast", Journal of VLSI and Signal Processing, Vol. 2, No. 3, 1991
2. Fisher & Kunq: "Synchronizing Large VLSI Processor Arrays", IEEE Transactions on Computers, Vol. C-34, No. 8, 1985
3. Dikaiakos & Steiglitz: Comparison of Tree and Straight-Line Clocking for Long Systolic Arrays",Journal of VLSI and Signal Processing, Vol. 3, No. 4, 1991

Some Small Exercises

Jo C. Ebergen

This note contains some small exercises in designing asynchronous circuits. All problems are stated informally and are often (intentionally so) incomplete. One part of the exercise is to come up with a formal specification in the notion of your choice. The other part of the problem is to find a asynchronous, speed-independent, or delay-insensitive implementation of your specification. If possible, explain how you found your implementation and what delay constraints need to be satisfied in the implementation. Each of the problems allows for several solutions; some solutions are trivial, some are nontrivial. (For most problems, solutions have been published in the literature. These solutions are not necessarily the most efficient ones.) Try to find an efficient solution. Explain also what kind of efficiency measure you consider.

Problem 1: An Unbounded Stack

In this exercise we consider an unbouded stack on which two operations can be performed: put and get. Design (a part of) the control part of the stack that dictates the data movements in the stack. You may choose the data movements for your implementations, but you do not have to design the data part itself.

Problem 2: A Bounded Stack

The exercise is similar to problem, but now the stack is bounded. You have to take care that no put will be performed when the stack is full, and no get will be performed when the stack is empty. You may assume that each put and get is acknowledged by a communication action indicating whether the stack is full, empty, or neither.

Problem 3: Up/down Counter

Two operations can be performed on an up/down counter: up and down. Let count represent the number of up minus the number of down operations. For an up/down counter of size N, count is always at least 0 and at most N, and initially the count is 0. After each up or down the up/down counter responds whether the counter is full (count = N), empty (count = 0), or neither. You may assume that no up operation will be attempted when the counter is full and no down operation will be attempted when the counter is empty.

(This exercise indeed has some similarity with the bounded stack. Notice, however, that in the up/down counter there are no data movements to be considered.)

Problem 4: The Committee Problem

Given are a number of committees consisting of persons. Persons may be a member of more than one committee. Persons are either busy in a committee meeting or free, and each person can be busy in at most one committee meeting at a time. A committee can meet only if all persons of that committee are free. Specify and design a committee schedular for the committees given below. The schedular receives from each person a signal indicating that he is free to meet, and it outputs signals indicating which committee can meet. After a person has signalled that he is free, he will be notified, in due time, in what committee he will meet. After the meeting is over, all committee members are free again and, after some rest, each of them will signal his availability to the schedular again, and so on. Notice that persons act independent of each other when they signal

their availability to the schedular. (Obviously, in the above 'he' may be replaced by 'she', if so desired.)

Design the schedular for the following sets of committees. Committees are denoted by capitols, persons by lower-case letters.

1. There is only one committee A given by A = {a; b ; c; d}
2. There are two committees A and B given by A = {a; b; c}, and B = {b ; e; c}.
3. There are three committees given by A = {a; b}, B = {b ; c} and C = {c; a}

Problem 5: The Transition Arbiter

A transition arbiter (of RGB arbiter) is used for realizing mutual exclusion between two processes that communicate with the arbiter using a transition signalling protocol. The transition arbiter has four inputs r0; r1; d0; and d1, and two outputs g0, and g1. The symbols r0; g0, and d0 stand for request, grant, and done for process 0. An informal specification of the RGD arbiter reads (see Sutherland's Tuning Award Lecture in Comm. ACM. Vol. 32, No 6. p.725.)'[RGD] ARBITER grants service, g0 or g1, to only one input request, r0 or r1, at a time, delaying subsequent grants until after the matching done event, d0 or d1.' Give a specification of the transition arbiter (or RGD arbiter) and an implementation.

You have several choices in implementing the transition arbiter. For example, you may implement it by a network of other primitive components, or you may implement it by means of a (CMOS) transistor network. (The latter is more challenging.)

Specification and Compilation of Interface Circuits

Peter Vanbekbergen, Bill Lin

A method to compile asynchronous circuits starting from the Signal Transition Graph (STG) formalism is mapped. First extensions of a classical STG formalism are mapped to overcome a number of problems like

- Do not care behavior.
- Using signal levels instead of signal transitions.
- Metastable behavior.
- Complex conditional behavior.

Second, a synthesis method starting from the state graph is proposed. We propose a "global assignment theory" for encoding state graph transformations. A constraint satisfaction framework is proposed that can guarantee necessary and sufficient conditions for a state graph assignment to result in transformed state graph that is free of critical races. The transformations achievable using the proposed framework correspond to very complex transformations on signal transition graphs.

Hazard Elimination in Logic From STG's

Cho Moon and R. Brayton

We discuss the problem of technology mapping logic obtained from implementing STG's into

standard gates consisting of AND and OR gates. We want to do this in such a way that the multi-level logic obtained is guaranteed free of all hazards. We assume that all gates may have unbounded delay, but all wire delays are 0. Further the environment must be well behaved in the sense that no new input comes in until the transitive fanout of the input signals changing, is guaranteed to be stable. This can always be insured by adding sufficient delay at the outputs.

The logic obtained from live STG's with the complete state assignment (CSC) property can be shown to have no critical races if the state assignment given by the set of all signals is used. Further the sum-of products (SOP) logic derived from the state graph of the STG has no static 0-hazards. The only hazards that have to be removed therefore are the static 1-hazards and the dynamic hazards. We give a method for removing all the static 0-hazards by using redundant primes but with the least number of primes possible. For dynamic hazards, it is shown that they can only occur as 0-hazards in product terms (cubes) in the SOP expression. We give a factoring method that factors out "characteristic literals" from the SOP expressions. It can be shown for each characteristic literal in a cube with a 0-hazard associated with a set of concurrently enabled signal transitions, the quotient of the SOP expression with respect to that literal is free of 0-hazards for that set of transition. However, the quotient may have other 0-hazards due to other concurrently enabled transitions. We give a method which can find a factoring which eliminates all 0-hazards if such a factoring exists. In this case, all hazards are eliminated without adding any further redundancy. For all examples done until now, this factoring exists. If no such factoring exists, we give a method which adds redundancy which we conjecture will always succeed in eliminating the remaining hazards.

Thus a hazard-free multi-level logic implementation in terms of simple gates and with few redundancies is produced.

Delay-Insensitive Multi-Ring Structures

Christian D. Nielson & Jens Sparso

Our work represents an attempt to establish a design technique which (1) ensures that a design is delay-insensitive by construction, (2) enables the designer to do a performance analysis, and (3) makes it possible to synthesize circuity automatically.

The design technique is based on a static data-flow concept, and the structure of the cirquits consist of pipelines and rings that are connected into multi-ring structures by joining and forking of signals. The designs are implemented using a small set of building blocks (latches, functional blocks, switches, joins and forks) that are realized using C-elements and signal gates.

Using this technique we have successfully designed, fabricated and tested a number or non-trivial delay insensitive VLSI circuits. At the seminar the technique was illustrated on the following problem:

Problem 182: An (un)bounded stack (also with Michael Kishinevski)

We present a solution to the stack problem based on multi-ring structures.Each stage in the stack consists of a small ring. With three (or more) latches to communicate with its closest neighbour only. The main component in each stage is a switch: In each cycle it recieves an element from both its left and right neighbors. Depending on the command,'put' or 'get', the element from either the left or the right is saved. In this way we can ensure a DI protocol between neighboring

cells.

In order to obtain constant response time, the put/get commands are distributed through a parallel Muller pipeline. This together with the elasticity of the rings ensure that the cycle time of the stack is bounded by the local stage cycle times.

We further discuss simple extensions to the design in order to include full and empty detection for the bounded stack.

The Formal Verification of a Delay Insensitive (DI) -Circuit Synthesis Algorithm

M. Yoeli, N. Shintel, H. Belhadj, G. Saucier

This paper defines the concepts of asynchronous modular network and network behavior. It introduces a precise mathematical framework to deal with the verification problem for asynchronous / DI realizations from trace structure specifications.

It then presents an algorithm for the synthesis of modular DI-networks from a restricted class of Petri-net specifications.

It is claimed that the outcome of this synthesis algorithm is "correct-by-construction". A formal proof of this claim is about to become available.

Synthesis of Asynchronous Modules

Bernd Kleinjohann

A formal model of hierarchical synthesis of complex asynchronous modules is introduced. The application of this model offers the possibility to synthesize more complex specifications by using hierarchical synthesis algorithms. A mapping of usual VHDL specifications onto the formal model is possible. The model consists of a trace structure defined by

$$TS = (L, \{\Sigma^i_1, ..., \Sigma^i_n\}, \{\Sigma^o_1, ..., \Sigma^o_n\})$$

The input alphabets Σ^i and output alphabets Σ^o allow the specification of a netlist of n modules. The trace set L specifies the behavior between the modules. For the specification of the trace set L labeled Petri-nets are introduced

A lot of restrictions, required for usual models are dropped in the introduced model. Trace sets that are not prefix closed are allowed. A finite trace specifies that the circuit may stop. Non-deterministic trace sets for the specification of design freedom are allowed as well.

For a given synthesis specification consiting of an environment (TS_{env}) and a hardware specification (TS_{spec}) a realization relation I is defined. The semantic of a synthesis process can be defined by $TS_{spec} \perp TS_{env} \supset TS_{HW} \perp TS_{env}$ where $\perp$ denotes the nearing operator. TS_{HW} denotes the trace set of the implemented hardware. The realization relation guarantees that for every behavior alternative in $TS_{spec} \perp TS_{env}$ it exists at least one alternative in TS_{HW}. Based on the labeled Petri-nets that recognize the trace sets constraints are formulated that allow a partition-

ing of a specification. Such a partitioning may be denoted by $TS_{env} \perp TS_{spec} \supset TS_{env} \perp TS_{HW1} \perp ... \perp TS_{HWn}$. The constraints are formulated in such a way that $TS_{HW1} \supset TS'_{HW1} \Rightarrow TS_{env} \perp TS_{spec} \supset TS_{env} \perp TS'_{HW1} \perp ... \perp TS'_{HWn}$ holds. The interconnections of hardware modules have to be realized by wires. The behavior of wires are specified by a special trace structure. This trace structure describes the behavior of stray and inertial delay in a speed independent manner by non-determinism.

Design of Communicating Asynchronous Circuits from a Petri - Net Specification of Interface Behavior

J. Beister, R. Wollowski

The asynchronous circuit to be designed -typically a controller for concurrent discrete processes- is embedded in a given environment, with which is required to interact in a precise manner.

In the first step of the proposed design procedure, the interaction across the interface between circuit and environment, is modeled by the Petri-net equivalent to a signal transition graph. Concurrancy is presented, and no references to internal events occur.The circuit is then thought to be decomposed into concurrently operating subcircuits such that concurrently changing output signals are assigned to separate subcircuits, and the outputs of any subcircuit may be used as inputs to the others. On the behavioral level, this step is carried out by decomposing the Petri-net into subnets with input concurrency only.

From each subnet, a primitive flow table for the corresponding subcircuit is constructed via the step (or reachability) graph. The flow table -automatically obtained in minimal equivalent primitive standard form- may be viewed as canonical initial solution for the subcircuit, leaving open all options of implementation. The primitive flow tables are then processed by the classical methods of asynchronous design. Input concurrency maps onto non-fundamental mode operation of the circuit and must be handled by appropriate measures.

Naive Design of Unbounded Stack

M. Kishinevski, C. Nielson

We present this design example as an attempt to active the result (an unbouded stack with constant responce time) in an straightforward manner, by means of stepwise transformations of the initial structural idea and initial specification of the process in a Change Diagram. The main aim is to demonstraite

* a decomposition technique based on an m event model (Change Diagram) and
* a technique to localize control structures.

We start with synchronous Master-Slave design with two phase clock, than put in global completion detectors to indicate completion of the processes in the data path. After decomposition of the specification into three parts (a phase control structure, a direction control structure and a data path cell) we synthesise the two phase local control circuity, pipelined structure for the direction

control path and the Master-Slave structure for the data path. The final solution has a constant response time, DI interface and speed-independent inner structure with local connection between neighbour cells.

Finally, we discuss possible quantitive measures to estimate the ratio of delay insensitivity.

An Up - / Down - Counter

Christian D. Nielson & Hartmud Schmeck

We have designed an up-/down- counter with the following properties:

- Counts in the range [0..N-1] for any N
- Constant response time: $\Theta \in O(1)$
- Logarithmic area $A \in O(\log N)$
- Bounded amortized power consumption or count: $\zeta \in O(1)$

The design is based on a pipelined binary encoded up-/down counter. The pipeline is extended with additional means for back propagation of full- or empty- signals from the more significant bit stages to ensure the constant response time.

With a redundant binary encoding we optain the bounded amortized power consumption for any up-/down sequens.

The Synthesis of Deterministic Delay-Insensitive Circuits from behavioral Specifications

Roger Sayle

This paper outlines a fully automatic description of specifications into delay insensitive network models of primitive modules. This work is similar to previous attempts at such an approach. C. H. Huang describes a method for implementing finite state machines; Ebergen, Udding and Josephs describe methods for the manual transformation of specifications and Brunwand, Brown and Martin describe syntax-directed translations of suitable specifications into circuits. These existing methods suffer from restricted generality of specification and for some poor efficiency.

This paper begins by outlining the three fundamental components that form basis or target for the general circuits; these are the merge (XOR gate), the muller C-element and the Keller select element. From these common components such as toggles, calls and asymetric C-elements may be constructed.

The method begins by translating the specification into a transition diagram (automaton diagram). These specifications may be Petri-nets, process algebra agents, event systems or truth tables. The use of a transition diagram permits very expressive generality. This graph is minimized and checked for monogenicity and delay-insensitive properties. If the specification is non-orbitrating, it is transformed into a "stable state graph". From this representation, a final circuit may be generated by applying a series of expert system rules. The resulting implementation is peep-hole optimized using circuit-to-circuit transformations to improve the quality of the circuit.

This method is applied to be the example of a bounded stack and is shown to give results com-

parible to those designed by hand. Finally the implementation of a up/down-counter and a one phase buffer are also presented.

Synthesis Methods in Self - Timed Design Compilation Approach

Alex Kondratyev

Two somewhat opposite approaches to a synthesis problem are considered. First is based on ensuring the CSC property and other implementation requirements by formal transformations of initial behavioral description. Another supposes a direct translation of behavioral description into a speed-independent circuit from a set of library modules. The latter is demonstrated on examples of dual-rail technique and distributer-based implementations.

It is shown that implementations obtained are of initial description that allows to estimate the quality of future solution beforehand.

On the Analysis and Optimization of Self-Timed Processor Arrays

Lothar Thiele

The talk deals with systematic methods for analyzing and designing self-timed regular arrays of processors. Methods are presented for deriving measures of efficiency and for verifying the computational behavior of a given array. It is shown that the optimization of a given self-timed processor array, with respect to its processor utilization, can be given mathematically in the form of linear programs or integer linear programming problems, whose sizes are independent of the size of the array.

Specification and Verification of Self-Timed Circuits behavior

- A Bottom - Up Approach to Description Tools -

A. Taubin

Three description levels are distinguished in VLSI design by analogy in programming: circuit code level, circuit assembler and high level language. Transition diagrams are suggested to use as a circuit code. Correctness properties are formulated in their terms for which the simple verification algorithms are known. The main shortcoming of transition diagrams is an exponential complexity of their size from the size of circuit specified. As a next step to use more compact specifications "Change Diagram" model is suggested. Speed-independence properties can also be analyzed in terms of this model and the complexity of each analysis is shown to be polynomial ($O(n^4)$) from the size of CD. To save a reliable correspondence between circuit assembler level (CD) and high level language all primitives of high level language presented are specified in CD terms also.

List of Self - Timed Chips (Designs):

1.) Identification
2.) Short description
3.) References
4.) Person to be contacted

1.) RISC-Microprocessor
2.) Self-timed, double-rail
more advanced architecture than Martin's
4.) M. Yoeli

1.) CORDIC Processor
2.) Self-timed using DCVSL and and controolers (STG), Calculates angle and length of a vector
4.) P. Vanbekbergen

1.) Multiplier, Queues, etc.
3.) Chips designed by Sutherland and Sproull through Austex corporation
4.) Ivan Sutherland, SUN, Mountain View, CA, C. E. Molnar

1.) Macromodules-CMultic-chip (1965-1975)
2.) 15 types -control and data functions
3.) Clark and Molnar, "Macromodules", Computers in biomedical research, Vo. IV, Stacyt Wax Man, ed. Academic Proc. 1974.
Also - Final Report on Macromodules, Project, SD-302, 14 Volumes, Through ASTIA-microfilm- (Washington University project)
4.) C. E. Molnar

1. + 2.) Several chips designed by Van Berkel's group at Philips
3.) Van Berkel's Ph.D. Thesis
4. Kees van Berkel at Philips

1.+2.) Microprocessor by Martin et al.
3.) Caltech VLSI Conference 1989
4.) Alain Martin

1.+2.) Vector multiplier
3.) Paper in EURODAC proceedings
4.) Jens. Sparso, Techn. University of Denmark, Lyngby

1.+2.) Neuron Proc. for NN engine
3.) Conference on NN and AI, 1990
4.) Christian Nielson, Techn. University of Denmark. Lyngby
1.+2.) Vector-Matrix Multiplier using Caltech approach
3.) HICSS 1993
4.) Christian Nielson, Techn. University of Denmark. Lyngby

1.+2.) Chaos Router
3.) SPPA 1989
4.) Hary Snyter, University of Washington

1.) Delay-insensitive vector multiplier.
2.) The design implements an iterative serial-parallel vectormultiplication algorithm with carry-save representation of the temporary result. In addition to the core multiply-accumulate circuitry the design includes a control unit and a couple of shift registers. The design is based on a static data flow concept, and the structure of the circuit consists of pipelines and rings that are connected into multi-ring structures by forking and joining of signals. A test chip has been fabricated on EUROCHIP's October 1991 run in a 1.5 micron CMOS technology. This chip multiplies vectors with 4-bit elements, and it has a 10-bit accumulator. The chip contains 12.450 transistors.
3.) J. Sparso, J. Staunstrup, and M. Dantzer-Sorensen. Design of delay insensitive circuits using multi-ring structures. In G. Musgrave, editor, Proc. of EURODAC~'92, European Design Automation Conference, Hamburg, Germany, September 7-10, 1992, IEEE Computer Society Press, 1992, pp. 15-20. September 1992.
J. Sparso and J. Staunstrup, "Design and Performance Analysis of Delay-Insensitive Multi-Ring Structures," To appear in: Proceedings of HICSS-26, January 5-8, 1993.} IEEE Computer Society Press, 1993.
4.) Jens. Sparso, Techn. University of Denmark, Lyngby

1.) Delay-insensitive multiply-accumulate unit. CONTACT: Christian D. Nielsen
2.)The design implements an iterative serial-parallel vector multiplication algorithm with carry-save representation of the temporary result. A test chip has been fabricated via MOSIS in a 2.0 micron CMOS technology. The chip multiplies vectors with 4-bit elements, and it produces an 8-bit result. The chip contains 3.124 transistors.
3.) Christian D. Nielsen and Alain J. Martin, "Design of a delay-insensitive multiply-accumulate unit." In Proceedings from HICSS-26, January 5-8, 1993. IEEE Computer Society Press, 1993. (To appear)
Christian D. Nielsen and Alain J. Martin, "A delay-insensitive multiply-accumulate unit." Computer Science Department, California Institute of Technology, Report Caltech-CS-TR-92-03, 1992.
4.) Jens. Sparso, Techn. University of Denmark, Lyngby

1.) A Zero-Overhead Self-timed 160 ns. 54-bit CMOS divider.
3.) Ted E. Williams. A zero-overhead self-timed 160 ns. 54 bit CMOS divider. IEEE Journal of Solid State Circuits, 26(11):1651--1661, 1991.
4.) Ted E. Williams

Dagstuhl-Seminar 9249:

List of Participants

Jochen **Beister**
Universität Kaiserslautern
FB Elektrotechnik
Postfach 3049
W-6750 Kaiserslautern
Germany
tel.: +49-631-205-28 40

Robert **Brayton**
University of California at Berkeley
Dept. of Electrical Engineering
and Computer Science
571 Evans
Berkeley CA 94720
USA
brayton@ic.berkeley.edu
tel.: +1-510-643-98 01

Reiner **Durchholz**
GMD-St. Augustin
Gesellschaft für Mathematik und
Datenverarbeitung mbH
Schloß Birlinghoven
5205 St. Augustin 1
Germany
durchholz@gmd.de
tel.: +49-2241-14 23 60

Jo C. **Ebergen**
University of Waterloo
Department of Computer Science
Waterloo Ontario N2L 3GI
Canada
jobergen@maytag.waterloo.edu
tel.: +1-519-888-46 66

Mark R. **Greenstreet**
University of British Columbia
Department of Computer Science
Vancouver BC V6T 1Z2
Canada
mrg@cs.ubc.ca
tel.: +1-604-822-30 61

Wolfgang **Hebgen**
Universität Kaiserslautern
Fachbereich Informatik
Postfach 3040
6570 Kaiserslautern
Germany
hebgen@informatik.uni-kl.de
tel.: +49-631-205-3143

Simon **Jones**
University of Loughborough
Dept. of Electrical & Electronical
Eigineering
Leicestershire LE11 3TU
Great Britain
s.r.jones@lut.ac.uk
tel.: +44-509-222-8 33

Michael **Kishinevsky**
Danmarks Teknikske Hojskole
Instituttet for Datateknik
Building 344
DK-2800 Lyngby
Denmark
mik@id.dth.dk
tel.: +45-45-93-12-22

Bernd **Kleinjohann**
CadLab Paderborn
Bahnhofstr. 32
W-4790 Paderborn
bernd@cadLab.cadLab.de
tel.: +49-5251-284-1 14

Markus **Kohn**
Universität Karlsruhe
Institut für Angewandte Informatik
Vincenz-Prießnitz-Str. 1
W-7500 Karlsruhe
mko@aifb.uni-karlsruhe.de

Alex **Kondratyev**
"Trassa" R&D Coop
Socialisticheskaya st. 9-31
191126 St.Petersburg
Russia
vico@trassa.spb.su

Bill **Lin**
IMEC Laborarory
Kapeldreef 75
3001 Leuven
Belgium
billlin@imec.be
tel.: +32-16-28 15 41

Charles **Molnar**
Washington University
Institute for Biomedical Computing
Computer Science Laboratory
St.Louis MO
USA
cem@wusbl.WUstl.edu

Christian D. **Nielsen**
Danmarks Teknikske Hojskole
Instituttet for Datateknik
DK-2800 Lyngby
Denmark
cdn@id.ith.dk
tel.: +45-45-93-33-32

Franz J. **Rammig**
Universität GH Paderborn
FB 17 - Mathematik/Informatik
Warburgerstr. 100
W-4790 Paderborn
Germany
franz@uni-paderborn.de
tel.: +49-5251-602-0 69

Roger **Sayle**
University of Edinburgh
Dept. of Computer Science
Mayfield Road
Edinburgh EH9 3JZ
Great Britain
ros@dcs.ed.ac.uk
tel.: +44-31-650-5163

Hartmut **Schmeck**
Universität Karlsruhe
Institut für angewandte Informatik
Vincenz-Prießnitz-Str. 1
W-7500 Karlsruhe
Germany
schmeck@aifb.uni-karlsruhe.de
tel.: +49-721-608-42 42

Einar **Smith**
GMD Schloß Birlinghoven
Postfach 13 16
W-5205 Sankt Augustin 1
Germany
Einar.Smith@gmd.de
tel.: +49-2241-14-27 80

Jens **Sparsoe**
Danmarks Teknikske Hojskole
Instituttet for Datateknik
DK-2800 Lyngby
Denmark
jsp@id.dth.dk
tel.: +45-45-93-33-32

Jorgen **Staunstrup**
Danmarks Teknikske Hojskole
Instituttet for Datateknik
DK-2800 Lyngby
Denmark
jst@id.dth.dk
tel.: +45-45 93 33 32

Alexander R. **Taubin**
"Trassa" R&D Coop
Socialisticheskaya st. 9-31
191126 St.Petersburg
Russia

Lothar **Thiele**
Universität des Saarlandes
Fachbereich 16 - Elektrotechnik
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
thiele@ll.uni-sb.de
tel.: +49-681-302-35 84

Peter **Vanbekbergen**
IMEC
Kapeldreef 75
B-3001 Leuven
Belgium

Gerd **Venzl**
SIEMENS AG - ZFE IS
Zentralabt. Forschung und Entwicklung
Otto-Hahn-Ring 6
W-8000 München 83
Germany
venzl@ztivax.siemens.de
tel.: +49-89-636-4 41 44

Michael **Yoeli**
Israel Institute of Technology
Computer Science Departement
Technion City
Haifa 32000
Israel
myoeli@cs.technion.ac.il
tel.: +972-4-29 43 67

Gerhard **Zimmermann**
Universität Kaiserslautern
FB Informatik
Postfach 3049
W-6750 Kaiserslautern
Germany
zimmerma@informatik.uni-kl.de
tel.: +49-631-205-26 28

Zuletzt erschienene und geplante Titel:

K. Compton, J.E. Pin , W. Thomas (editors):
Automata Theory: Infinite Computations, Dagstuhl-Seminar-Report; 28, 6.-10.1.92 (9202)

H. Langmaack, E. Neuhold, M. Paul (editors):
Software Construction - Foundation and Application, Dagstuhl-Seminar-Report; 29, 13..-17.1.92 (9203)

K. Ambos-Spies, S. Homer, U. Schöning (editors):
Structure and Complexity Theory, Dagstuhl-Seminar-Report; 30, 3.-7.02.92 (9206)

B. Booß, W. Coy, J.-M. Pflüger (editors):
Limits of Modelling with Programmed Machines, Dagstuhl-Seminar-Report; 31, 10.-14.2.92 (9207)

K. Compton, J.E. Pin , W. Thomas (editors):
Automata Theory: Infinite Computations, Dagstuhl-Seminar-Report; 28, 6.-10.1.92 (9202)

H. Langmaack, E. Neuhold, M. Paul (editors):
Software Construction - Foundation and Application, Dagstuhl-Seminar-Report; 29, 13.-17.1.92 (9203)

K. Ambos-Spies, S. Homer, U. Schöning (editors):
Structure and Complexity Theory, Dagstuhl-Seminar-Report; 30, 3.-7.2.92 (9206)

B. Booß, W. Coy, J.-M. Pflüger (editors):
Limits of Information-technological Models, Dagstuhl-Seminar-Report; 31, 10.-14.2.92 (9207)

N. Habermann, W.F. Tichy (editors):
Future Directions in Software Engineering, Dagstuhl-Seminar-Report; 32; 17.2.-21.2.92 (9208)

R. Cole, E.W. Mayr, F. Meyer auf der Heide (editors):
Parallel and Distributed Algorithms; Dagstuhl-Seminar-Report; 33; 2.3.-6.3.92 (9210)

P. Klint, T. Reps, G. Snelting (editors):
Programming Environments; Dagstuhl-Seminar-Report; 34; 9.3.-13.3.92 (9211)

H.-D. Ehrich, J.A. Goguen, A. Sernadas (editors):
Foundations of Information Systems Specification and Design; Dagstuhl-Seminar-Report; 35; 16.3.-19.3.9 (9212)

W. Damm, Ch. Hankin, J. Hughes (editors):
Functional Languages:
Compiler Technology and Parallelism; Dagstuhl-Seminar-Report; 36; 23.3.-27.3.92 (9213)

Th. Beth, W. Diffie, G.J. Simmons (editors):
System Security; Dagstuhl-Seminar-Report; 37; 30.3.-3.4.92 (9214)

C.A. Ellis, M. Jarke (editors):
Distributed Cooperation in Integrated Information Systems; Dagstuhl-Seminar-Report; 38; 5.4.-9.4.92 (9215)

J. Buchmann, H. Niederreiter, A.M. Odlyzko, H.G. Zimmer (editors):
Algorithms and Number Theory, Dagstuhl-Seminar-Report; 39; 22.06.-26.06.92 (9226)

E. Börger, Y. Gurevich, H. Kleine-Büning, M.M. Richter (editors):
Computer Science Logic, Dagstuhl-Seminar-Report; 40; 13.07.-17.07.92 (9229)

J. von zur Gathen, M. Karpinski, D. Kozen (editors):
Algebraic Complexity and Parallelism, Dagstuhl-Seminar-Report; 41; 20.07.-24.07.92 (9230)

F. Baader, J. Siekmann, W. Snyder (editors):
6th International Workshop on Unification, Dagstuhl-Seminar-Report; 42; 29.07.-31.07.92 (9231)

J.W. Davenport, F. Krückeberg, R.E. Moore, S. Rump (editors):
Symbolic, algebraic and validated numerical Computation, Dagstuhl-Seminar-Report; 43; 03.08.-07.08.92 (9232)

R. Cohen, R. Kass, C. Paris, W. Wahlster (editors):
Third International Workshop on User Modeling (UM'92), Dagstuhl-Seminar-Report; 44; 10.-13.8.92 (9233)

R. Reischuk, D. Uhlig (editors):
Complexity and Realization of Boolean Functions, Dagstuhl-Seminar-Report; 45; 24.08.-28.08.92 (9235)

Th. Lengauer, D. Schomburg, M.S. Waterman (editors):
Molecular Bioinformatics, Dagstuhl-Seminar-Report; 46; 07.09.-11.09.92 (9237)

V.R. Basili, H.D. Rombach, R.W. Selby (editors):
Experimental Software Engineering Issues, Dagstuhl-Seminar-Report; 47; 14.-18.09.92 (9238)

Y. Dittrich, H. Hastedt, P. Schefe (editors):
Computer Science and Philosophy, Dagstuhl-Seminar-Report; 48; 21.09.-25.09.92 (9239)

R.P. Daley, U. Furbach, K.P. Jantke (editors):
Analogical and Inductive Inference 1992 , Dagstuhl-Seminar-Report; 49; 05.10.-09.10.92 (9241)

E. Novak, St. Smale, J.F. Traub (editors):
Algorithms and Complexity for Continuous Problems, Dagstuhl-Seminar-Report; 50; 12.10.-16.10.92 (9242)

J. Encarnação, J. Foley (editors):
Multimedia - System Architectures and Applications, Dagstuhl-Seminar-Report; 51; 02.11.-06.11.92 (9245)

F.J. Rammig, J. Staunstrup, G. Zimmermann (editors):
Self-Timed Design, Dagstuhl-Seminar-Report; 52; 30.11.-04.12.92 (9249)

B. Courcelle, H. Ehrig, G. Rozenberg, H.J. Schneider (editors):
Graph-Transformations in Computer Science, Dagstuhl-Seminar-Report; 53; 04.01.-08.01.93 (9301)

A. Arnold, L. Priese, R. Vollmar (editors):
Automata Theory: Distributed Models, Dagstuhl-Seminar-Report; 54; 11.01.-15.01.93 (9302)

W. Cellary, K. Vidyasankar , G. Vossen (editors):
Versioning in Database Management Systems, Dagstuhl-Seminar-Report; 55; 01.02.-05.02.93 (9305)

B. Becker, R. Bryant, Ch. Meinel (editors):
Computer Aided Design and Test , Dagstuhl-Seminar-Report; 56; 15.02.-19.02.93 (9307)

M. Pinkal, R. Scha, L. Schubert (editors):
Semantic Formalisms in Natural Language Processing, Dagstuhl-Seminar-Report; 57; 23.02.-26.02.93 (9308)

H. Bibel, K. Furukawa, M. Stickel (editors):
Deduction , Dagstuhl-Seminar-Report; 58; 08.03.-12.03.93 (9310)

H. Alt, B. Chazelle, E. Welzl (editors):
Computational Geometry, Dagstuhl-Seminar-Report; 59; 22.03.-26.03.93 (9312)

J. Pustejovsky, H. Kamp (editors):
Universals in the Lexicon: At the Intersection of Lexical Semantic Theories, Dagstuhl-Seminar-Report; 60; 29.03.-02.04.93 (9313)

W. Straßer, F. Wahl (editors):
Graphics & Robotics, Dagstuhl-Seminar-Report; 61; 19.04.-22.04.93 (9316)

C. Beeri, A. Heuer, G. Saake, S.D. Urban (editors):
Formal Aspects of Object Base Dynamics , Dagstuhl-Seminar-Report; 62; 26.04.-30.04.93 (9317)