# CONCURRENT HARDWARE

## *The Theory and Practice of Self-timed Design*

**Michael Kishinevsky, Alex Kondratyev, Alexander Taubin and Victor Varshavsky**

WILEY SERIES IN PARALLEL COMPUTING

# CONCURRENT HARDWARE

## The Theory and Practice of Self-timed Design

**Michael Kishinevsky**
*R&D Coop TRASSA, Russia*
*Technical University of Denmark*

and

**Alex Kondratyev, Alexander Taubin**
and
**Victor Varshavsky**
*R&D Coop TRASSA, Russia*
*University of Aizu, Japan*



Translated by
***Alex Yakovlev, Eric Napelbaum***
and
***Olga Reva***

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto · Singapore

Baffins Lane, Chichester
West Sussex PO19 1UD, England
Telephone (+44) (243) 779777



*Other Wiley Editorial Offices*

John Wiley & Sons, Inc., 605 Third Avenue,
New York, NY 10158-0012, USA

Jacaranda Wiley Ltd, 33 Park Road, Milton,
Queensland 4064, Australia

John Wiley & Sons (Canada) Ltd, 22 Worcester Road,
Rexdale, Ontario M9W 1L1, Canada

John Wiley & Sons (SEA) Pte Ltd, 37 Jalan Pemimpin 05-04,
Block B, Union Industrial Building, Singapore 2057

***Library of Congress Cataloging-in-Publication Data***

Concurrent hardware : the theory and practice of self-timed design / M. Kishinevsky ... [et al.].
p. cm.
Includes bibliographical references (p. ) and index.
ISBN 0 471 93536 0
1. Parallel computers—Design and construction. I. Kishinevskiĭ M. A. (Mikhail Aleksandrovich)
QA76.58.C666 1994
004'.35—dc20 93-29284
CIP

***British Library Cataloguing in Publication Data***

A catalogue record for this book is available from the British Library

ISBN 0 471 93536 0

Produced from camera-ready copy supplied by the authors using LaTeX
Printed and bound in Great Britain by Bookcraft (Bath) Ltd

# CONCURRENT HARDWARE

# WILEY SERIES IN PARALLEL COMPUTING

## SERIES EDITORS:



---

**Carey (ed.):** Parallel Supercomputing: Methods, Algorithms and Applications

**de Bakker (ed.):** Languages for Parallel Architectures: Design, Semantics, Implementation Models

**Axford:** Concurrent Programming: Fundamental Techniques for Real-Time and Parallel Software Design

**Gelenbe:** Multiprocessor Performance

**Treleaven (ed.):** Parallel Computers: Object-oriented, Functional, Logic

**Williams:** Programming Models for Parallel Systems

**Raynal and Helary:** Synchronization and Control of Distributed Systems and Programs

**Takeuchi:** Parallel Logic Programming

**Eliens:** DLP — A Language for Distributed Logic Programming: Design, Semantics and Implementation

**Kacsuk and Wise (eds):** Implementations of Distributed Prolog

**Pitas (ed.):** Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks

**Kishinevsky *et al*:** Concurrent Hardware: The Theory and Practice of Self-timed Design

*Моя мечта надменна и проста!*
*Схватить весло, поставить ногу в стремя*
*И обмануть медлительное время ...*
**Н.С. Гумилев, "Дон Жуан"**

*My simple dream is haughty and sublime!*
*To take an oar, to spur a steed of mine*
*And to deceive His Majesty the Time...*
**N.S. Gumilev**

*To speak a language is to commit ourselves to the double indeterminacy due to our reliance both on its formalism and on our own continued reconsideration of this formalism in its bearing on experience.*
**Michael Polanyi, "Personal Knowldege"**

# Preface

*Why should I know what I think?*
*Let me say it first – then I know.*
**B.Zakhoder**
The Foreword to the Russian version of
"Alice in Wonderland" by L.Carroll.

Perhaps, we should have cut this preface to a haughty phrase, "We have written this book just because we wanted to write it".

In our case this confession would have been true. The chief thing for us was that we really longed to work together. Of course, the current situation in science was also conducive to the creation of this volume. It prompted the direction in which we had "to take the offensive".

In 1984 we completed our work on the book entitled "Self-timed control of concurrent processes" (the Russian version was published in 1986, with the English translation in 1990). As usual, only after we had finished it, we realized that everything should have been done in a different way. But later it became clear that it was not just a result of a spiritual bankruptcy caused by the parting with the fostered piece of writing (the state that can probably be best described by the term "post-natal depression"). It was rather a reflection of the objective necessity. The theory of self-timing, was living through its usual kind of crisis: almost everything was solvable in theory but the tools proposed here and there filled us with a feeling of dissatisfaction. The exponential barrier of complexity hindered practical implementation of the analysis and synthesis methods. We really felt as if we had poured an old vintage into a modern high-tech container.

We were confronting the problem of translating the results of the self-timed circuit theory into a new language (not from Russian into English, of course, but into the language of compact event-based models). The purity of the scientific field here was a reminder of the glorious times of conquering the Wild West. We were working easily and freely, without looking back to any authorities.

During the past decade the "ideology of event-based specifications" has firmly taken hold of the masses and created its gurus. This inspires the hope that our book will find its interested reader.

No doubt "the book which is not worth reading twice is not worth reading even once". We believe that we have come close to the requirements of this old universal criterion since few readers could grasp the involved picture of the results at first reading. This is not a consequence of an ill-intentioned confusion but rather of a striving for formal proofs and substantiations.

In fact, the main ideas of the book are simple and few in number (approximately one for each chapter).

We shall try to set our reader's imagination free suggesting a somewhat flippant associative row to illustrate our main ideas (we all once took an interest in psychology and remembered the importance of the first impression or, if you wish, imprinting).

A relatively scarce "foreign experience" of the authors suggests that the only thing the West is short of, but Russia abounds in, is anecdotes.

The remainder of this foreword is built on the following pattern: a title of the chapter, an anecdote to illustrate it and the comment of a clever narrator. So, ...

*Chapter 1. The Models for Parallel Processes Specification...*

At an international competition of dwarfs a dwarf from America won the first prize. The Russian delegation appealed against the jury's decision by saying, "The decision is not fair for our dwarf is bigger".

The pivot of the chapter is the event-based model called change diagram (CD). The necessity to introduce our own original model can be accounted for by the following two reasons:

1) At that time (we mean 1982-1984) "the American dwarf" (a model of Signal Transition Diagram by T.A.Chu) had not yet been born. 2) "Our dwarf" indeed proved bigger. The CD model allows description of a wider scope of self-timed circuits and is supplied with effective analysis tools. But this brings us to the next chapter.

*Chapter 2. Verification of Parallel System Behavior with Formal Models.*

The scientists wanted to compare the quickness of a monkey's with that of a man. High up on the tree a banana was hung. The monkey began to shake the tree - to no avail. Then it scratched its head, took a stick and knocked the banana down.

A student of a Computer Science faculty volunteers for the test. He shakes the tree again and again. The student is sweating, the experimenters are in despair, the banana does not fall down. A young apprentice shouts to the student, "Hold on, buddy, just give it a bit of thought!". But without a break in his action the volunteer says, gasping, "Come on ... Why should I think? ... I have been taught to shake..."

Let not the followers of the verification methods by global state models take offence for our innocent analogy with the tenacious student. Without their continuous effort we would have never obtained our present results. The transition from the state-based to the event-based models demands, indeed, a certain "toggle" in thinking, but it is fully justified allowing us to reduce the complexity of the problem from exponential to polynomial.

*Chapter 3. Relationship Between State-Based and Event-Based Models.*

Albert Einstein failed to create integral theory of space and time. But Sergeant Petrov succeeded. This morning he gave his young soldiers a command, "Dig a trench from this post to lunch!".

The laurels of Sergeant Petrov would not let us have a moment's peace. Without elucidating the relations between habitual global and event-based models we could not get an integral idea of the processes in self-timing. The establishment of a one-to-one correspondence between CDs and transition diagrams justifies the use of CDs as a main tool for the design of self-timed circuits. In practice it gives a tempting possibility of handling the models of both types together.

*Chapter 4. Behavior Identification and Circuit Analysis.*

The teacher told the class to bring ten roubles each for the starving people of Donako. Everybody brought the money except Little Vovotchka who explained that his father said there were no starving people in Donako.

Next time the teacher collected the money to protect the Nature of Donako, but Little Vovotchka again gave nothing. This time his father said that everything was OK with Nature in Donako.

On the third occasion a charity was set up to support the Fraternal Communist Party of Donako. To everyone's surprise Little Vovotchka brought twenty roubles, "My father said that if there are communists in Donako, then there should also be both starving people and dying nature."

In Chapter 4 circuits are analyzed through the reconstruction of the event-based model. The method operates in naive confidence that such a model can be generated by the circuit. But with any incorrectness detected, the algorithm behaves like Little Vovotchka - rejects the illusions (part of the reconstructed behaviour) and reports when the circuit is "bad". If this analysis is successful, it is then possibile to compare the circuit and the model of its behavior (circuit identification).

*Chapter 5. Transformation of the Behavior Specification.*

Everlasting peace was declared in the jungle, but the very next day a tiger attempted to kill a goat. Luckily, the victim managed to escape. The animals summoned a court to try the tiger. "Cut off his head!" demanded the monkey. "Not the whole head, not straight off", pronounced the presiding elephant, "let's cut off his tail." "OK", agreed the monkey. "Let it be the tail... But up to his ears!".

CD is a compact language. But high-level languages are even more compact. They are usually worked out hierarchically, top-down. The language constructs are traditionally introduced so as to suit the potential user who pays little attention to the implementations compared to the problem descriptions ("from head to tail"). We are concerned about circuits and we prefer departing from them, so our language meccano generalizes the practice of the design upon the CD basis and are built mainly bottom-up. This way yields the specifications of sufficiently high level ("from the tail but as far as the ears").

*Chapter 6. Methods of Direct Translation of Specifications into Circuits.*

Second World War. Front line. Heavy fire. A soldier broke through to the inspecting general with a suggestion to build a gun which would shoot with a range of 200 miles. The soldier is immediately taken to Stalin. "Well, well, well! Now tell us how to make it!" demands the generalissimo. "My business is only to suggest. But how to do it ... You've got these designers", was the reply.

We do not want to be that soldier, that is why our book contains Chapters 6 and 7 instructing how to build circuits using these models. This task turned out to have a rather simple solution; as in a meccano: "a brick to a brick" - and the circuit is ready. This is a syntax-directed, modular or structural synthesis. Chapter 6 presents the scope of such synthesis methods whereby the CD graph is directly translated into a circuit structure.

*Chapter 7. Formal Methods of Synthesis...*

A Soviet leader, Nikita Khruschev once did not like some orator's speech at the UN Assembly. He took off his shoe and began knocking it on the table. When reporters asked

Churchill what he thought about such behavior, he said, "I don't mind him using his shoe, the British parliament has seen even worse actions. But to have put on yellow shoes with such a suit is really impossible!"

Frankly speaking, we are fond of the English classical style. The formal and strict synthesis methods presented in this chapter will hopefully balance a certain engineering eclecticism akin to the structural approach of the preceding chapter. Not every CD implies a physical circuit. The formal theory enables not only analysis of the shortcomings of this or that CD ("yellow shoes") but also suggests effective ways for its "modernization" (not to say "perestroika") to achieve implementability.

*Chapter 8. Review of the State-of-the-Art in Self-Timing.*

At a history lesson the teacher asks, "Who was Stalin?", a girl answers, "A communist leader who messed up the USSR economy in the 40s." "Sit down, your mark is 'excellent'. And who was Breznev?" Another girl answers, "Breznev was a communist leader who messed up the USSR economy in the 70s." "Sit down, excellent. And who is Gorbatchev?" Vovotchka who has been holding out his hand for a long time, stands up to blurt out, "Gorbatchev is a communist leader who messed up the USSR economy in the 80s!" "Sit down, your mark is 'bad' ... so far."

We did not know how soon this joke would have become out of date. Maybe our review will face the same destiny. Its main observations and conclusions are not very likely to suffer from time, but any evaluation of the latest achievements is changeable and dynamic.

*Chapter 9. The FORCAGE System and Self-Timed Circuit Design.*

(It supplements the material of the book together with a CAD-demo disk).

A psychiatric patient boasts of the advantages of his new asylum, "We have a swimming pool there with a ten meter high tower for diving. And our Doctor promises us that if we learn to dive well, he will let us have it filled with water."

Deprived of the practical application, our book would be like diving into a swimming pool without water. To avoid the criticisms that our book is fruitless and abstract we are supplying with it a demo-version of a software system called FORCAGE in order to support the self-timed design. The results, in theory, have laid the foundation for the development of this system.

Prefaces are usually concluded with *acknowledgements*, but we dared not express them in such an off-hand manner. We shall do it later, in our *introduction*, by the end of which we hope to have gained a sufficient degree of seriousness.

# Introduction

*Why did time exist? Why always this idiotic succession of one thing after another, and not a roaring surfeiting simultaneity?*
**H.Hesse, "Klingsor's Last Summer"**

The rapid advance in semiconductor technology (IC, LSI, VLSI, ULSI...) has generated a euphoria among the designers of computing systems. Before very long the difference between "I want" and "I can" seems to have vanished altogether. Despite the physical limit for miniaturization which has now become quite near, the desire to build more complex and faster chips remains unabated, and "the eyes are still greedier than the mouth". As a result of this "physiological" dissatisfaction, the designers have refocussed their effort into the search for new architectures.

In the title of this book we have used two terms: *concurrency* and *self-timing*. The first is very popular in modern computer science. "More parallelism - good and varied" seems to be the motto of today's design. But the collocation "concurrent hardware" must be explained.

It is rather usual to see the notions of concurrency, concurrent process, concurrent system etc., in the context related to the fields of software engineering and computer system architecture, both at the modelling and application level.

The concept of self-timing encompasses both parallelism in the behavior of the system and asynchronism in the interaction of the components at the "lowest" hardware level, where possible ordering in the switching of logical elements (gates) can be considered. In that sense, all the results of the concurrency theory are applicable in the theory of self-timed systems. We believe that the converse is also true – the results of the self-timing theory can be useful and necessary in the theory of concurrency.

The term "self-timing" is well-known. For the last 35 years, since the publication of the pioneering works by David E. Muller and his followers *(the speed-independent circuits theory)*, self-timing still remains somewhat exotic in theory and design. But, especially during the last decade, the scope of research, development and design in this field has been gradually broadened. It may be because the exotic "islands" have always been attractive not only for the tourists. We are the natives of those lands, the majority of us "originated" there and we hope that the description of the living conditions and opening horizons can prove interesting and attractive for the reader.

We must confess that our argument about a possible influence of self-timing upon the development of general architectural ideas is not new. The ideas of data-flow architecture had found their roots in the general ideas of self-timing and it was not incidental that the

authors of the data-flow concept, e.g. Jack B. Dennis, made their contribution into the development of self-timing principles.

The fundamental idea of a self-timed circuit design is a *principle of a two-phase handshake operation* . The process of any complexity can be modelled as a delay between the signals of its initiation and completion, i.e. between the changes of its phase states. As a result, if such a process is included in a series with some gate taken in a self-timed circuit, it will not change the partial ordering of the events in the original circuit, whose behavior does not depend on the delay of its elements. Two important conclusions can be drawn from this statement:

- for every concurrent system one can construct a self-timed circuit, in which the states of the outputs of its gates model (represent) the phase states of the processes, i.e. for every concurrent system there can be constructed its hardware model;
- such a hardware model can be used as a control device coordinating a concurrent system.

Thus, the term *"concurrent hardware"* has a dual meaning. On the one hand, a self-timed circuit is a concurrent system with respect to its components. On the other hand, a self-timed circuit is the hardware to coordinate the processes (control) in a concurrent system.

It would be fair to state that the above ideas emerged long after the beginning and spreading of the work on self-timing. The true impact to such research and the introduction of the term "self-timing" (speed-independent, delay-insensitive) were due to real difficulties in the external timing of circuits and high complexity systems.

Meanwhile the obstacles against constructing the global timing systems kept growing.

One of the main problems in superminiaturization is the necessity to allow for the duration and variation of the delays in gates and interconnections between chips.

For the 3 micron technology, the gate delay is about 3 nsec, and the device performance is not seriously affected by the variations in time parameters at the level of several hundred picoseconds. For the 1.5 micron technology, the gate delay and the variations in the time parameters turn comparable, which often causes the violation of the circuit's functionality. In submicron technology this grows into a major trouble – the delays at the interconnections become a dominating factor (the circuit proves to be made "not of the gates connected by wires but of the wires connected through the gates").

The abovementioned circumstances made the designers focus their attention on the behavioral modelling of circuits. Unfortunately, as often in such cases, the hopes are pinned on the "front attack". It is suggested, for example, that a simulation program must emulate the timing waveforms of a real circuit in the operation mode allowing for the variations in values of the physical parameters of a particular chip [1]. In practice this implies enumerating the values of the delays of the real gates and interconnections, which is bound to result in "combinatorial explosion" thus making the solution absolutely hopeless.

*The only alternative to the enumeration of values of real component delays is to guarantee that the circuit acts correctly for any combination of the delay values, i.e. that the circuit is self-timed.*

Thus, undoubtedly, we are the adherents to the self-timed circuits (STCs). What are the major "pros and cons" of self-timed design?

1. As integration scale goes up the traditional clock systems are becoming bulkier and less efficient. Today, in the large circuits the number of synchro-threads is over a dozen and the clocks specially adjustable to the circuit parameters are being constructed. It has

even come to "suspending" a laser above the chip [2] to provide the simultaneous start of the clocks in several equichronic regions (such a region is an area on the chip within which reliable external clocking can be implemented without loss of speed). The size of equichronic regions deceases with the increase of integration scale. Therefore, the number of regions per chip gets larger and so constructing an integral clock system becomes more and more problematic.

*Self-timed systems do not require any common clocks at all.*

2. The idea of self-timing is based upon the possibility of controlling the duration of the phase transition (the analogue of clock signal) by the built-in circuits indicating the completion of the transient processes. With this capability, STCs can operate correctly under any deviations of the component delays from the standard values and are *degradation failure-insensitive*. Moreover, they fully utilize the speed properties of the elements since they operate on the actual rather than maximum delays.

3. STCs proved to be *self-checking with respect to stuck-at faults* [3,4]. (The circuit simply halts when the output of any element gets stuck at a constant 0 or 1). This allows for *self-repair* in a simple way.

4. The latter issue is especially important in constructing Ultra-LSI or wafer-scale integrated circuits. It is the possibility of warding off the defects on the wafer through reconfiguration which is acknowledged as the decisive condition in the organization of such circuits, guaranteeing a *sufficiently high yield* [5].

5. From the standpoint of the popular concept of silicon compilation, it seems most attractive *how easy would be to reimplement an STC in a technology with a different feature size* (the layout scaling does not cause any degradation in the STC functionality).

However, all these nice features are only achievable at the expense of the following:

- The area cost increases. (Usually, the increase is considered to be twofold, but we would recommend to look at this pessimistic estimation with some care. For a wide range of applications, such as counters, pipelines and memory, to name but a few, we have obtained the solutions whose complexity exceeded that of their synchronous analogues by 10-20% only.)

- The design is more difficult in comparison with the standard techniques for synchronous circuits. It was the complexity of their design that hindered the proliferation of STCs. Manual STC design ("on the piece of paper") demands rather high skills. For the design of such circuits to be automated it is necessary to have a powerful theoretical foundation. And so, the STC theory is the major subject of this book.

The basic tool for the STC study and design is a behavioral model since semantically self-timing can be best defined in terms of behavior: through the causal relations between the switchings of the elements. According to the experts' opinion, it is namely the "behavioral design" that is to become the pivot of the CAD development in the 90s [6]. With submicron technology all the designers will face the tasks of specification, verification and implementation of the circuit behavior refined to the level of the transitions of signals in gates and wires. As a result, the complexity in obtaining synchronous implementations may even outgrow the complexity of the STC design.

This enables us to speak about the complexity in the design of self-timed and other VLSI circuits as levelling.

As we have already mentioned, D.E. Muller was the first to study the circuits whose behavior does not depend on the relative times of the elements' reactions. At the end of the 50s, he developed the theory of speed-independent circuits and suggested a method for specifying their behavior, which is now known as the Muller model.

Muller also investigated some methods of analysis and synthesis for such circuits [7-12].

Although ahead of its time, Muller's work found no practical success, chiefly because the level of semiconductor technology was inadequate (discrete elements) and the subject of practical circuit engineering was not sufficiently studied.

The concept of "aperiodic automata with self-synchronization" [13] based on the two-phase handshake principle of behavioral organization seems to be the next stage in the STC development. This concept took its final shape during the period of 1972-1975 and the first book on STC design saw its publication in 1976 [14].

The advancements in the STC theory and practice of the next decade were reflected in a collective monograph [4]. The main results of that period were due to the development of basic circuitry, methods of STC modular design and STC analysis methods based upon the global states model (transition diagram of the Muller model). For the first time the self-checking properties of STCs and the methods for the selfrepair were thoroughly investigated.

For STCs, as well as for computer technology on the whole, the modern period of development is characterized by the total "assault" of the models, methods and algorithms, specially targeted at design automation.

With enormous variety and complexity of the circuits now being developed, the VLSI design discipline is increasingly more like "software engineering". Such an approach usually suggests a "top-down design" of hardware, from the behavior specifications to the circuit layout [15]. Behavioral design [6] is now becoming the kernel of modern CAD.

The experts in STCs more than anyone else have been in need of the CAD tools which would enable an "amateur" to design efficient circuits ("VLSI programming"). Long ago they were confronted with the problems in behavioral modelling. It is no wonder that four groups, in the California Institute of Technology [16], Stanford University [17], Technical University of Denmark [18] and the Philips Research Laboratories [19] have stated their intention to construct CAD for STC "programming".

In such an approach, the key issue is to choose and formally investigate some model that would allow the specification of finely grained dynamic phenomena in the self-timed circuit behavior.

The models studied by the above-named groups, such as trace structures, abstract circuits, production rules and models of synchronized transitions are aimed exactly at this objective.

To overcome the verification difficulties, some original criteria of the circuit behavior correctness are introduced into such models of the lower level. But, what is typical for them, the question of the correspondence of these criteria to the Muller theory, is ignored. As a result, the class of implementable processes becomes artificially narrowed, and quite "good" circuits, falling into this class, are claimed to be incorrect (and, moreover, nonsemi-modular [20]). Thus the authors, being unaware of the existence of the types of behavior rejected by them (e.g. in [20,21]), virtually ignore the existence of the subclass of semi-modular circuits with the term takeover.

Choosing the formal specification language is always a pivotal question. On the one hand, the desire and effort to make the model as simple as possible is quite natural. But

it is dangerous to overdo that effort, impoverishing the "instrumental opportunities" of the designer. The "short-sightedness" of the model may strongly diminish the yield of efficient implementations.

On the other hand, overly powerful models generate complex algorithms of synthesis and analysis. To a certain extent it refers, for example to the classical Muller's *transition diagrams (TD)*, which are practically impossible to apply for the real tasks due to the exponential (to the size of the initial specification) number of the states in them. (It is a general shortcoming of models in global states.)

Balancing between these two extremes, we have tried to suggest a new event-based model, called *the change diagram (CD) (Chapter 1)*.

CD is a formalism which guarantees the complexity of the verification algorithms to be polynomial, but not exponential to the specification size *(Chapter 2)*. This enables to work, both in the analysis *(Chapter 2 and 4)* and formal synthesis *(Chapter 7)* procedures, with the circuits as complex as hundreds of gates. At the same time, the modular synthesis methodology, based on the CD language *(Chapter 6)*, the decomposition methods and the methods for the direct translation of CD into circuits *(Chapter 5 and 6)* all enable the automated design *(Chapter 9)* of systems of practically any complexity.

No less important is the theoretically proved fact that correct CD and semi-modular TD possess equal descriptive power *(Chapter 3)*. And thus, passing from TD to a compact CD model, we lose no specification details. CD is the only event-based model to possess such attractive qualities. Hence it has been chosen to become the basic description language within the scope of the present book.

Numerous interesting publications on self-timing have arrived in an avalanche during the last few years, with gifted young researchers emerging on the "self-timing horizon" and new knowledge "crystallizations centres" springing up everywhere - all this provides indirect evidence that the interests of applied microelectronics and pure science have finally matched the STC design automation. The groups dealing with the study and design of STC have been organized in most of the leading academic centers of the USA, Europe and Japan [16-30] (cf. *Table 8.1. "BRIEF INFORMATION ON MAIN CENTERS OF RESEARCH IN SELF-TIMING" and Chapter 8*).

Such firms as Intel, Digital Equipment Corp., Sun Microsystems, SGS-Thomson, Philips, Hewlett-Packard, Mastek etc., began to finance the work on STC.

In this wave the main practical achievements have been obtained on the way of "evolutionary penetration" into synchronous environment. The bestknown of such devices are: self-timed mesh-routing chips for the Intel Touchstone Delta multicomputer (Ch. Seitz *et al.*, Caltech), micropipeline registers (Turing Award lecture by Ivan E. Sutherland), pipeline and ring iterative structures (a division chip by T. Williams), interface blocks in the data transmission processor (Mitsubishi Corp.) and modules made by Austek Microsystems Ltd.

The present book is envisaged by the authors as a contribution, within their powers, to a common theoretical "box", making the foundation for such studies.

The above-stated scope of questions concerning the parallel STC design has already been touched in the monographs [14] and [4]. Those books are the snapshots of their respective up-to-the-date results and are not characterized by a sufficient theoretical or conceptual integrity.

This book makes an attempt to systematically present an integral design theory for concurrent self-timed systems, from their general specification and to concrete implementation algorithms. Most of our results on concurrency are fall beyond the scope of the STC theory

and from the authors' viewpoint may be of interest for the general theory of the formal models of concurrent processes.

## Acknowledgments

First of all we would like to state that we are much indebted to our colleagues without whose advice we would never have become professionals and this volume would not have been possible.

They are: Leonid Rosenblum, Vyacheslav Marakhovsky, Valery Peschansky and Boris Tsirlin. We would also like to mention here Nikolai Starodubzev, who took an active part in the discussions which led to the creation of Change Diagram as a model.

We are grateful to the Russian specialists: Alexander Astanovsky, Alexander Yakovlev, Hary Tani, Dmitry Pospelov, Vadim Kotov, Pavel Parkhomenko, Adolf Filin, Yuri Stepchenkov, Anatoli Chebotarev for their attention to our work.

We would also like to thank the scientists whose work or advice produced a stimulating effect on the preparation of this book: David Muller, Charles Molnar, Tam-Anh Chu, Robert Brayton, Luciano Lavagno, Michael Yoeli, Teresa Meng, Jørgen Staunstrup, Gert Goossens, Peter Vanbegbergen, Alain Martin, David Dill, Jo Ebergen, John Brzozowski, Martin Rem, Jan Udding, Mark Josephs, Dines Bjørner and many many others.

Our thanks to Luciano Lavagno, Jorgen Staunstrup, Christian Nielsen and Ganesh Gopalakrishnan for their remarks to the "Table of Self-timing World" (Table 8.1).

We highly appreciate titanic efforts of Alex Yakovlev, who translated the Russian English of this book into proper English. He became the first reader of our book and his comments were extremely helpful in preparing the final version. Eric Napelbaum and Olga Reva produced the first translation of this book.

Special thanks to Vadim Yakker and two Elenas (Kishinevsky and Varshavsky) for their technical assistance and software support in the development of our CAD system "FORCAGE".

We are especially grateful to IRCA (Instituto per la Riserca Applicata, S.p.A., Milano, Italy) and Professors Gian Paolo Caliguri and Umberto Pellegrini for support of our work (including book preparation) during the period of 1989 – 1992.

M. Kishinevsky is grateful to the "Mogens Balslevs Fond" which supported his stay at the Department of Computer Science, Technical University of Denmark. The preliminary version of the book was used for a course on Self-timed Design given at the Technical University of Denmark, where the students did their best in finding errors in the text.

Finally we want to express our deepest gratitude to our relatives, friends and colleagues for their patience in coping with the outbursts of our cantankerous characters. This was entirely due to the tight schedule of our work.

# Contents