DIGITAL DESIGN AUTOMATION: FINITE STATE MACHINE DESIGN

Marek Perkowski
Department of Electrical Engineering
Portland State University
P. O. Box 751
Portland, OR 97207

I. INTRODUCTION.

The currently used technologies for design of digital circuits include, among others, gate arrays, standard cells, custom and semi-custom circuits and Programmable Logic Devices. Many design automation tools are currently available on the market for these technologies from different vendors. They include: schematic capture, logic minimization, geometrical cell layout, placement, routing, design rule checking, simulation and other. In this paper we will concentrate on high-level tools that are relatively independent of the technology and are applied to all the above technologies. Tools of this type have been built for a number of years by large hardware companies and at universities and are starting to arrive also from companies that produce commercially available CAD software: ENDOT Inc., and Daisy being the first examples [73, 43].

While the circuit/system design on most current workstation systems starts either from the netlist of the circuit and/or logic equations of a Boolean function described in a textual form or from the schematic capture of a logic circuit, the new generation of software design tools will start from higher level descriptions and will generate these data in resultant or intermediate formats only. Since the lower design stages are well known, we will concentrate in this paper on the upper ones and we will briefly present design methods and software which produces data for the current CAD systems in the form of netlists, minimized truth tables and logic expressions.

High level descriptions available in the in-house tools from large companies, like IBM, GE, GTE, RCA, INTEL include: register-transfer languages, finite-state machines, Boolean expressions and structural languages. Universities (Carnegie-Mellon, Stanford, UCLA, Kiel) experiment with various system description hardware languages, regular expressions, Petri Nets, nondeterministic and parallel automata, recursive equations, general purpose languages like Lisp, C, Concurrent Prolog or Ada as the high-level design automation source formats.

It is not clear these days, which direction design automation will follow with respect to the high level languages and design methodologies. Some trends are however clearly visible while attending the premiere conferences like the Design Automation Conference or International Conference on Computer Aided Design and watching the new software tools presented, and especially talking to the company's representatives about future plans. These trends include: integration of design/analysis/verification tools into comprehensive systems; giving the designer choices to select various design styles, and providing him/her with tools to build his own CAD software; mastering user-friendly interfaces based on windows and graphics; incorporating Boolean minimization methods and technology adaptation methods for various design styles, not only PLAs; using finite-state machine description for custom cell design.

In this paper I will concentrate on the topics of finite state machine design tools, that in my opinion are one of the most important tasks in CAD tools development for the coming years. At Portland State University we investigate various algorithms and programs for designing finite state machines (FSMs) with the goal of building a comprehensive FSM design system. The following selection of topics will present a subjective perspective; from large number of approaches we disscuss those, which are related to methods and algorithms applied in our system. Other important high-level issues like simulation, verification, and data-path design will be not discussed in this paper due to size restrictions.

Good tutorials on CAD VLSI tools can be found in [131, 154, 115, 135]. The recommended logic design textbooks, useful from the point of view of designing CAD tools for logic design are: [26, 173, 148, 68, 123, 95, 77, 176, 112, 127, 179, 186, 38].

First, various methods of describing sequential circuits realized as finite-state machines will be presented. Next, various methods to optimize FSMs will be presented, including minimization of the number of states of machines, state assignment of machines' internal states, decomposition, and partitioning of machines. We will mention asynchronous circuits design methodologies that will be also gaining importance in coming years. Finally we will discuss modern CAD tools for logic minimization.

Papers presented in this session will be all related to implementation of software tools for FSM design. The first paper will present optimization methods for designing with PLDs (Programmable Logic Devices) using industrial tools. The author discusses both the implementer's and the user's aspects of PLD design tools. Such devices permit for very fast and inexpensive design (programming) of logic and sequential circuits from small to medium size circuits. They are expected to have a massive impact on the market and are of increasingly applied by electrial engineers who are not able or do not require using more expensive highly integrated technologies. The next paper will describe the user's interface to the design automation system Diades, under design at Department of Electrical Engineering at PSU. Designing better user interfaces is very important from the practical point of view, not only in industrial, but also in the University environment, where most of the system's users are only casual digital designers and have troubles with quickly learning how to use many various CAD tools. The system is implemented in Lisp, Prolog, Pascal, C and Fortran on a Vax 11/750 computer under Unix and cooperates with the widely used UC Berkeley CAD VLSI tools. The system can be a front-end to both PLDs and Custom VLSI circuits design tools. The last two papers describe two tools, implemented in C and Pascal, which are not only useful, but can be easily implemented on personal computers. They will be used by everybody interested in logic design, but having no access to expensive mainframes - undergraduate PSU students being the first, to simulate and design logic circuits on their personal home computers. Portable register-transfer simulator, described in the third paper will permit for simulation of logic circuits on logic and register-transfer levels. Visualization of logic values in time or geometrical domains is helpful to understand intuitively a circuit's behavior and was proven a very useful design/debugging tool for both novice and experienced designers. Finally , the last paper describes a Boolean minimizer based on the new principle of graph coloring. It can be useful for PAL minimization on personal computers with small memory. All papers, including this one, will be illustrated in lectures with many practical examples.

Availability of the described above software tools, together with inexpensive PAL and PLD programming devices (widely advertized in journals like Byte) permits small and medium businnesses, as well as the individual hobbyists, to make his/her own "poor man's LSI CAD/CAM design shop".

II. INITIAL SPECIFICATION OF FINITE-STATE MACHINES.

Finite state machines (FSMs), as we all know them from basic Digital Design courses are represented on inputs to the CAD systems by **state graphs** or by **state tables.** Most of the current systems use either tabular descriptions or textual descriptions corresponding to the FSM tables. Sometimes tables are specified as **sets of transitions**:

(present state, input state, next state, output state),

where present state and next state are the internal states of the machine, and the input and output states are values of combinations of input and output signals, respectively. The internal states are either names or numbers or in the case of encoded (assigned) machines, they are strings of ones and zeros. The Number of elements in the string correspond to the number of flip-flops in a machine.

State tables are very useful for many types of machines, however they have two disadvantages: they cannot be used for large machines and also are not easily modifiable - adding new input signals or transitions often requires rewriting the whole table.

Three approaches are then used in the systems:

- user decomposition of the machine,
- higher level description,
- graphics high-level schematic capture.

In the **first approach,** which is mainly caused by systems' and technology's inability to deal with oversized machines, the user is responsible for specifying the machine not as a single entity, but as a collection of mutually communicating smaller machines, each of them with smaller numbers of internal states, input and output signals. Unfortunately, little guidance is given as how to decompose the machine. Some ancient theoretical papers are well known [116, 188], see also their listing in [114], but as far as we are aware of, no working implementations of these methods are available as usable computer programs. Recently, a new approach to decomposition of machines is advocated by both university and industrial researchers that can possibly lead to better software tools [32]. It is based not on classical partition-based FSM decomposition theory, developed first by Harrison and Hartmanis/Stearns, but on graph-partitioning methods applied to FSM graph.

The **second approach,** high level description of FSM, is currently used mostly in academical systems and in some systems from large companies. Behavioral register-transfer (rt) languages are the most commonly used medium at this point [159]. The methods to convert rt-description to FSM state tables are well known [110, 28, 164]. This description makes it also easy to integrate the CAD system around it: with tools like rt-simulators, that simulate the circuit on behavioral level; flow-graph optimizers, that optimize speed and cost of both data-path and control unit parts of the circuit's realization; data-path allocators, that help to find optimum structures/floor plans of data paths, and other tools to design both the control unit and the data path parts of the system under design [135, 138, 139].

Recently, there has also been an interest in other forms of high-level description: regular expressions and Petri Nets. Regular expressions are expressions that describe a class of regular languages. They compose expressions E1 and E2 with use of operations of *sum of expressions*: E1 or E2, *concatenation:* E1 next E2 and *iteration*: E1 repeated arbitrary number of times. The systems to design FSMs from regular expressions have been designed in some universities, including: Stanford [75], Carnegie-Mellon and Columbia [76], McGill [78], and Portland State [144]. The Stanford system uses a special language, which is a combination of state machines and regular expressions. The first version used certain rules to directly map expressions to layout; the improved, second version uses a more classical approach where the expression is first converted to FSM, encoded and realized with Boolean functions [75]. The system from PSU uses the Brzozowski's method of derivatives of regular expressions [30], to convert a vector of regular languages to a Mealy or Moore machine. The program is written in the Artificial Intelligence language Prolog [33]. The regular expressions in this system can be of extended type, which means that operations of product of expressions, difference and negation of expressions are available, as well as those of sum, iteration and concatenation of expressions used in most of other systems [144].

The systems to design from Petri Nets are implemented in Universities of Paderborn and Dortmund in Germany [28]. Petri Nets are a form of graphs with parallelism and concurrency of execution of separate paths. They are commonly used to describe operating systems, controllers, and real time systems. Applications of Petri Nets in hardware descriptions on many levels has recently been gaining momentum.

Another interesting form of initial description includes predicate calculus clauses [102] (similiar to Prolog programs) and recursion equations [101] (similiar to Lisp programs).

The **third method** of FSM specification - graphical interface, seems to be very promising for future systems and can be used in conjunction with all other specification methods discussed above. Graphical schematic capture and window/menu systems are used for capture of logic schemata, block schemata and real-time system diagrams for software design. In two existing systems: one from Daisy and one from Intel, schematic capture is used to capture FSM graphs. However, a very similiar method can be applied also to capture flow-diagrams, Petri Nets, regular expressions or non-deterministic automata. We are not aware of any actual implementations, but supposedly they are coming. The graphical interfaces for these data should have all standard capabilities, like panning, zooming, pull-down menus, separate windows for graphics and text that can be selected from higher level pictorial descriptions. They should permit the introduction of hierarchially organized obiect-oriented data like the hierarchical, parallel automata of Harel [Hare 84].

Numerous papers and algorithms are devoted to the topic of converting high-level FSM descriptions to either FSM tables, or to lower level descriptions like logic networks. Some methods of direct conversion of flow-graphs, regular expressions or Petri Nets, even to the level of schematic geometrical layout have been also created.

Some of the conversions are NP-complete combinatorial problems [81], which make them difficult to apply for larger machines, and the decomposition is again a must. It also seems that not enough research was done on finding efficient transformation methods, which would make more use of algorithmic and heuristic methods developed recently both in other areas of VLSI CAD and in Artificial Intelligence and Mathematical Programming. Introduction of more powerful "personal supercomputers", like iPSC from Intel, should soon dramatically improve the prospects, since many of the conversion algorithms are very well amenable for parallelization.

III. STAGES OF FINITE STATE MACHINE DESIGN.

When the machine is already available as a table or a set of 4-tuples (3-tuples for Moore machines), the systems apply one of two approaches, a simple one or the advanced one.

In most of the current systems this conversion is done in a straightforward - primitive method. Widely available and used is the program Peg from UC Berkeley, which accepts FSM description in simple hardware desrcuption language on input and produces logic equations on output [89]. This program is integrated into UC Berkeley VLSI tools so finally PLA artwork can be generated from it. State symbols are replaced with codes of states and the state table is rewritten to the truth table format for D type flip-flops. The assignment of codes to states is done either according to the order of their specification, which is practically random, or the user can declare his/her own codes.

In more advanced systems not yet available commercially, several optimization and analysis operations are done on the state table first to the truth table conversion. We will first concentrate on synchronous machines.

The optimization operations include some (or usually one) of the following:

- minimization of number of internal states,
- state splitting for better assignment [114, 92, 93, 91, 94],
- minimization of number of input and output states,
- assignment of internal states,
- assignment of input, output and internal states,
- conversion from Mealy machine form to Moore machine form and vice versa,

The analysis includes simulation of the machine and testability analysis, as well as analysis of hazards in output functions of machines' realization.

In some systems the state table is enhanced with additional columns and/or rows to increase its testability with methods like LSSD or other [80, 1].

Many other FSM analysis methods have been theoretically investigated [114], and some of them were implemented in the academic environment, but industrial systems do not yet incorporate these methods.

Finally, the analysed and optimized state table is converted to a truth table. This truth table can be created for various types of flip-flops and has the primary input signals and outputs from flip-flops as inputs, and primary outputs and inputs to flip-flops as outputs. In most of current systems only the type D flip-flops are assumed, in some others the user has a choice of flip-flop types (RS, JK, T, D, RST) realized in an array. Selection of an appropriate flip-flop type can essentially decrease the area of logic realization for some machines, (like counters). In yet another system each flip-flop can be of a different type to further minimize silicon area for both logic and flip-flop components. Some interesting work on new types of flip-flops for VLSI implementation have been also published that will possibly even further reduce area for Boolean logic implementation, increasing slightly the area for flip-flops.

In the next stage the truth table is minimized and realized as a logic network. In most of the current systems a two-level realization of logic is assumed, but the recently realized systems also investigate the multi-level designs for many various technologies. Developments in this area will have also an essential influence on FSM optimization. For instance, the popular state assignment program Kiss [55] gives worse results when the assigned machine is realized with many levels of logic, than when it is realized in a two-level PLA. Existence of fast multi-level circuit minimizers will have the same positive effect on next generation assignment programs as PLA minimizer Espresso had on a creation of Kiss.

Below, we will discuss state assignment, minimization of number of states and other approaches to design FSMs, as well as design of asynchronous machines. State assignment will be discussed with most details, since it seems that this is an issue of most immediate practical applications.

FSM design systems are described in [2, 174, 15, 48, 27, 73, 36, 100, 123, 138, 139].

IV. STATE ASSIGNMENT FOR FINITE STATE MACHINES.

A. *Exsisting approaches to state assignment.*

A State Assignment Problem is to assign codes to the internal states of FSM in order to mimimize some cost function related to the circuit's realization (like PLA area or number of gates).

Most of the research papers discuss only the problem of assigning internal states, assuming that the user specifies codes for input and output states. However, there are efforts based on very similiar principles which assign input states and output states (specified initially by names) with binary codes. They have found, for instance, application in minimization of microprogrammed control units [57] or in designing control units with PLAs [103].

The problem is a classic in Switching Circuits Theory, but relatively few programs have been widely available until recent years. In the early sixties the basic main approaches to state assignment and structural theory of FSMs were formulated. Few of them were programmed and the programming results were rather unsatisfactory. Recently the state assignment problem is again gaining momentum because of widespread attempts to realize a logic level silicon compiler for VLSI [52] - [61], [121, 122], [138] - [143].

We will not characterize and evaluate in more detail the existing approaches here. The reader can find respective discussion in [52, 55, 59, 122]. However, some comparison of the known approaches will serve to present the main theoretical and practical issues that must be solved to make the respective software tools useful.

There are basically seven approaches to state assignment:

1. **Partition theory of Hartmanis, Stearns, Kohavi** [92, 93, 91, 168, 106, 113]. The theory is based on a concept of the partition of states of machine into blocks. Partitions are found from state tables and used to find sets of partitions producing unique and optimal encodings. This theory is very elegant from mathematical point of view, gives an insight into the nature of structural properties of machines, as well it is very general (it permits also for decomposition of FSMs, state minimization, realization with shift-registers,

etc.). However, the published results of programs that use this theory, as well as our own experience with this approach prove that in general, this approach is intractable for computer solutions of FSMs with more than 10 states, 10 inputs, and 10 outputs.

2. **Column evaluation approach of Dolotta and Mc Cluskey** [69], extended by Curtis [40, 41], Weiner [182], Vavilov [179], Torng [176]. The columns of the state table are scored with respect to many various criteria influencing quality of assignment. The scores are used to find good partitions and next the assignment. This approach can produce very good realizations (also for various types of flip-flops); the results are even better than those from approach 1, but is also intractable for machines of more than 12 states.

3. **Enumerative approach of Story** [170]. All posible partitions are evaluated as candidates for assignment separately by calculating the complexities of realizations of the corresponding Boolean functions, and assuming that all other partitions have been optimally selected for them. The subset of partitions with the best scores, also mutually matching, is selected for assignment. This approach gives better results than the approach from 2, but it is even slower.

4. **Branch and bound approach of Perkowski, Lee and Zasowska** [138, 187, 121, 122, 141]. This approach has two variants. The first permits realization of machines with 8 input, 8 internal and 8 output states and gives optimum realizations for arbitrary technology and flip-flops. However, in this variant nearly all possible partitions have to be evaluated. This is perhaps the program that generates most optimum solutions of all the published programs, but is very slow. The approximate method based on this method [122] does not evaluate all partitions, but only heuristically selected best partitions. It permits for realization of machines with 12 states, but can be extended to 18 - 20 states. Both approaches permit for state assignment combined with state minimization.

5. **Quadratic assignment approaches of Armstrong, De Micheli and Perkowski** [6, 7, 52, 144]. All these approaches are based on embedding some graphs created from FSM table to hypercube graphs. This approach permits for realization of machines to 100 states but according to evaluations from [6], it gave solutions too far from optimum. Some theoretical improvements were made in [7], but not programmed. De Micheli implemented two algorithms for state assignment, while at UC Berkeley. The first of them was based on

the quadratic assignment method - state assignment is reduced to the graph embedding problem. The second algorithm was based on other principles since he was not satisfied with quality of results. The results from [144], where both creation of the graph for embedding and the embedding algorithm were done on new principles, seem to be very satisfactory (machines with 136 states have been assigned), but no detailed comparison with other approaches is yet available.

6. **Approach of Moroz** [128]. This is a very fast constructive embedding algorithm that is widely used in design automation systems in Soviet Union. The author has implemented this algorithm and observed that it can find assignments for machines with more than 100 states, but that the quality of solutions for small machines was far from optimum. This is perhaps the fastest program currently available. The speed results from the fact that it does not solve the quadratic assignment but simple edges embedding problem, and also the graph for embedding is created directly from the state graph.

7. **Approach of De Michelli, Brayton and Sangiovanni-Vincentelli** [52] - cite[DeMi 85b], [23, 151] (KISS program). This approach is based on minimization of multiple-valuated Boolean functions to find state groups and then constructive assignment by embedding of these groups to the faces of a hypercube. Its strategy is very innovative and the computer results are very good. It is perhaps the best product currently available from the point of quality/time ratio. It was applied to machines with up to 100 states, but the largest published result is for 27 states. The groups of states are found with the use of multi-valued Boolean minimization program Espresso-mv [23, 150, 151] applied to FSM before state assignment. This method of finding groups of states (blocks), combined with fast Boolean minimization was a source of program's success. The Kiss program is now widely available in universities and is used by many companies to build commercial software. A program extending this approach was built in Intel [36].

B. Problems with existing approaches and requirements for a practical state assignment system.

We would like to have a program as fast as that of approach 6, as good as in 4 and as useful in VLSI design as approach 7. However, this is impossible. The user then has to implement a method selected among the above for his class of machines and systems's speed and performance requirements, or implement many algorithms and select among them interactively for any particular problem [143].

Let us consider in more detail what are the advantages and disadvantages of the above methods.

The method of De Micheli fails when there are few or no groups of internal states that transit to the same state under some input state. Unfortunately, such type of machines often occur in industrial applications [36]. The new program, developed by De Micheli at IBM gives results about 20constraint assignment problems [61].

The idea of applying Boolean minimization to the non-assigned machine is not new. It was used in the systems of Story, Harrison and Reinhard [170] and of Perkowski and Zasowska at Warsaw Technical University [187, 138], but because of the lack of fast Boolean minimizers, like Espresso-mv, this approach was applicable only to machines smaller than 9 internal states. However, the also method also a secondary assignment, minimizes numbers of gates and can be used for multi-level logic realization. The enumerative approach of the method allows finding an absolutely minimum solution to the problem for various technologies, since the cost function for realization is user-defined. Approximate variant for larger machines was also created. Availability of fast minimizers permits for improvement of methodologies from [121, 122]. The new methods and algorithms will be also presented in the forthcoming papers. The methods give better results than Kiss, but still cannot be applied to machines of more than 20 states.

The disadvantages of the original approach of De Micheli et al [55], are the following:

- they do not take into account output assignment and various structures,
- they do not take into account modern methods of solving the quadratic assignment problem.
- they do not minimize secondary assignment.

Attempts to improve Kiss were successful [36, 61]. However, program still cannot be applied for machines with more than 100 states [60]. Also, it does not take into account output states for assignment.

The disadvantages of the approach of Armstrong are the following:

- the method of reduction to quadratic assignment was doubtful,
- he didn't take into account the possibilities of fast logic minimizers (they didn't exist at that time),
- he didn't take into account possibilities of modern approximate approaches to quadratic assignment (they didn't exist either).

- he didn't took into account the assignment of outputs.

The disadvantages of Moroz approach are the following:

- the method of creating the assignment graph can be essentially improved,
- the method of solving the quadratic assignment problem can be used instead of his embedding, which should produce results of better quality,
- he didn't take into account assignment of outputs.

Different papers applying the embedding methods use different approaches to create the assignment graph.

Saucier [157] creates a nonoriented weighted graph, whose edges correspond to transitions between states of FSM.

Moroz [128] creates an oriented graph, whose edges correspond to oriented transitions between states. He writes about embedding, but his work can be treated as approximate solution to quadratic assignment of a particular type, where the graph is oriented, costs the of edges are equal, and the cost function is defined as in the quadratic assignment.

Armstrong [6, 7] formulates a nonoriented graph, whose edges are created according to several principles of adjacency.

The above authors use different constructive algorithms for embedding those graphs on hypercubes. They do not give any, other than heuristic, explanations of adequacy of the proposed assignment (embedding) techniques. Also, program of Saucier is designed for asynchronous machines.

Perkowski uses a combination of factors that take into account adjacency of states, inputs, and outputs, both from the view of primary adjacency (like De Micheli), and also secondary adjacency with use of methods similiar to partition theory and tabular methods.

Although many interesting approaches to state assignment have been recently proposed, the careful comparison of algorithms is necessary on large benchmarks of industrial machines. Many new interesting algorithmic concepts arise from the papers and evaluation results recently available. Large sets of industrial FSMs compiled by us anda other authors allows comparisons of methods. Perhaps in future systems, various algorithms will be used for small (less than 8 states), medium (9-40), and large (40-120) and very large (more than 120) machines to obtain good speed/performance ratios. Detailed analysis of reduction methods is necessary, as well as evaluation of the complexity of optimal and approximate algorithms for related mathematical problems, like hypercube embedding, embedding to faces and quadratic assignment [36, 144]. The possible use of general purpose parallel computers as well as research on special hardware accelerators seem to be also very promising.

V. MINIMIZATION OF NUMBER OF STATES OF FINITE STATE MACHINE.

A. *Do we really need state minimization in CAD systems?*

Minimization of number of states of FSMs is a classical topic of most undergraduate textbooks [114, 127, 125, 186] on logic design and sequential machines. It consists of merging groups of compatible states of a machine into single states of the new machine, in order to achieve an equivalent machine with reduced (and possibly minimal) number of states. It is assumed, that a machine with less states yields better realization. Compatible are the states that exhibit identical input/output behavior. Problem is difficult in the general case since the groups of compatible states are not disjoint and compatibility of some group of states can imply compatibilities of other groups of states. A number of mathematically interesting papers have been published [185, 134, 145, 184, 71, 3, 14, 16, 63, 64, 165, 152, 153, 107, 169, 84, 85, 86, 124]. However, it seems that no system is used in American industry, which would apply this advantageous option. This is to the contrary of practise in Europe and Soviet Union, where the present author has had an opportunity to observe several systems of this type in industrial applications. In USA responsible managers do not see a need for this kind of systems. Why is it so? In my opinion there are two reasons for it. First of them is that in the States the high level (architectural) design is often separated from logic minimization/layout design and is done by separate groups of engineers. When the "logic minimization people" receive the specification of the circuit from the "architecture people" they do not see many possibilities for improvement, for instance those that can result from introducing don't cares or minimizing the machine. The machines are small, because the initial decomposition is done intuitively by architecture designer. The other reason is that there never was a large market for industrial control systems realized as FSMs, since the programmable controllers and microprocessors took it early. In Eastern Europe for instance, these types of sequential circuits, which are usually very large, were realized as finite state machines with PLAs, so a need for software for state minimization and state assignment for machines of hundreds states existed.

It is expected that a need for state minimization will be more evident when the new high level tools for FSM description (like for regular expressions, Petri

Nets , parallel program schemata) will be incorporated into CAD systems and conversions to state tables will be done automatically. For instance, minimization of a completely specified machine is a part of procedures for: conversion of regular expression to FSM state table, convertion of parallel automaton, nondeterministic automaton or Petri Net to state table, equivalence verification of two FSMs [172] and many others. Various new algorithms lead to state tables that are strongly incomplete (have many don't cares) and therefore can be substantially minimized (for instance, when the invariants of the flow-graph are used to create don't cares in the state table).

B. The current approaches to state minimization.

The programs for state minimization can be divided into two categories: for completely specified machines and for incompletely specified machines. The first problem is relatively easy. The compatibility relation on states of the machine [114] is in such a case an equivalence relation and therefore efficient $O(n \cdot log(n))$ algorithms have been found [97, 114]. The problem to minimize an incompletely specifed machine is much more difficult, and the solution is not unique to an isomorphism (as it is for the completely specified machines). Very little information on programmed state minimization algorithms can be found in literature available in the U.S. All papers attempt to find a minimum solution and use a variant of covering/closure algorithm for groups of compatible states (so called **compatibles**). The point is to select such a subset of compatibles, that is full and closed, which means that all state numbers of the initial machine are included in some groups, and that each group implied by some selected group is also selected. Improved variants of the classical algorithms were created but not programmed. The only published references to fast approximate programs for large machines are [11, 12] and [141]. Both these algorithms are based on the branch-and-bound principle to find a closed and complete subset of groups of compatibles The program described in [141] minimizes 7-8 state machines (20-30 compatibles) in 6-10 second of Cyber. Generation of solutions for 10-state machines required 60-100 seconds. These programs cannot yet be used for interactive design of FSMs with more than 30 - 40 states on Vax-class machines.

A lot of basic research and especially programming effort is necessary to design practical state minimization algorithms for machines with hundreds of states, inputs and outputs, that occur in current circuits, like for instance control units of microprocessors.

VI. OTHER APPROACHES TO FSM DESIGN AND OPTIMIZATION.

Because the traditional approach to FSM design either involves solving at least two difficult combinational problems: state minimization and state assignment, or produces possibly poor solutions, many attempts to implement other FSM design methodologies have been undertaken. They attempt to minimize total area/chip count and include:

- design of general-purpose FSMs based on shift-registers or other elementary machines instead of flip-flops [49],
- design with special flip-flops with multiplexed inputs,
- design with regular arrays of elementary tunable machines,
- decomposition of FSMs into structures of FSMs [116, 188],
- converting the form of machine (Mealy to Moore and Moore to Mealy) to select one of better performance,
- concurrent state minimization and state assignment [121, 122],
- special structures of FSMs, like different type of micro-programmed control units with many ROMs and multiplexers, or partitioned realizations of logic [166, 109],
- direct conversion of high level description like parallel FSMs to symbolic or geometric layout [99].

The above approaches are used selectively for various categories of machines: some of them can be used for all machines, some of them are reasonable for large machines only, some other can be applied only to small machines.

A. Concurrent state minimization and state assignment to improve area.

Below we will discuss one of the methods to improve a chip's area. Current approaches to structural synthesis of finite state machines are nonminimal. As described above, the currently used design approach is first to minimize the number of machine's internal states and follow it with the states' assignment.

The aim of these methods is generally to decrease the semiconductor area of the realization for some selected implementing technology. However, the methods apply very abstract cost approximations to achieve this. It would be more desirable to minimize some generalized technology-dependent cost functions.

Minimization of the number of internal states results from the adopted assumption: "the more internal states, the more complicated is the realization, hence more memory elements are needed, there are more excitation functions, and therefore their realization is more complicated". Practical examples bear evidence that the flow table with the minimum number of internal states is not necessarily the appropriate starting point to achieve the circuit realization of the minimum total complexity (computed for instance as the weighted sum of the number of flip-flops and the number of combinational gates). Please note, that many contemporary technologies (among others - MOS) do not require first of all minimization of the number of memory elements, as is assumed in the classical methods.

Practical examples show evidence that we should not seek a FSM with the minimum number of internal states, nor one with the excitation functions depending on the minimum number of variables Examples of FSMs can be easily found that have minimum realizations with the greater than minimum number of flip-flops (because of much simpler excitation functions). Also, the attempt to find a realization of the set of excitation functions with the minimum number of argument variables is often useless, because such realizations can have more gates or connections than other the realizations of these functions. Moreover, these assumptions do not take into account the realization of excitation functions for flip-flops different than D flip flops.

One of the possible approaches is to replace **two stages:**

- minimization of the number of the machine's internal states,
- state assignment of the machine's internal states,

with **one stage** of joint minimization and state assignment. In this joint process, the cost function, related to the realization of the excitation functions in a selected technology, is optimized. The optimum and approximate methods of this type are presented in [121, 122].

B. Methods based on decomposition and partitioning of FSMs.

Another group of methods is related to decomposing the machine into a group of machines. Various approaches have been investigated. The classical methods, based on partition theory, seek in general certain partitions that permit description of a machine as a parallel, cascade or parallel/cascade composition of other machines. Such procedure can be done recursively. Unfortunately, such types of decomposition are very laborious to find and their nontrivial cases occur rarely for industrial machines. Some useful variants of classical theories discuss decomposition with use of important blocks like shift registers and counters.

New methods of decomposition try to use general graph partitioning methods [Kern 70] to partition a state graph. Such methods are used in practical systems as a neccessity, but their behavior seems to be unsatisfactory. Yet another group of methods assumes certain structure of the realization of the machine, for instance a microprogrammed unit, in which some outputs from the FSM are on the address input of some multiplexer and control selection of input signals to this FSM (the multiplexer's output is one of inputs of a FSM). Separate PLAs are realized for encoding of input and encoding of output signals. The main PLA realizing excitation and output functions is partitioned into many PLAs. These approaches combined can yield many various decomposed structures with essentially different realization costs.

Another possible approach is to create methods, that are particularly suitable for some selected method of machine's realization. For instance, methods of FSM design especially suitable for PLA minimization are discussed in [79, 117, 132, 166, 133].

VII. ASYNCHRONOUS MACHINES.

All the previously discussed methods for synchronous machines can be modified for asynchronous machines, but the requirements for asynchronous design are more difficult, the circuit should not produce hazards, races and oscillations, that can occur mostly because of different delays in gates. This means that special design tools must be built for asynchronous machines.

State minimization methods for synchronous state machines can be used without modification for asynchronous machines, but more efficient methods can be implemented for asynchronous machines, which make use of the specific forms of tables in such machines.

In assignment of asynchronous machines the race-less condition must be additionally taken into account. This constraint limits the number of applicable partitions (codes) and sometimes permits finding an optimum design for small machines faster than for synchronous machines of the same size. However, for large asynchronous machines, the problem is very complex and good algorithms are not currently available. The most advanced algorithms I am aware of permit for assignment of machines of about hundred states, but no benchmark comparison were run and results were hard to evaluate [88].

When the excitation functions are realized for asynchronous machines, the hazard must be also avoided; static hazard in asynchronous circuits is very dangerous indeed because changes not only dynamic but also static behavior of the machine.

The methods of logic circuit design to avoid both static and dynamic hazard are well known and have been incorporated into some systems (see a paper by Loc Nguyen/Perkowski in this issue). However, they increase the realization by adding more gates and/or reducing number of logic levels, increasing the circuit's redundancy and therefore making it less testable.

The logic methods of races and hazard removal calculate for the worst case combination of gates' delays - some better results can be obtained when the delays of particular gates are known from layout analysis. This method is however technology dependent, and cannot be used in a technology-independent high-level system.

The algorithms for asynchronous machine design are given in [178, 157], and [140].

VIII. MINIMIZATION OF LOGIC NETWORKS.

A. *Why we need various logic minimization tools.*

The portion of a chip implementing a set of Boolean functions usually represents a major contribution to chip's area or system's chip count. Obviously, there are many circuits which realize the same Boolean function. Unfortunately, at present there is no general theory that provides designers (and design automation programs) with lower bounds for the total area of logic implementations in integrated systems. Therefore, the main task for computer optimization programs appear in choosing the circuit with the most convenient layout, minimizing the area but also fulfilling many other constraints like timing, power consumption, and use of standard cells. Various approaches are used for different technologies and design styles (standard cells, ROM, PLA, Weinberger layout, SLA, gate arrays, etc.).

There are many sources of non-optimality of initial specification of logic circuits. If the logic is created from FSM description, it was not minimized by the designer at all. If it is specified as a source description - it bears all non-optimalities of the way of how the designer thinks about his idea. The description of logic in integrated design automation systems is a result of hardware compilation from the high level front end system or the register-transfer language; this description is then usually non optimal and should thus be next optimized with technology independent and next technology dependent transformations based on Boolean algebra.

The logic minimization is then always a must, but different methods are applied, according to the design stage, design goals, and target technology. Therefore, two stages are incorporated in the logic optimization systems - technology independent, followed by the mapping from generic to target technology and technology-dependent optimization.

The logic circuit design methods include basically two categories: two-level and multi-level design. Two-level circuits are realized usually with Programmable Logic Arrays (PLA) or Programmable Array Logic (PAL).

B. Minimization of PLAs and two-level circuits.

PLAs are used in many design to realize the logic part of FSMs or blocks of combinational functions. If not minimized, PLAs can become extremely large, consume a lot of power and are slow. For PLA design the problems of Boolean minimization partitioning and folding are addressed. Logic minimization consists of minimization of numbers of terms and sometimes also number of literals in terms (for less power consumption, reliability and foldability). The Boolean minimization algorithms for PLAs are optimum or approximate. The first can be used for not more than 14 variables in general case and up to 20 variables for most industrial functions [42]. The commonly used approximate program, Espresso, [23] introduced a number of new theoretical and implementation ideas that are now being tuned and improved both in industry and in universities. The topological optimization includes partitioning [104], and folding (Hachtel) to decrease the area of unused space inside a PLA. Folding attempts to find such a permutation of rows and/or columns of AND and OR planes in the PLA, that as many as possible rows or columns can be combined (folded) into single rows or columns. Various folded architectures and folding algorithms have been implemented in UC Berkeley, IBM and other places. Topological partitioning methods are described in [52]. Recently the simulating annealing method [111] has been used by several authors as a general method of solving combinatorial problems in CAD, including all problems related to PLA optimization as well as many problems related to multilevel optimization. The important partial problems related to (mostly) two-level synthesis are that of complementation of a Boolean function [156], and recognizing and minimizing unate functions [9, 24] (unate function is a sum of products of non-negated variables).

Since operations on arrays of cubes representing Boolean functions are relatively slow and are repeatedly used in the design process, hardware accelerators for them have been recently proposed. Sasao had constructed a specialized accelerator to verify logic tautologies [156], Perkowski proposed a general purpose accelerator for solving combinatorial problems, based on reduction of problems to generic combinatorial problems like "graph coloring" or satisfiability and implemented in a parallel systolic architecture [141].

C. Multi-level logic minimization.

There are several possible design style alternatives for multilevel logic design:

- domino logic,
- Cmos static cells,
- standard cells,
- cascaded PLAs,
- Weinberger layouts,
- gate arrays,
- multilevel PLAs,
- special regular layouts (like for symmetric functions).

Currently, optimization programs are designed or are under design for most of these approaches. Structure of most of the systems is the following:

- front-end software converts the input high-level language design description (functional,behavioral,or structural) to internal representation of logic networks,
- logic optimizer minimizes network independently on technology, optimizing such criteria as number of gates, number of inputs, number of transistors,
- technology-dependent logical transformations are performed,
- topological optimizer (like folding of PLA, wire packing in Weinberger layout, etc.) is used to generate symbolic layout,

The systems that we are aware of include:

- LSS (Logic Synthesis System) of IBM [44, 45, 46, 47, 17, 19], for gate arays,
- Yorktown Silicon Compiler [20] - [25],
- Cascode Voltage Switches [74].
- MAMBO system [96],
- Domino Logic,

Three approaches currently exist to multilevel logic design:

- global optimization, (Yorktown Silicon Compiler, part of Angel [98] and FDS [72].
- local optimization (part of Angel, LSS, MAMBO).
- combination of the above.

Global optimization approaches include:

- decomposition of Boolean functions [5, 114, 147, 160, 161, 175]. An example of global optimization approach is given in [20] and was found efficient for single sided Cascade Voltage Switches. The approach of Brayton completely redesigns the network.

Algorithms used include:

1. extraction - common subexpressions are recognized and replaced with one variable,
2. collapsing - intermediate variables which do not offer savings are eliminated.
3. simplification - a two level Boolean expression is minimized by use of logic minimization algorithms,
4. decomposition - a function too large to be implemented in a target technology is factored and decomposed (FDS of Bell Labs).

The result of the first stages of compilation of high-level behavioral description program is the nonoptimized logical network. At this stage of design some transformations to optimize the network are possible, that hold true for arbitrary technology of network's realization. These transformations consist, generally speaking, in removal of some parts of the network which do not influence its behavior (no path goes from them to an output of the network) and in the reduction of the number of inputs, according to the rules of the Boolean algebra. The parts of the network not connected to outputs can be found which is an indirect effect of some transformations executed at the previous stages of the structural implementation. The fact that such parts of network occur can be either the designer's error or rather a result of high level initial specification.

Reductions of the number of gates' inputs are also possible, as well as the number of gates in some cases. They result from constant ones and zeros on gates' inputs; it is usually an effect of applying, on some previous design stages, of the standard blocks (like registers, adders, multiplexers, comparators), with certain fixed input data. For instance application of an adder to add a variable, (like a four-bit bus), and a constant, (a four bit binary sequence), gives possibility of simplifying the adder. The adder in the resultant network will not retain its universality. The transformations applied to minimize such networks are rule-based. Each of them is very simple, however their successive usage sometimes permits for essential simplification of a network [Wiec 84a].

Next transformations optimize the network by applying a rule-based approach and this is done also independently of technology. An excellent example of this kind of system is one of GE, described in [51, 82, 87, 34]. Another system of this type was designed in IBM and its description can be found in [44] - [47], [10]. The transformational approach to NAND network design is presented in [120]. Other rule-based systems for arbitrary negative gates are described in [180, 181] and [162].

Another approach, which usually starts from a set of Boolean equations, is based on factorization [65, 20]. The basic factoring rules (like ab+ac=a(b+c)) can be usually applied in many places of the set of expressions, and applying some factorization prohibits another one. The point is to select optimum factorizations for large sets of logic equations. Factorization is strongly related to decomposition. Both the disjoint and the disjoint decompositions have been studied. The function shall be presented as a composition of some other functions, possibly operating on disjoint sets of input variables. The classical papers on decomposition of Boolean functions are [37, 39, 62] and [105]. A branch and bound algorithm to design optimum multilevel NAND networks was described by Davidson [50]. Another approach to multilevel design is presented in [118]. Approaches to design multilevel multi-output functions are discussed in [171]. A modern approach to decomposition and factoring was introduced in papers of Brayton and respective software is currently being implemented in IBM. Some other special structures of functions' realization are : three level circuits with NANDs [136], trees, cascades, and arrays of tunable gates. Problems of mapping circuits to different technologies, selecting modules, and partitioning are discussed in [83, 119, 35, 108]. A dynamic programming approach to optimal mapping of logic network to a set of modules is discussed in [158].

Interesting approaches to several partial problems in logic design are included in [8, 29, 31] and [66]. It seems that they can be used for efficient implementation of some new methods in a comprehensive global/local logic design system with various representations of logic functions.

Since EXOR gates are used in most arithmetical and many coding/telecommunication operations, design with this type of gates is important. Circuits that use many exors usually produce PLAs with large areas, research in multilevel synthesis methods with EXORs and ANDs, XNORS and ORs, or any other functional system including EXORs is very important from practical point of view. Many papers have been devoted to this topic in the past [4, 13, 189, 129, 146], but results are not widely used in industry. New research emphasizes programming efficiency and possibility of using these methods for wider categories of circuits in comprehensive general purpose global/local approaches.

In some systems logic minimization is done in conjunction with layout minimization, best examples are perhaps [163] and [149].

ACKNOWLEDGMENTS

Discussions with Karl Lieberherr, Jeff Fox, Rick Rudell, Alan Coppola, Loc Bao Nguyen and Giovanni De Micheli were very instructive for me while preparing this paper. I would like also to thank Richard Rudell for providing me with Kiss, Espresso and FSM examples.

References

[1] **Abadir, M.S., Breuer. M.A.**: "A Knowledge-Based System for Designing Testable VLSI Chips", IEEE Design & Test, pp.56-68, August 1985.

[2] **Agrawal, P., Meyer, M.J.**: "Automation in the Design of Finite-State Machines", VLSI Design, Vol.5, pp. 74-84, Sept. 1984.

[3] **Albicki, A.**: "Computer Method for Minimization of Number of States of an Automaton", Ph.D. Thesis, Faculty of Electronics, Warsaw Technical University, Warsaw, 1973 (in Polish).

[4] **Almaini, A.E.A., Moosa, M.A., Aziz, N.M.**: "Computer Aided Design of Groups of Exclusive Logic Functions", Digital Processes, Vol.6, pp. 227-243, 1980.

[5] **Ashenhurst, R.L.**: "The Decomposition of Switching Functions", Proceedings. Int. Symp. on the Theory of Switching, pp.74-116, April 1957.

[6] **Armstrong, D.B.**: "A Programmed Algorithm for Assigning Internal Codes to Sequential Machines", IRE Transactions Electr. Comp., Vol. EC-11, pp. 466-472, August 1962.

[7] **Armstrong, D.B.**: "On the Efficient Assignment of Internal Codes to Sequential Machines", IRE Transactions on Electronic Computers, Vol. EC-11, pp. 611-622, October 1962.

[8] **Barthel, D.**: "A Remark Concerning Minimization of Expressions of Boolean Algebra in the Case of DON'T-CARE Conditions", Elektron. Informationsverarb. Kybern. , Germany, Vol. 19., No.7-8, pp.393-396.

[9] **Baugh, C.R.**: "Generation of Representative Functions of the NPN Equivalence Classes of Unate Boolean Functions", IEEE Trans. on Comp., Vol. C-21, No.12., pp. 1373-1379, December 1972.

[10] **Bendas, J.B.**: "Design Through Transformation", Proc. 20th Design Automation Conference, June 1983, pp. 253-256.

[11] **Bennets, R.G.**: "An Improved Method of Prime C-Class Derivation in the State Reduction of Sequential Networks", IEEE Trans. Computers, Vol. C-20, pp.229-231, 1971.

[12] **Bennets, R.G., Washington, J.L.,Lewin, D.W.**: "A Computer Algorithm for State-Table Reduction", Radio Electron. Engn. Vol.42, pp.513-520, 1972.

[13] **Besslich, P.W.**: "Efficient Computer Method for EXOR Logic Design", IEE Proc. , Vol. 130, Pt.E, No.6, November 1983.

[14] **Biswas, N.N.**: "Implication Trees in the Minimization of Incompletely Specified Sequential Machines", Int. J. Electron. (GB), No.2, pp.291-298, August 1983.

[15] **Blank, J., Fox, J., Blackman, T., Ciesielski, M., Markov, L.**: "The Silc Silicon Compiler", GTE Laboratories Profile , September 1984.

[16] **Bouchet, A.**: "An Algebraic Method for Minimizing the Number of States in an Incomplete Sequential Machine", IEEE Trans. Computers, Vol. C-17, pp. 795-798, 1968.

[17] **Brand, D.**: "Redundancy and Don't Cares in Logic Synthesis", IEEE Trans. Computers, Vol.C-32, No.10, pp.947-952, October 1983.

[18] **Breitbart, Y., Vairavan, R.:** "The Computational Complexity of a Class of Minimization Algorithms for Switching Functions", IEEE Trans. on Comp., Vol. C-20, No. 12, pp. 941-943, December 1979.

[19] **Brand, D., Griffin W.**: "Synthesis of Circuit Families with Open Book Set", Proceeding Second Int. Symp. on VLSI Tech. Syst. and Appl. Taipei, Taiwan, May 1985.

[20] **Brayton, R.K., McMullen, C.T.**: "The Decomposition and Factorization of Boolean Expressions", Proc. of 1982 ISCAS Symp., Rome, pp.49-54, May 1982.

[21] **Brayton, R.K., Hachtel, G.D., Hemachandra, L.A., Newton, A.R., Sangiovanni-Vincentelli, A.L.M.**: "A Comparison of Logic Minimization Strategies Using ESPRESSO: An APL Program Package for Partitioned Logic Minimization", Proc. of 1982 ISCAS Symp., Rome, pp.42-48, May 1982.

[22] **Brayton, R.K., Cohen, J.D., Hachtel, G.D., Trager, B.M., Yun, D.Y.Y.**: "Fast Recursive Boolean Function Manipulation", Proc. of 1982 ISCAS Symp., Rome, pp.58-62, May 1982.

[23] **Brayton, R.K., et al**: "Automated Implementation of Switching Functions as Dynamic CMOS Circuits", Proc. 1984 IEEE Cust. Int. Circ. Conf. Rochester, NY., May 1984.

[24] **Brayton, R.K., Hachtel, G.D., McMullen, C.T., Sangiovanni-Vincentelli, A.L.**: "Logic Minimization Algorithms for VLSI Synthesis", Kluwer Academic Publishers, 1984.

[25] **Brayton, R.K., McMullen, C.T.**: "Synthesis and Optimization of Multistage Logic", Proc. 1984 Int. Conf. on Comp. Des., pp.23-30, Rye, NY ,October 1984.

[26] **Breuer, M.A. (ed.).**: "Design Automation of Digital Systems". Vol. 1, Prentice Hall, Englewood Cliffs, New Jersey, 1972.

[27] **Brown, D.**: "A State Machine Synthesizer", Proc. 18th Des. Aut. Conf. Nashville, June 1981.

[28] **Bruck, R., Kleinjohann, B., Kathofer, T., Rammig, F.J.**: "Synthesis of Concurrent Modular Controllers from Algorithmic Descriptions, Proc. 23rd Design Automation Conf., pp. 285-292, June 29-July 2, Las Vegas 1986.

[29] **Bryant, R.E.**: "Symbolic Manipulation of Boolean Functions Using a Graphical Representation", Proc. 22nd Design Automation Conference, IEEE 1985, pp. 688-694.

[30] **Brzozowski,**: "Derivatives of Regular Expressions", J. Assoc. Computing Machinery, Vol. 11, pp. 481-494, 1964.

[31] **Cerny, E., Marin, M.A.**: "A Computer Algorithm for the Synthesis of Memoryless Logic Circuits", IEEE Trans. on Comp. , Vol. ??, pp. 455-465, May 1974.

[32] **Cioffi, g., Constantini, E., DeJiulio, S.**: "A New Approach to the Decomposition of Sequential Systems", Digital Processes, Vol. 3, pp. 35-48, 1977.

[33] **Clocksin, W.F., Mellish, C.S.**: "Programming in Prolog". Springer-Verlag, Berlin, Heidelberg, New York. 1981.

[34] **Cohen, W., Barlett, K., deGeus, A.J.**: "Impact of Metarules in a Rule Based Expert System for Gate Level Optimization", Proc. Intern. Symp. on Circuits and Systems, May 1985.

[35] **Cohoon, J., Sahni, S.**: "Heuristics For The Circuit Realization Problem", Proc. 20th Design Automation Conference, IEEE 1983, pp. 560-566.

[36] **Coppola, A.J.**: "An Implementation of a State Assignment Heuristic", Proc. of the 23-rd Design Automation Conference, pp. 643-649, June 29 - July 2, 1986.

[37] **Crist, S.C.**: "Synthesis of Combinational Logic Using Decomposition and Probability", IEEE Trans. on Comp., Vol. C-29, No. 11, pp. 1013-1016, November 1980.

[38] **Curtis, H.A.**: "Design of Switching Circuits", Van Nostrand, Princeton N.J., 1962.

[39] **Curtis, H.A.**: "Generalized Tree Circuit - The Basic Building Block of an Extended Decomposition Theory", JACM, 1963.

[40] **Curtis, H.A.**: "Systematic Procedures for Realizing Synchronous Sequential Machines Using Flip-Flop Memory: Part 1", IEEE Trans. on Comp. Vol. C-18, pp. 1121-1127, December 1969.

[41] **Curtis, H.A.**: "Systematic Procedures for Realizing Synchronous Sequential Machines Using Flip-Flop Memory: Part 2", IEEE Trans. on Comp. Vol. C-19, pp. 66-73, January 1970.

[42] **Dagenais, M.R., Agarwal, V.K., Rumin N.C.**: "The McBoole Logic Minimizer", Proc. 22nd Design Automation Conference, IEEE 1985, pp. 667-673.

[43] **Daisy Corp.**: "Personal Logician", 700 Middlefield Road, P.O. Box 7006, Mountain View, CA 94039-7006.

[44] **Darringer, J., Joyner, W.H., Jr.**: "A New Look at Logic Synthesis", Proceedings of 17th DAC, Minneapolis, pp.543-549 , 1980.

[45] **Darringer, J., Joyner, J.W., Berman, C., Trevillyan, L.**: "Experiments in Logic Synthesis", Proc. IEEE Intern. Conf. on Circuits and Computers ICCC 80, pp.234-7A, 1980.

[46] **Darringer, J.A., Joyner, J.W., Berman, C., Trevillyan, L.** : "Logic Syntesis Through Local Transformations", IBM J. of Res. and Devel. Vol. 25, no.4., July 1981.

[47] **Darringer, J.A., et al.**: "LSS: A System for Production Logic Synthesis", IBM J. of Res. and Dev. vol 128, no.5, pp. 537-545, Sept. 1984.

[48] **DATA I/O** "Product Guide, U.S. Edition, January 1985, DATA I/O, 10525 Willows Road N.E., P.O. Box 97046, Redmond, WA 98073-9746

[49] **Davis, W.A.**: "Single Shift-Register Realization for Sequential Machines", IEEE Trans. on Comp., Vol 1-17, No. 5, pp. 421-431, May 1968.

[50] **Davidson, E.S.**: "An Algorithm for NAND Decomposition Under Network Constraints", IEEE Trans. on Comp., Vol. C-18, pp. 1098-1109, 1969.

[51] **deGeus, A.J., Cohen, W.**: "A Rule-Based System for Optimizing Combinational Logic", IEEE Design and Test, August 1985, pp.22-32.

[52] **De Michelli, G., Sangiovanni-Vincentelli, A., Villa, T.**: "Computer-Aided Synthesis of PLA-Based Finite State Machines", Proc. IEEE 1983 Intern. Conf. on Computer Aided Design, pp. 154-156, September 1983.

[53] **De Micheli, G.**: "Computer-Aided Synthesis of PLA-based Systems" Ph.D Dissertation, University of California, Berkeley 1983.

[54] **De Micheli, G., Santomauro, M.**: "Topological Partitioning of Programmable Logic Arrays", Proc. of IEEE Intern, Conf. on Computer-Aided Design, pp. 182-183, Santa Clara , California 1983.

[55] **De Micheli, G., Brayton, R., Sangiovanni-Vincentelli, A.L.**: "Optimal State Assignment for Finite State Machines" IBM Research Report RC 10599.

[56] **De Micheli, G., Hoffman, M., Newton, A.R., Sangiovanni-Vincentelli, A.L.**: "A Design System for PLA-based Digital Circuits", in Advances in Computer Engineering Design, Jai Press, 1984

[57] **De Micheli, G.**: "Optimal Encoding of Control Logic" Int. Conf. on Circ. and Comp. Des. Rye NY, Sept. 1984.

[58] **De Micheli, G. ,Brayton, R., Sangiovanni-Vincentelli, A.L.**: "KISS: A Program for Optimal State Asignment of Finite State Machines" Int. Conf. on Comp. Aid. Design., Santa Clara, November 1984.

[59] **De Micheli, G., Brayton, R., Sangiovanni-Vincentelli, A.L.**: "Optimal State Assignment for Finite State Machines". IEEE Transactions on Computer-Aided Design, Vol. CAD-4, No.3, pp. 269-285, July 1985.

[60] **De Micheli, G.,** Private communication.

[61] **De Micheli, G.**: "Symbolic Minimization of Logic Functions", Proc. ICCAD 85, Santa Clara, California, pp.293-295, Nov.18-21, 1985.

[62] **Deschamps, J.P.**: "Binary Simple Decomposition of Discrete Functions", Digital Processes, Vol. 1, pp. 123-140, 1975.

[63] **de Sarkar, S.C., Basu, A.K., Choudhury, A.K.**: "Simplification of Incompletely Specified Flow Tables With the Help of Prime Closed Sets", IEEE Trans. Computers, Vol. C-18, pp.953-955, 19969.

[64] **de Sarkar, S.C., Basu, A.K., Choudhury, A.K.**: "On the Determination of Irredundant Prime Closed Sets", IEEE Trans. Computers, Vol. C-20, pp.933-938, 1971.

[65] **Dietmeyer, D.L., Schneider, P.R.**: "A Computer-Oriented Factoring Algorithm for NOR Logic Design", IEEE Trans. on Electr. Comp., Vol. EC-14, pp. 868-874, 1965.

[66] **Dietmeyer, D.L., Schneider, P.R.**: "Identification of Symmetry, Redundancy and Equivalence of Boolean Functions", IEEE Trans. on Electr. Comp. , Vol. EC-16, pp. 804-817, December 1967.

[67] **Dietmeyer, D.L., Su, Y.H.**: "Logic Design Automation of Fan-In Limited NAND Circuits", IEEE Trans. on Comp., Vol. C-18, pp. 11-22, 1969.

[68] **Dietmeyer, D.L.**: "Logic Design of Digital Systems". Boston, Mass: Allyn and Bacon,1971.

[69] **Dolotta, T.A., McCluskey, E.G.**: "The Coding of Internal States of Sequential Machines", IEEE Trans. Electr. Comp., Vol. EC-13, pp. 549-562, October 1964.

[70] **Doshi, M.S., Dietmeyer, D.C.**: "Automated PLA Synthesis of DDL Descriptions", Univ. of Wisconsin at Madison Report, ECE 78-17, 1978.

[71] **Dunworth, A., Hartog, H.V.**: "An Efficient State Minimization Algorithm for Some Special Classes of Incompletely Specified Sequential Machines", IEEE Trans. Comp., Vol. C-28, No. 7, p. 531-535, July 1979.

[72] **Dussault, J., Liaw, C.C., Tong M.**:" A High Level Synthesis Tool for MOS Chip Design". Proc. 21-nd Design Automation Conf., Albuquerque, 25-27 June 1984.

[73] **ENDOT, Inc.**: "Technical Data", ENDOT, Inc., 11001 Cedar Ave., Cleveland, Ohio, 44106.

[74] **Erdelyi, C.K., Griffin, W.R., Kilmoyer, R.D.**: "Cascode Voltage Switch Design", VLSI Design, pp. 78-86, October 1984.

[75] **Floyd, R.W., Ullman, J.D.**: "The Compilation of Regular Expressions into Integrated Circuits." JACM, Vol. 29., No.3, July 1982, pp. 603 - 622.

[76] **Foster, M.J.**: "Specialized Silicon Compilers for Language Recognition", PhD Thesis, Department of Computer Science, Carnagie-Mellon University, July 1984.

[77] **Friedman, A.D., Menon, P.R.**: "Theory and Design of Switching Circuits", Woodland Hills, California, Computer Science Press, Inc., 1975.

[78] **Friedman, N., Chandrasekhar, M.**: "Area Efficient PLA's for Recognizing Regular Languages", Proc. 5-th Annual Intern. Conf. on Computers and Communications, pp.680-685, March 26-28, 1986, Scottsdale, Arizona.

[79] **Fritsnovich, G.F.**: "Synthesis of a Microinstruction Decoder Using Programmable Logic Arrays", Avtomatika i Vychislitel'naya Tekhnika, Vol. 15, No.2, pp. 45-53, 1981.

[80] **Fujiwara, H.**: "Logic Testing and Design for Testability", The MIT Press, 1985.

[81] **Garey, M.R., Johnson, D.S.**: "Computers and Intractability. A Guide to the Theory of NP-Completeness", W.H. Freeeman and Company, San Francisco 1979.

[82] **Garrison, K., Gregory, D., Cohen, W., deGeus, A.**: "Automatic Area and Performance Optimization of Combinatorial Logic", ICCAD 1984, pp.212-214, 1984.

[83] **Gilkinson, J.L., Lewis, S.D., Winter, B.R., Hekmatpour, A.**: "Automated Technology Mapping", IBM J. Res. Develop. Vol. 28, no.5, September 1984.

[84] **Ginsburg, S.**: "A Synthesis Technique for Minimal State Sequential Machines", IRE Trans. Electron. Computers, Vol. EC-8, No.1, pp.13-24, March 1959.

[85] **Ginsburg, S.**: "On the Reduction of Superflous States in a Sequential Machine", J. Assoc. Computing Machinery, Vol.6., pp.259-282, April 1959.

[86] **Grasselli, A., Lucio, F.**: "A Method of Minimizing the Number of Internal States in Incompletely Specified Sequential Networks", IEEE Trans. Electr. Comp., Vol. EC-14, No. 3, pp. 330-359, June 1965.

[87] **Gregory, D., Bartlett, K., deGeus, A.J.**: "Automatic Generation of Combinatorial Logic from a Functional Specification", Proc. 1985 Int. Symp. on Circ. and Syst. Kyoto, Japan, June 1985.

[88] **Hallbauer, G.**: "Procedures of State Reduction and Assignment in One Step in Synthesis of Asynchronous Sequential Circuits", Proc. Intern. IFAC Symp. on Discrete Systems, pp. 272-282, Riga, September 30 - October 4, 1974.

[89] **Hamachi, G.**: "Peg Tutorial", VLSI Tools, Univ. of California, Berkeley, 1984.

[90] **Harel, D.**: "Statecharts: A Visual Approach to Complex systems", Report, Dept. of Applied Math, The Weizmann Institute of Science, Rehovot, Israel, Dec. 1984.

[91] **Harrison, M.**: "Introduction to Switching and Automata Theory", McGraw-Hill, New York, 1965.

[92] **Hartmanis, J.**: "On the State Assignment Problem for Sequential Machines, I", IRE Trans. Electr. Comp., Vol. EC-10, pp. 157-165, June 1961.

[93] **Hartmanis, J., Stearns, R.E.**: "Algebraic Structure Theory of Sequential Machines", Prentice-Hall, New York, 1966.

[94] **Hennie, F.C.**: "Finite-State Models for Logical Machines", John Wiley, New York, 1968.

[95] **Hill, F.J., Peterson, G.R.**: "Introduction to Switching Theory and Logical Design", 2nd Ed., New York, John Wiley & Sons, Inc., 1974.

[96] **Hoffman, M., Newton, R.**: "A Synthesis System for CMOS Domino Logic", Proc. IEEE Int. Symp. Circuits and Systems, pp. 986-989, May 1984.

[97] **Hopcroft, J.E.**: "An nlogn Algorithm for MInimizing States in a Finite Automaton", in Kohavi,Z., Paz, A. (eds), Theory of Machines and Computations, Academic Press, New York, pp. 189-196, 1971.

[98] **Hoshino, T., Endo, M., Karatsu, O.**: "An Automatic Logic Synthesizer for Integrated VLSI Design System", Proc. 1984 Cust. Int. Circ. Conf., pp. 356-360, Rochester, NY, May 1984.

[99] **Hurson, A.R.**: "A VLSI Design of the Parallel Finite State Automaton and its Performance Evaluation on a Hardware Scanner", Intl. J. of Comp. and Info. Science, Vol. 13, No. 5, pp. 481-505, 1984.

[100] **INTEL** Solutions, March/April 1986, p.14.

[101] **Johnson, S.D.**: "Synthesis of Digital Designs from Recursion Equations", The MIT Press, 1984.

[102] **Kabat, W.C. Wojcik, A.S.**: "Automated Synthesis of Combinational Logic Using Theorem Proving Techniques", Proc. of the 12th Intern. Symp. on Multiple-valued Logic, Paris, May 1982, pp. 178 - 199.

[103] **Kalnberzin, A.Ya., Chapenko, V.P.**: "Method of Input State Assignment for Digital Devices Implemented Using Programmable Logic Devices", Avtomatika i Vychislitel'naya Tekhnika, Vol. 17, No.1, pp. 41-47, 1983.

[104] **Kang, S.**: "Synthesis and Optimization of Programmable Logic Arrays", Technical Report No 216, July 1981. Computer Systems Laboratory, Department of Electrical Engineering, Stanford University, 1981.

[105] **Karp, R.M.**: "Functional Decomposition and Switching Circuit Design", J.SIAM, 1963.

[106] **Karp, R.M.**: "Some Techniques of State Assignment for Synchronous Sequential Machines", IEEE Trans. Electr. Comp., Vol. EC-13, No. 5, pp. 507-518, October 1964.

[107] **Kella, J.**: "State Minimization of Incompletely Specifed Sequential Machines", IEEE Trans. Computers, Vol. C-19, pp.342-348, 1970.

[108] **Kernighan, B.W., Lin , S.**: "An Efficient Heuristic Procedure for Partitioning Graphs", Bell Syst. Tech. J. Vol.??, Feb. 1970, pp.291-307.

[109] **Kidson, D.**: "A Consideration of the Use of Standard MSI Units in the Realization of Sequential Machines", Digital Processes, No. 4, pp. 85-107, 1978.

[110] **Kim, J.H., Siewiorek, D.P.**: "Issues in IC Implementation of High Level, Abstract Designs", Proceedings of 17th DAC, Minneapolis, pp.85-91, 1980.

[111] **Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.**: "Optimizaton by Simulated Annealing", Science, Vol. 220, no. 4598, pp.671-680, 13 May 1983.

[112] **Klir, G.J.**: "Introduction to the Methodology of Switching Circuits", D. Van Nostrand Co., New York , 1972.

[113] **Kohavi, Z.**: "Secondary State Assignment for Sequential Machines" IEEE Trans. on Elect. Comp. pp. 193-203, June 1964.

[114] **Kohavi, Z.**: "Switching and Finite Automata Theory", McGraw-Hill, New York, 1970.

[115] **Kowalski, T.J., Geiger, W.H., Wolf, W.H, Fichtner, W.**: "The VLSI Design Automation Assistant: From Algorithms to Silicon", IEEE Design & Test, pp. 33-43, August 1985.

[116] **Krohn, K., Rhodes, J.L.**: "Algebraic Structure Theory of Machines.I: the Decomposition Results", Trans. American Math Soc., CXVI 1965.

[117] **Lange ,E.E.**: "Lower Bound For The Number of Terms in The System of DNF That Describes The Logical Structure of An Automaton" Avtomatika i Vychislitel'naya Tekhnika, Vol. 15, No.4, pp. 27-32, 1981.

[118] **Lawler ,E.L.**: "An Approach to Multilevel Minimization", JACM, Vol. 11, No. 3, pp.283-295, July 1964.

[119] **Leive, G.W., Thomas, D.E.**: "A Technology Relative Logic Synthesis and Module Selection System", Proc. 18th Design Automation Conf., June 1981, pp. 479-485.

[120] **Lee, H.P., Davidson, E.S.**: "A Transform for NAND Network Design", IEEE Trans. on Comp. , Vol. C-21, N0.1, pp.12-20, January 1972.

[121] **Lee, E.B., Perkowski, M.**: "A New Approach to Structural Synthesis of Automata". University of Minnesota, Department of Electrical Engineering, report, 1982.

[122] **Lee, E.B., Perkowski, M.**: " Concurrent Minimization and State Assignment of Finite State Machines " . Proceedings of the 1984 Intern. Conf. on Systems, Man, and Cybernetics, IEEE, Halifax, Nova Scotia, Canada, October 9 - 12, 1984.

[123] **Lewin, D.**: "Computer-Aided Design of Digital Systems", Crane Russak, New York, 1977.

[124] **Lucio, F.**: "Extending the Definition of Prime Compatible Classes of States in Incomplete Sequential Machine Reduction", IEEE Trans. Computers, Vol. C-18, pp. 537-540, 1969.

[125] **Majorov, S.A.**: "Design of Digital Computers" Energia, Leningrad 1972 (in Russian).

[126] **Meyer, M.J., Agraval, P., Pfister, R.G.**: "A VLSI FSM Design System", Proc. 21sr Design Automation Conference, pp.434-440, Albuquerque, New Mexico, June 25-27, 1984.

[127] **Miller, R.E.**: "Switching Theory. Vol. 1 + 2 ", John Wiley, New York 1965.

[128] **Moroz, D.Z.**: "An Algorithm for Encoding the States of an Automaton", Avtomatika i Vychislitel'naya Tekhnika, Vol.4., No. 4, pp. 21-24, 1970.

[129] **Muller, D.E.**: "Application of Boolean Algebra to Switching Circuit Design and Error Detection", IRE Trans. 1954, pp. 6-12.

[130] **Muroga, S., Ibaraki, T.**: "Design of Optimal Switching Networks by Integer Programming", IEEE Trans. on Comp., Vol. C-21, pp. 573-582, June 1972.

[131] **Newton, A.R., Sangiovanni-Vincentelli, A.L.**: "Computer-Aided Design for VLSI Circuits", Computer, April 1986, pp. 38 - 60.

[132] **Page, E.W, Marinos, P.N.**: "Programmable Array Realizations of Synchronous Sequential Machines", IEEE Trans. on Computers, Vol.C-26, No.8, pp. 811-818. August 1977

[133] **Papachristou Ch. A., Sarma, D.**: "An Approach to Sequential Circuit Construction in LSI Programmable Arrays", IEEE Proceedings, Vol. 130, Pt. E, No. 5, pp. 159-164, September 1983.

[134] **Paull, M.C., Unger, S.H.**: "Minimizing the Number of States in Incompletely Specified Sequential Switching Functions," IEEE Trans. Electr. Comp. Vol. EC-8, 1959.

[135] **Parker, A.C., Thomas, D.E., Siewiorek, D.P., Barbacci, M., Hafer, L. Leive, G., Kim, J.**: "The CMU Design Automation System - An Example of Automated Data Path Design", Proc. 16th Design Automation Conf., June 1979, pp. 73-80.

[136] **Perkowski, M.**: "Synthesis of multioutput three level NAND networks". Proceedings of the Seminar on Computer Aided Design. Budapest, Hungary, 3-5 November 1976, pp. 238-265.

[137] **Perkowski, M.**: "The state-space approach to the design of multipurpose problem-solver for logic design". Proceedings of the IFIP WG.5.2 Working Conference "Artificial Intelligence and Pattern Recognition in Computer-Aided Design". Grenoble, France, 17-19 March 1978, J. C. Latombe (ed.), North Holland, Amsterdam, pp. 124-140, 1978.

[138] **Perkowski, M.**: "A System for Automatic Design of Digital Systems." Magyar Tudomanyos Akademia. Szamitastechnikai Es Automatizalesi Kutato Intezete. Budapest Tanulmanyok 99/1979 pp. 93-112. Hungary, 1979 (in Russian).

[139] **Perkowski, M.**: "Automatischer Entwurf von MOS-LSI-digitalen Schaltungen in System DIADES". Messen, Steuern, Regeln, No. 6, 1979, pp. 346-350, East Germany, (in German).

[140] **Perkowski, M., Zasowska. A.,**: "Minimal Area MOS Asynchronous Automata". Proceedings of the International Symposium on Applied Aspects of Automata Theory, Warna, Bulgaria, 14-19 May 1979, pp. 284-298.

[141] **Perkowski, M., Nguyen, N.**: "Minimization of Finite State Machines in SuperPeg". Proceedings of the Midwest Symposium on Circuits and Systems. Luisville, Kentucky, 22-24 August 1985.

[142] **Perkowski, M.**: "Systolic Architecture for the Logic Design Machine", Proc. Intern. Conference on Computer Aided Design, pp. 133-135, Santa Clara, November 1985.

[143] **Perkowski, M.**: "A New Approach to the Structural Design of Finite State Machines", Department of EE, PSU, Report, 1986.

[144] **Perkowski, M, Smith, D., Krzywiec, R.**: "Logic Simulation/Design/Verification Environment in Prolog", Proc. of 17th Annual Pittsburgh Conference on Modeling and Simulation, April 24-25, 1986.

[145] **Pfleger, C.F.**: "Complete Sets and Time and Space Bounded Complexity", Doctoral Dissertation, Computer Science Dept., Pensylvania State Univ., University Park, PA 1974.

[146] **Ramamoorthy, C.V.**: "Procedures for Minimization of "Exclusive-Or" and "Logical-Equivalence" Switching Circuits", Report.

[147] **Risch, R.H.**: "Staggered Input Networks: An Approach to Automatic Logic Decomposition", Proc. of 1982 ISCAS Symp., Rome, pp.55-57, May 1982.

[148] **Roth, J.P.**: "Computer Logic, Testing, and Verification", Computer Science Press, Potomac, MD., 1980.

[149] **Rowen, C.** : "Multi-Level Logic Array Synthesis", Technical Report No. 85-279, Computer Systems Lab., Stanford University, July 1985.

[150] **Rudell, R.**: "Finite State Machine Synthesis. State Assignment. Logic Minimization," Report, University of California, Berkeley, 1984.

[151] **Rudell, R.L., Sangiovanni-Vincentelli, A.L.**: "Espresso-MV : Algorithms for Multiple-Valued Logic Minimization", IEEE 1985 Custom Integrated Circuits Conference, pp. 230-234.

[152] **Russo, G.V., Palama, G.**: "Minimization of Incompletely Specified Finite State Machines", Digital Proceses, Vol. 6., pp. 199-206, 1980.

[153] **Russo, G.V., Palama, G., Neve, A.C.**: "Really Prime Classes Implying only Really Prime Classes," Electr. Letters, Vol. 19, No. 13, 23 June 1983.

[154] **Sangiovanni-Vincentelli, A.L.**: "An Overview of Synthesis Systems", Proc. IEEE 1985 Custom Integrated Circuits Conference, pp.221-225, May 1985.

[155] **Sasao, T.**: "An Algorithm to Derive the Complement of a Binary Function with Multiple-valued Input", IEEE Trans. on Comput., Vol. C-34, No.2, pp.131-140, Febr. 1985.

[156] **Sasao, R.**: "HART: A Hardware for Logic Minimization and Verification", IEEE Proc. of the Intern. Conf. on Computer Design: VLSI in Computer, October 7-10, 1985, Port Chester, NY.

[157] **Saucier, G.**: "State Assignment of Asynchronous Sequential Machines Using Graph Techniques", IEEE Trans. on Comp. Vol. C-21, pp. 282-288, March 1972.

[158] **Schmidt, D.C., Metze, G.**: "Modular Replacement of Combinational Switching Networks", IEEE Trans. on Comp., Vol. C-24, pp. 29-48, 1975.

[159] **Shahdad, M.**: "An Overview of VHDL Language and Technology", Proc. of the 23-rd Design Automation Conference, ACM and IEEE, Las Vegas, pp. 320-326, June 29- July 2, 1986.

[160] **Shen, V., Mc Kellar, A.**: "An Algorithm for the Disjunctive Decomposition of Switching Functions", IEEE Trans. on Comp. Vol. C-19, pp. 239-248, 1970.

[161] **Shen, V., Mc Kellar, A.**: "A Fast Algorithm for the Disjunctive Decomposition of Switching Functions", IEEE Trans. on Comp. Vol. C-20, pp. 304-309, 1971.

[162] **Shinsha, T., et al.**: "POLARIS: Polarity Propagation Algorithm For Combinational Logic Synthesis", 21st Design Automation Conference, IEEE 1984.

[163] **Shirakawa, I., Okuda, N., Harada, T., Tani, S., Ozaki, H.**: "A Layout System for Random Logic Portion of MOS LSI", Proceedings of 17th DAC, Minneapolis, pp.92-99 , 1980.

[164] **Shiva, S.G.**: "Combinational Logic Synthesis from HDL Description", Proceedings of 17th DAC, Minneapolis, pp. 550-555 , 1980.

[165] **Sinha Roy, P.K., Sheng, C.L.**: "A Decomposition Method of Determining Maximum Compatibles", IEEE Trans. Computers, Vol. C-21, pp.309-312, 1972.

[166] **Sklyarov, V.A.**: "Design of Automata Using Programmable Logic Arrays with Memory", Kibernetika, pp.840-848, Plenum Publishing Corp., 1985.

[167] **Southard, J.R., Domic, A., Crouch, K.W.**: "Report on the Lincoln Boolean Synthesizer", Digest of ICCAD , pp.192-193, IEEE, September 1983. Springer Verlag, Berlin-New York, 1981.

[168] **Stearns, R.E., Hartmanis, J.**: "On the State Assignment Problem for Sequential Machines, II", IRE Trans. Electr. Comp., Vol. EC-10, No. 4, pp. 593-603, December 1961.

[169] **Stentiford, F.W.H., Lewin, D.W.**: "Heuristic Procedure for Reduction of Finite-State Machines", IEE Electron. Lett., Vol.7, pp.700-702, 1971.

[170] **Story, J.R., Harrison, H.J., Reinhard, E.A.**: "Optimum State Assignment for Synchronous Sequential Circuits", IEEE Trans. on Comp., Vol. C-21, No. 12, pp. 1365-1373, December 1972.

[171] **Su, S.Y.H., Nam, C.W.**: "Computer Aided Synthesis of Multiple Output Multilevel NAND Networks With Fan-In and Fan-Out Constraints", IEEE Trans. on Comp., Vol. C-20, pp. 1445-1455, 1971.

[172] **Supovit, K., Friedman, S.J.**: "A New Method for Verifying Sequential Circuits", Proc. 23rd Design Automation Conf., pp. 200-207, June 29-July 2, Las Vegas 1986.

[173] **Svoboda, A.**: "Advanced Logical Circuit Design Techniques", Garland STMP Press, New York, 1979.

[174] **Teel, B., Wilde, D.**: "A Logic Minimizer for VLSI PLA Design", Proc 19-th Design Automation Conference, ACM-IEEE, June 1982.

[175] **Thayse, A.**: "A Fast Algorithm for Proper Decomposition of Boolean Functions", Philips Res. Rep. No. 27, pp. 140-150, 1972.

[176] **Torng, H.C.**: "Introduction to the Logical Design of Switching Systems," Addison-Wesley, Reading, Massachusetts, 1964.

[177] **Torng, H.C.**: "An Algorithm for Finding Secondary Assignments of Synchronous Sequential Circuits", IEEE Trans. on Comp. Vol. C-17, pp. 416-469, May 1968.

[178] **Tracey, J.H.**: "Internal State Assignment for Asynchronous Machines" IEEE Trans on Electr. Comp. Vol. EC-15, pp. 551-560, August 1966.

[179] **Vavilov, E.N., Portnoy, G.P.**: "Synthesis of Circuits of Electronic Computers," Energia Publishers, Moscov, 1966 (in Russian).

[180] **Wieclawski, A., Perkowski, M.**: " Optimization of Negative Gate Networks Realized in Weinberger-like layout in a Boolean Level Silicon Compiler". Proceedings of 21st Design Automation Conference, ACM and IEEE, Albuquerque, 25 - 27 June, 1984.

[181] **Wieclawski, A., M. Perkowski.**: "Optimization of Negative Gate Networks," Department of Electrical Engineering, Portland State University, Technical Report, version 2, May 1984.

[182] **Weiner, P., Smith, E.J.**: "Optimization of Reduced Dependencies for Synchronous Sequential Machines", IEEE Trans. on Electr. Comp., Vol. EC-16, pp. 835-847, December 1967.

[183] **Viewlogic Systems, Inc**: "WorkView", 33 Boston Post Road, West Marlboro MA 01752, 1986.

[184] **Yang, C.**: "Closure Partition Method for Minimizing Incomplete Sequential Machines", IEEE Trans. Computers, Vol. C-29, No.8, pp. 732-736, August 1980.

[185] **Yamamoto, M.**: "A Method for Minimizing Incompletely Specified Sequential Machines," IEEE Trans. Comp. Vol C-29, No. 8, pp. 732-736, August 1980.

[186] **Zakrevskij, A.A.**: "Algorithms of Discrete Automata Synthesis", Nauka, Moscow, 1971 (in Russian).

[187] **Zasowska, A., Perkowski, M.**: "The Computer-Oriented Method for Joint Minimization and State-Assignment of Synchronous and Asynchronous Automata", Proc. of the Conf., "Application of Computers in Engineering Design", Katowice, Poland, 1979.

[188] **Zeiger, H.P.**: "Cascade Decomposition of Automata using Covers", In the Algebraic Theory of Machines, Languages, and Semigroups, by M.A. Arlib, Academic Press, Netherlands, 1968.

[189] **Zhang, Y.Z., Rayner, P.J.W.**: "Minimisation of Reed-Muller Polynomials with Fixed Polarity", IEE Proceedings, Vol. 131, Pt.E, No. 5, September 1984.