THESIS ON INFORMATICS AND SYSTEM ENGINEERING C19

LOW POWER FINITE STATE MACHINE SYNTHESIS

ELENA FOMINA

TALLINN 2005

Faculty of Information Technology
Department of Computer Engineering
Chair of Computer Engineering
TALLINN UNIVERSITY OF TECHNOLOGY

Dissertation is accepted for the defence of the degree of Doctor of Philosophy in Computer Engineering.

The commencement of the thesis will take place on the 15 December 2005 in Tallinn University of Technology, Ehitajate Street 5, Tallinn, Estonia.

Supervisor:
Alexander Sudnitson,
Associated professor
Department of Computer Engineering, Tallinn University of Technology

Opponents:
Arkadi Zakrevski
Professor, Doctor of Science
Corresponding Member of NAS of Belarus,
United Institute of Informatics Problem, Minsk, Belarus

Vladimir Hahanov
Professor, Doctor of Science
Kharkov National University of Radioelectronics (KhTURE), Kharkov, Ukraine

Declaration

Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for any degree or examination.

ISSN
ISBN

To my parents

Abstract

Low power finite state machine synthesis

Low power synthesis has gained significant attention in recent years. The thesis is a result of the investigations into development of energy-efficient Finite State Machine (FSM) design. The current research focuses on a technique to reduce dynamic power dissipation of the FSM. FSM partitioning and state encoding are two virtually identical concepts that lie at the heart of the presented approach of synthesis of a low power FSM.

We target the reduction of the average switching activity for an FSM in the state variables by minimizing the number of bit changes during state transitions. The technique introduces a fundamentally new methodology of finding state encoding for an FSM described by the State Transition Graph (STG).A new state encoding technique for a low power FSM based on the concept of weakly crossed edge cuts is presented. The basic idea of this approach is that the code length is equal to the number of encoding partitions on the set of states and to the number of edge cuts on a set of state transitions in the STG. The set of edge cuts is constructed with aim to minimize the Boolean (Hamming) distance between the codes of the neighbor states (connected with a transition).

Using a probabilistic description of an FSM, we have adapted our strategy to propose a state encoding algorithm that minimizes the Hamming distance between the codes of the states with high transition probability. The framework we have developed is a general formulation of a state encoding problem that links a probabilistic description of an FSM to its power dissipation.

Minimizing the switching activity by modifying the state encoding of an FSM by itself does not always guarantee reduced total power dissipation, because the power consumed in the combinatorial part is not accounted for. The most popular technique to reduce power in the FSM is Dynamic Power Management (DPM). We specify the proposed state encoding strategy using technique of FSM partitioning which is one of DPM techniques. Decomposition has been shown as a very effective technique for synthesis of a low power FSM.

A new complex approach to a low power FSM design which consists of two phases: FSM decomposition and FSM state encoding is presented. Moreover, these phases are tightly connected, which gives the possibility of solving the task of construction a network of interacting and interconnected machines with given restrictions on the structure.

The most important result of the presented research is the complete framework developed for synthesis of a low power FSM. The key to success of this work is a combination of a novel heuristic approach and well-known approved techniques and methods.

Lühike ülevaade

Lõplike automaatide madala energiatarbega süntees

Käesoleva doktoritöö on pühendatud madala energiatarbega lõplike automaatide (edasises automaat; inglise keeles *Finite State Machine*) projekteerimise alastele arendusuuringutele. Uuringute keskseks põhiteemaks on automaatide dünaamiliste võimsuskadude vähendamise meetodite ja algoritmide väljatöötamine. Töös on esitatud madala energiatarbega automaatide sünteesi käsitlus, mis baseerub automaadi tükeldamisel ja tema olekute kodeerimisel, mis on käesolevas kontekstis kaks peaaegu kattuvat mõistet.

Doktoritöös esitatakse uus madala energiatarbega automaatide olekute kodeerimise meetod, mis baseerub automaadi siirdegraafi (*State Transition Graph*) tükeldamisel nõrgalt seotud lõigete alusel. Pakutud meetod toob sisse põhimõtteliselt uue olekute kodeerimise metodoloogia graafiga kirjeldatud automaatide jaoks.

Töö eesmärgiks on automaatide olekumuutuste keskmise ümberlülitumise aktiivsuse vähendamine, minimeerides olekusiirete käigus muutvate olekukoodi järkude arvu. Seejuures on olekukoodi pikkus määratud olekute kooditükelduste hulga võimsusega, mis omakorda võrdub automaadi siirdegraafi lõikete arvuga. Lõigete hulk konstrueeritakse nii, et siirdega seotud naaberolekute vaheline loogiline e. Hammingu kaugus oleks minimaalne.

Kasutades automaatide tõenäosuslikku kirjeldust on väljatöötatud kodeerimisstrateegiat edasi arendatud ning pakutud välja olekute kodeerimisalgoritm, mis minimeerib kõrge siirdetõenäosusega olekukoodide vahelise Hammingu kauguse. Väljatöötatud raamistik annab olekute kodeerimise üldise määratluse, mis seob automaatide tõenäosusliku kirjelduse võimsuskadudega.

Automaadi olekute ümberlülitamise aktiivsuse minimeerimine ei garanteeri iseenesest alati üldise võimsuskao vähenemist, kuna pole arvestatud automaadi realisatsiooni kombinatoorse osa poolt tarbitavat energiat. Automaadi võimsuskadude summaarsel minimeerimisel kasutatakse automaadi dünaamilise võimsuskontrolli meetodeid (*Dynamic Power Management*), milliste hulka kuulub ka automaadi dekomponeerimine. Baseerudes dekomoneerimismeetodil on käesolevas töös esitatud automaadi olekute kodeerimisstrateegia ning uus kompleksne lähenemine madala energiatarbega automaatide projekteerimisele, mis koosneb kahest etapist: automaadi dekomponeerimisest ja automaadi olekute kodeerimisest. Veelgi enam, need etapid on omavahel väga tihedalt seotud, mis antud struktuuripiiranguid arvestades võimaldab lahendada omavahel ühendatud ja vastastiktoimega komponentautomaatide võrgu projekteerimise ülesandeid.

Käesoleva doktoritöö oluliseimaks tulemuseks ongi täieliku raamistiku loomine madala energiatarbega automaatide sünteeks. Nimetatud tulemus saavutati eelkõige tänu uue heuristilise lähenemise õnnestunud seostamisele varasemate hästituntud meetoditega.

Acknowledgements

I have many people to thank for their direct or indirect contribution to this research work.

First, I would like to thank the main advisor, Dr. Alexander Sudnitson of Tallinn University of Technology forgiving me the opportunity of affording this research work. I am especially grateful to him for guiding my efforts, for constantly monitoring my progresses on the research, and for reading these pages.

My deepest gratitude is to Professor Dr. Andres Keevallik of Tallinn University of Technology for his constant availability and guidance, which enable my research to achieve fruitful results. I am also grateful him for the friendship he demonstrated to me at difficult moments.

Then, I would like to thank Dr. Margus Kruus of Tallinn Technical University for giving me his support.

Many special thanks to my opponents Professor Dr. Arkadi Zakrevskij of United Institute of Informatics Problem, Minsk, Belarus and Dr. Vladimir Hahanov of Kharkov National University of Radioelectronics, Ukraine for the time and patience spent in carefully reading this dissertation and other previous reports and papers.

I also wish to thank my colleagues of Computer Department of Tallinn University of Technology, mainly Professor Raimund Ubar, Dr. Marina Brik, Sergei Devadze and Roman Vassilyev for their support to me in the research.

My warmest special thanks I would like to thank to those without whom this would have never been possible, my parents: especially to my mother for being a constant and loving support at every difficult moment and believing in me.

Lastly, I lovely wish to thank all my friends, especially Alexander Rastovtsev, Oleg Balabanov, Elena Prelova and Ruslan Moris, for their unconditioned love and constant presence.

With all my heart, this thesis is dedicated to all of them.

Elena Fomina
Tallinn, 2005

List of publications

Part of contents of this dissertation has been published in the following papers:

[DDECS'02] E. Fomina, A. Keevallik, and A. Sudnitson, "*Entropic Analysis of Finite State Machines' Networks*", in Proc. IEEE 5th International Workshop on Design and Diagnostics of Electronic Circuits and Systems, Brno, Czech Republic, pp. 244-251, 2002.

[MIEL'02] E. Fomina, A. Keevallik, and A. Sudnitson, "*Lower Power Synthesis Based on Information Theoretic Measures*", in Proc. IEEE 23rd International Conference on Microelectronics, Nis, Yugoslavia, NJ, USA: IEEE, vol.2, pp. 699-702. 2002.

[StZag'02] E. Fomina, "*Entropy Evaluations of Information in Finite State Machines' Networks*", in Proc. National Conference with International Participation, Stara Zagora, Bulgaria, pp. 85-91, 2002.

[BEC'02] E. Fomina, A. Keevallik, M. Kruus, and A. Sudnitson, "*Web-based Tools for Finite State Machine Decomposition with Analysis of Information Flows*", in Proc. of the 8th International Baltic Electronics Conference, Tallinn, Estonia, pp. 165-168, 2002.

[EUROCON'03] S. Devadze, E. Fomina, A. Keevallik, M. Kruus, and A. Sudnitson, "*Web-Based System for Sequential Machines Decomposition*", in Proc. of IEEE EUROCON 2003 International Conference on Computer as a Tool, V.1., Ljubljana, Slovenia, v.1, pp. 57-61, 2003.

[CompSysTech'03] E. Fomina, A. Keevallik, M. Kruus, and A. Sudnitson, "*A Decomposition Procedure for Register-Transfer Level Power Management*", in Proc. of International Conference on Computer Systems and Technologies, Sofia, Bulgaria, v.1, pp.21-26, ACM Press New York, NY, USA, 2003.

[SAER'04] E. Fomina, P. Ellervee, M. Kruus, A. Sudnitson, and K. Tammemae, "*Digital Synthesis Tools for Education and Research*", in Proc. 18th International Conference on Systems for Automation of Engineering and Research, Varna, Bulgaria, pp. 160-164, 2004.

[EWDTW'04] E. Fomina, and A. Sudnitson, "*Information Relationships for Decomposition of Finite State Machine*", in Proc. of East-West Design & Test Workshop, Crimea, Ukraine, pp. 41-47, 2004.

[WsBP'04] E. Fomina, A. Sudnitson, and R. Vassilyev, "*FSMs Network Coding Guided by Informational Relationship Measure*", in Proc. of the 6th International Workshop Boolean Problems, Freiberg, Germany, pp. 55-62. 2004.

[BEC'04] E. Fomina, A. Sudnitson, and R. Vassilyev, "*Optimization of FSMs Network by New Encoding Strategy*", in Proc. of the 9th International Baltic Electronics Conference, Tallinn, Estonia, pp. 119-122, 2004.

[CompSysTech'05] E. Fomina, "*State Encoding Technique for Low Power FSM*", in Proc. of International Conference on Computer Systems and Technologies, Varna, Bulgaria, pp. V.1-1-V.1-6, 2005.

[EWDTW'05a] E. Fomina, R. Vassilyev, M.Brik, and A. Sudnitson "*New Approach to State Encoding of Low Power FSM*", in Proc. of the IEEE East West Design Test Workshop, Odessa, Ukraine, pp. 21-26, 2005.

[EWDTW'05b] M. Brik, E. Fomina and R. Ubar, "*A Proposal for Optimization of Low-Powered FSM Testing*", in Proc. of the IEEE East West Design Test Workshop, Odessa, Ukraine, pp. 15-20, 2005.

[SAER'05] E. Fomina and A. Sudnitson, "*Extended Finite State Machine Decomposition for Low Power*", in Proc. of the 19th International Conference on Systems for Automation of Engineering and Research, Varna, Bulgaria, pp. 126-131, 2005.

Abbreviations

CAD – Computer-Aided Design
CMOS – Complementary Metal-Oxide Semiconductor
DPM – Dynamic Power Management
FSM – Finite State Machine
FSMD – Finite State Machine with Data-path
FPGA – Field Programmable Gate Array
ICs – Integrated Circuits
STG – State Transition Graph
RTL – Register Transfer Level
VLSI – Very Large Scale Integration

CONTENTS

1 Introduction 1

1.1 Switching activity as main factor for low power FSM synthesis 1
1.2 FSM state encoding targeting reduction of switching activity 3
1.3 FSM decomposition for RTL power management 7
1.4 Outline of the thesis 11

2 Preliminaries 12

2.1 Basic automata theory concepts 12
2.2 Basic algebraic structure theory concepts 15

3 State Encoding for a Low Power FSM 20

3.1 Introduction 20
3.2 A new state encoding technique 22
3.2.1 Problem statement 22
3.2.2 Weakly crossed edge cuts encoding algorithm 28
3.2.3 Comparison of encoding methods 43
3.2.4 Further improvement of the received encoding 49
3.3 Experimental results 51
3.4 Summary 53

4 Decomposition for a Low Power FSM 54

4.1 Introduction 54
4.2 FSM decomposition technique 56
4.2.1 Decomposition constraints 59
4.2.2 Information relationship measures 63
4.2.3 Decomposition procedure 67
4.2.4 Encoding of the network of machines 76
4.3 Experimental results 81
4.4 Summary 88

5 Conclusion and Future Direction 90

5.1 Thesis summary 90
5.2 Future work 92

References 93

Other bibliography 99

1 INTRODUCTION

Power consumption has become a primary concern in the design of Integrated Circuits (ICs). Two independent factors have contributed for this. On one hand, low power consumption is essential to achieve longer autonomy for portable devices. On the other hand, increasingly high circuit density and higher clock frequencies are creating heat dissipation problems which in turn raise reliability concerns and lead to more expensive packaging. Huge effort has been invested to come up with a wide range of design solutions that help solve the power dissipation problem for different types of electronic devices, components and systems.

The thesis focuses on a technique to solve the problem of reducing the power dissipated in synchronous sequential circuits range from Register Transfer Level (RTL) power management. More precisely, we present techniques applicable at RT-level that have proven to hold good potential for power optimization in practical design environments.

1.1 Switching activity as main factor for low power FSM synthesis

We propose a novel approach for the low power synthesis of synchronous Finite State Machines (FSM) starting at the RT-level of design specification. Since in Complementary Metal-Oxide Semiconductor (CMOS) technology the largest fraction of power dissipation is caused by signal switches, our approach deals with the reduction of switching activity.

In the CMOS technology, overall power consumption can be partitioned in three main components [57]: $P=P_{dyn}+P_{sc}+P_{lk}$. P_{dyn} is the dynamic or switching power. It is due to charging and discharging load capacitance. P_{sc} (short-circuit power) is caused by the currents flowing from supply to ground when pairs of PMOS/NMOS transistors are conducting simultaneously. Finally, P_{lk} (leakage power) is static in nature and it originates mainly from sub-threshold MOS conduction. In most current CMOS IC technologies, P_{dyn} is dominant [1], [7].

During normal operation of well designed CMOS circuits, power consumption is determined by the switching activity in the circuit [27], [41]:

$$P_{dyn} = \frac{1}{2} C_L A_{SW} V_{dd}^2 f_{clk} \quad (1)$$

Here C_L is the load of a circuit node, V_{dd}^2 is the supply voltage, f_{clk} is the clock frequency and A_{SW} is the switching activity of the node, defined as the expected number of logic transitions during one clock cycle.

P_{dyn} reduction targets the minimization of one or more factors in the equation above. The parameters that influence the dynamic dissipation are the voltage, the capacitance, the frequency of the switching activities. Voltage and

capacitance are limited by the technology used; hence, ways to reduce dynamic dissipation are focused on decreasing frequencies of switching activities [41].

The average power dissipation is proportional to the average switching activity [6], [66]. A good approximation of the average switching activity is the switching probability). Given the input switching probability it is possible to calculate the probability of the state transitions in an FSM [26], [45].

At the gate level the average switching activity at the output of a gate i in a time period T is the average number of signal transitions [6]: $n_i(T) = \frac{n_{trans}}{T}$, where n_{trans} is the number of transitions during time period T.

The switching (transition) probability is the limit value of the switching activity when the observation time goes to infinity [17]: $p_i = \lim_{T\to\infty} n_i(T)$.

According to [27], [41], the power dissipation at the gate level can be regarded to be proportional to its switching activity. Then the average overall power dissipation P_{tot} is a function of the average power dissipated by each gate g_i during one clock cycle T_{cycle} as follows:

$$P_{tot} = \frac{\frac{1}{2}V_{dd}^2}{T_{cycle}} \sum_{i=1}^{G} C(g_i)E(g_i) \qquad (2)$$

Here V_{dd}^2 is the supply voltage, $C(g_i)$ is the capacitive load at the output of gate g_i, $E(g_i)$ is the switching activity of the gate g_i and G is the set of all gates in the circuit [76].

The intention of low power design is to diminish the value of P_{tot}. Voltage V_{dd} and clock cycle T_{cycle} are assumed to be fixed. The thesis concentrates on reducing the power consumption at the gate level by reducing of the term $\sum_{i=1}^{G} C(g_i)E(g_i)$ in (2). This can be done by determining a register configuration and a combinational structure [14].

Basically, our approach provides a workbench-like method taking in two particular low power optimization techniques and a strategy for their application. The two techniques are deactivation of the parts of the circuit which are not doing useful work and shut them down by either turning off the power supply or the clock signal and appropriate state encoding.

Using our strategy, we consider the data-dependent or DPM technique [51] in order to exploit both techniques in a suitable way. Depending on the special application-specific requirements presented state encoding can be applied separately or in a combination with the power management technique manner. In the case of combined application our state encoding approach is capable to consider state encoding constrains resulting from the disabling the inactive parts of the circuit.

1.2 FSM state encoding targeting reduction of switching activity

The currently used technologies for design of sequential circuits usually consist of several independent phases [6], [27] among which the step of encoding is one of significance. In general, an encoding (or assignment) problem is to assign (binary) codes to attribute to symbolic states, satisfying the cost metrics, through the minimization of a given cost function. The problem is NP-complete, indeed the optimum encoding can be found by exhaustive enumerating all the possible assignments, carrying out logic synthesis for each assignment and then picking the one that has the least area. This method is computationally too expensive. Hence, different heuristic methods are used to obtain a solution. The problem is a classic in switching circuit's theory [58]. However, recently the encoding problem is again gaining momentum. The one of predominant reasons of that is a demand of energy efficient designs. In the last few years the problem of increasing the power consumption through the phase of encoding during sequential circuit design has become a primary and a major concern. Many various works devoted to encoding use different models, methods and heuristics have been reported to address the power-efficient design at different levels of abstraction [1], [57].

The main investigation object in the current thesis is an FSM that was generated and presented in the form of State Transition Graphs (STG) or State Transition Tables (STT). Currently is discussed only the problem of assigning the *internal states* of an FSM with unique binary codes. *State encoding* determines the number of flip-flops that are required to hold the state and influences the complexity of the combinational logic used to realize the next state and output of a synthesized state machine. The number of flip-flops must be sufficient to represent the number of states as a binary number. A machine with n states will require at least $\log_2 n$ flip-flops to store the encoded representation of the states, but it could have more. The task of finding the state encoding with minimum required code length is most widespread. In the presented research the task of state encoding with minimal required number of bits is also considered.

A code that changes by only one bit between adjacent codes will reduce the simultaneous switching of adjacent physical signal lines in a circuit, thereby minimizing the possibility of electrical crosstalk. These codes also minimize transitions thought intermediate states, when state changes occur in the operation of the actual hardware. The problem of intermediate transitions arises because flip-flops in the state register do not change simultaneously. When more that one bit changes to make a state transition and the bit do not switch simultaneously, an intermediate state is present momentarily in the state register. This could have undesirable consequences. Hence, the strategic aim of the investigation is the development of the optimal (code length $\rightarrow$ minimum code length) FSM state encoding that minimizes the number of state variables that changes their value when an FSM moves between two adjacent states.

Ideally, if we can guarantee that each state transition results in a single state variable change, than we will have optimally reduced the switching activity associated with registers in the given FSM.

Modifying of the state encoding is most of the popular technique to reduce power in an FSM [6], [46], [50], [71], [77]. The state encoding has been extensively studied because it is a crucial step in the synthesis of the controller circuitry. Early researches were focused on finding a state encoding that minimizes area of the circuit [13], [17], [20], [67], [71]. More recently, a number of low power state encoding techniques target the reducing of switching activity have been proposed [37], [47], [55], [56], [75], [77]. The problem of Minimum Weighted Hamming Distance [61] was firstly formulated in 1992. This problem considers the reducing of switching activity of input state lines on next state logic during state encoding. The objective function in encoding techniques is to minimize the Weighted Hamming Distance. Despite of that the Minimum Weighted Hamming Distance does not exhibit a high absolute accuracy this metric is still relevant and quite effective [4], [6], [55], [56], [66].

The state encoding for power dissipation in an FSM has received a lot of attention. It would be impossible to report in detail on all different approaches that have been proposed. We restrict our attention to those state encoding methods that are either very commonly used or representative of a class of techniques.

In 1994, Olson et al. used a linear combination of switching activity of the next state lines and the number of literals as the cost function [56].

In 1995, Benini and De Michelli presented the aim of the low power state assignment is the minimization of register switches in the synthesized circuit which is approximated by the register switching rate. Usually this minimization is performed by a simulated annealing procedure utilizing weighted state transitions [6].

In 1997, Surti at al. presented the Huffman-code architecture to realize encoding using two different code lengths [69]. Switching activity is reduced by decreasing the expected number of state bits switched less than $\lceil log_2|S| \rceil$. The state set S of the FSM is decomposed into two sets based on the limited state probabilities. The state set with very high probability is encoded with less than $\lceil log_2|S| \rceil$ bits. The other state set, being less probable, is encoded using more than $\lceil log_2|S| \rceil$ bits. Therefore authors use two code lengths for one state machine.

In 1998, Tsui et al. [72] used simulated annealing as a search strategy to find a low power state encoding that accounts for both the switching activity of the next state lines and switched capacitance of the next state and output logic.

In 1999 it was proposed that the register switching activity can be reduced by an adapted state assignment [29]. The state assignment procedure for power and complexity consists of the following three steps: selection of a suitable subset of all possible facets (trade-off between complexity and power), based on this subset criterion of a partial state encoding for complexity

reduction, completion the partial encoding by exploiting the remaining optimization apace to minimize the register switching rate.

In 2000, Silvano provided a general frame work for low power state assignment, starting from a probabilistic description of the FSM. The authors consider the state assignment process as composed of two tasks: symbolic state ordering and state encoding. The state ordering determines a propriety list of symbolic states to be used during the successive phase of encoding. The weight of each edge in the STG reflects the transition probabilities inside the corresponding pairs of states. Authors define encoding techniques to assign binary codes to the symbolic states to reduce the switching activity of state registers [4], [66].

Area and power can be reduced using a number of encoding variables over the minimum required to distinguish among the states. A state assignment algorithm for the synthesis of multi-level low power controllers is presented in [46]. The proposed algorithm follows a two step strategy. Each one works with a different cost function. The first step targets area minimization. A partial encoding is derived using area oriented criteria. In the second step, this partial code is completed with the aim to reduce the register switching activity. The multi-criteria approach is taken and the increasing in the number of state bits over the minimum is explored.

Wu and Pedram [78] presented the state assignment technique called priority encoding which uses multi-code assignment plus clock gating to reduce power dissipation in sequential circuits. During the low-power design of combinational circuits they have found that blocking the redundant signals and shutting off the redundant parts in circuit is effective method to low the energy dissipation [77]. The priority-based state assignment exploits the redundant state codes to mask the clock to some of the flip-flops. Some states do not require binary assignment of all state variables. When the system is in such state, the unused state variables become redundant. Because the corresponding flip-flops are not used, they can be isolated from the clock to reduce their power dissipation.

Combined parameter “area and registers switching rate” is presented in [38]. For providing a trade-off between two parameters authors estimate the efficiency of variant for area reduction, and restrict the solution space by means of two thresholds. They increase the probability to find a power reduced design in the remaining space.

Venkataraman et al. use Genetic Algorithm (GA) for simultaneous partitioning and state encoding of an FSM with power reduction as the objective [74]. They constructed GALLOP – an FSM synthesis tool targeting low power. It differs from previous works by performing state assignment and partitioning simultaneously using GA and using steady state probabilities to bias the initial population to wards giving better results. The tool yields circuits which consume considerably less power than those obtained by the use of earlier known low power synthesis procedures.

The method proposed in [62] is not mutually exclusive with any of these state assignment techniques, but complements them. A representative of existing low power state assignment algorithms attempts to minimize switching activity by reducing the Hamming distance between the states with high state transition probability between them. Various cost functions combined with these probabilities are suggested to control both area and power.

In 2003, Eggermont et al. introduce the profiling-based state-assignment technique for low power that utilizes dynamic loop information extracted from an FSM profiling data [21]. Authors proposed three different loop-based state assignment algorithms. The depth-first search (DFS) algorithm performs an exhaustive search over the FSM encoding space using the loop information for intermediate cost estimates of an encoding. The loop-based DFS algorithm performs a similar search on a loop-by-loop basis, where the loops are ordered in the descending order of weight. The per-state algorithm encodes the states individually, on the same loop-by-loop basis.

1.3 FSM decomposition for RTL power management

The development of Computer-Aided Design (CAD) techniques for power minimization has been a very active area of research [41], [57] because power dissipation has recently emerged as one the most critical design constraints [52]. A wide range of techniques has already been proposed for the optimization of logic circuits for low power.

Power management methods are among the most effective techniques for power reduction [57]. The most representative data-dependent power management techniques that have recently been proposed are pre-computation, guarded evaluation, gated-clock FSM and FSM decomposition. Each of these techniques uses a different approach to identify the input conditions for which the circuit (or part of) can be disabled. These methods detect periods of time during which parts of the circuit are not doing useful work and shut them down by either turning off the power supply or the clock signal. Several methods have been presented that perform shutdown on a clock-cycle base. Depending on the input conditions at the beginning of a clock-cycle, the clock driving some of the registers in the circuit can be inhibited, thus reducing the switching activity in the fan-out of those registers. These techniques are referred to as data-dependent or dynamic power management techniques. *Dynamic power management* is a concept that includes various design methods and techniques and is based on shutting down the parts of the circuit that are not currently active [8], [42].

High-level logic synthesis produces a combined description of data-path and control logic. The latter is normally in the form of a transition structure, whose most familiar representation is an FSM or a collection of machines. The translation of such an FSM into a structural description presents opportunities for reducing power consumption [41].

In static CMOS circuits, the probabilistic average switching activity of the circuit is a good measure of the average power dissipation of the circuit. Methods that can efficiently compute the average switching activity, and thus power dissipation, in CMOS combinational [54] and sequential [72] circuits have been developed.

Power consumption in a synchronous FSM can be reduced by partitioning it into a number of coupled component machines where only the part that is involved in a state transition is clocked [14].

In sequential circuit design, an effective approach to reduce power dissipation is to “turn off” portions of the circuit [12], and hence reduces the switching activities in the circuit. This approach is motivated by the observation that, for an FSM, active transitions occur only within a subset of states in a period time. Therefore, it is possible to synthesize an FSM in such way that only the part of the circuit which computes the state transitions and outputs will be turned on while all other parts will be turned off, power consumption will be reduced.

The design of state machines can be made with a special state machine design tool and companion optimizer, then the necessary of optimizing the state

assignment for the performing constraints is arising. Machine *decomposition* or *partitioning* can be regarded as a first phase of the state machine encoding. In other words, machine decomposition and state encoding are virtually identical concepts. Decomposition of an FSM into a group of component state machines requires additional state encoding constraints because of assigning binary codes to each state of all component machines. Composite encoding of all component machine (or encoding of a network of component machines) with aim to minimize the size of representations is considered as extended optimization problem.

In the next review some of approaches present different strategies for low power consumption which is based on the principle of decomposition [12], [39], [41], [42], [50]-[53], [60], [65], [70], [74], and [78].

In [12], the combinational logic block is partitioned and the active part is decided basing on the encoding of the present state. The states selected for one of the component machines are all encoded in such a way that the enable signal is always on for first combinational logic while it is off for the second combinational logic. Conversely, for all states in the other sub-FSM, the enabled signal is always off for the first combinational logic while it is on for the second combinational logic. Consequently, for all the transitions within the first component machine, only the first combinational logic will be active and vice-versa.

The basic idea of the clock-gating technique in [51] is to decompose the STG of an FSM into two component machines that jointly produce an input-output behavior which is equivalent to that of the original machine. Power is saved because, except for transitions between the two component machines, only one of the component machines needs to be clocked. The technique follows a standard decomposition structure. The states are partitioned by searching for a smaller subset of states with high probability of transitions among these states and a low probability of transitions between the other states. This subset of states will then constitute a small sub-FSM that is active most of the time. When the small sub-FSM is active, the other larger sub-FSM can be disabled. Consequently, power is saved because most of the time only the smaller, more power-efficient, component machine is clocked.

In 2000 orthogonal partitioning with the gated clock architecture was used in [65] for low power realization of FSM. An FSM with n state is decomposed into two approximately $\sqrt{n}$ state machines interacting with each other and running concurrently. When one or both the machines have a self-loop, then clock and primary inputs are disabled for the respective machine/machines. Therefore for all the self-edge conditions, the inputs and clock of the respective machine are disabled to reduce the switching activity and thereby the power.

The clock gating technique based on FSM decomposition presented in [50] has been modified in [52]. A serious limitation of the previously proposed techniques is that they require the STG of the FSM to be given or extracted from the circuit. Since the size of the STG can be exponential on the number of

registers in the circuit, explicit techniques can be applied to relatively small sequential circuits. The authors present an approach to perform FSM decomposition by direct manipulation of the circuit. This methodology allows avoiding both disadvantages of the previous method: the explicit extraction of the STG and computation of the transition relations. The computation of the exact transition probabilities changes to simulation of approximate transition probabilities and this approximation uses in the partition algorithm. Register-disabling signal are added to the decomposed circuit, hence the overall switching activity is minimized.

In 2002 the problem of optimizing FPGA (Field Programmable Gate Array)-based FSM circuits for low power has been considered [70]. The decomposition architecture like in [12] was evaluated in terms of area-time-power. Using the transition probability distribution an FSM is partitioned into two or more component machines such that minimize the sum of transition probabilities between component machines. Only one component machine is active at a time, meanwhile the other is disabled to save power. The transference of control between the machines is based on the values of the inputs and actual states.

Unlike previous works which focused only on either controller or data-path in 2003 authors present a decomposition technique that takes both controller and data-path into consideration [40]. The extended Finite State Machine was decomposed into several extended component machines taking state probability and resource sharing into account. At any time also only one component machine is active while the others are idle. By turning off idle circuits, the switching activity reduced and power consumption minimized.

In [74] an approach based on a Genetic Algorithm (GA) for simultaneous partitioning of an FSM with power reduction as objective was presented. The partition scheme decomposes the set of states of an FSM into two subsets implemented by two component machines. Such partition is a partition of the combinational logic of the sequential circuit into two sub-circuits each of which computes the sequential circuit outputs and next states for different state-transitions. Using GA and steady state probabilities authors proposed a methodology to estimate cost function for designing of a low power FSM.

Objectives and motivations

To define objectives of the thesis let us do some conclusions.

Conclusion 1 – the logic synthesis for achieving low-power consumption is still one of the most important problems in the energy-efficient design of ICs. Nowadays researchers show several reasons [27]. First, many ICs are employed in mobile battery-powered systems, where the lifetime of the battery decreases as the power consumption of ICs and peripherals grows. Second, low-power design is required to either satisfy technical feasibility from a thermal profile standpoint, or to reduce the cost of the package and cooling means.

Thus, economic, ecological and ethical reasons mandate the development of energy-efficient ICs.

Conclusion 2 – during the minimization of switching activity the circuit is transformed by adding logic that localizes computation in such a way that switching is substantially reduced. The cost of the added logic is amortized by significant switching activity reduction on many circuit nodes at the same time. In the case of minimization of switching capacitance, transformations directly optimize logic-level approximations of dynamic power consumption. These techniques are generally local in scope, and overall power reduction is the compound effect of a large number of local transformations.

The goal is to find a proper trade-off regarding combined parameter "area and switching rate" in the encoding procedure. The simultaneous consideration of area and register switching rate in a common low power design strategy is based on the common optimization of both parameters.

Conclusion 3 – low power FSM synthesis is conventionally identified with low power state encoding [6], [72]. The minimization of the register transitions has to be combined with an appropriate implementation of the combinational logic for obtaining a global power saving [66], so the state encoding can be the starting point for further optimization of the combinational part.

The power-oriented cost function should account for the minimization of the number of logic transitions of the state registers between two successive clock cycles, assuming the power consumption is proportional to the switching activity on the state bit lines of the machine (2). Hence, an effective state encoding technique should assign adjacent binary codes to state pairs characterized by very high transition probability.

Conclusion 4 – partitioning has been shown as effective technique for reducing power in an FSM [12], [9], [39], and [52]. Partitioning decomposes a given FSM into two or more coupled component machines. When one submachine is active, the inputs to the other component machine are turned off to reduce power dissipation. Hence, except when there is a transition between two different component machines, only one component machine is active at time.

There is a need for a development of general approach to an FSM partitioning and state encoding with power reduction as the objective.

1.4 Outline of the thesis

The rest of the thesis is organized as follows. It is divided into five main chapters. The thesis contains description of the investigated problem, implementation discussions, examples and conclusions.

Chapter 2 presents some basic concepts from the classic machine theory and algebraic structure theory of sequential machines that will be used throughout the thesis.

Chapter 3 deals with the problem of state encoding for a low power FSM. This chapter consists of four subchapters. Introduction discusses the traditional sequential logic synthesis technique for low power – state encoding. The main subchapter of the current chapter introduces a new encoding strategy for low power FSM based on the concepts of weakly crossed edge cuts. A proposed algorithm is illustratively described in detail. The comparison among several encoding methods is added. A practical application was confirmed by series of experimental results.

Chapter 4 describes FSM decomposition as an effective technique of DPM for power reduction. Three types of decomposition were presented. The first type is multiplicative decomposition which is based on the algebraic theory and is constructed by a set of partitions on the set of states of a decomposable machine. The second and the third types are additive and generalized additive decompositions, which are based on the identification in the STG of sub-routines or co-routines. A sub-routine/co-routine corresponds to a fragment of the STG augment with a wait state. Additive decomposition is constructed by a partition on the set of states of decomposable machine, while generalized additive decomposition is constructed by a cover on the set of states of decomposable machine.

The low power FSM optimization task consists of decomposition and further encoding. Decomposition applies additional restrictions – decomposition constraints. After multiplicative decomposition encoding of the network of component machines is a composite encoding of component machines with decomposition constraints – blocks of partitions from a complete system of partitions. After additive and generalized additive decomposition we apply an independent encoding of component machines based on the heuristic described in the previous chapter.

Chapter 5 summarizes main contributions and outlines possible directions for future investigations.

2 PRELIMINARIES

2.1 Basic automata theory concepts

Over the years, many important problems in sequential circuit synthesis and optimization have been approached using concepts from automata theory. Traditionally, FSM is a discrete dynamical system translating sequences of input vectors into sequences of output vectors. An FSM has a set of states and of transitions between states; the transitions are triggered by input vectors and produce output vectors. The states can be seen as recording the past input sequence, so that when the next input is seen a transition can be produced based on the information of the past history. If a system is such that there is no need to look into past history to decide what output to produce, it has only one state and therefore it yields a combinational circuit. From the other side, systems whose past history cannot be condensed in a finite number of states are not physically realizable.

It is widely recognized that the FSM is used to model the work of digital devices, and reachability analysis is a powerful approach to retrieve information from such model. Because of their finite nature, FSM yield better to describe, to analyze and to synthesize of intricate digital systems than any other alternative models. Moreover, the FSM is often the formalism of choice for specifying the behavior of sequential components [22].

Behavior representation of an FSM

Traditionally, associated with a circuit, an FSM is represented as algebraic quintuple: sets of states, inputs and outputs, and two functions – transition and output.

Definition 2.1 A *Finite State Machine* is a discrete dynamic system translating sequences of input vectors into sequences of output vectors and defined as $A=(S,I,O,\delta,\lambda)$:

- S is a finite nonempty set of states;
- I is a finite nonempty set of inputs;
- O is a finite nonempty set of outputs;
- δ: $S\times I\rightarrow S$ is called the transition (or next state) function;
- λ: $S\rightarrow O$ (Moore FSM) and $S\times I\rightarrow O$ (Mealy FSM) is called output function.

States S, I and O are nonempty. Functions δ and λ are multiple-output Boolean functions: an l input, r output Boolean function F, is a mapping from an l-dimensional Boolean space to an r-dimensional Boolean space F: $B^l\rightarrow B^r$, where B={0,1}. B^l is called the *domain* of F, and B^r is called the *co-domain* of F. If $r>1$ the F is *multiple output function*. The sets of inputs $I=\{x_1, x_2, \ldots, x_l\}$, where $x_i\in I$ is a binary input variable and outputs $O=\{y_1, y_2, \ldots, y_r\}$, where $y_i\in O$ is a binary output variable are considered as structural inputs and outputs of an FSM. The domain of next state function is $D(\delta)\rightarrow S\times D^l$, $D=\{0,1\}$. The domain

of output function is $D(\lambda)\rightarrow D^r$, $D=\{0,1\}$ (Moore FSM) and $D(\lambda)\rightarrow S\times D^r$, $D=\{0,1\}$ (Mealy FSM).

An FSM can be represented by two equivalent structures, a State Transition Graph (STG) and, a State Transition Table (STT). The first is graphical, second is tabular representation form. The two representations are equivalent.

Definition 2.2 Given an FSM $A=(S,I,O,\delta,\lambda)$, the State Transition Graph STG(A)=(V,E) is a labeled directed graph where each state in S corresponds to a vertex in V labeled v and each transition in δ corresponds to a directed edge in E labeled e.

The STG for the FSM "bbara" [48] is depicted in Figure 2-1, where e_1:$\neg x_3x_4\vee x_3\neg x_4\vee\neg x_3\neg x_4$, e_2:$\neg x_1\neg x_2x_3x_4$, e_3:$x_2x_3x_4$, and e_4:$x_1\neg x_2x_3x_4$. Each edge in the STG corresponds to an entry in the state table.

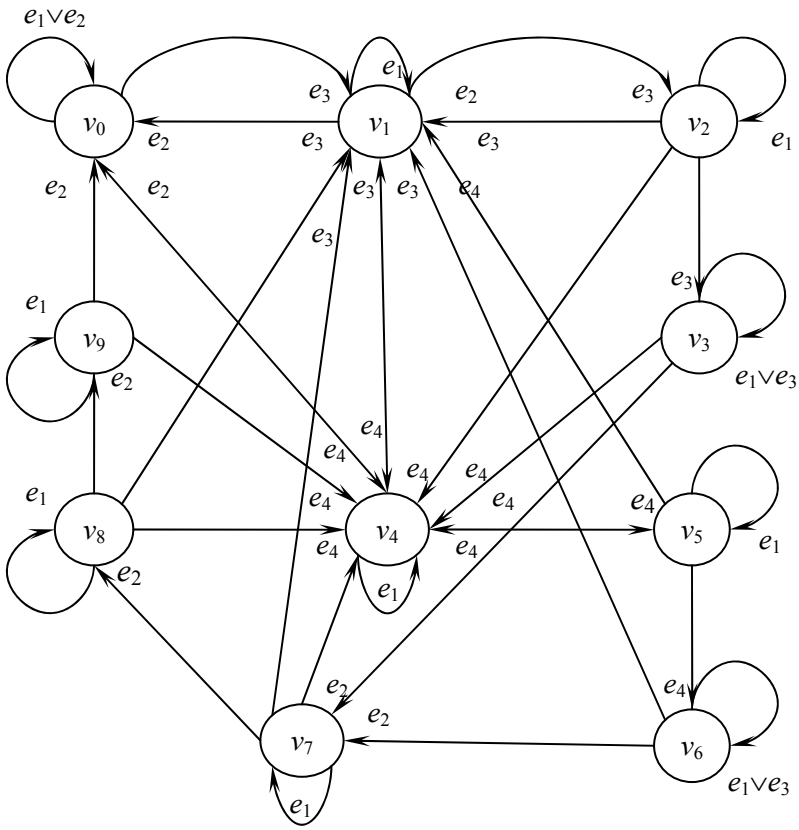

Figure 2-1 State transition graph for the FSM "bbara"

The states of the STG are labeled with the unique symbolic state names, whereas the edges are labeled with the corresponding inputs and outputs values.

The state table is simply the list of edges of the STG. An example of STT for the FSM "bbara" is presented in Table 2-1.

Table 2-1 State transition table for the FSM “bbara”

№	Present state	Input	Next state
1	*st*0	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4 \vee \neg x_1\neg x_2x_3x_4$	*st*0
2		$x_2x_3x_4$	*st*1
3		$x_1\neg x_2x_3x_4$	*st*4
4	*st*1	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4$	*st*1
5		$\neg x_1\neg x_2x_3x_4$	*st*0
6		$x_2x_3x_4$	*st*2
7		$x_1\neg x_2x_3x_4$	*st*4
8	*st*2	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4$	*st*2
9		$\neg x_1\neg x_2x_3x_4$	*st*1
10		$x_2x_3x_4$	*st*3
11		$x_1\neg x_2x_3x_4$	*st*4
12	*st*3	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4 \vee x_2x_3x_4$	*st*3
13		$\neg x_1\neg x_2x_3x_4$	*st*7
14		$x_1\neg x_2x_3x_4$	*st*4
15	*st*4	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4$	*st*4
16		$\neg x_1\neg x_2x_3x_4$	*st*0
17		$x_2x_3x_4$	*st*1
18		$x_1\neg x_2x_3x_4$	*st*5
19	*st*5	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4$	*st*5
20		$\neg x_1\neg x_2x_3x_4$	*st*4
21		$x_2x_3x_4$	*st*1
22		$x_1\neg x_2x_3x_4$	*st*6
23	*st*6	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4 \vee x_1\neg x_2x_3x_4$	*st*6
24		$\neg x_1\neg x_2x_3x_4$	*st*7
25		$x_2x_3x_4$	*st*1
26	*st*7	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4$	*st*7
27		$\neg x_1\neg x_2x_3x_4$	*st*8
28		$x_2x_3x_4$	*st*1
29		$x_1\neg x_2x_3x_4$	*st*4
30	*st*8	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4$	*st*8
31		$\neg x_1\neg x_2x_3x_4$	*st*9
32		$x_2x_3x_4$	*st*1
33		$x_1\neg x_2x_3x_4$	*st*4
34	*st*9	$\neg x_3x_4 \vee x_3\neg x_4 \vee \neg x_3\neg x_4$	*st*9
35		$\neg x_1\neg x_2x_3x_4$	*st*0
36		$x_2x_3x_4$	*st*1
37		$x_1\neg x_2x_3x_4$	*st*4

Notice that both the STG and the STT completely define the input-output behavior of the FSM, but they do not provide any information regarding the circuit implementation. In that sense, STG and STT can be considered as *behavioral* representation of an FSM.

2.2 Basic algebraic structure theory concepts

In the early sixties Hartmanis was one of the first to work on an algebraization of the notion of logical or functional dependences in an FSM. His "Algebraic Structure Theory of Sequential Machines" [28] presented fundamental tools for describing two concepts, namely, "information" and "information dependence". The importance of this theory lies in the fact that it provides a direct link between algebraic relationships and physical realizations of machines. The formal techniques are very closely related to modern algebra. It has an abstract beauty combined with the challenge of physical interpretation and application. It falls squarely in the interdisciplinary area of applied algebra, which is a part of engineering mathematics.

Partitions and partition pair algebra

We regard a partition on a finite set as an instrument for analyzing this set.

Definition 2.3 A *partition* π on S is a collection of disjoint nonempty subsets of S whose set union is S, i.e. $\pi=(B_a)$ such that $B_a \cap B_b=\varnothing$ for $a \neq b$ and $\cup(B_a)=S$. We refer to the sets of π as blocks of π and designate the block which contains s by $B_\pi(s)$.

Otherwise a set π of subsets of S, is a partition of S if

- No element of π is empty;

- The union of the elements of π is equal to S;

- The intersection of any two elements of π is empty (the elements of π are pair-wise disjoint)

Hence, a partition on a finite set can be interpreted as an algebraic form of the notion of information. According to this interpretation, the zero partition contains maximum information while the unit partition contains minimum information about the set.

The notion of partition is the particular case of the notion of cover: a *cover* φ of a set S is a collection of subsets B of S whose union is S. More generally, if $B \subseteq S$ and φ is a collection of subsets of S whose union contains B, then φ is said to be a cover of S.

Now we describe how partitions on a set can be "multiplied" and "added". These operations and the subsequently defined ordering of partitions play a central role in the structure theory of sequential machines and form a basic link between machine concepts and algebra.

If π_1 and π_2 are partitions on S, then:

First, $\pi_1 \cdot \pi_2$ is the partition on S such that $s \equiv t(\pi_1 \cdot \pi_2)$ if and only if $s \equiv t(\pi_1)$ and $s \equiv t(\pi_2)$.

Second, $\pi_1+\pi_2$ is the partition on S such that $s \equiv t(\pi_1+\pi_2)$ if and only if there exists a sequence in S $s=s_0, s_1, \ldots, s_n=t$ for which either $s_i \equiv s_{i+1}(\pi_1)$ or $s_i \equiv s_{i+1}(\pi_2)$, $0 \leq i \leq n-1$.

The basic research object in structure theory of sequential machines is a partition pair [48]. To study of machine structure is begun in following definition with a formal notion of concept "information dependence".

Definition 2.4 A *partition pair* (π, π') on the machine $A=(S,I,O,\delta,\lambda)$ is an ordered pair of partitions on S such that $s\equiv t(\pi)$ implies $\delta(s, x)\equiv\delta(t, x)(\pi')$ for all x in I.

Thus (π, π') is a partition pair (p.p.) on S if and only if the blocks of π are mapped into the blocks of π' by S. That is, for every x in I and B_π in π, there exists a $B_{\pi'}$ in π' such that $\delta(B_\pi, x)\subseteq B_{\pi'}$.

The concept of partition pairs based on the idea, that the first partition in a pair has enough information to calculate the second one.

Next we determine for a given partition π, which partitions π' can be used to make a partition pair (π, π') on S.

Definition 2.5 If π is a partition on S of A, let $m(\pi)=\prod(\pi_i|(\pi, \pi_i)$ is a p.p. on A) and $M(\pi)=\Sigma(\pi_i|(\pi_i, \pi)$ is a p.p. on A).

Informally speaking, for a given partitionπ, the partition $m(\pi)$ describes the largest amount of information which we can compute about the next state of A knowing only π (i.e. the block of π which contains the present state of A). Similarly, for a given partition π', the partition $M(\pi')$ describes the least amount of information we must have about the present state of A to compute π' for the next state.

Partition pairs and a component machine

Partitioning methods allow transforming a source FSM into a set of smaller interconnecting and interacting machines. Let consider a circuit C with m flip-flops, i.e., there are m state variables of the set of states S in the FSM, A, associated to C, the entire state space for machine A is S^m. Each state variable corresponds to a coordinate vector of this Boolean m-space. And let π_i a *state partition* of A is a partition of S. Each component of the state partition represents a Boolean subspace consisting of the coordinate vectors corresponding to the state variables in the partition component.

Each block of the state partition π_i identify a set of components of the next state function. This set of components can be seen as the next state function of a sub-FSM of the given machine.

Let π_i be the state partition inducting *state decomposition* on machine A. For each π_i, there is an associated FSM, $A_i=(S_i,I_i\delta_i)$ where δ_i is the partitioned next state function.

Definition 2.6 A *component machine* is a triple $A=(S,I,\delta)$:

- $S=(s_1, s_2, \ldots, s_M)$ is a set of the component machines *states*;
- $I=(x_1, x_2, \ldots, x_L)$ is a set of primary input symbolic variables of the component;
- δ: $D(\delta)\rightarrow S$ is a multiple valued next state function for component with domain $D(\delta)=D_1\times\ldots\times D_L\times S$ and co-domain S. Here, D_i represents a set of values each x_i may assume.

The set ($A_i \mid i \in I=(1, \ldots, n)$) of all the component machines A_i represents the *decomposed FSM network* obtained from the original machine A when its set of states is decomposed according of π_i.

Definition 2.7 A *FSM network* we treat as a system $N=(S_N,I_N,O_N,\delta_N,\lambda_N)$, where:

- $S_N=\{A_i=(S_i,I_i\delta_i)|i=\{1, \ldots ,n\}\}$ is a set of state machines referred to as component machines;
- I_N is a set of network external inputs;
- O_N is a set of network external variables;
- δ_N: $(\times S_j)\times I_N \rightarrow I_i$ $1 \le i, j \le n$ – machine connecting rules;
- λ_N: $(\times S_i)\times I_N \rightarrow O$ – network output function.

As mentioned above, on the set of states S of the source original FSM A we chose a state partitions π_i. Next we define some information partitions which are induced on A by a network that defines A. These "associated" partitions on A may be thought of as a global characterization (on A) of the information used and computed in a component machine of network. It is natural correspondence between local and global properties that allows us to approach the structure of machines with partition pair algebra.

For each component machine can be separated two partition pairs. The partition pair I-S determines the dependence between external input of the network and state of the component machine. The pair S-S reflects the dependence between previous and next states of the component. Partitions $\pi(S)$ and $\tau(S)$ are partitions on the set of states and $\eta(X)$ is a partition on the set of inputs for all $x \in X$ and for all $s \in S$. The equivalencies specifying $\tau(S)$ and $\eta(X)$ are: $s \equiv t(\tau) \leftrightarrow \delta(s,x) \equiv \delta(t,x)(\pi)$ and $a \equiv b(\eta) \leftrightarrow \delta(s,a) \equiv \delta(s,b)(\pi)$. According to [28] these pairs form pair algebras associated with the component machine.

Suppose that the state behavior of a machine A is realized by network and suppose that $s=\alpha(s_1, s_2, \ldots, s_n)$ and $t=\alpha(t_1, t_2, \ldots, t_n)$ are states of A; a and b are inputs to A, than let

$s \equiv t(\pi_i)$ if and only if $s_i=t_i$;

$s \equiv t(\rho_{i,j})$ if and only if $f_{i,j}(s_i)=f_{i,j}(t_i)$;

$a \equiv b(\mu_i)$ if and only if $f_{i,j}(i(a))=f_{i,j}(i(b))$.

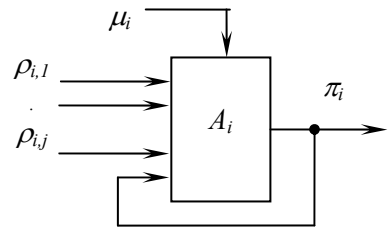

Figure 2-2 Associated partitions for the component machine A_i

Main conditions of decomposition of an FSM

It is not an overstatement to tell that the decomposition task is one of the most intricate and actual problem at complex discrete devices synthesis. Commonly, the FSM decomposition task is representing of an original FSM as its network realization. It means that, we construct such network of interconnecting and interacting component machine that it must realizes the work of an original FSM. As mathematical tool for do this we chose partition pair algebra. In our opinion it is expedient algebraic system for research of FSM structural properties. The basic element of the theory is partition which can be interpreted as a measure or equivalent of information the source set. Thus, character of possible FSM partitions is caused by properties of partitions on the set of states of the original FSM. In other words, the process of finding a good decomposition of the original machine into set of component machines can be called machine partitioning [28].

There, we should emphasize that at the decision of practical problems we are not satisfied with any machine partitioning. Depending on conditions of the given task, on construction of a network various additional conditions are imposed. Moreover, the decomposition approach can be widely used to investigate all kinds of internal functional dependences of an FSM. As frameworks of this work do not assume of search of effective decomposition, then we consider necessary and sufficient condition for existence of FSM decomposition in general.

Next, we present a fundamental theorem of machine decomposition.

Given a machine $A=(S,I,O,\delta,\lambda)$ and partitions π_i and $\rho_{i,j}$ on S and μ_i on I for $1\le i,j\le n$; then there exists a network such that realizes the state behavior of A, and π_i, $\rho_{i,j}$, μ_i are associated partitions on A if and only if the following conditions hold:

$$\pi_i \prod_i \rho_{j,i} \le M_{S-S}(\pi_i) \text{ for all } i;$$

$$\mu_i \le M_{I-S}(\pi_i) \text{ for all } i;$$

$$\rho_{i,j} \ge \pi_i \text{ for all } i \text{ and } j;$$

$$\prod_i \pi_i = 0 .$$

This theorem show that a network can be built to certain specifications on what information is to be stored where and what carry information is to be used in computing states. Additional associated partition can be defined to study carries to output logic.

Summarizing, the main condition of general FSM decomposition is equality to zero partition (zero partition is the partition whose elements are the singleton subsets of the set) of product of all selected partitions on the set of the states of the FSM. These partitions are called *complete set of partitions*. From informational point of view, while a partition on the set of states of a source FSM is some measure of information about corresponding component sub-FSM, the zero partition on the set of states of decomposed FSM contains

complete information about it. Partitions $\rho_{i,j}$ are additional information partitions for partitions π_i, which represent the flows of information from other component machines. Thus, component sub-FSM A_i receives information flow from itself, which is illustrated by partition π_i and from other component machines, which is illustrated by product of partitions $\prod_j \rho_{ji}$. The problem of search suitable partitions $\rho_{i,j}$ rests on the state assignment problem of network [28]. It is well known that the selection of binary codes to represent the internal states of the machine is one of the central problems in the physical realization of sequential machines.

Partition $M_{S\text{-}S}(\pi_i)$ describes amount of information we must have about the present state of A_i to compute π_i for the next state. In other words, amount of information received by the given machine from itself and from other machines should be sufficient to compute the next state. In this way, the first and the third conditions of main decomposition theorem are executed. The second condition can be interpreted as follows. Partition μ_i illustrates how much information from input need to work of component FSM A_i.

Partitions search problem

Two products C and C' are in the relations of consensus (C con C') if and only in they have opposite values (0 and 1) exactly in one bound component. Two covers φ_1 and φ_2 are in consensus if and only if there are $C \in \varphi_1$ and $C' \in \varphi_2$ which are in consensus [14].

For every $x \in I$ we define such symmetric binary relation ω on S that $s_p \omega s_q$ ($p \neq q$) if and only if for some s_r exist α-transitions (s_r, s_p, α_{rp}) and (s_r, s_q, α_{rq}) such that correspondent input conditions α_{rp} and α_{rq} are in consensus [68]. As a result of transitive closure operations of relation ω we will receive symmetric and transitive relation on S which we represent as partition with don't care (PDC) [28] and call primary α_i-*partition* on S.

If $x \in I$, than $\alpha_i(x)$ is PDC on S such, that $s_i \sim s_j(\alpha_i(x))$ means that transitions from state s_i to state s_j are the same if input variable x is masked (s_i and s_j are "indistinguishable" by the input channel x).

For every PDC τ we put in accordance partition $\pi \in G(\tau)$, which defines component machine A_i in the network N. The number of states of component machines is equal to the number of blocks in corresponding partition π [35].

Let is the sum $\alpha(I)$ (the least upper bound) of all α-partitions $\alpha_i(x)$ such that $x \in I$ and A_i is a component FSM which is constructed in accordance with some partition from $G(\alpha(I))$ than behavior of A_i does not depend on all prime inputs of network from I if and only if $\pi \geq \alpha(I)$.

3 STATE ENCODING FOR A LOW POWER FSM

3.1 Introduction

In the "Future of Logic Synthesis and Verification" [27] Brayton predicts that in future most sequential synthesis methods will not be used for two reasons. First reason, in his opinion, is that only relatively small designs can be handled. Second, sequential synthesis will not be used because it is hard to verify if the changed design matches the original. He writes in one of his challenge: "it is known how to drive, for a node in a network of interacting FSMs, the set of all permissible behaviours that can be placed at the node without changing the functionality of the network. However, that is very expensive computation in additions; the method gives all possible solutions, and the next task to drive a good one. This is also an expensive computation. In many situations, there already exists a (particular) solution. Thus instead of finding all possible solutions, operate directly on the particular one to find a good one. This may circumvent both of the computationally expensive tasks mentioned above…"

Despite of such pessimistic prediction this work tries to defence sequential synthesis methods targeting the reduction of power dissipation. Power can be minimized by appropriate synthesis of logic. The goal in this case is to minimize the switched capacitance of the circuit by low power driven logic minimization techniques [1].

State Encoding is one of the traditional techniques for sequential logic synthesis for low power. Synthesis of sequential circuits for low power is an area of research that promises to result in large power savings. The step that translates a representation where some variables are symbolic into one where they are all binary-valued is called encoding. An encoding must at least be correct, which means that the encoded representation must behave as the symbolic representation (usually an encoding must establish an injection from symbols to codes); but more interestingly, it is often required that the encoded implementation satisfies some further condition or optimality criteria.

Encoding plays an important role in determining the structure and complexity of the resulting FSM in terms of the number of nodes required to implement the output and next logic. Encoding also affects the switching activity of the state variables and hence the internal signals in the circuit [66].

Classically, the problem of the encoding of an FSM which arises during the design of controllers is formulated as that of obtaining a binary code for each state of the FSM so that given design criteria are optimized. Typical design constrains are the construction if the circuit with a minimal amount of logic or that the circuit can be easily testable, or it consumes low power [75]. Recently reduced power consumption has become a critical design parameter because of several well known reasons [57] and synthesis algorithms as well as design techniques targeted towards low power have been developed at different levels [14].

Concerning encoding aiming low power, research was targeted to reduce switching activity in the state registers. Codes should be given such that the Hamming distance of the codes of those states with a high transition probability of a transition between them is minimized. Reducing the switching activity of state bits contributes to reduce the dynamic power dissipation of the state register and in CMOS circuits the major contributor to power consumption is dynamic power.

The dynamic power dissipation in the combinational part of the circuit is very difficult to estimate, even after the state encoding is determined [55]. At the beginning there are already several different realizations to choose from, depending on what kind of technology will be used. Later, when the gate level implementation is known, the exact computation of the dynamic power dissipation including glitches is often intractable, since it requires the examination of all possible pairs of input patterns of the combinational logic.

In CMOS circuits, power is consumed during charging and discharging of the load capacitances. Average power dissipation is proportional to the average switching activity [6]. A good approximation of the average switching activity is the switching probability. In order to estimate the power consumption the signal and transition probabilities are calculated [49], [66].

These probabilities depend on the input patterns, the delay model, and the circuit structure. Given the input switching probability, it is possible to calculate the probability of the state transitions in an FSM.

Power and switching activity estimation for sequential circuits are significantly more difficult, because the probability of the circuit being in any of its possible states has to be computed. Given a circuit with n flip-flops there are 2^n possible states. At any given instant, the probability that the circuit is in a particular state can be distinct across all the states.

To compute the exact state probabilities of the machine we use the Chapmann Kolmogorov equations for discrete-time Markov Chains [41]. The method requires the solution of a linear system of equations of size 2^n, where n is the number of flip-flops in a machine.

The main idea of this chapter is to present an approach to optimize the state encoding for low power embedded controllers, given the probabilistic model of the FSM. If we press for a general solution, we need to find a method that does not assume a particular STG structure and is not heavily constrained on the number of state variables to use. Thus we use the probabilistic model of an FSM to obtain state assignments that minimize the average number of signal transitions on the state lines for a general STG.

3.2 A new state encoding technique

3.2.1 Problem statement

In this section we introduce a new FSM state encoding technique. The problem of FSM state encoding can be formulated by the following way. For the given FSM=$\{I, S, O, \delta, \lambda\}$ an encoding is a process of assigning to each value of a symbolic variable $\{S\}$ a unique combination of values of a set of logic variables defined on $\{0,1\}^k$ [14].

Given the set of symbols $S=(s_1, s_2, \ldots, s_n)$ for an FSM a ***state encoding*** is given by an integer k and an injective functions e: $S \rightarrow \{0,1\}^k$.

Encoding Matrix

The codes of the symbols are represented by a Boolean ***state encoding matrix*** $E \in B^{|S| \times k}$ with n rows presenting state codes and k columns corresponding to state variables, the umber of which is the unknown of the problem. Each row of E is the encoding of a symbol.

An important degree of freedom to be exploited during the state encoding of an FSM is the choice of the number of state variables, k. To have enough codes, it is necessary that $\lceil \log_2 k \rceil \le k \le |S|$, where $|S|$ is the cardinality of S.

The classical combinatorial problem is search for the FSM encoding with minimum code length $k \rightarrow min$. The strategy of most power-driven state encoding algorithms consists of introducing the minimum possible number of state bits k to minimize the corresponding number of registers [14].

As an example, for the FSM "bbtas" from [48] an arbitrary state encoding matrix E is presented in the Table 3-1.

Table 3-1 Arbitrary state encoding matrix for the FSM "bbtas"

states	codes
*st*0	001
*st*1	011
*st*2	111
*st*3	110
*st*4	100
*st*5	000

STG Representation

FSM state encoding problem can be described as a process of embedding codes of states of an FSM into a complete Boolean graph.

An ***undirected graph*** $G(V, E)$ is a pair (V, E), where $V=\{v_1, v_2, \ldots, v_n\}$ is a set of vertices and $E=\{e_1, e_2, \ldots, e_m\}$ is a set of edges (unordered pairs from V). It presents a binary relation on V.

For the FSM "bbtas" the undirected graph is presented on the Figure 3-1.

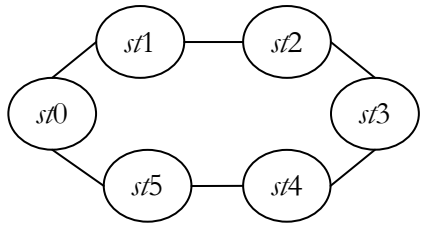

Figure 3-1 Undirected graph for the FSM "bbtas"

A symmetric Boolean graph where vertices correspond to Boolean space elements and edges connect adjacent vertices is a ***complete Boolean graph*** [92].

On the Figure 3-2 the complete Boolean graph for 3-dimension Boolean space is depicted.

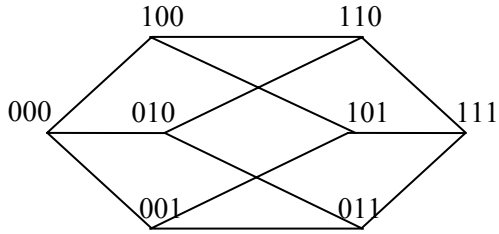

Figure 3-2 Complete Boolean graph for 3-dimension Boolean space

Matrix Representation

Below we present a matrix description of an FSM given by STG [19], [93].

The ***incidence matrix*** of an undirected graph G is an $n \times m$ matrix $[b_{ij}]$ where n and m are the number of vertices and edges respectively, such that $a_{ij}=1$ if the vertex v_i and edge e_{ij} are incident and 0 otherwise.

The incidence matrix is related to the adjacency matrix of a graph.

The ***adjacency matrix*** for a finite undirected graph G on n vertices is an $n \times n$ Boolean matrix where the entry $a_{ij}=1$ if and only if there exists an edge joining vertex i and vertex j. The adjacency matrix of an undirected graph is symmetric.

The *degree*, $d_G(v_i)$ of a vertex v_i in a graph G is the number of edges incident to v_i.

The matrix description: the incidence matrix and adjacency matrix corresponding to the FSM "bbtas" are presented in the Table 3-2.

Table 3-2 Adjacency and incidence matrices for the FSM "bbtas"

$v_i \backslash v_i$	*st*0	*st*1	*st*3	*st*4	*st*5	$d(v_i)$
*st*0		1			1	2
*st*1	1		1			2
*st*2		1		1		2
*st*3			1		1	2
*st*4				1		2
*st*5	1				1	2

$v_i \backslash e_i$	0 1	1 2	2 3	3 4	4 5	0 5
*st*0	1					1
*st*1	1	1				
*st*2		1	1			
*st*3			1	1		
*st*4				1	1	
*st*5					1	1

The set of *neighbors*, called a **(**open**)** neighborhood $N_G(v_i)$ for a vertex v_i in a graph G, consists of all vertices adjacent to v_i but not including v_i. When v_i is also included, it is called a closed neighborhood, denoted by $N_G[v_i]$.

Optimization & Cost Functions

An encoding is a typical logic synthesis procedure which includes FSM restructuring to obtain a logic description that can be mapped optimally into a target technology. Often optimization is done first independent from technology. Optimization depends not only on the target technology, but also on the cost functions: besides area, speed and power consumption are growing importance [41].

The state encoding problem is an optimization problem whose solution can be measured in terms of a cost function and such that the cost functions attains a minimum value.

In general, in the coding theory [75] the following three properties of a code are analyzed: code length, total number of valid codes and the minimum Hamming distance between two adjacent codes. We consider an encoding cost function based on the third parameter, the minimum Hamming distance between two adjacent codes c_i and c_j:

An ***encoding cost function*** O is $O = \sum_{i,j} H(c_i, c_j)$.

The *Hamming distance* is the number of positions in two strings of equal length for which the corresponding elements are different. Put another way, it measures the number of substitutions required to change one for the other.

Attention has also been paid for FSM state encoding for low power. A power-oriented cost function accounts for the minimization of the number of logic transitions of the state registers between two successive clock cycles, assuming the power consumption is proportional to the switching activity on the state bit lines of the machine [66]. The minimization of the register transitions has to be combined with an appropriate implementation of the combinational logic for obtaining a global power saving, so the state encoding can be the starting point for further power optimization of the combinational part.

The cost function consider the sum of the Hamming distances $H(c_i,c_j)$ between the codes c_i, c_j being assigned to all pairs of states s_i, s_j among which a transition can occur [42].

A ***power-oriented cost function*** O_{power} is $O_{power} = \sum_{i,j} w_{i,j} \cdot H(c_i, c_j)$,

where $w_{i,j}$ is a weight of transition between states s_i and s_j.

Graph Weighing

To compute $w_{i,j}$ we use the probabilistic model of an FSM described below.

Given the FSM description and the input probabilities, we estimate the transition probabilities for each edge in the STG, by modeling the FSM as a Markov chain. The input probability distribution can be obtained by simulating the FSM at a higher level of abstraction in the context of its environment or by direct knowledge from the designer [6].

An FSM is described by an STG defined by a vertex (state) set $S=\{s_1, s_2, .., s_n\}$ and a related directed edge set representing the set of transitions from one state to another. The STG of the FSM "bbtas" with six states and two inputs is presented on the Figure 3-3. The "-"symbol represents don't care entry.

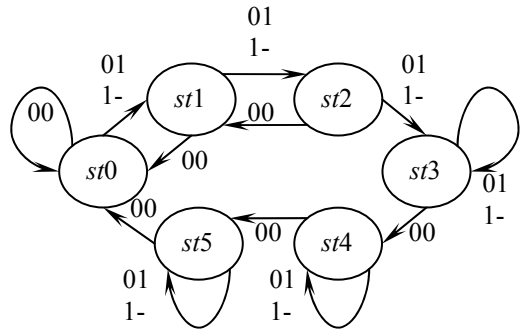

Figure 3-3 Directed STG for the FSM "bbtas"

We use information about probabilities to compute a static probabilistic model of the FSM which will give the transition probabilities for the FSM. We do this by interpreting the STG as a Markov chain. A Markov chain is a representation of a finite Markov process, a stochastic model where the probability distribution at any time depends only on the present state and do not on how the process arrived in that state [45]. The Markov chain model for the STG can be described by a directed graph with a structure isomorphic to the STG and with weighted edges. For a transition from state s_i to state s_j, the weight $p_{i,j}$ on the corresponding edge represents the conditional probability of the transition (i.e., the probability of a transition to state s_j given that the machine was in state s_i). Symbolically this can be expressed as: $p_{i,j}$=prob(next=s_i|present=s_j). The edges with zero conditional probability are never drawn in the graph representation of the Markov chain.

The conditional transition probabilities assuming equiprobable and independent input signals for the FSM "bbtas" are presented on the Figure 3-4.

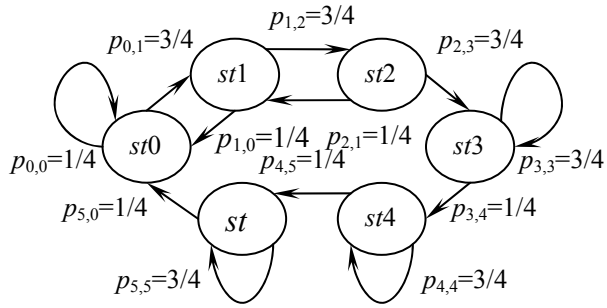

Figure 3-4 Conditional probability distribution for the FSM "bbtas"

The set of values of all condition probabilities is called the conditional probability distribution. The conditional probability distribution is easily found from the input probability distribution and by observing for which input configurations the FSM performs its state transitions [9], [25].

Conditional transition probabilities are used as a rough approximation to the transition probabilities [6]. We need to calculate the probability of a transition taking the present state into account. These probabilities are called total transition probabilities, $P_{i,j}$, and can be calculated from the state probabilities, where the state probability, P_i represents the probability that the machine is in a given state s_i [43]: $P_{i,j}=p_{i,j}\cdot P_i$.

The next step is computation the state probabilities. These values are not time-dependent [6]. This implies that as the observation time increases, the probability that the machine is in each of its states converges to a constant (stationary) set of real numbers. In other words, we receive a steady state probability vector *vect* whose elements are stationary state probabilities.

We do not discuss in the current work such STG's for which the stationary state probabilities do not exist and refer readers for more information to [45].

Let P be the conditional transition probability matrix whose entries $p_{i;j}$ are the conditional transition probabilities, and *vect* the steady state probability vector whose components are the state probabilities P_i. Then we compute the steady state probabilities by solving the system of n+1 Chapman Kolmogorov equations [6], [45]: $vect^T\cdot P=vect^T$, $\sum_{i=1}^{n} P_i = 1$.

The stationary state probabilities calculated solving the system above for the FSM "bbtas" are shown in Figure 3-5. The Figure shows the total transition probabilities (the products $p_{i,j}\cdot P_i$) on the edges. Note that the probabilities for self-loops ($P_{0,0}$=0.029, $P_{3,3}$=0.176, $P_{4,4}$=0.176, $P_{5,5}$=0.176) are not shown only because we are not interested in edge that do not imply any state transition.

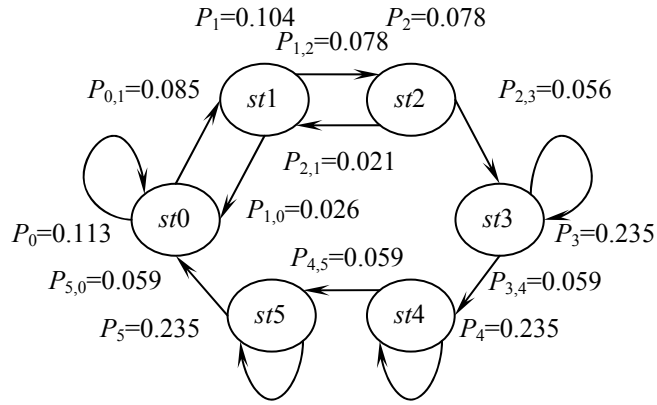

Figure 3-5 State probabilities and total transition probabilities for the FSM "bbtas"

Once the total transition probabilities have been calculated, we transform the original STG into a weighted graph which preserves only the relevant information needed for state encoding. All the unreachable states and self-loops are eliminated from the graph. The STG is transformed into an undirected graph by converting all multiple-directed edges into a single undirected edge.

The weighted STG will be the starting point for the power-oriented state encoding algorithm. For the FSM "bbtas" the weighted STG is shown on the Figure 3-6.

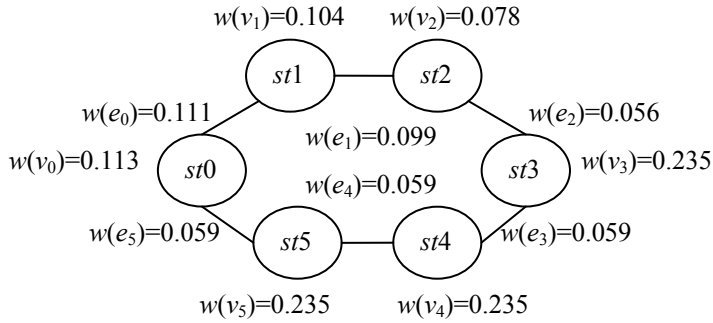

Figure 3-6 Weighted graph for the FSM "bbtas"

3.2.2 Weakly crossed edge cuts encoding algorithm

The main idea of our approach is an economical covering of the set of transitions by weakly crossed edge cuts. To form such edge cuts we construct a set of two blocks partitions called as encoding partitions on the set of states of an FSM. The number of encoding partitions corresponds to the code length. We consider the minimum number of state variables to find a set of distinct codes.

Necessary definitions

Definition 3.1 An *encoding partition* π on S is a collection $\pi = \{B_1(s), B_0(s)\}$ of two disjoint subsets $B_1 \cap B_0 = \varnothing$ of S whose set union is S: $B_1 \cup B_0 = S$ (B_1 is the unit block and B_0 is the zero block).

Note that we mark out two notions of a decision of FSM state encoding problem. The encoding e which has been received during applying of an encoding algorithm is called as *received encoding*. A *perfect encoding* (or ideal encoding) is an encoding such that the sum of Hamming distances between two adjacent states is equal to the number of all transitions of FSM. In other words, the perfect encoding is the encoding where the Hamming distance between two adjacent states is equal to 1.

Definition 3.2 A transition between states s_i and s_j with Hamming distances $H(c_i,c_j)$ between the codes c_i, c_j more than 1 is called *complicated transition*.

Definition 3.3 An additional switching that is necessary to be done to change state variable besides required switching (single state variable change) is called *redundant switching*.

Next we introduce an evaluation to estimate the efficiency of received encoding.

Definition 3.4 A *summary defect of an encoding e* is defined by a number of complicated transitions and the sum of their redundant switching.

Heuristic algorithm

The STG representation of an initial FSM is given: ($G(V, E)$), $V=(v_1, v_2, \ldots, v_n)$ – set of vertices and $E=(e_1, e_2, \ldots, e_m)$ – set of edges.

Preliminary step

- The adjacency and the incidence matrices are constructed.

Example: we illustrate how our encoding algorithm works on the FSM "bbara" [48]. The undirected graph for the FSM "bbara" is depicted on the Figure 3-7. Table 3-3 and Table 3-4 are presented the adjacency and incidence matrices for the considered FSM.

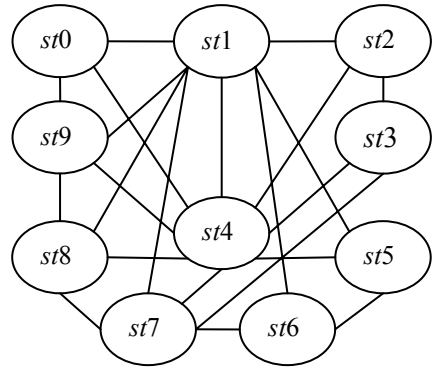

Figure 3-7 Undirected graph for the FSM "bbara"

Table 3-3 Adjacency matrix for the FSM "bbara"

$v_i \backslash v_i$	0	1	2	3	4	5	6	7	8	9	$d(v_i)$
0		1			1					1	3
1	1		1		1	1	1	1	1	1	8
2		1		1	1						3
3			1		1			1			3
4	1	1	1	1		1		1	1	1	8
5		1			1		1				3
6		1				1		1			3
7		1		1	1		1		1		5
8		1			1			1		1	4
9	1	1			1				1		4

Table 3-4 Incidence matrix for the FSM "bbara"

$v_i \backslash e_{ij}$	0-1	0-4	0-9	1-2	1-4	1-5	1-6	1-7	1-8	1-9	2-3	2-4	3-4	3-7	4-5	4-7	4-8	4-9	5-6	6-7	7-8
0	1	1	1																		
1	1			1	1	1	1	1	1	1											
2				1							1	1									
3											1		1	1							
4		1			1							1	1		1	1	1	1			
5						1									1				1		
6							1												1	1	
7								1						1		1				1	1
8									1								1				1

The algorithm constructs a set of encoding partitions. The number of encoding partitions is equal to $k=\lceil log_2 n \rceil$, where n is a number of vertices in STG and k is a code length. Each encoding partition π_r consists of two blocks: the unit and the zero blocks which are represented by variable sets $B^1{}_r$ and $B^0{}_r$ respectively, $1 \le r \le k$.

First Step

- A partition π_Π on the set of states S of the machine is equal to unit partition $\pi_\Pi=\pi_1$ (unit partition is the partition where all elements of S are in one block, $\pi_\Pi=\pi_1=\{s_1, s_2, \ldots, s_n\}$).
- The sets $B^1{}_r$ and $B^0{}_r$ are empty at the beginning, i. e. $B^1{}_0=\varnothing$ and $B^0{}_0=\varnothing$.
- A vertex v_i with maximal $d(v_i)$ from the adjacency matrix is selected as a starting point.
- Selected vertex v_i is placed in the set $B^1{}_1=\{v_i\}$.

The procedure of calculation the edge cut weight increment $\gamma^1(B^1_r, v_c)$ by adding vertex v_c to the set B^1_r

For each undistributed vertices v_c this increment of the set $B^1{}_r$ is calculated by the following way:

$$\gamma^1(B^1{}_r,v_c)=d(v_c)-2|N'(v_c)| \qquad (3)$$

where $d(v_c)$ is the degree of the vertex v_c and $N'(v_c)$ is the set of neighbors for the vertex v_c, which consists of all vertices from the set $B^1{}_r$ adjacent to v_c. From the degree of the vertex v_c we subtract the double number of its neighbors.

A cardinality of an edge cut is the number of edges which has been cut. The increment $\gamma^1(B^1{}_r,v_c)$ allow observing on the cardinality of constructed edge cut.

By using the adjacency matrix the first vertex v_c with the minimal increment is selected. The selected vertex v_c is added to the set $B^1{}_r$, so it becomes equal to $\{v_i, v_c\}$.

Usually the procedure of calculation of the increment by adding vertex v_c to the set $B^1{}_r$, choosing the best vertex and extending the set $B^1{}_r$ repeats until a cardinality of $B^1{}_r$ reaches approximately the half of a cardinality of V. However, as it was mentioned above, we observe on the cardinality of constructed edge cut with aim to minimize it. If adding of the next vertex increases a cardinality of constructed edge we do not add this vertex to the considered set.

The vertex v_c in the set $B^1{}_r$ is also checked on a nearness to the set $B^0{}_r$ in case if this set is not empty: $\gamma^0(B^0{}_r,v_c)=d(v_c)-2|N''(v_c)|$, is the increment by adding vertex v_c to the set $B^0{}_r$, $d(v_c)$ is the degree of the vertex v_c and $N''(v_c)$ is the set of neighbors for the vertex v_c, which consists of all vertices from the set $B^0{}_r$ adjacent to v_c.

If the vertex v_c has the same increment $\gamma^1(B^1{}_r,v_c)=\gamma^0(B^0{}_r,v_c)$ for both sets $B^1{}_r$ and $B^0{}_r$, we place this vertex to such set $B^1{}_r$ or $B^0{}_r$ that minimizes the cardinality of constructed edge cut.

An encoding partition π_r has the unit block equal to the set $B^1{}_r$ and the zero block equal to the set $B^0{}_r$. From the incidence matrix all rows which correspond to the vertices of the set $B^0{}_r$ are deleted. The rest rows (rows which correspond to the vertices of the set $B^1{}_r$) are summed component-wise by modulo two. The set of edges marked with 1 in the resulting Boolean vector form the edge cut of the encoding partition π_r.

Example, $\pi_\Pi=\pi_1=\{st0, st1, st2, st3, st4, st5, st6, st7, st8, st9\}$; $B^1{}_1=B^0{}_1=\varnothing$; selected vertex v_1, $d(v_1)=8$; $B^1{}_1=\{v_1\}$, |EC| – the cardinality of the edge cut.

Using (3) vertex v_0 with minimal increment, 1 (column $\gamma^1{}_1$ in Table 3-5: $d(v_0)=3$, $N'(v_0)=1$ and hence $\gamma^1{}_1(v_0)=3-2\cdot1=1$) is selected; $B^1{}_1=\{v_1,v_0\}$.

Then we calculate the increment to the set $B^1{}_1$ for all rest vertices. Next we select the vertex v_9 with the increment $\gamma^1{}_0(v_9)=4-2\cdot2=0$, $B^1{}_1=\{v_1,v_0,v_9\}$ and then the vertex v_8 with the increment $\gamma^1{}_9(v_8)=4-2\cdot2=0$, $B^1{}_1=\{v_1,v_0,v_9,v_8\}$. The next appropriate vertex to add to the set $B^1{}_1$ is vertex v_6 with the increment $\gamma^1{}_8(v_6)=3-2\cdot2=-1$, but we need to check this vertex on the increment to the set $B^0{}_1=\{v_2,v_3,v_4,v_5,v_7\}$: $\gamma^0{}_6(v_6)=3-2\cdot2=-1$. The cardinality of the set $B^1{}_1$ is equal to $|B^1{}_1|=9$, if we add the vertex v_6 to the set $B^1{}_1$ the cardinality of the set $B^1{}_1=\{v_1,v_0,v_9,v_8,v_6\}$ will be equal to $|B^1{}_1|=10$. Thus, we do not add the vertex v_6 to the set $B^1{}_1$.

$\boldsymbol{\pi_1=\{\{st0,st1,st8,st9\};\{st2,st3,st4,st5,st6,st7\}\}}$

Table 3-5 Construction of the first encoding partition for the FSM "bbara"

$v_i\backslash v_i$	0	**1**	2	3	4	5	6	7	8	9	$d(v_i)$	$\gamma^1{}_1$	$\gamma^1{}_0$	$\gamma^1{}_9$	$\gamma^1{}_8$	$\gamma^0{}_6$
0		**1**			1					1	3	**1**				
2		**1**		1	1						3	1	1	1		
3			1		1				1		3	3	3	3		
4	1	**1**	1	1		1		1	1	1	8	6	4	2		
5		**1**			1		1				3	1	1	1		
6		**1**				1			1		3	1	1	1	**-1**	-1
7		**1**		1	1		1	1			5	3	3	3		
8		**1**			1				1	1	4	2	2	**0**		
9	1	**1**			1			1		0	4	2	**0**			
$\boldsymbol{B^1{}_1}$											**1**	**0**	**9**	**8**		
$\boldsymbol{B^0{}_1}$														**2,3,4,5,7**	**6**	
|EC|											8	9	9	9		

Table 3-6 First edge cut for the FSM "bbara"

$v_i\backslash e_{i,j}$	01	04	09	12	14	15	16	17	18	19	23	24	34	37	45	47	48	49	56	67	78	89
1	1			1	1	1	1	1	1	1												
0	1	1	1																			
9			1							1								1				1
8									1								1				1	1
		1		**1**	**1**	**1**	**1**	**1**									**1**	**1**			**1**	

Next Steps

- The construction of the next encoding partition, $r=r+1$.
- The partition π_Π is equal to $\pi_\Pi=\pi_\Pi\times\pi_r$.
- Select an arbitrary edge (v_i, v_j) from the previous edge cut.
- Set $B^1{}_k=\{v_i, v_j\}$.
- Repeat until $r \le k$.

Example, construction of the second encoding partition, $r=2$; $\pi_\Pi=\{\{st0, st1, st8, st9\};\{st2, st3, st4, st5, st6, st7\}\}$; an arbitrary edge from the first edge cut (v_7,v_8); $B^1{}_2=\{v_7,v_8\}$, Table 3-7.

The vertices v_3 and v_6 have the same increment $\gamma^1{}_{7,8}(v_3/v_6)=3-2\cdot1=1$, the vertex v_3 is added to the set $B^1{}_2=\{v_7,v_8,v_3\}$. The column $\gamma^1{}_3$ demonstrates the increment of all rest vertices to the set $B^1{}_2$. The vertices v_2 and v_6 have also the same increment $\gamma^1{}_3(v_2/v_6)=3-2\cdot1=1$, vertices v_6 is added to the set $B^1{}_2=\{v_7,v_8,v_3,v_6\}$. $B^0{}_2=\{v_2,v_4,v_5\}$ because the second block of partition π_Π is blocked. $\gamma^1{}_6(v_1)=8-2\cdot3=2$ and $\gamma^1{}_6(v_9)=4-2\cdot1=2$ we check these vertices on the increment to the set $B^0{}_2=\{v_2,v_4,v_5\}$: $\gamma^0{}_1(v_1)=8-2\cdot3=2$ and $\gamma^0{}_1(v_9)=4-2\cdot1=2$. The vertex v_1 is added to the set $B^1{}_2=\{v_7,v_8,v_3,v_6,v_1\}$.

π_2*=*{{st1,st3,st6,st7,st8};{st0,st2,st4,st5,st9}}

Table 3-7 Construction of the second encoding partition for the FSM "bbara"

$v_i\backslash v_i$	0	1	2	3	4	5	6	**7**	**8**	9	$d(v_i)$	$\gamma^1{}_{7,8}$	$\gamma^1{}_3$	$\gamma^1{}_6$	$\gamma^0{}_1$
0		1			1					1	3	3	3	3	
1	1		1		1	1	1	**1**	**1**	1	8	4	4	**2**	2
2		1		1	1						3	3	1		
3			1		1			**1**			3	**1**			
4	1	1	1	1		1		**1**	**1**	1	8	4	2		
5		1			1		1				3	3	3		
6		1				1		**1**			3	1	**1**		
9	1	1			1				**1**		4	2	2	2	2
B^1_2											**7,8**	**3**	**6**	**1**	
B^0_2													**2,4,5**	**0,9**	
|EC|											7	8	9	11	

Table 3-8 Second edge cut of the FSM "bbara"

$v_i\backslash e_{i,j}$	01	04	09	12	14	15	16	17	18	19	23	24	34	37	45	47	48	49	56	67	78	89
7								1						1		1				1	1	
8									1								1				1	1
3											1		1	1								
6							1												1	1		
1	1			1	1	1	1	1	1	1												
	1			**1**	**1**	**1**				**1**	**1**		**1**			**1**	**1**		**1**			**1**

Example, construction of the third encoding partition, r=3; π_Π={{*st*0, *st*9};{*st*1, *st*8};{*st*2, *st*4, *st*5};{*st*3, *st*6, *st*7}}; an edge from the second edge cut is (v_0,v_1); $B^1{}_3$={v_0,v_1}. The first two blocks of the partition π_Π are blocked, then $B^0{}_3$={v_8,v_9}. The vertices v_1 and v_5 have the same minimal increment $\gamma^1{}_{0,1}(v_1/v_5)$=3-2·1=1, we select the vertex v_5 and $B^1{}_3$={v_0,v_1,v_5}, and $B^0{}_3$={v_8,v_9,v_2,v_4} since the third block of the product partition is blocked. Finally we add the vertex v_6 with minimal increment $\gamma^1{}_5(v_6)$=3-2·2=-1 to the set $B^1{}_3$={v_0,v_1,v_5,v_6}, Table 3-9 and Table 3-10.

π_3={{*st*0,*st*1,*st*5,*st*6};{*st*2,*st*4,*st*3,*st*7,*st*8,*st*9}}

Table 3-9 Construction of the third encoding partition for the FSM "bbara"

$v_i \backslash v_i$	**0**	**1**	2	3	4	5	6	7	8	9	$d(v)$	$\gamma^1{}_{0,1}$	$\gamma^0{}_5$	$\gamma^1{}_5$
2		**1**		1	1						3	1		1
3			1		1				1		3	3		3
4	**1**	**1**	1	1		1		1	1	1	8	4		2
5		**1**			1		1				3	**1**	3	
6		**1**				1			1		3	1		**-1**
7		**1**		1	1		1	1			5	3		3
8		**1**			1				1	1	4			
9	**1**	**1**			1			1			4			
$\boldsymbol{B^1{}_3}$											**0,1**	**5**		**6**
$\boldsymbol{B^0{}_3}$											**8,9**	**2,4**		**3,7**
|EC|											9	10		9

Table 3-10 Third edge cut for the FSM "bbara"

$v_i \backslash e_{i,j}$	01	04	09	12	14	15	16	17	18	19	23	24	34	37	45	47	48	49	56	67	78	89
0	1	1	1																			
1	1			1	1	1	1	1	1	1												
5						1									1				1			
6							1												1	1		
		1	**1**	**1**	**1**			**1**	**1**	**1**					**1**					**1**		

Example, construction of the forth encoding partition, r=4; π_Π={{*st*0};{*st*1};{*st*2, *st*4};{*st*3, *st*7};{*st*5};{*st*6};{*st*8}; {*st*9}}; an edge from the third edge cut (v_0,v_4); $B^1{}_4$={v_0,v_4}; $B^0{}_4$={v_0,v_4}, Table 3-11 and Table 3-12.

π_4={{*st*0, *st*1, *st*4, *st*5, *st*6, *st*7, *st*8, *st*9};{*st*2, *st*3}}

Table 3-11 Construction of the fourth encoding partition for the FSM "bbara"

$v_i \backslash v_i$	**0**	1	2	3	**4**	5	6	7	8	9	$d(v)$	$\gamma^1{}_{0,4}$	$\gamma^1{}_9$	$\gamma^1{}_8$	$\gamma^1{}_1$	$\gamma^1{}_5$	$\gamma^1{}_6$
1	**1**		1		**1**	1	1	1	1	1	8	4	2	**0**			
2		1		1	**1**						3						
3			1		**1**			1			3	1	1	1	1	1	1
5		1			**1**		1				3	1	1	1	**-1**		
6		1				1		1			3	3	3	3	1	**-1**	
7		1		1	**1**		1		1		5	3	3	1	-1	-1	**-3**
8		1			**1**			1		1	4	2	**0**				
9	**1**	1			**1**				1		4	**0**					
$\boldsymbol{B^1{}_4}$											**0,4**	**9**	**8**	**1**	**5**	**6**	**7**
$\boldsymbol{B^0{}_4}$											**2**						**3**
|EC|											9	9	9	9	8	7	4

Table 3-12 Fourth edge cut for the FSM "bbara"

$v_i \backslash e_{i,j}$	01	04	09	12	14	15	16	17	18	19	23	24	34	37	45	47	48	49	56	67	78	89
0	1	1	1																			
4		1			1							1	1		1	1	1	1				
9			1							1								1				1
8									1								1				1	1
1	1			1	1	1	1	1	1	1												
5						1									1				1			
6							1												1	1		
7								1						1		1				1	1	
				1								**1**	**1**	**1**								

Concluding Step

- Check if the partition π_Π is equal to the zero partition: $\pi_\Pi=\pi_1\times\pi_2\times\ldots\times\pi_r=\pi_0$ (zero partition is partition where each symbol of S is in one block: π_0={{s_1}, {s_2}, …, {s_n}})
- Calculate the number of complicated transitions, the number of redundant switching and the summary defect of encoding.

Example, π_{Π}={{*st*0}, {*st*1}, {*st*2}, {*st*3}, {*st*4}, {*st*5}, {*st*6}, {*st*7}, {*st*8}, {*st*9}}=π_0. The encoding matrix for the FSM "bbara" is presented in Table 3-13.

Table 3-13 Encoding matrix for the FSM "bbara"

states	codes
*st*0	0100
*st*1	0000
*st*2	1111
*st*3	1011
*st*4	1110
*st*5	1100
*st*6	1000
*st*7	1010
*st*8	0010
*st*9	0110

Table 3-14 Hamming distance of all edge cuts for the FSM "bbara"

01	04	09	12	14	15	16	17	18	19	23	24	34	37	45	47	48	49	56	67	78	89
	1		1	1	1	1	1									1	1			1	
1			1	1	1				1	1		1			1	1		1			1
	1	1	1	1			1	1	1					1					1		
			1								1	1	1								
1	**2**	**1**	**4**	**3**	**2**	**1**	**2**	**1**	**2**	**1**	**1**	**2**	**1**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**

Number of complicated transitions is equal to 8; number of redundant switching is equal to11; $O(e_{perfect})$=22, $O(e_{received})$=33 and hence the defect of the encoding 50%, Table 3-14.

Power-oriented algorithm

The STG representation of an FSM is given: ($G(V, E)$), $V=(v_1, v_2, \ldots, v_n)$ – set of vertices and $E=(e_1, e_2, \ldots, e_m)$ – set of edges.

Preliminary step

- The weights of vertices and the weights of edges are calculated. The set of vertices and the set of edges are resorted by decreasing order.
- The adjacency and incidence matrices are constructed.

Example, the set of vertices and the set of edges of FSM "bbara" are putted in decreasing order by the weights; and the adjacency and incidence matrices are presented in Table 3-15 and Table 3-16.

Table 3-15 Adjacency matrix for the FSM "bbara"

	$w(v_i)$	0.267	0.197	0.155	0.134	0.134	0.049	0.037	0.016	0.009	0.002	
$w(v_i)$	$v_i \backslash v_i$	1	4	0	2	3	5	7	6	8	9	$d(v_i)$
0.267	1		1	1	1		1	1	1	1	1	8
0.197	4	1		1	1	1	1	1		1	1	8
0.155	0	1	1								1	3
0.134	2	1	1			1						3
0.134	3		1		1			1				3
0.049	5	1	1						1			3
0.037	7	1	1			1			1	1		5
0.016	6	1					1	1				3
0.009	8	1	1					1			1	4
0.002	9	1	1	1						1		4

Table 3-16 Incidence matrix for the FSM "bbara"

$v_i \backslash e_{ij}$	1-2	1-4	0-1	0-4	2-3	4-5	2-4	3-4	3-7	1-5	1-7	5-6	1-6	4-7	7-8	1-8	6-7	4-8	8-9	1-9	0-9	4-9
1	1	1	1							1	1		1			1				1		
4		1		1		1	1	1						1				1				1
0			1	1																	1	
2	1				1		1															
3					1			1	1													
5						1				1		1										
7									1		1			1	1		1					
6												1	1				1					
8															1	1		1	1			

The algorithm constructs a set of encoding partitions. The number of encoding partitions is equal to $k=\lceil log_2 n \rceil$, where n is a number of vertices in STG and k is a code length. Each encoding partition π_r consists of two blocks: the unit and the zero blocks which are represented by variable sets $B^1{}_r$ and $B^0{}_r$ respectively, $1 \le r \le k$.

First Step

- A partition π_Π on the set of states S of the machine is equal to unit partition $\pi_\Pi=\pi_1$.
- The sets $B^1{}_r$ and $B^0{}_r$ are empty at the beginning, i. e. $B^1{}_0=\varnothing$ and $B^0{}_0=\varnothing$.
- The first edge (v_i, v_j) from the incidence matrix is selected as a starting point.
- Vertices of the selected edge (v_i, v_j) are placed in the set $B^1{}_1$, so the latter takes value $\{v_i, v_j\}$.

The procedure of calculation the edge cut weight increment $\gamma^1(B^1_r, v_c)$ by adding vertex v_c to the set B^1_r

For each undistributed vertices v_c this increment of the set $B^1{}_r$ is calculated by the following way:

$$\gamma^1(B^1{}_r,v_c)=d(v_c)-2|N'(v_c)| \quad (4)$$

where $d(v_c)$ is the degree of the vertex v_c and $N'(v_c)$ is the set of neighbors for the vertex v_c, consists of all vertices from the set $B^1{}_r$ adjacent to v_c but not including v_c. From the degree of the vertex v_c we subtract the double value of its neighbors because

A cardinality of an edge cut is the number of edges which has been cut. The increment $\gamma^1(B^1{}_r,v_c)$ allow observing on the cardinality of constructed edge cut.

By using the adjacency matrix the first vertex v_c with the minimal increment is selected. The selected vertex v_c is added to the set $B^1{}_r$, so it becomes equal to $\{v_i, v_j, v_c\}$.

Usually the procedure of calculation of the increment by adding vertex v_c to the set $B^1{}_r$, choosing the best vertex and extending the set $B^1{}_r$ repeats until a cardinality of $B^1{}_r$ reaches approximately the half of a cardinality of V. However, as it was mentioned above, we observe on the cardinality of constructed edge cut with aim to minimize it. If adding of the next vertex increases a cardinality of constructed edge, we do not add this vertex to the considered set.

The vertex v_c in the set $B^1{}_r$ is also checked on a nearness to the set $B^0{}_r$ in case if this set is not empty: $\gamma^0(B^0{}_r,v_c)=d(v_c)-2|N''(v_c)|$, is the increment by adding vertex v_c to the set $B^0{}_r$, $d(v_c)$ is the degree of the vertex v_c and $N''(v_c)$ is the set of neighbors for the vertex v_c, consists of all vertices from the set $B^0{}_r$ adjacent to v_c but not including v_c.

If the vertex v_c has the same increment $\gamma^1(B^1{}_r,v_c)=\gamma^0(B^0{}_r,v_c)$ for both sets $B^1{}_r$ and $B^0{}_r$, we place this vertex to such set $B^1{}_r$ or $B^0{}_r$ that minimizes the

cardinality of a constructed edge cut. In case of for some vertex v_c we have the same increment and the adding of this vertex do not change the cardinality of a constructed edge we do not increase the set B^1_r.

An encoding partition π_r has the unit block equal to the set B^1_r and the zero block equal to the set B^0_r. From the incidence matrix all rows which correspond to the vertices of the set B^0_r are deleted. The rest rows (rows which correspond to the vertices of the set B^1_r) are summed component-wise by modulo two. The set of edges marked with 1 in the resulting Boolean vector form the edge cut of an encoding partition π_r.

Example, $\pi_\Pi=\pi_1=\{\{st0\}, \{st1\}, \{st2\}, \{st3\}, \{st4\}, \{st5\}, \{st6\}, \{st7\}, \{st8\}, \{st9\}\}$; $B^1{}_1=B^0{}_1=\varnothing$; the selected edge (v_1,v_2); $B^1{}_1=\{v_1,v_2\}$.

Vertex v_0 is the first vertex with minimal increment, 1 (column $\gamma^1{}_{1,2}$ in Table 3-17, $\gamma^1{}_{1,2}(v_0)=3-2\cdot1=1$, (4)); $B^1{}_1=\{v_1,v_2,v_0\}$. Then we calculate the increment of the set $B^1{}_1$ for all rest vertices. Thus vertex v_9 combines into two edges $(v_9;v_1)$ and $(v_9;v_0)$ with respect to the set $B^1{}_1=\{v_1,v_2,v_0\}$ and the minimal number of increment for vertex v_9 is $\gamma^1{}_0=4-2\cdot2=0$. We stop of forming the set $B^1{}_1=\{v_1,v_2,v_0,v_9\}$ on the vertex v_9, because for the next vertex v_4 we have the same increment to both sets $\gamma^1{}_4=\gamma^0{}_4$. Moreover, the cardinality of the edge cut is equal to 10, see Table 3-18, if we add the last vertex v_4, $B^1{}_1=\{v_1,v_2,v_0,v_9,v_4\}$ the cardinality of the edge cut is also would be equal to 10, so we do not add the vertex v_4 to the set $B^1{}_1$.

π_1={{*st*0, *st*1, *st*2, *st*9};{*st*3, *st*4, *st*5, *st*6, *st*7, *st*8}}

Table 3-17 Construction of the first encoding partition for the FSM "bbara"

$v_i \backslash v_i$	**1**	4	0	**2**	3	5	7	6	8	9	$d(v_i)$	$\gamma^1{}_{1,2}$	$\gamma^1{}_0$	$\gamma^1{}_4$	$\gamma^0{}_4$
4	**1**		1	**1**	1	1	1		1	1	8	4	2	0	0
0	**1**	1								1	3	1			
3		1		**1**			1				3	1	1	1	
5	**1**	1						1			3	1	1	1	
7	**1**	1			1			1	1		5	3	3	3	
6	**1**					1	1				3	1	1	1	
8	**1**	1					1			1	4	2	2	0	
9	**1**	1	1						1		4	2	0		
$B^1{}_1$											**1,2**	**0**	**9**		
$B^0{}_1$													**3,5,6,7,8**		**4**
|EC|											9	10	10	10	

Table 3-18 First edge cut for the FSM "bbara"

$v_i \backslash e_i$	12	14	01	04	23	45	17	56	19	16	47	78	18	67	24	34	37	48	89	15	09	49
1	1	1	1				1		1	1			1							1		
2	1				1										1							
0			1	1																	1	
9									1										1		1	1
		1		**1**	**1**		**1**			**1**			**1**		**1**				**1**	**1**		**1**

Next Steps

- The construction of the next encoding partition, $r=r+1$.
- The partition is equal to $\pi_\Pi=\pi_\Pi\times\pi_r$.
- Select the first edge (v_i, v_j) from the previous edge cut.
- Set $B^1{}_k=\{v_i, v_j\}$.
- Repeat until $r \le k$.

Example, construction of the second encoding partition, $r=2$; $\pi_\Pi=\{\{st0, st1, st2, st9\}; \{st4, st5, st6, st7, st8, st9\}\}$; the first edge from the first edge cut (v_1,v_4); $B^1{}_2=\{v_1,v_2,v_4\}$.

To the set $B^1{}_2=\{v_1,v_2,v_4\}$ the vertices v_0 and v_3 are added because they have the smallest increment, see Table 3-19. For the last vertex we have the increment $\gamma^1{}_5$ to the set $B^1{}_2$: $\gamma^1{}_5(B^1{}_2)$=3-2*2(edges (v_1,v_5) and (v_4,v_5))=-1 and the cardinality of the edge cut is equal to 10, Table 3-19 and Table 3-20.

***π_2*={{*st*0, *st*1, *st*2, *st*3, *st*4, *st*5};{*st*6, *st*7, *st*8, *st*9}}**

Table 3-19 Construction of the second encoding partition for the FSM "bbara"

$v_i\backslash v_i$	**1**	**4**	0	**2**	3	5	7	6	8	9	$d(v)$	$\gamma^1{}_{1,2,4}$	$\gamma^1{}_0$	$\gamma^0{}_0$	$\gamma^1{}_5$	$\gamma^0{}_5$
0	**1**	**1**								1	3	-1				
3		**1**		**1**			1				3	-1	-1	3		
5	**1**	**1**						1			3	-1	-1	3	-1	1
7	**1**	**1**			1			1	1		5	1	1	5		
6	**1**					1	1				3	1	1	3		
8	**1**	**1**					1			1	4	0	0	2		
9	**1**	**1**	1						1		4	0				
$\boldsymbol{B^1{}_2}$											**1,2,4**	**0**	**3**		**5**	
$\boldsymbol{B^0{}_2}$												**9**			**6,7,8**	
|EC|											13	12	11		10	

Table 3-20 Second edge cut of the FSM "bbara"

$v_i\backslash e_i$	12	14	01	04	23	45	81	24	34	37	51	71	56	74	78	67	61	84	89	91	90	94
1	1	1	1				1				1	1					1			1		
2	1				1			1														
4		1		1		1		1	1					1				1				1
0			1	1																	1	
3					1				1	1												
5						1					1		1									
							1			**1**		**1**	**1**	**1**			**1**	**1**		**1**	**1**	**1**

Example, construction of the third encoding partition, r=3; $\pi_{\prod}$={{*st*0, *st*1, *st*2}; {*st*3, *st*4, *st*5}, {*st*6, *st*7, *st*8}, {*st*9}}; the first edge from the first edge cut (v_8,v_1); $B^1{}_3$={v_1,v_2,v_4,v_8}, Table 3-21 and Table 3-22.

π_3={{*st*1,*st*2,*st*3,*st*4,*st*7,*st*8,*st*9};{*st*0,*st*5,*st*6}}

Table 3-21 Construction of the third encoding partition for the FSM "bbara"

$v_i \backslash v_i$	**1**	**4**	0	**2**	3	5	7	6	**8**	9	$d(v)$	$\gamma^1{}_{1,2,4,8}$	$\gamma^1{}_9$	$\gamma^0{}_9$	$\gamma^1{}_7$	$\gamma^1{}_7$
0	**1**	**1**								1	3	-1				
3		**1**		**1**			1				3	-1	-1	3		
5	**1**	**1**						1			3	-1	-1	3		
7	**1**	**1**			1			1	**1**		5	-1	-1	3	-3	3
6	**1**					1	1				3	1	1	1		
9	**1**	**1**	1						**1**		4	-2				
$B^1{}_3$											**1,2,4,8**	**9**	**3**		**7**	
$B^0{}_3$												**0**	**5**		**6**	
|EC|											13	11	10		7	

Table 3-22 Third edge cut for the FSM "bbara"

$v_i \backslash e_i$	12	14	01	04	23	45	81	24	34	37	51	71	56	74	78	67	61	84	89	91	90	94
1	1	1	1				1				1	1					1			1		
2	1				1			1														
4		1		1		1		1	1					1				1				1
8							1								1			1	1			
9																			1	1	1	1
3					1				1	1												
7										1		1		1	1	1						
			1	**1**		**1**					**1**					**1**	**1**				**1**	

Example, forth encoding partition, r=4; $\pi_{\prod}$={{*st*0}, {*st*1, *st*2}, {*st*3, *st*4}, {*st*5}, {*st*6}, {*st*7, *st*8}, {*st*9}}; the first edge from the first edge cut (v_0,v_1); $B^1{}_4$={v_0,v_1}. To the set $B^1{}_4$={v_0,v_1} we subsequently add the vertices v_9, v_8, v_6, and v_4, because they have the smallest increment, Table 3-23. The last undistributed vertex v_5 has the increment to the both sets: $B^1{}_4$={v_0,v_1,v_9,v_8,v_6,v_4} and $B^1{}_4$={v_2,v_7,v_3}, $\gamma^1{}_5(B^1{}_4)$=3-2*1(edge(v_4,v_5))=1 and $\gamma^1{}_5(B^0{}_4)$=3-2*2(edges (v_3,v_5) and (v_3,v_7))=-1. Despite of that the increment of vertex v_5 to the set $B^0{}_4$ is smaller than to the set $B^1{}_4$ we decide to add the vertex v_5 to the set $B^1{}_4$={v_0,v_1,v_9,v_8,v_6,v_4,v_5}, because the cardinality of the edge cut is equal to 7, Table 3-24. If we do not add the vertex v_5 to the set $B^1{}_4$, the cardinality would be equal to 10.

π_4={{*st*0,*st*1,*st*4,*st*5,*st*6,*st*8,*st*9};{*st*2,*st*3,*st*7}}

Table 3-23 Construction of the fourth encoding partition for the FSM "bbara"

$v_i \backslash v_i$	**1**	4	**0**	2	3	5	7	6	8	9	$d(v)$	$\gamma^1_{0,1}$	γ^1_9	γ^1_8	γ^1_4	γ^1_5
4	**1**	1	**1**	1		1		1	1	1	8	4	2	0		
3		1		1			1				3	3	3	3		
5	**1**	1						1			3	1	1	1	-1	
7	**1**	1			1			1	1		5	3	3			
6	**1**					1	1				3	1	1	1	1	-1
8	**1**	1					1			1	4	2	0			
9	**1**	1	**1**						1		4	0				
$\boldsymbol{B^1_4}$											**0,1**	**9**	**8**	**4**	**5**	**6**
$\boldsymbol{B^0_4}$											**2**		**7**	**3**		
|EC|											9	9	9	9	8	7

Table 3-24 Fourth edge cut for the FSM "bbara"

$v_i \backslash e_i$	12	14	01	04	23	45	81	24	34	37	51	71	56	74	78	67	61	84	89	91	90	94
0			1	1																	1	
1	1	1	1				1				1	1					1			1		
9																			1	1	1	1
8							1								1			1	1			
4		1		1		1		1	1					1				1				1
5						1					1		1									
6													1			1	1					
	1							**1**	**1**			**1**		**1**	**1**	**1**						

Concluding Step

- Check if the partition π_Π is equal to the zero partition: $\pi_\Pi=\pi_0$.
- Calculate the number of complicated transitions, the number of redundant switching and the summary defect of the encoding.

Example, π_Π={{*st*0}, {*st*1}, {*st*2}, {*st*3}, {*st*4}, {*st*5}, {*st*6}, {*st*7}, {*st*8}, {*st*9}}=π_0. The encoding matrix for the FSM "bbara" is presented in Table 3-25.

Table 3-25 The encoding matrix for the FSM "bbara"

states	codes
*st*0	1101
*st*1	1111
*st*2	1110
*st*3	0110
*st*4	0111
*st*5	0101
*st*6	0001
*st*7	0010
*st*8	0011
*st*9	1011

Table 3-26 Hamming distances of all edge cuts for the FSM "bbara"

12	14	01	04	23	45	18	24	34	37	15	17	56	47	78	67	16	48	89	19	09	49
	1		1	1		1	1			1	1					1		1			1
						1			1		1	1	1			1	1		1	1	1
		1	1		1					1					1	1				1	
1							1	1			1		1	1	1						
1	**1**	**1**	**2**	**1**	**1**	**2**	**2**	**1**	**1**	**2**	**3**	**1**	**2**	**1**	**2**	**3**	**1**	**1**	**1**	**2**	**2**

Number of complicated transition is equal to 10; number of redundant switching is equal to 12; $O_{power}(e_{perfect})$=0.2215, $O_{power}(e_{received})$=0.2757 and hence the defect of the encoding 24.4%, Table 3-26.

3.2.3 Comparison of encoding methods

The encoding problem is *NP*-complete; therefore most of the proposed state assignment techniques rely on heuristic solutions, which are using problem-solving techniques based on experience [57].

In this sub-section we select five state-of-the-art encoding methods to compare with the proposed encoding. These methods are Huffman-Style Encoding by Surti [69], One Level Tree (OLT) by Silvano [66], POW3 by Benini [6], Spanning Tree Based (STB) by Nőth [55], and Deep First Search (DFS) by Eggermont [21]. All of these approaches have the aim to reduce the dynamic power dissipation in synchronous sequential circuits by reducing the register switching activity. The reduction of the average switching activity targets the minimization of the number of bit changes during the FSM state transitions.

Common to these approaches is the starting data that includes the initial description of an FSM as the STG and a probabilistic model to present the stochastic behavior of an FSM. The initial STG is transformed into the weighted undirected graph. The graph is weighted using the probabilistic description of the FSM. A good approximation of the average switching activity is the switching probability (or transition probability) [6]. Given the input switching probability it is possible to calculate the probability of the state transitions in an FSM. This information is used to find an encoding that minimizes the switching activity of the variables. The Huffman encoding algorithm works with the vertex-weighted undirected graph, while POW3, OLT and STB use the edge-weighted undirected graph.

1. Using **Huffman-Style Encoding** higher probability states have shorter code length while lover probability states have longer code length [69]. Huffman encodings for the FSM "bbara" is presented in Table 3-27.

This approach may not always be practical since variable code lengths have significant management overhead.

Table 3-27 Huffman encoding for the FSM "bbara"

$w(v_i)$	v_i	k_1	k_2	k_3	k_4	k_5	k_6
0.268	*st*1	1	1	-	-	-	-
0.195	*st*4	1	0	-	-	-	-
0.155	*st*0	0	1	0	-	-	-
0.134	*st*2	0	1	1	-	-	-
0.134	*st*3	0	0	1	-	-	-
0.049	*st*5	0	0	0	1	1	-
0.037	*st*7	0	0	0	1	0	-
0.016	*st*6	0	0	0	0	1	-
0.009	*st*8	0	0	0	0	0	1
0.003	*st*9	0	0	0	0	0	0

2. **One Level Tree** encoding uses an edge-weighted undirected graph to state ordering and thus to assign a priority to the symbolic states. When the order is defined the well-known Gray code is used [4]. For example, the edge $e_{1,2}$ has maximal weight, let the state has code 0000, then state *st*2 has code 0001, the next edge ordered by weights is the edge $e_{1,4}$ then the state *st*4 can be encoded as 0010. Next follows the edge $e_{0,1}$ and the state *st*0 has the code 0100 and etc. Table 3-28 presents the OLT encoding for the FSM "bbara".

The problem arises if several edges have the same weight.

Table 3-28 OLT encoding for the FSM "bbara"

e_i	$w(e_i)$	states	codes
1-2	0,668	*st*1	0000
		*st*2	0001
1-4	0,661	*st*4	0010
0-1	0,577	*st*0	0100
2-3	0,268	*st*3	0011
4-5	0,246	*st*5	0110
3-7	0,134	*st*7	1011
5-6	0,049	*st*6	0111
7-8	0,037	*st*8	1010
8-9	0,009	*st*9	1110

3. **POW3** encoding minimizes the Hamming distance between the codes of the states connected by transition with high probability. The encoding approach assigns the adjacent codes for the states correspond to weights of edges, $w(e_i)$. These weights are recomputed in the cost function after each previous variable assignment [6]. For example, the edges $e_{5,6}$, $e_{7,8}$, and $e_{8,9}$ have smallest weights, let k_1=1 for the states *st*6, *st*8 and *st*9, then only two edges $e_{5,6}$ and $e_{7,8}$ have the Hamming distance equals to 1. Next the weights which correspond to selected edge are redoubled. Table 3-29 demonstrates the construction of POW3 encoding for the FSM "bbara".

The solution is not globally optimal because the method is not considering the impact that the choice of one state variable has on the other state bits.

Table 3-29 POW3 encoding for the FSM "bbara"

e_i	$w^1(e_i)$	v_i	$\mathbf{k_1}$	$H(e_i)$	$w^2(e_i)$	$\mathbf{k_2}$	$H(e_i)$	$w^3(e_i)$	$\mathbf{k_3}$	$H(e_i)$	$w^4(e_i)$	$\mathbf{k_4}$
1-2	0.668	*st*1	**0**	0	0.668	**0**	0	0.668	**0**	1	1.336	**1**
		*st*2	**0**			**0**			**1**			**1**
1-4	0,661	*st*4	**0**	0	0,661	**0**	0	0,661	**0**	0	0,661	**0**
0-1	0,577	*st*0	**0**	0	0,577	**1**	1	1.154	**0**	1	2.308	**1**
2-3	0,268	*st*3	**0**	0	0,268	**0**	0	0,268	**1**	0	0,268	**0**
4-5	0,246	*st*5	**0**	0	0,246	**1**	1	0.492	**0**	1	0.984	**0**
3-7	0,134	*st*7	**0**	0	0,134	**1**	1	0.268	**1**	1	0.536	**0**
5-6	0,049	*st*6	**1**	1	0,098	**0**	1	0.196	**1**	3	0.784	**0**
7-8	0,037	*st*8	**1**	1	0,074	**0**	1	0.148	**0**	3	0.592	**0**
8-9	0,009	*st*9	**1**	0	0,009	**1**	0	0,009	**1**	2	0.027	**1**

4. The **Spanning Tree Based** encoding construct a maximum spanning tree of the undirected graph and formulate the state encoding problem as an embedding of the spanning tree into a Boolean hypercube of unknown dimension [55]. In presented approach the limitation to a predetermined number of bits for the state encoding is removed. In case of several edges have the highest weight the division of spanning tree is ambiguous. Figure 3-8 illustrates the maximum weighted spanning tree for the FSM "bbara". The highest weighted edge $e_{1,2}$ is selected, the degree of node v_1 is equal $d(v_1)$=3 and the degree of node v_2 is equal to $d(v_2)$=2, d_{max}=3.

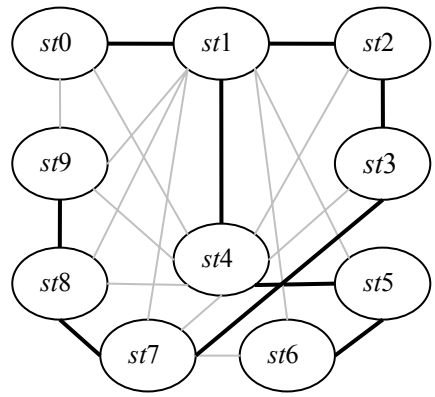

Figure 3-8 Maximum weighted spanning tree for the FSM "bbara"

The edge $e_{1,2}$ is removed and it is leaved two sub-trees, Figure 3-9. When we divide the maximum weighted spanning tree into the two sub-trees we select the highest weighted edge, $e_{1,2}$, which does not increase the degree of the nodes. In case of some edges have the highest weight the division of spanning tree is ambiguous.

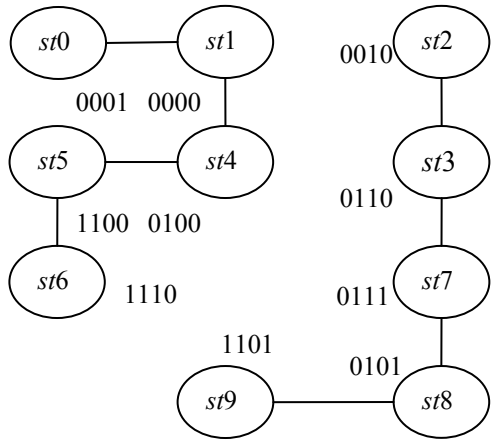

Figure 3-9 Two sub-trees for the FSM "bbara"

Next, adjacent codes are applied to both sub-trees, Figure 3-9 and Table 3-30.

Table 3-30 STB encoding for "bbara" FSM

states	codes
*st*0	0001
*st*1	0000
*st*2	0010
*st*3	0110
*st*4	0100
*st*5	1100
*st*6	1110
*st*7	0111
*st*8	1111
*st*9	1101

5. The **Deep First Search** (DFS) encoding utilizes dynamic loop information extracted from FSM profiling data [21]. First, FSM run under a relevant input data, state profiling collects information about a state register trace. Second, loop detection searches for loops in the state trace. Loops are identified by the repeated occurrence of the same state in the trace, and each discovered loop is stored and counted to obtain the frequency of the loops. Third, to each state assign a unique code. For some circuits proposed algorithms gives more power dissipation due to the larger state register. Hence the code length is bigger than expected. The DFS can not be used for very large circuits. For the FSM "bbara" the loop 1→2→1 has the weight 100, while the loop 1→2→3→4→1 has the weight 50 and the loop 1→4→1 is a (nested) inner loop with weight 25. These loops are encoded using adjacent codes, Table 3-31.

Table 3-31 DFS encoding for “bbara” FSM

states	codes
*st*0	0100
*st*1	0000
*st*2	0001
*st*3	0011
*st*4	0010
*st*5	0110
*st*6	0111
*st*7	1011
*st*8	1010
*st*9	1000

Table 3-32 summarizes the comparison among several discussed power-oriented encoding methods for the FSM "bbara".

Table 3-32 Comparison among several encoding methods for the FSM "bbara"

		Hamming Distances $H(c_i,c_j)$					
e_i	$w(e_i)$	WCEC	DFS	OLT	POW3	STB	Huffman
1-4	0,042	1	1	1	1	1	1
1-2	0,041	1	1	1	1	1	1
0-1	0,036	1	1	1	1	1	1
0-4	0,022	2	2	2	2	2	2
2-3	0,017	1	1	1	1	1	2
4-5	0,015	1	1	1	1	1	1
2-4	0,008	2	2	2	2	2	2
3-4	0,008	1	1	1	1	1	1
3-7	0,008	1	1	1	1	1	1
5-1	0,006	2	2	2	2	2	2
7-1	0,005	3	3	3	3	3	2
5-6	0,003	1	1	1	1	1	1
6-1	0,002	3	3	3	3	3	2
7-4	0,002	2	2	2	2	2	1
7-8	0,002	1	1	1	1	1	1
8-1	0,001	2	2	2	2	2	2
6-7	0,001	2	2	2	2	2	1
8-4	0,001	1	1	1	1	1	1
8-9	0,001	1	1	1	3	1	1
9-1	0,0003	1	1	3	3	3	2
9-0	0,0001	2	3	2	2	2	2
9-4	0,0001	2	2	2	4	2	1
Complicated Transitions		10	10	11	12	11	9
Redundant Switching		12	13	14	18	14	9
Cost Function O_{power}		0.276	0.277	0.278	0.279	0.277	0.283
Defect (%)		24.4	24.5	24.7	25.7	24.5	27.7

3.2.4 Further improvement of the received encoding

As it was mentioned above, the proposed state encoding technique is based on a greedy choice of an appropriate vertex added to a constructed edge cut. In this subsection we present some ways to improve the received encoding.

In computer science a search algorithm takes a problem as input and returns a solution to the problem, usually after evaluating a number of possible solutions. Searching algorithms can be divided into uninformed search algorithms which use the simplest, most intuitive method of searching and informed search algorithms which use heuristic to apply knowledge about the structure of the search space try to reduce the amount of the time spent for searching.

List search, tree search and graph search are uninformed search algorithms. The goal of list search algorithm is to find one element of a set by some key. Tree search is specialized version of search algorithms, which takes the properties of trees into account. Many of the problems in graph theory can be solved using graph search algorithms which are extensions of the tree search algorithms [81].

Tree search techniques

Tree search algorithms are the heart of searching techniques [88]. These search nodes of trees, whether that tree is explicit or implicit (generated on the go). The basic principle is that a node is taken from a data structure, its successors examined and added to the data structure. By manipulating the data structure, the tree is explored in different orders for instance level by level or reaching a leaf node first and backtracking.

A tree is a widely-used computer data structure that emulates a tree structure with a set of linked nodes. Each node has zero or more child nodes, which are below it in the tree. A node that has a child is called the child's parent node. A child has at most one parent; a node without a parent is called the root node (or root). Nodes with no children are called leaf nodes.

The root of the tree corresponds to an initial situation of the problem. Other vertices associate with situations which can be achieved in process of solution of the problem. There are two basic types of tree. In an unordered tree, there is no distinction between the children of a node --- none is the "first child" or "last child". A tree in which such distinctions are made is called ordered tree. Ordered trees are by far the most common form of tree data structure. The root selection gives an orientation to the tree when all paths are from the root to the other vertices. An arc, or directed edge, is an ordered pair of vertices. In tree arcs correspond to simple operations that represent the steps in process of solution of the problem. Arcs connect the vertices, which correspond to situations. A situation presented by a vertex characterizes a wide variety of different next steps that described by arcs outgoing from the considered vertex. Some situations present solutions of the problem.

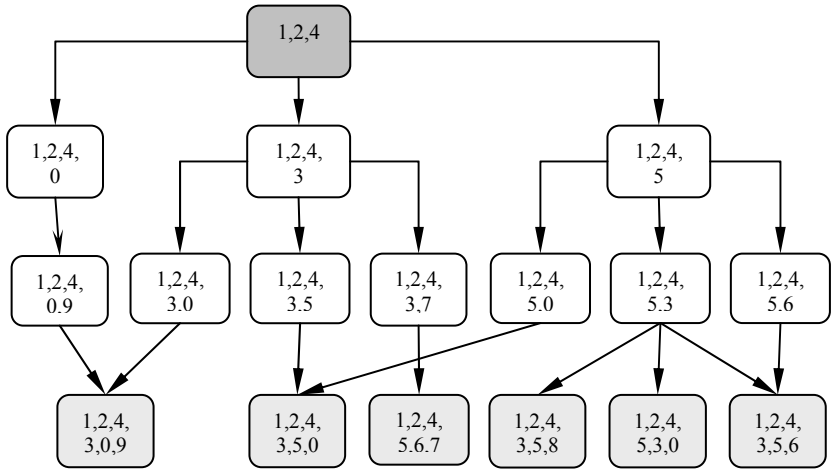

Figure 3-10 A fragment of tree search for the second encoding partition for the FSM "bbara"

Example, Figure 3-10 presents a fragment of tree search for the constructing the second encoding partition for the FSM "bbara". The root is {1, 2, 4}. Arcs correspond to variants of choice of the next vertex to add to the constructed set. We have received ten leaf nodes that correspond to the six various unit blocks of the second encoding partition.

Randomization

Randomization is a process of making something random. Randomization is a core principle in the statistical theory of design of experiments which is a methodology for solving optimization problem [81].

Under randomization of our encoding technique we imply a process of making a list of rest undistributed vertices random. We form a register of rewriting of rest vertices and then choose more appropriate vertex to construct an encoding partition.

Example, Table 3-7 demonstrates the construction of the second encoding partition for the FSM "bbara". We start the construction with a selection of an arbitrary edge ($e_{7,8}$) from the previous edge cut. Obviously, that the choice of other edge as a starting point perhaps has different result of construction from the received partition. We can form a register of rewritings of edges from the previous edge cut to find better decision. Also we form a register of rewritings of vertices (v_3,v_6) with minimal increment ($\gamma^1_{7,8}(v_3/v_6)=1$) with aim to select the most appropriate vertex to construct of an encoding partition. We select the first vertex v_3. Next we choose between two vertices v_2 and v_6 ($\gamma^1_3(v_2/v_6)=1$), and prefer the vertex v_6 because we receive the best decision.

Using randomization at a selection of next appropriate vertex to construct an encoding partition greatly reduces a search among rest undistributed vertices.

Tree search and randomization techniques were used for machine realization of the proposed encoding method.

3.3 Experimental results

To evaluate the performance of the proposed state encoding algorithm we use the benchmark set [48]. This is an industry-standard benchmark suite. Table 3-33 shows the benchmarks, the number of states each FSM consists of, and the number of codes available to the state encoding algorithms for the minimum size of the state register. Furthermore, the table lists the number of transitions.

Table 3-33: Benchmarks statistics

benchmark	states	codes	transitions
bbara	10	16	60
bbsse	13	16	56
beecount	7	8	28
cse	16	16	91
dk14	7	8	56
dk15	4	4	32
dk16	27	32	108
dk17	8	8	32
dk27	7	8	14
dk512	15	16	30
donfile	24	32	96
ex1	20	32	138
ex4	14	16	21
ex6	8	8	34
keyb	19	32	170
log	17	32	28
mark1	13	16	33
opus	10	16	22
planet	48	64	115
pma	24	32	74
s1	20	32	107
s8	5	8	32
sand	32	32	184
shiftreg	8	8	16
sse	13	16	48
tbk	32	32	104
tma	20	32	96
train11	11	16	15

We have applied our encoding algorithm Weakly Crossed Edge Cut to all FSM benchmarks and we present a summary of the results, Table 3-34. FSM state encoding for low power dissipation do not target the circuit switching activity directly, but instead minimize only the state register switching activity.

The switching activity is expressed as the number of bit changes in the state register, or the number of signal changes in the circuit, that occurred during the simulation. During the simulation, each state transition will cause at least one bit to change in the state register. Therefore, the number of state transitions forms a lower bound for the state register switching activity.

Table 3-34 Defect of the encodings ([a] - longer code)

benchmark	WCEC	DFS	OLT	POW3	STB	Huffman
bbara	**24.0**	24.1	24.6	25.0	26.0	27.2
bbsse	**15.1**	**15.1**	15.7	16.0	16.1[a]	19.0
beecount	**5.3**	**5.3**	5.5	5.5	5.4[a]	6.9
cse	**4.6**	5.6	7.8	8.0	4.9[a]	10.3
dk14	**34.1**	**34.1**	44.8	44.8	**34.1**[a]	52.0
dk15	**19.8**	**19.8**	**19.8**	**19.8**	**19.8**[a]	21.1
dk16	80.2	-	85.6	91.5	**73.7**[a]	104.0
dk17	**22.7**	22.8	23.5	23.5	23.4[a]	25.5
dk27	**16.3**	18.8	17.8	19.8	19.8	23.9
dk512	**18.4**	**18.4**	29.4	35.2	25.2[a]	39.8
donfile	83.0	-	91.8	110.1	**74.8**[a]	111.8
ex1	**29.8**	30.4	49.5	55.5	34.8[a]	68.8
ex4	**10.2**	**10.2**	12.0	14.2	11.4	14.8
ex6	**25.8**	**25.8**	29.6	27.9	-	27.9
keyb	**1.3**	**1.3**	**1.3**	**1.3**	**1.3**	1.6
log	**4.0**	-	-	4.7	**4.0**[a]	-
mark1	**29.8**	30.2	-	-	-	44.4
opus	**30.8**	**30.8**	-	32.4	32.4[a]	-
planet	32.8	-	52.3	64.7	**27.7**[a]	80.1
pma	**22.3**	**34.3**	-	-	**22.3**[a]	-
s1	54.8	-	66.1	75.2	**48.9**[a]	88.4
s8	**14.4**	14.5	29.3	33.7	14.5	38.9
sand	33.7	-	47.7	56.4	**18.1**[a]	80.0
shiftreg	**14.1**	**14.1**	35.6	46.7	**14.1**[a]	50.1
sse	**15.1**	**15.1**	-	-	16.0[a]	-
tbk	78.0	-	88.8	90.4	**35.5**[a]	100.3
tma	**14.6**	26.6	40.1	45.5	28.9	50.9
train11	**13.3**	13.4	46.8	50.7	23.5	54.9

The benchmarks *bbtas*, *kirkmann*, *lion*, *lion*9, *mc*, *modulo*12, *tav*, *train*4 are encoded perfectly, resulting in an average state register switching activity of 100%, or only one bit change for each state transition.

As expected, DFS [21] encoding finds the optimal state encoding solution for some FSM for a minimal width state register. The reason for this is the relatively low number of transitions for the number of states. The STB [55] guarantees to find the optimal state encoding solution because this algorithm utilizes (where needed) wider state registers, which relieves one of the constraints of the other low power state encodings. This allows the algorithm to obtain lower state register switching activities, at the cost of a larger chip area for the state register.

3.4 Summary

The research presented in the Chapter 3 focuses on the problem of minimizing the power consumption in synchronous sequential circuits, which is an indispensable component of any power optimization and synthesis environment for digital circuits. Based on the concept of an economical covering of the set of transitions by weakly crossed edge cuts we propose an effective technique to solve the state encoding problem for an FSM targeting power minimization.

Weakly Crossed Edge Cut encoding method uses the STG to describe of an FSM. The probabilistic model of an FSM is constructed by using the stochastic behavior of an FSM. For the given FSM description and the knowledge of the input probabilities the transition probabilities for the STG are calculated. The input probability distribution can be obtained by simulating the FSM at a higher level of abstraction in the context of its environment, or by direct knowledge from the designer. Transition probability information for each edge in the STG can then be determined by modeling the STG as a Markov chain. We describe the Markov chain model for the STG by a directed graph with a structure isomorphic to the STG and with weighted edges. The conditional probability distribution is found from the input probability distribution. The total transition probabilities are calculated. Then, we compute the steady state probabilities by solving the system of equations. Using the total transition probabilities the original STG is transformed into weighted graph which preserves only the relevant information needed for state encoding. Two matrices, the adjacency matrix and incidence matrix, are constructed by using the vertex-weighted and the edge-weighted graphs simultaneously.

The main idea of our technique is to find a state encoding that minimizes the number of state variables that change their value when FSM moves between two adjacent states. We construct a set of encoding partitions that corresponds to a set of weakly crossed edge cuts for the STG of an FSM. The framework of constructing of a set of weakly crossed edge cuts was developed in general enough to update for exploration of the algorithm for the optimization of the FSM that takes into account the reducing of switching activity. Power-oriented encoding algorithm was implemented and ran on standard benchmark circuits. We found that the presented state encodings allows low power FSM synthesis. Our results confirm that state encoding has an impact on power dissipation in the overall circuit.

4 DECOMPOSITION FOR A LOW POWER FSM

4.1 Introduction

In the previous chapter the problem of FSM state encoding has received considerable attention, because it is an important step in the process of sequential circuit synthesis. The FSM state encoding problem can be also viewed with relation to the FSM decomposition problem. Recently decomposition of the FSM has attracted attention of researchers for power reduction [9], [12], [51].

FSM decomposition is an organic part of logic synthesis process of sequential circuits [14], [41], and [58]. A complex FSM can be decomposed into simpler ones in order to recieve more efficient implementation. The task of decomposition has been a classic problem of the discrete system theory for many years [3], [28]. Decomposition of the FSM is a topic that waxes and wanes in importance. The fundamental works were done in the 1960s [28], became less interesting during the era of Very Large Scale Integration (VLSI) [27], and is becoming more important again with pervasive use of programmable logic and low power applications in digital design [1].

A large hardware behavioral description is decomposed into several smaller ones. One goal is to make the synthesis problem more tractable by providing smaller sub-problems that can be solved efficiently. Another goal is to create descriptions that can be synthesized into a structure that meets the design constraints. In the past, the synthesis focused on quality measures based on area and performance [64]. The continuing decrease in feature size and increase in chip density in recent years have given rise to consider decomposition theory for low power as a new dimension of design process [9], [12], [29], [30], [39], [41], [42], [50]-[53], [60], [65], [70], [74], and [78].

Usually, the task of FSM decomposition is a representation of the original FSM as its network realization. It means that, we construct such a network of the connected and interacted component machines which should realize behavior of the original FSM. Moreover, each component machine should satisfy some of given conditions.

Indeed, such a statement of the FSM decomposition task is common in sense of understanding of an overall object of machine decomposition. This work is concentrated on investigations of the pair method of decomposition [28]. The structural properties of machine are described in pair algebra terms. The character of possible decomposition of machine is caused by properties of partitions on the set of the states of this machine. Interaction of several machines can be imagined by mutual information depending on what is reflected by elements of pair algebra. Moreover, the task of FSM decomposition and problem of states encoding are virtually identical concepts. Thus, the problem of assigning a binary code to all states of original machine leads to the problem of assigning a binary code to the blocks of partitions. Therefore, the

task of FSM decomposition can be formulated in a manner of apparatus of partitions.

Given the STG of a circuit controller, the optimization task consists of decomposing and encoding the graph in order to prepare it for logic synthesis. Decomposition techniques produce interconnected machines from one large FSM, and they can be divided into two categories [42]: those based on the algebraic theory [28] and those based on the identification in the STG of sub-routines or co-routines [15]. A sub-routine/co-routine corresponds to a fragment of the STG augment with a wait state. Shutdown techniques can be applied to the individual machines because only one is active at a given point of time [10]. Both approaches to decomposition try to minimize the activity along the lines connecting the component machines, which tend to drive heavier loads. Decomposition naturally helps tackling the complexity issue [42].

We regard three categories of FSM decompositions: multiplicative, additive and generalized additive.

Multiplicative Decomposition is the general method of FSM decomposition. The decomposition is called "multiplicative" because the graph of the source FSM is embedded into a product of smaller graphs. This method enables synthesis of a network of interacting component machines corresponding to a complete set of partitions on the set of states of the source FSM. Each component machine corresponds to a partition on the set of the states. All the states belonging to a single block in a submachine are given the same code in that component machine. Therefore, there is no way of distinguishing between two states belonging to a single block in a submachine without recourse to information from other component machines. A block of states in a partition effectively corresponds to the state in the submachine associated with that partition. The functionality of the source machine is maintained in the decomposed machine if the partitions associated with the decomposition such that their product is the zero-partition on the set (every block of partition consists exactly of one state).

The idea of the *Additive Decomposition* is to introduce an additional "idle" state into component machines. The graph of an FSM is partitioned into node disjoint subsets. The network of machines consists of components working alternatively in time, i.e. all components except one are suspended in one of extra state (the "wait" state). The network of interacting component machines corresponds to a given partition on the set of the states of the source FSM. The number of the component machines in the network is equal to the number of the blocks and the number of the states of the component machines is equal to the number of the states in the corresponding block of a given partition plus 1.

The *Generalized Additive Decomposition* proceeds from a given cover on the set of the states of the source FSM. Each component machine corresponds to a subset of the set of the states of the source FSM. This method is a more generalized approach based upon the previous method (more than one machine could be active executing a computation at any given time while the other component machines are idle).

4.2 FSM decomposition technique

Because the traditional approach to FSM design involves solving at least two difficult combinatorial problems (state minimization and state encoding [58]), the attempt to implement a united strategy has been undertaken. The FSM decomposition problem is classic in the machine theory [28] just like the state encoding problem. Various approaches based on different heuristics have been investigated [1], [27], [58].

An FSM can be decomposed into smaller interacting machines in order to achieve different optimizations, such as area, performance, power consuming and testability. The range of our interests comes to power optimization. We proceed from the idea that proposed heuristics for state encoding and techniques of logic optimization work better for smaller circuits as opposed to larger ones. In general, if a good decomposition can be found, smaller areas for a decomposed FSM over a single lumped circuit will result. Taking into consideration the performance, the decomposed circuits can be clocked faster than the source machine, due to smaller critical path delays. At the same time, FSM decomposition may provide a solution that partitions the whole circuit into parts working alternatively. In this case, the switching activity of the circuit can be greatly reduced.

The main problem of the decomposition procedure is the choice of a partition system [14]. We have also emphasized that in the current work we deal only with decomposition methods based on the algebraic structure partition theory [28]. In the introduction to the current chapter, we have mark out three variants of partition systems: a complete system of partitions, a partition and a cover of the set of the states of the decomposable FSM. In general, decomposition using a partition and a cover on the set of the states of the decomposable machine can be regarded as a particular case of more common decomposition method using a complete system of partitions. Thus, the results received during multiplicative decomposition can be extended into additive and generalized additive decompositions.

The way to select (or construct) a partition (a set of partitions) is not discussed in the thesis. We leave this question open for future investigations. However, it is needed to underlain that the key condition of existing decomposition for a given FSM must be executed. In subchapter 2.2 we present the main condition of FSM decomposition in a general way.

To avoid significant complications during the decomposition procedure, it is necessary to keep certain tactics. One of the essential foundations of optimization methods of the decomposition synthesis is a strategy of global and local transformations [35]. The strategy consists of two logical transformations of the source FSM. The global or primary (coarse) transformation fragments the source FSM into a group of its pieces. Clearly, the way of this fragmentation depends on given conditions, or is dictated by a required basis. As a rule, more important requirements are expected to be executed at this stage.

The stage of global transformations will entail the formation of decomposition constraints. Consequently, already at this stage the main

restrictions on the basis and structure of the network should be executed. Among such restrictions, we are able to allocate, for example, the restriction on the number of component machines in the network, the restriction of the number of states of component machines, the restriction on the number of binary inputs of component machines, the restriction on the number of binary inputs of each component machines, etc. Every one of such restrictions demands the decision of the certain subtask at the stage of choosing the partition system. Moreover, depending on the given task, we can have the necessity to build a network with several restrictions. All above-mentioned is an occasion of creation a flexible method of global transformations realization of the source FSM on the partitions basis. The second step of the decomposition procedure is the step of determining of structure of the network. Here we define the number of machines in the network, inputs and outputs of each components and also connections between them.

When the system gets accustomed after impact of global transformations, here arises a question of more delicate (fine) optimization. The accurate definition of the received parts, detection and the specification of information connections and final clarifying of the decomposable FSM as a whole are ultimate aims of action of local transformations.

During the action of local transformations, we are interesting in component-wise optimization. After determining the structure of the network, here arises the task of optimum encoding, a task of searching for optimum input and output alphabets of the component machines, connection functions, and the network output. Generally speaking, the stage of local transformation means working directly with each component machine. During this process takes place an original swapping of complexity from the connection functions and the network output to the output functions of the component machines. Obviously, that optimization on the stage of local transformations also requires a local criterion.

The strategy of global and local transformations at FSM decomposition based on the concept of pair method also consists of two consistent steps. On the global transformation step we realize rough decomposition of the source FSM. Thus, the source FSM is strictly divided into a set of component machines, which satisfy the given restrictions. As a result of global transformations, we have a network of component machines which realizes the source FSM.

The next stage is local transformations. During this phase, we have the possibility for a more detailed and delicate specification of the parts received on the previous step. Local transformations of the component machines allow constructing of its internal inputs and outputs. Moreover, local transformations give us resources for the optimization of internal connections between component machines.

The main goal of FSM decomposition is division on a symbiosis of global and local transformations is an opportunity to avoid full excess at the optimization. It is well known that exact solution of optimal decomposition is

practically not feasible. Therefore, the strategy of global and local transformations gives us an opportunity of more differentiated approach to the problem of searching for optimum decomposition.

We illustrate how works the proposed procedure of FSM decomposition on the FSM "opus" [48]. The state transition and output table the FSM "opus" is presented in Table 4-1.

Table 4-1 State transition and output table for the FSM "opus"

№	Present state	Input	Next state	Output
1	*init*0	x_3	*init*0	y_1y_2
2		$\neg x_3$	*init*1	y_1y_2
3	*init*1	x_3	*init*0	y_1y_2
4		$\neg x_3\neg x_4$	*init*1	y_1y_2
5		$\neg x_3x_4$	*init*2	$y_1y_2y_6$
6	*init*2	x_3	*init*0	y_1y_2
7		$\neg x_3$	*init*4	$y_1y_2y_4$
8	*init*4	x_3	*init*0	y_1y_2
9		$\neg x_3x_4$	*init*4	$y_1y_2y_4$
10		$\neg x_3\neg x_4$	*IOwait*	-
11	*IOwait*	x_3	*init*0	y_1y_2
12		$x_1\neg x_2\neg x_3\neg x_4$	*init*1	y_1y_2
13		$\neg x_3x_4$	*init*2	$y_1y_2y_6$
14		$\neg x_1\neg x_2\neg x_3\neg x_4$	*IOwait*	-
15		$\neg x_1x_2\neg x_3\neg x_4\neg x_5$	*read*0	y_1y_3
16		$x_1x_2\neg x_3\neg x_4\neg x_5$	*write*0	y_1y_5
17		$\neg x_1x_2\neg x_3\neg x_4x_5$	*RMACK*	y_1
18		$x_1x_2\neg x_3\neg x_4x_5$	*WMACK*	y_1
19	*RMACK*	x_3	*init*0	y_1y_2
20		$\neg x_3x_5$	*read*0	y_1y_3
21		$\neg x_3\neg x_5$	*RMACK*	y_1
22	*WMACK*	$\neg x_3$	*init*0	y_1y_2
23		$x_3\neg x_5$	*write*0	y_1y_5
24		$\neg x_3x_5$	*WMACK*	y_1
25	*read*0	x_3	*init*0	y_1y_2
26		$\neg x_3$	*read*1	$y_1y_3y_6$
27	*read*1	x_3	*init*0	y_1y_2
28		$\neg x_3$	*IOwait*	-
29	*write*0	x_3	*init*0	y_1y_2
30		$\neg x_3$	*IOwait*	-

4.2.1 Decomposition constraints

In the previous we describe the notion of restrictions which are applied on the structure of the network. For example, we would like to apply the restriction on the number of inputs in each component machine in the network. Such restrictions have effect on the construction of each partition from the complete system of partitions for decomposition of the source FSM. Each partition from the complete system of partitions represents the work of one component machine in the network. The blocks of the partition describe the states of the correspondent component machine. The set of all blocks of all partitions from the complete set of partitions present the global states of the network of machines and form the set of decomposition constraints.

We introduce tree heuristic methods for global transformations executing.

In our scenario, the first heuristic method corresponds to the primary transformation of an FSM should begin with a separation of its inputs. Mainly, it is explained by desire to mark out some inputs, which are not essential for some component machine. The follows method breaks up into two consecutive steps, each of which allows the process of complete set of partitions choice.

The second heuristic method for global transformation of the source FSM allow to identify the most probable behaviors in a sequential component.

The third method is global transformation of the source FSM with aim to separate outputs. The method provides a systematic way to separate outputs among component machines.

Various decomposition techniques based on partition on the set of states of decomposable machine produce various sets of constraints.

The set of decomposition constraints for the network of machines defined by a complete system of partitions $\{\pi_i\}$, $1 \le i \le n$ is a set $\left\{B_{\pi_i}^j \mid 1 \le i \le n, 1 \le j \le |\pi_i|\right\}$, where B_{π_i} are the blocks of partitions from $\{\pi_i\}$ and $|\pi_i|$ is the number of blocks in π_i. For the constructed network of machines the number of decomposition constraints is equal to the total number of blocks in the complete system of partitions.

The set of decomposition constraints for the network of machines defined by a partition π is a set $\left\{B_\pi^j \mid 1 \le j \le |\pi|\right\}$, where B_π are the blocks of partition π and $|\pi|$ is the number of blocks in π.

Similarly the set of decomposition constraints for the network of machines constructed by a cover on the set of states of decomposable machine is defined.

In general we present a matrix formulation for the decomposition constraints as follows [36]: for the given set of states S of the decomposable machine the constraint matrix is a matrix with n_c rows and n_s columns. n_c is equal to the number of constrains and n_s is equal to the number of symbols in S. Entry (i, j) is 1 if and only if the i^{th} constraint contains symbol j, otherwise it is 0.

Tables 4-2, 4-3 and 4-4 present tree matrices of decomposition constraints for the FSM “opus”.

Decomposition with a separation of inputs (multiplicative decomposition)

Decomposition of the source FSM with separation of inputs implies partial or full division of all input signals between the component machines.

First step

- Calculating of α-partitions on the set of inputs of a decomposable FSM

Example, we illustrate the calculation of α-partitions on the FSM "opus":
$\alpha(x_1)$={{*init*1,*IOwait*}; {*read*0,*write*0}; {*RMACK*,*WMACK*};
{*init*0,*init*2,*init*4,*read*1}},
$\alpha(x_2)$={{*init*1,*write*0,*WMACK*}; {*IOwait*,*read*0,*RMACK*};
{*init*0,*init*2,*init*4,*read*1}},
$\alpha(x_3)$={{*init*0,*init*1,*init*2,*init*4,*IOwait*,*read*0,*write*0,*RMACK*,*WMACK*,*read*1}},
$\alpha(x_4)$={{*init*0,*read*1};{*init*1,*init*2,*init*4,*IOwait*,*read*0,*write*0,*RMACK*,*WMACK*}}
$\alpha(x_5)$={{*read*0,*RMACK*}; {*write*0,*WMACK*};
{*init*0,*init*1,*init*2,*init*4,*IOwait*,*read*1}}.

Second Step

- Generating of a complete system of partitions $\{\pi\}$ on the set of states S of a decomposable FSM

Intuitively, at the current step we have a certain freedom at the choice of complete system of partitions. On the first step of this method, by search of α-partitions we have been strictly limited to their definition. Quite opposite, the second step does not limit us to a rule of addition of α-partitions. Each step generates one next partition which fills up set $\{\pi\}$ empty at the beginning. At the forming of the first partition π_1 from the set of partitions we consecutively enlarge the zero partition π_I by addition with some α-partitions. However, there are some variants of partitions increasing. It is clear, that imposing additional restrictions, we can make process of forming partitions more flexible. Thus, the restriction on the number of component machines dictates the number of partitions in complete set of partitions. The number of blocks in partitions determines the number of states of the corresponding component machines.

Example, the partition π_1 describes the work of the first component machine "opus_1" which does not depend on the inputs x_2 and x_5; the partition π_2 describes the work of the second component machine "opus_2" which does not depend on the input x_1; the product of π_1 and π_2 is equal to zero partition and the complete system of partitions for the FSM "opus" is π=$\{\pi_1,\pi_2\}$:
π_1={{*init*1,*write*0,*WMACK*};{*IOwait*,*read*0,*RMACK*};{*init*0,*init*2,*init*4,*read*1}} and **π_2={{*init*1,*IOwait*,*init*0}; {*write*0,*read*0,*init*2}; {*WMACK*,*RMACK*,*init*4}; {*read*1}}.**

The sets of inputs for both component machines "opus_1" and "opus_2" are I^1=(x_1, x_3, x_4) and I^2=(x_2, x_3, x_4, x_5).

Table 4-2 Constraint matrix for the network "opus" after multiplicative decomposition

	*init*0	*init*1	*init*2	*init*4	*IOwait*	*Rmack*	*Wmack*	*read*0	*read*1	*write*0
*c*0	0	1	0	0	0	0	1	0	0	1
*c*1	0	0	0	0	1	1	0	1	0	0
*c*2	1	0	1	1	0	0	0	0	1	0
*c*3	1	1	0	0	1	0	0	0	0	0
*c*4	0	0	1	0	0	0	0	1	0	1
*c*5	0	0	0	1	0	1	1	0	0	0
*c*6	0	0	0	0	0	0	0	0	1	0

Decomposition based on state probability distribution (additive decomposition)

First Step

- Calculating of the distribution of steady state probabilities

Example, for the FSM "opus" we have the distribution of steady state probabilities calculated using the probabilistic model of the FSM described above:

P_{init0}=0.500
P_{init1}=0.334
P_{init2}=0.088
P_{init4}=0.058
P_{IOwait}=0.017
P_{read0}=0.0006
P_{write0}=0.0006
P_{RMACK}=0.0006
P_{WMACK}=0.0006
P_{read1}=0.0006

After performing sensitive analysis of state probability distribution of the considered FSM "opus", we decide to decompose it into the network of two component machine where the first component machine consists of states with probability greater than *P* times the probability of the most frequently occurring state.

Second Step

- Generating a partition π_p on the set of states of a decomposable FSM

Example, the 0.9-ordered partition π_p for the FSM "opus" is

$\pi_p(S)$={{***init*0,*init*1,*init*2,*init*4,*IOwait*};
{***read*0,*write*0,*RMACK*,*WMACK*,*read*1}}

Table 4-3 Constraint matrix for the network “opus” after additive decomposition

	*init*0	*init*1	*init*2	*init*4	*IOwait*	*Rmack*	*Wmack*	*read*0	*read*1	*write*0
*c*0	1	1	1	1	1	0	0	0	0	0
*c*1	0	0	0	0	0	1	1	1	1	1

Decomposition with separation of outputs (generalized additive decomposition)

First Step

- Calculating a partitions on the set of outputs of a decomposable FSM

Example, we divide the FSM “opus” into the network of two component machines, the partitions on the set of inputs are:
$O^1=\{y_1,y_2,y_4\}$
$\pi(O^1)=\{\{init0,init2,write0,read1\};$
$\{init1,init4,IOwait,read0,RMACK,WMACK\}\}$
$O^2=\{y_1,y_2,y_3,y_5,y_6\}$
$\pi(O^2)=\{\{init0,init1,IOwait,read0,read1,write0,WMACK,WMACK\};$
$\{init2,init4\}\}$.

Second Step

- Generating a cover φ on the set of states of a decomposable FSM

Example, the cover φ on the set of states S of the FSM “opus” is
$\boldsymbol{\varphi(S)=\{\{init0,init2,init4,write0,read1\};}$
$\boldsymbol{\{init0,init1,IOwait,read0,write0,read1,RMACK,WMACK\}\}}$.

Table 4-4 Constraint matrix for the network “opus” after generalized additive decomposition

	*init*0	*init*1	*init*2	*init*4	*IOwait*	*Rmack*	*Wmack*	*read*0	*read*1	*write*0
*c*0	1	0	1	1	0	0	0	0	1	1
*c*1	1	1	0	0	1	1	1	1	1	1

4.2.2 Information relationship measures

The problem of low power synthesis corresponds to an optimal decomposition of an FSM reduced to choice of partitions on the set of states of prototype machine [65]. For evaluation of the networks, the informational modeling based on entropy measure is considered. It enables to enhance the decomposition partition search for low power synthesis [32], [33].

We propose measures to enable qualitative and quantitative analysis of the information structure and information flows of a FSM network to control low-power synthesis of a sequential circuit [DDECS'02], [MIEL'02], [StZag'02].

Let $E = \{e_1, e_2, \ldots, e_n\}$ be a set of events which may occur with the probabilities $p_1, p_2, \ldots, p_n$. This set of events is complete that is $p_1+p_2+\ldots+p_n=1$.

Entropy of E (denoted by $H(E)$) is given by [11], [45]:

$$H(E) = -\sum_{e \in E} p(e) \cdot \log_2 p(e)$$

As an event, we can consider the state of occupation or state transition of FSM. The entropy of partition π is [35]:

$$H(\pi) = -\sum_{B \in \pi} p(B) \cdot \log_2 p(B)$$

Here the probability of the block $B \subseteq S$ is defined as the cumulative occupation probability of the states in B.

Entropy of the network of machines [35] corresponding to the set of partitions, $N=\{\pi_1,\pi_2, \ldots, \pi_n\}$, is equal to the sum of entropies of partitions of N.

For partition pair $\langle \pi_i, \pi_j \rangle$ the conditional entropy is:

$$H(\pi_i, \pi_j) = H(\pi_i \cdot \pi_j) - H(\pi_i)$$

To estimate the power consumption (1), one has to calculate the switching factor of the circuit. Entropy is related to switching activity that is if the signal switching is high, it is likely that entropy is high also [44]. Theoretically confirmed high correlation proves that partition entropy is suitable for estimating corresponding component machines [73] which make it a good measure for partition choice for appropriate decomposition [35]. For estimation of switching activity of an FSM as complete set of events, we consider the set of all transitions (corresponding to edges of the STG) in the FSM, T_i. To estimate switching activity, we have to take into account only events related to changing of states. Criterion of switching activity of component machine A_i, is $H(T_i)$ decreased by self-loop component:

$$H^{sw}(A_i) = \left[-\sum_{1 \le j \le m} (p_i \cdot q_{ij}) \cdot \log_2 (p_i \cdot q_{ij}) \right] - \left[-(p_i \cdot q_{ii}) \cdot \log_2 (p_i \cdot q_{ii}) \right]$$

Affirmation. For FSM A, $H^{sw}(A)$ is maximal if transitions of the FSM are equiprobable:

$$R^{sw}(N) = \frac{\sum_{1 \le i \le n} H^{sw}(A_i)}{H^{sw}(A)} \tag{5}$$

To be useful for high-level power analysis, the switching activity has to be used in conjunction with an estimator for the complexity of the target circuit [2]. Since we want to keep the independence from the actual implementation of the circuit, we need an encoding independent measure of the complexity of the circuit. We suppose, that the measure for complexity of a controller (which is synthesized from the FSM description in the design process) approximated by the number of rows in the state transition table of the corresponding FSM description. It is shown [38] that for low-power design this measure is good as parameter of area estimation.

To calculate this parameter we need to find the symbolic cover of the discrete function F_i: $D(\delta) \rightarrow \pi_i$. Given the FSM, we first assign one-hot codes to all states. Then symbolic minimization is applied to the one-hot coded machine using multi-valued logic minimization. The result is a symbolic cover, K_i, of the F_i. Each element of the symbolic cover is a symbolic prime implicant, that is a triple $\langle \beta, B', B \rangle$ where B' is the set of states (block of partition $M(\pi)$) which transit to the next state contained in the same block B of the partition π under input condition β. The number of prime implicates, $|K_i|$, is proportional to number of rows in the transition table of the corresponding component machine.

From this convincing argument:

$$C^{compl}(A_i) = |K_i|$$

Next, we define the complexity measure for the network relative to the source FSM:

$$R^{compl}(N) = \frac{\sum_{1 \le i \le n} |K(A_i)|}{|K(A)|} \tag{6}$$

In additional, we present the integral criterion (including two parameters (5) and (6)) for search decomposition partitions as geometric mean of switching measure of order and area:

$$\Omega(N) = \sqrt{R^{sw}(N) \cdot R^{compl}(N)}$$

The value of this estimation takes into account the parameters area (capacitance) and register switching rate simultaneously.

Example, we apply introduced notions to the example FSM "dk15" [48] for which the exact state occupation probabilities are computed, Figure 4-1.

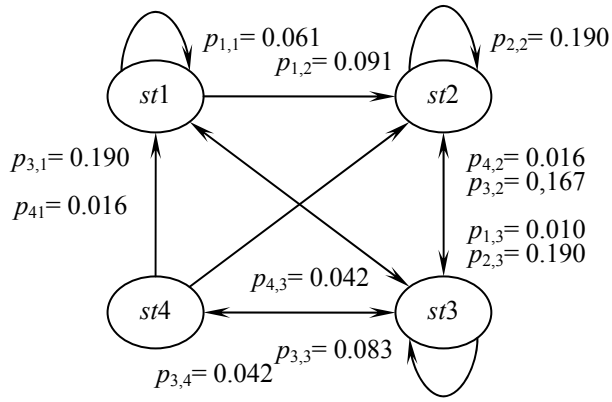

Figure 4-1 STG of the FSM "dk15" with transition probabilities

Final or limiting probabilities of the states for the FSM "dk15" are $P(S)$={0.2435; 0.3809; 0.3339; 0.0417}, entropy of this complete set is H=1.7463. We decompose the FSM "dk15" into a network of two component machines. We select two different partitions systems: [π_1={{*st*1,*st*4},{*st*2},{*st*3}}, π_2={{*st*1},{*st*2,*st*4},{*st*3}}] and [τ_1={{*st*1,*st*2,*st*3},{*st*4}}, τ_2={{*st*1},{*st*2},{*st*3,*st*4}}].

According to [28] $M(\pi_1)=M(\pi_2)$={{*st*1},{*st*2},{*st*3},{*st*4}}, and $M(\tau_1)$={{*st*1,*st*2,*st*3},{*st*4}}, $M(\tau_2)$= {{*st*1},{*st*2},{*st*3},{*st*4}}. Entropy of partitions in the first case $H(\pi_1)$=1.575 and $H(\pi_2)$=1.550; total entropy of states after decomposition $H(N_1)$=3.125. In case of the second decomposition system $H(\tau_1)$=0.250, $H(\tau_2)$=1.557, $H(N_2)$=1.808. Transition tables of component machines for both cases are presented in Table 4-5 and Table 4-6.

Table 4-5 Transition tables for the component machines in the network N_1

Present state	Next state	$q_{i,j}$	$p_i q_{i,j}$
*st*1	***st*1**	**0.333**	**0.095**
	*st*2	0.375	0.107
	*st*3	0.292	0.083
*st*2	***st*2**	**0.500**	**0.190**
	*st*3	0.500	0.190
*st*3	***st*3**	**0.125**	**0.042**
	*st*1	0.625	0.209
	*st*2	0.250	0.083

Present state	Next state	$q_{i,j}$	$p_i q_{i,j}$
*st*4	***st*4**	**0.25**	**0.061**
	*st*5	0.375	0.091
	*st*6	0.375	0.091
*st*5	***st*5**	**0.453**	**0.191**
	*st*4	0.141	0.060
	*st*6	0.406	0.172
*st*6	***st*6**	**0.125**	**0.042**
	*st*4	0.500	0.167
	*st*5	0.375	0.125

Table 4-6 Transition tables for the component machines in the network N_2

Present state	Next state	$q_{i,j}$	$p_i q_{i,j}$
*st*1	*st*1	**0.977**	**0.936**
	*st*2	0.023	0.022
*st*2	*st*1	1.000	0.042

Present state	Next state	$q_{i,j}$	$p_i q_{i,j}$
*st*3	*st*3	**0.25**	**0.061**
	*st*4	0.375	0.091
	*st*5	0.375	0.091
*st*4	*st*4	**0.500**	**0.190**
	*st*5	0.500	0.190
*st*5	*st*5	**0.250**	**0.094**
	*st*3	0.495	0.186
	*st*4	0.255	0.096

Total entropy of transitions before decomposition $E_{sw}(A)$=5.873 and total entropy of transitions after decomposition in the first case $E_{est}(N_1)$=3.125, in the second case $E_{est}(N_2)$=1.808. The integral criterions for the both networks are $C_{est}(N)$=0.368 and $C_{est}(N)$=1.083 and it is illustrate the complexity of implementation.

We have presented the approach for implementation-independent low power partitioning synthesis that attempts to minimize the average number of signal transitions at the sequential circuit nodes through DPM. It is shown that decomposition yields attractive power reduction in the final implementations. What makes entropy especially useful from decomposition and coding point of view is fact, that partitions, which are incomparable under the least upper bound and the largest lower bounds in classic lattice, usually do have different entropy values, so they become comparable from the power consumption point of view. The idea of using entropy based on informational measures can be also extended to other phases of logic synthesis also.

4.2.3 Decomposition procedure

The synthesis of the network of machines in the proposed decomposition procedure implies of determining the structure of the network and encoding of the network of machines.

The structure of the network under multiplicative decomposition

For the FSM $A=\{S,I,O,\delta,\lambda\}$ and the complete system of partitions $\{\pi_i\}$, $1\le i\le n$ on the set of states S of A we define the network N with n component machines $A^i=\{S^i,I^i,\delta^i\}$ in accordance to the pairs (S,π_i), $1\le i\le n$ [28]. The number of component machines is equal to the number of partitions in the complete system of partitions. Each partition π_i defines one component machine A^i. The number of states of A^i is equal to the number of blocks in π_i. The component machine A^i is defined as follows:

The set of states $A^i=B^j$, $1\le j\le m$, where $B^j\in\pi_i$, $1\le i\le n$ is the j^{th} block of the partition π_i.

The sets of inputs I^i for each component machines A^i defines by using the operator $M_{I\text{-}S}(\pi_i)$ on the set of inputs of FSM A [28].

The set of internal binary outputs Z_o^i for each component machines A^i describes by partitions π_i.

The set of internal binary inputs Z_I^i for each component machines A^i defines by using operator $M_{S\text{-}S}(\pi_i)$ [28]. $M_{S\text{-}S}(\pi_i)$ describes the information received from the other components of the network sufficient for the component machine A^i to compute its next state.

Example, for the FSM "opus" we have

$A^1(\pi_1)=\{\{init1,write0,WMACK\}; \{IOwait,read0,RMACK\}; \{init0,init2,init4,read1\}\}$, $A^2(\pi_2)=\{\{init1,IOwait,init0\}; \{write0,read0,init2\}; \{WMACK,RMACK,init4\}; \{read1\}\}$.

$A^1(s)=\{st0,st1,st2\}$,

$A^2(s)=\{st3,st4,st5,st6\}$.

$M_{I\text{-}S}(\pi_1)=\{x_1, x_3, x_4\}$,

$M_{I\text{-}S}(\pi_2)=\{x_2, x_3, x_4, x_5\}$,

$M_{S\text{-}S}(\pi_1)=\{\{RMACK,read1,write0\}; \{init4\}; IOwait\}; \{WMACK,init0\}; \{init1\}; \{init2,read0\}\}$,

$M_{S\text{-}S}(\pi_2)=\{\{read0\}; \{init2\}; \{IOwait\}; \{init1\}; \{init4\}; \{init0,read1,write0\}; \{RMACK,WMACK\}\}$.

$Z_o^i(\pi_1)=\{z^1_1, z^1_2\}$

$Z_o^i(\pi_2)=\{z^2_1, z^2_2\}$

$Z_I^i(\pi_1)=\{z^2_1, z^2_2\}$

$Z_I^i(\pi_1)=\{z^1_1, z^1_2\}$

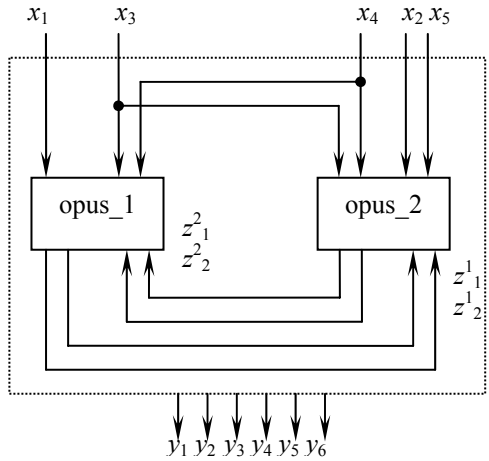

Figure 4-2 Structure of the network “opus” under multiplicative decomposition

Table 4-7 State transition table for the component machine “opus_1”

№	Present state	Input	Next state
1	*st*0	$x_4\mathbf{z}^2_1\neg\mathbf{z}^2_2\vee\neg\mathbf{z}^2_1\mathbf{z}^2_2\vee x_3\mathbf{z}^2_1$	*st*0
2		$\neg x_3\mathbf{z}^2_1\mathbf{z}^2_2$	*st*1
3		$\neg x_3\neg\mathbf{z}^2_1\neg\mathbf{z}^2_2\vee\neg x_3\neg x_4\neg\mathbf{z}^2_2$	*st*2
4	*st*1	$x_3\mathbf{z}^2_1\vee x_3\mathbf{z}^2_2\vee x_4\mathbf{z}^2_1\mathbf{z}^2_2$	*st*0
5		$\neg x_3\vee\neg x_3\neg x_4\mathbf{z}^2_2$	*st*1
6		$\neg x_3\neg\mathbf{z}^2_1\mathbf{z}^2_2$	*st*2
7	*st*2	$\neg\mathbf{z}^2_1\mathbf{z}^2_2\vee x_3\mathbf{z}^2_1\vee x_4\mathbf{z}^2_2$	*st*0
8		$x_1\neg x_3\neg x_4\mathbf{z}^2_1\mathbf{z}^2_2$	*st*1
9		$\neg x_3\neg\mathbf{z}^2_1\mathbf{z}^2_2\vee\neg x_1\neg x_3\neg x_4\mathbf{z}^2_1$	*st*2

Table 4-8 State transition table for the component machine “opus_2”

№	Present state	Input	Next state
1	*st*3	$x_2x_3\mathbf{z}^1_1\vee\neg x_2x_3\mathbf{z}^1_2\vee\mathbf{z}^1_1\mathbf{z}^1_2\vee x_2\neg x_4\mathbf{z}^1_2\vee\neg x_2\neg x_4\mathbf{z}^1_1$	*st*3
2		$x_2\neg x_3x_4\neg\mathbf{z}^1_1\mathbf{z}^1_2\vee\neg x_2\neg x_3x_4\mathbf{z}^1_1\neg\mathbf{z}^1_2\vee x_2\neg x_3\neg x_5\mathbf{z}^1_1\neg\mathbf{z}^1_2$	*st*4
3		$x_2\neg x_3\neg x_4x_5\mathbf{z}^1_1\neg\mathbf{z}^1_2$	*st*5
4	*st*4	$\neg\mathbf{z}^1_1\mathbf{z}^1_2\vee x_3\mathbf{z}^1_1$	*st*3
5		$\neg x_3\mathbf{z}^1_1\mathbf{z}^1_2$	*st*5
6		$\neg x_3\mathbf{z}^1_1\neg\mathbf{z}^1_2$	*st*6
7	*st*5	$x_3\mathbf{z}^1_1\vee x_3\mathbf{z}^1_2\vee\neg x_4\mathbf{z}^1_1\mathbf{z}^1_2$	*st*3
8		$\neg x_3x_5\mathbf{z}^1_1\neg\mathbf{z}^1_2\vee\neg x_3x_5\neg\mathbf{z}^1_1\mathbf{z}^1_2$	*st*4
9		$\neg x_3x_4\mathbf{z}^1_1\mathbf{z}^1_2\vee\neg x_3\neg x_5\mathbf{z}^1_1\neg\mathbf{z}^1_2\vee\neg x_3\neg x_5\neg\mathbf{z}^1_1\mathbf{z}^1_2$	*st*5
10	*st*6	$\mathbf{z}^1_1\mathbf{z}^1_2$	*st*3

The structure of the network under additive decomposition

For the FSM $A=\{S,I,O,\delta,\lambda\}$ and the partition $\pi=\{B^1,\ldots,B^n\}$ on the set of states of A, where $B^i(i=1,\ldots,n)$ is a block of this partition we define the network N with n component machines $A^i=\{S^i,I^i,O^i,\delta^i,\lambda^i\}$ in accordance to the pair (S,π) [5]. The number of component machines is equal to the number of blocks in the partition π.

The FSM A^i is defined as follows. The set of states $S^i=B^i\cup s_{idle}$ where B^i is the i^{th} block of the partition π and s_{idle} is the additional (extra wait [70]) state in A^i, such state exists in each component machine. To define the sets of inputs I^i and outputs O^i in A^i we put the sets $I^i(s)$, $O^i(s)$, $G^i(s)$, and $T^i(s)$ in accordance to each state s of the FSM A: $I^i(s)$ is the set of inputs in all conjunctions for the transitions from s, $O^i(s)$ is the set of outputs for all transitions from the states s, $G^i(s)$ is the set of states from which there are transitions to the state s and a in not included in $G^i(s)$: $G^i(s)=\{s_j|\delta^i(s_j,x)=s,\ x\in I,\ s_j\neq s\}$, $T^i(s)$ is the set of states to which there are transitions from the state s; s is not included in $T^i(s)$: $T^i(s)=\{s_i|\delta^i(s,x)=s_i,\ x\in I,\ s_i\neq s\}$.

For each block $B^i(i=1,\ldots,n)$ of the partition π we define the following sets [5]: $I(B^i)$ is the set of inputs at all transitions from the states of the block B^i in the transition table of the FSM A, $O(B^i)$ is the set of outputs at all transitions from the states of the block B^i in the transition table of the FSM A, $G(B^i)=\{s_j|\delta^i(s_j,x)=s,\ x\in I,\ s_j\notin B^i,\ s\in B^i\}$ is the set of states not included in B^i, form which there are transitions to the states of B^i in the FSM A, and $T(B^i)=\{s_i|\delta^i(s,x)=s_i,\ x\in I,\ s_i\notin B^i,\ s\in B^i\}$ is the set of states not included in B^i, to which there are transitions from the states of B^i in the FSM A. $Q(B^i)=\{s|\delta^i(s_j,x)=s,\ x\in I,\ s_j\notin B^i,\ s\in B^i\}$ is the subset of states of B^i, to which there are transitions from the states not included in B^i (from the states of $T(B^i)$) in the FSM A.

Next we define the set of inputs I^i in the component machine A^i: $I^i=I(B^i)\cup Z^i_I$, where Z^i_I is the set of additional inputs which connect other component machines with the machine A^i. To define Z^i_I we put the additional inputs z of the component machine in accordance to each state $s\in Q(B^i)$. The number of elements in Z^i_I is the number of additional inputs in the component machine A^i and is equal to the number of states in $Q(B^i)$: $Z^i_I=\{z|\delta^i(s_j,x)=s,\ x\in I,\ s_i\notin B^i,\ s\in B^i\}$. Thus, if there is a transition from s_j to s in the FSM A and $s\in B^i$ (s is the state of the component machine A^i) and $s_i\notin B^i$ (s_i is not the state of A^i), then there is an additional input z in the machine A^i.

Similarly, we define the set of additional outputs Z^i_O which connect the machine A^i A^i with other components in the network. To define Z^i_O we put the additional output z of the component machine A^i in accordance to each state $s\in T(B^i)$. The number of elements in Z^i_O is the number of additional outputs in the component machine A^i is equal to the number of states in $T(B^i)$. Thus, if there is a transition from s to s_i in the FSM A and $s\in B^i$ (s is the state of the component machine) and $s_i\notin B^i$ (s_i is not the states of A^i), then there is an

additional output z in the machine. The sets $Q(B^i)$ and $T(B^i)$ define the sets of additional input Z^i_I and output Z^i_O in the component machine A^i.

The transition function δ^i and the output function λ^i in the component machine A^i are defined as follows. Let $\delta(s_i,x)=s$; $\lambda(s_i,x)=y$ be the transition in the FSM A. If $s_i,s\in B^i$; s_i and s are the states of the same machine A^i, then $\delta^i(s_i,x)=s$; $\lambda^i(s_i,x)=y$ is the transition in the machine A^i. If $s_i\in B^i$, $s_j\in B^j$; s_i and s_j are the states of two different machines A^i and A^j, then the component machine A^i transits from the state s_i to the additional state s_{idle} with the output O^i and the additional output z_O: $\delta^i(s_i,x)=s_{idle}$; $\lambda^i(s_i,x)=y\cup\{z_O\}$. The additional output of the component machine A^i is the input of the component machine A^j. This input z_I produces the transition from the additional state s_{idle} to the state s with the output y_0 in the machine A^j: $\delta^j(s_{idle},z_I)=s$; $\lambda^j(s_{idle},z_I)=y_0$. y_0 corresponds to the output vector containing only zero components, none of $y_{1,\ ...,}\ y_m$ are written in the column "output" at the transition (s_{idle},s) in the machine A^j.

Example, for the FSM "opus" and the decomposition partition $\pi=\pi_p$, $\pi_p(S)$=[(*init*0,*init*1,*init*2,*init*4,*IOwait*);(*read*0,*write*0,*RMACK*,*WMACK*,*read*1)] we define the network "opus" based on state probability distribution under additive decomposition, Figure 4-3.

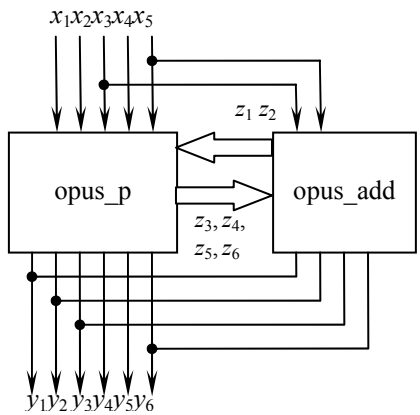

Figure 4-3 Structure of the network "opus" based on probability extraction

$B^1(\pi_p)$={*init*0,*init*1,*init*2,*init*4,*IOwait*},
$B^2(\pi_p)$={*read*0,*write*0,*RMACK*,*WMACK*,*read*1};
$B^1(s)$={*st*0,*st*1,*st*2,*st*3,*st*4,st_{idle}}, $B^2(s)$={*st*0,*st*1,*st*2,*st*3,*st*4,st_{idle}};

The sets $I^i(s)$, $O^i(s)$, $G^i(s)$, and $T^i(s)$ for the FSM "opus" are presented in Table 4-9.

$I(B^1)=\{x_1,x_2,x_3,x_4,x_5\}$, $I(B^2)=\{x_3,x_5\}$;
$O(B^1)=\{y_1,y_2,y_3,y_4,y_5,y_6\}$, $O(B^2)=\{y_1,y_2,y_3,y_6\}$;
$G(B^1)$={*read*0,*write*0,*RMACK*,*WMACK*,*read*1}, $G(B^2)$={*init*0,*IOwait*};
$T(B^1)$= {*read*0,*write*0,*RMACK*,*WMACK*}, $T(B^2)$={*init*0,*IOwait*};
$Q(B^1)$={*init*0,*IOwait*},
$Q(B^2)$={*read*0,*write*0,*RMACK*,*WMACK*}.

Table 4-9 The state parameters of the FSM “opus”

State	$I^i(s)$	$O^i(s)$	$G^i(s)$	$T^i(s)$
*init*0	x_3	y_1y_2	*init*1, *init*2, *init*4, *IOwait*, *RMACK*, *WMACK*, *read*0, *read*1, *write*0	*init*1
*init*1	x_3x_4	$y_1y_2y_6$	*init*0, *IOwait*, *read*0	*init*0
*init*2	x_3	$y_1y_2y_4$	*init*1, *IOwait*	*init*0,*init*4
*init*4	x_3x_4	$y_1y_2y_4$	*init*2	*init*0,*IOwait*
IOwait	$x_1x_2x_3x_4x_5$	$y_1y_2y_3y_5y_6$	*init*4, *read*1,*write*0	*init*0,*init*1,*init*2, *RMACK*,*WMACK*, *read*0,*write*0
RMACK	x_3x_5	$y_1y_2y_3$	*IOwait*	*init*0,*read*0
WMACK	x_3	$y_1y_2y_5$	*IOwait*	*init*0,*write*0
*read*0	x_3	$y_1y_2y_3y_6$	*IOwait*, *RMACK*	*init*0,*read*1
*read*1	x_3	y_1y_2	*read*0	*init*0,*IOwait*
*write*0	x_3	y_1y_2	*IOwait*, *WMACK*	*init*0,*IOwait*

Additional inputs: $Z^1_I=\{z_1, z_2\}$, $Z^2_I=\{z_3, z_4, z_5, z_6\}$;

Additional outputs: $Z^1_O=\{z_3, z_4, z_5, z_6\}$, $Z^2_O=\{z_1, z_2\}$.

Table 4-10 and Table 4-11 present the transition tables for the component machines “opus_p” and “opus_add”.

Table 4-10 State transition and output table for the component machine “opus_p”

№	Present state	Input	Next state	Output
1	*st*0	x_3	*st*0	y_1y_2
2		$\neg x_3$	*st*1	y_1y_2
3	*st*1	x_3	*st*0	y_1y_2
4		$\neg x_3\neg x_4$	*st*1	y_1y_2
5		$\neg x_3x_4$	*st*2	$y_1y_2y_6$
6	*st*2	x_3	*st*0	y_1y_2
7		$\neg x_3$	*st*4	$y_1y_2y_4$
8	*st*3	x_3	*st*0	y_1y_2
9		$\neg x_3x_4$	*st*3	$y_1y_2y_4$
10		$\neg x_3\neg x_4$	*st*4	-
11	*st*4	$\neg x_1\neg x_2\neg x_3\neg x_4$	*st*0	y_1y_2
12		$x_1\neg x_2\neg x_3\neg x_4$	*st*1	y_1y_2
13		$\neg x_1x_2\neg x_3\neg x_4\neg x_5$	*st*2	$y_1y_2y_6$
14		$x_1x_2\neg x_3\neg x_4\neg x_5$	*st*4	-
15		$\neg x_1x_2\neg x_3\neg x_4\neg x_5$	st_{idle}	$\mathbf{z_3}\,y_1y_3$
16		$x_1x_2\neg x_3\neg x_4\neg x_5$	st_{idle}	$\mathbf{z_4}\,y_1y_5$
17		$\neg x_1x_2\neg x_3\neg x_4x_5$	st_{idle}	$\mathbf{z_5}\,y_1$
18		$x_1x_2\neg x_3\neg x_4x_5$	st_{idle}	$\mathbf{z_6}\,y_1$
19	st_{idle}	$\mathbf{z_1}$	*st*0	-
20		$\mathbf{z_2}$	*st*4	-
21		$\mathbf{\neg z_1\neg z_2}$	st_{idle}	-

Table 4-11 State transition and output table for the component machine "opus_add"

№	Present state	Input	Next state	Output
1	$st0$	x_3	st_{idle}	$\mathbf{z_1}\, y_1 y_2$
2		$\neg x_3$	$st4$	$y_1 y_3 y_6$
3	$st1$	x_3	st_{idle}	$\mathbf{z_1}\, y_1 y_2$
4		$\neg x_3$	st_{idle}	$\mathbf{z_1}$
5	$st2$	x_3	st_{idle}	$\mathbf{z_1}\, y_1 y_2$
6		$\neg x_3 x_5$	$st0$	$y_1 y_3$
7		$\neg x_3 \neg x_5$	$st2$	y_1
8	$st3$	x_3	st_{idle}	$\mathbf{z_1}\, y_1 y_2$
9		$\neg x_3 x_5$	$st0$	$y_1 y_5$
10		$\neg x_3 \neg x_5$	$st3$	y_1
11	$st4$	x_3	st_{idle}	$\mathbf{z_1}\, y_1 y_2$
12		$\neg x_3$	st_{idle}	$\mathbf{z_1}$
13	st_{idle}	$\mathbf{z_3}$	$st0$	-
14		$\mathbf{z_4}$	$st1$	-
15		$\mathbf{z_5}$	$st2$	-
16		$\mathbf{z_6}$	$st3$	-
17		$\mathbf{\neg z_3 \neg z_4 \neg z_5 \neg z_6}$	st_{idle}	-

The structure of the network under generalized additive decomposition

For the FSM $A=\{S,I,O,\delta,\lambda\}$ and the cover $\varphi=\{B^1,\ldots,B^n\}$, $B^i\subseteq S$, $s\in B^i \Leftrightarrow O(s)\cap O^i\neq\varnothing$ on the set of states of A in accordance to the pair (S, π_y) we define the network N with n component machines $A^i=\{S^i,I^i,O^i,\delta^i,\lambda^i\}$ [5]. The number of component machines is equal to the number of blocks in the cover φ.

The state s will be in the block B^i of cover φ if there is at least one output from the block B^i of the partition π_y at the transitions from this state. The same state s will be in several blocks of φ, for example, in B^i and B^j, if $O(s)\cap O^i\neq\varnothing$ and $O(s)\cap O^j\neq\varnothing$, i.e. the output from B^i and B^j are produced at the transitions from s.

In exact the same way, we define the cover φ_{tt} on the set of rows of the transition table of A in accordance to the pair (S, π_y): $\varphi_{tt}=\{B^1{}_{tt},\ldots,B^n{}_{tt}\}$, $B^i{}_{tt}\subseteq S$, $r\in B^i{}_{tt} \Leftrightarrow O(r)\cap O^i\neq\varnothing$. The row r will be in the block $B^i{}_{tt}$ of the cover φ_{tt}, if at least one output from the block $B^i{}_{tt}$ of the partition π_y is written in this row r. Just as for φ, the same row r will be in several blocks of φ_{tt}, for example, in $B^i{}_{tt}$ and $B^j{}_{tt}$, if $O(r)\cap O^i\neq\varnothing$ and $O(r)\cap O^j\neq\varnothing$, i.e. the output from $B^i{}_{tt}$ and $B^j{}_{tt}$ are written in the row r.

Component machine A^i is defined as follows:

$S^i=B^i\cup s_{idle}$ is the set of states, where B^i is the i^{th} block of the cover φ and s_{idle} is the additional state in A^i, such state exists in each component machine.

$I^i=I(B^i{}_{tt})\cup Z^i{}_I$ is the set of inputs in the component machine A^i, where $I\left(B^i_{tt}\right)=\bigcup_{r\in B^i_{tt}} I^i(r)$, $I^i(r)$ is the set of inputs in the row r of the FSM A transition table and $B^i{}_{tt}$ is the i^{th} block of the cover φ_{tt}.

$Z^i{}_I=\{z|\delta^i(s_i,x)=s,\ x\in I,\ s_i\notin B^i,\ s\in B^i\}$, x is the input signal at the transition from s_i to s, $s\in B^i$. The additional input z in the component machine A^i in accordance to each $s\in B^i$ exists if there is at least one transition to this state s from the state s_j not included in B^i in the FSM A.

$O^i=O(B^i)\cup Z^i{}_O$ is the set of outputs in the component machine A^i, where, $O^i(B^i)$ is the i^{th} block of the partition π_y.

$Z^i{}_O=\{z|\delta(s,x)=s_j,\ x\in I,\ s\in B^i,\ s_j\in B^j,\ s\notin B^i,\ (i\neq j)\}$. The additional output z in the component machine A^i in accordance to each $s\notin B^i$ exists if the following two conditions are concurrent:

- there is a transition from the state s included in B^i to the state s_j not included in B^i in the machine A;
- there is at least one block B^j $(i\neq j)$ such that s_j is included in B^j and is s not included in B^j among the blocks of the cover φ.

Next we define the transition δ^i and output λ^i functions in component machines. Assume that here is a transition from s to s_j with the input x and the output O^i in the FSM: $\delta(s_i,x)=s$; $\lambda(s_i,x)=O^i$. Consider the corresponding transitions in component machine. Let $\sum_i$ be the set of component machine with the state s_i, $\sum_j$ be the set of component machine with the state s_j, and $\sum_{ij}=\sum_i\cap\sum_j$ be the set of component machine with the states s_i and s_j. If $A^k\in\sum_{ij}$, then in A^k:

$\delta^k(s_i,x)=s$. If $A^i \in \sum_i \backslash \sum_{ij}$ (s_i is the state of A^i and s_j is not the state of A^i), then in A^i: $\delta^i(s_i,x)=s_{idle}$.

Output function for the machine $A^l \in \sum_i$ (s_i is the state of A^l and it is also possible that s_j is the state of A^l): $\lambda^l(s_i,x)=O^i \cap O^l$. Moreover, one and only one machine from $\sum_i$, for example A^r, must have the output signal z_r, which forces each machines $A^v \in \sum_i \backslash \sum_{ij}$ (if this set is not empty) to transit from the additional state s_{idle} to s_j. In the machine $A^r \in \sum_i$: $\lambda^r(s_i,x)= O^i \cap O^r \cup \{z_r\}$. If $A^m \in \sum_i \backslash \sum_{ij}$ (s_j is the state of A^m and s_i is not the state of A^m), then in A^m: $\delta^m(s_{idle},z_k)=s_j$; $\lambda^m(s_{idle},z_k)=y_0$.

Example, for the FSM "opus" and the decomposition cover $\varphi(S)=\{\{init0, init2, init4, write0, read1\}; \{init0, init1, IOwait, read0, write0, read1, RMACK, WMACK\}\}$ we define the network "opus" under generalized additive decomposition, Figure 4-4.

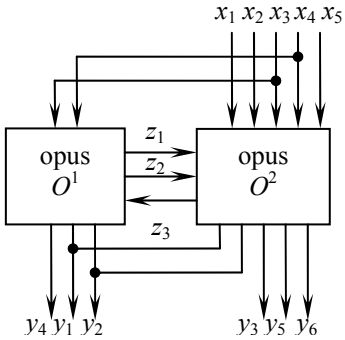

Figure 4-4 Structure of the network "opus" under generalized additive decomposition

Table 4-12 State transition and output table for the FSM "opus_O^1"

№	Present state	Input	Next state	Output
1	*st*0	x_3	*st*0	y_1y_2
2		$\neg x_3$	st_{idle}	$\mathbf{z_1}\, y_1y_2$
3	*st*1	x_3	*st*0	y_1y_2
4		$\neg x_3$	*st*2	$y_1y_2y_4$
5	*st*2	x_3	*st*0	y_1y_2
6		$\neg x_3x_4$	*st*2	$y_1y_2y_4$
7		$\neg x_3 \neg x_4$	st_{idle}	$\mathbf{z_2}$
8	*st*3	x_3	*st*0	y_1y_2
9		$\neg x_3$	st_{idle}	$\mathbf{z_2}$
10	*st*4	x_3	*st*0	y_1y_2
11		$\neg x_3$	st_{idle}	$\mathbf{z_2}$
12	st_{idle}	$\mathbf{z_3}$	*st*1	-
13		$\mathbf{\neg z_3}$	st_{idle}	-

$B^1(\varphi)$={*init*0,*init*2,*init*4,*write*0,*read*1},
$B^2(\varphi)$={*init*0,*init*1,*IOwait*,*read*0,*write*0,*read*1,*RMACK*,*WMACK*};
$B^1 \cap B^2$={*init*0,*write*0,*read*1}.
For the FSM "opus", Table 4-1, we have:
$B^1(\varphi_{tt})$={1,2,6,7,8,9,10,27,28,29,30},
$B^2(\varphi_{tt})$={1,2,3,4,5,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30};
$B^1 \cap B^2$={1,2,27,28,29,30}.
$B^1(s)$={*st*0,*st*1,*st*2,*st*3,*st*4,st_{idle}}, $B^2(s)$={*st*0,*st*1,*st*2,*st*3,*st*4,*st*5,*st*6,*st*7,st_{idle}}.
$I(B^1)=\{x_3, x_4\}$, $I(B^2)=\{x_1, x_2, x_3, x_4, x_5\}$.
$O(B^1)=\{y_1, y_2, y_4\}$, $O(B^2)=\{y_1, y_2, y_3, y_5, y_6\}$.
Additional inputs: $Z^1{}_I=\{z_3\}$, $Z^2{}_I=\{z_1, z_2\}$, outputs: $Z^1{}_O=\{z_1, z_2\}$, $Z^2{}_O=\{z_3\}$.
Transition tables for the component machines "opus_O^1"and "opus_O^2" are presented in Table 4-12 and Table 4-13.

Table 4-13 State transition and output table for the FSM "opus_O^2"

№	Present state	Input	Next state	Output
1	*st*0	x_3	*st*0	y_1y_2
2		$\neg x_3$	*st*1	y_1y_2
3	*st*1	x_3	*st*0	y_1y_2
4		$\neg x_3 \neg x_4$	*st*1	y_1y_2
5		$\neg x_3 x_4$	st_{idle}	$\mathbf{z_3}y_1y_2y_6$
6	*st*2	x_3	*st*0	y_1y_2
7		$x_1 \neg x_2 \neg x_3 \neg x_4$	*st*1	y_1y_2
8		$\neg x_3 x_4$	st_{idle}	$\mathbf{z_3}y_1y_2y_6$
9		$\neg x_1 \neg x_2 \neg x_3 \neg x_4$	*st*2	-
10		$\neg x_1 x_2 \neg x_3 \neg x_4 \neg x_5$	*st*5	y_1y_3
11		$x_1 x_2 \neg x_3 \neg x_4 \neg x_5$	*st*7	y_1y_5
12		$\neg x_1 x_2 \neg x_3 \neg x_4 x_5$	*st*3	y_1
13		$x_1 x_2 \neg x_3 \neg x_4 x_5$	*st*4	y_1
14	*st*3	x_3	*st*0	y_1y_2
15		$\neg x_3 x_5$	*st*5	y_1y_3
16		$\neg x_3 \neg x_5$	*st*3	y_1
17	*st*4	$\neg x_3$	*st*0	y_1y_2
18		$x_3 \neg x_5$	*st*7	y_1y_5
19		$\neg x_3 x_5$	*st*4	y_1
20	*st*5	x_3	*st*0	y_1y_2
21		$\neg x_3$	*st*6	$y_1y_3y_6$
22	*st*6	x_3	*st*0	y_1y_2
23		$\neg x_3$	*st*2	-
24	*st*7	x_3	*st*0	y_1y_2
25		$\neg x_3$	*st*2	-
26	st_{idle}	$\mathbf{z_1}$	*st*1	-
27		$\mathbf{z_2}$	*st*2	-
28		$\mathbf{\neg z_1 \neg z_2}$	st_{idle}	-

4.2.4 Encoding of the network of machines

The problem of assigning a binary code to all states of machine leads to the problem of assigning a binary code to the blocks of partitions. Extended logic optimization problem is to find an encoding of blocks of states of a machine with aim to modify the size of its minimum representation.

State encoding of the network of machines after multiplicative decomposition and state encodings in case of additive and generalized additive decompositions have one fundamental difference. When we assign a binary code to the states of decomposable FSM after multiplicative decomposition we are taking into consideration the fact that this state exerts on states of each component machine in the network.

In case of additive decomposition we apply independent encoding to each component machine in the network. It is because the state of decomposable FSM presents strongly only in one component machine.

We decompose an FSM using generalized additive decomposition we also encode each component machine independently. However in this case we have the situation when one state has the different code depending on the work of corresponding components.

State encoding of the network of machines after multiplicative decomposition

In [WsBP'04] we propose the encoding method for the network of machines based on the measure of information relationship and guarantee the minimum code length for given decomposition constraints. It is important to note that the solution of this task is tightly connected with classical combinatorial problem – face hypercube embedding [24].

We use the standard approach to define the problem of state encoding of the FSM which was described in the Chapter 3. State encoding of the network of machines can be modeled by a binary constraint matrix with $n_r \times n_s$ dimension, whose n_r rows are the optimized cover corresponding to the symbol set S under consideration. The matrix has as many columns as n_s symbols in the set of states S. The fundamentally difference between encoding of the FSM and the network of machines consists in meaning that the encoding of the network of machines should satisfy to decomposition constrains. Here we accentuate that we form the encoding partitions with stringent restriction from the complete system of partition (decomposition constraints). It means that we are definitely limited in generation of encoding partitions.

Encoding problem has always a solution satisfying the constraints [75]. The encoding of n_s symbols in a field may require more than $|log_2 n_s|$ bits. Hence the search for a minimum bit encoding is relevant. The problem of finding the minimum k and related encoding such that is decomposition constraints satisfied is called face hypercube embedding [14], [23], [24].

The constructing a Boolean hypercube with aim of state assignment (or encoding) of FSM is presented in [59]. The process is the sequence of n steps,

after which *n*-dimensional Boolean hypercube is obtained. The *n*-component Boolean vectors are assigned to the vertices of the hypercube where the neighborhoods relation between the vectors presented by the edges of the hypercube.

Approach of encoding of the network of machines presented in the current work is distinguished from [59] by two peculiarities. First is that we assigned each states of the network of machines to Boolean vectors, not the states of one FSM. Second peculiarity is that constructing a Boolean hypercube in our approach is with using the measure of informational relationship.

Our desire to evaluate the contents of information in encoding partitions with respect to information in the complete system of partitions is likewise the concept of information relationship by Jozwiak [31]. The information relationships and measures enable to analyze relationships between the modeled information streams and constitute an important analysis apparatus that can be used for analysis and synthesis of various information systems. In the field of fixed task, i.e. encoding of the network of machines, we are interesting in relationships between information in information streams described by coding partitions and completed information reflected by system of partitions.

Information in discrete systems is considered by values of some signals or variables which can be represented by a set system. A certain set system gives information about elements of a set on which this system was described. We deal with set system defined on some set as a compatibility relation.

When we think about relationships between information in information streams described by encoding partitions and complete system of partitions we are interesting in combined information about these streams. Encoding partition is the two-block partition on the set of states *S* of decomposable FSM and complete system of partitions is the set system on the set *S*. Then we interpret product of the set system and the partition as joint information [WsBP'04].

Example, for the network "opus" after multiplicative decomposition the encoding matrix is presented in Table 4-14.

Table 4-14 Encoding matrix for decomposable FSM "opus" after multiplicative decomposition and encoding matrix of the network "opus"

States	Codes
*init*0	1001
*init*1	000
*init*2	1011
*init*4	1101
IOwait	0001
Rmack	0101
Wmack	0100
*read*0	0011
*read*1	1111
*write*0	0010

Constraints	Codes
*c*0	0--0
*c*1	0--1
*c*2	1--1
*c*3	-00-
*c*4	-01-
*c*5	-10-
*c*6	1111

Independent state encoding of component machines in the network after additive and generalized additive decompositions

As a result of FSM decomposition stage, we receive the network of interacted and interconnect component machines. In contrast to the network of machines described in previous section in case of additive decomposition the network consists of components working alternatively in time. It means that all component machines except one are suspended in one of extra state – the "wait" or "idle" state. We do not need to keep in mind the information about global state of the network. In one particular period of time only one machine is active while the other machines are in "wait" state. It is gives us possibility to encode each machine in the network independently. Usually the number of states of component machines is less than the number of states of decomposable FSM. Hence we can find the encoding with shorter encoding length.

Next we apply the new low power state encoding method presented in Chapter 3 to the component machines in the network in case of additive and generalized additive decomposition.

Example, the encoding matrices after independent state encoding for the component machine "opus_p" and "opus_add" are presented in Table 4-15. Table 4-16 presents the same encoding for the component machines "opus_O^1" and "opus_ O^2".

Table 4-15 Encoding matrices for the component machines "opus_ p" and "opus_ add"

States	Codes
$st0$	010
$st1$	000
$st2$	001
$st3$	110
$st4$	100
st_{idle}	111

States	Codes
$st0$	101
$st1$	010
$st2$	001
$st3$	100
$st4$	110
st_{idle}	000

Table 4-16 Encoding matrices for the component machines "opus_O^1" and "opus_ O^2"

States	Codes
$st0$	000
$st1$	011
$st2$	---
$st3$	---
$st4$	---
st_{idle}	001

States	Codes
$st0$	0100
$st1$	0001
$st2$	0010
$st3$	0110
$st4$	1000
$st5$	1110
$st6$	1100
$st7$	1010
st_{idle}	0000

Optimization of the network under multiplicative decomposition

This step of decomposition procedure enables to optimize the network of machines [16] under multiplicative decomposition in terms of the number of internal binary connections. In [BEC'04] we illustrate that the proposed encoding technique allows minimizing the total number of internal binary variables between component machines by generation of new *M*-partitions [28] after encoding of the network of machines. $M(\pi_i)$ keeps amount of information needed for i^{th} component machine in the network. In other words, the operator $M(\pi_i)$ gives the maximum front partition of partition pair. Informally speaking, for a given partition π_i the partition $M(\pi_i)$ describes the least amount of information we must have about the present state of the machine to the next state (i.e., the block of π_i which contains the next state of the machine). If partition $M(\pi_i)$ is less or equal to multiplication of partitions from $\{\pi_i\}$ that it means that i^{th} component machine receives enough information from component machines with which it is connected accordingly relation of connection to compute the next state.

Example, we have selected the FSM "bbtas" [48]. To decompose the FSM we select an arbitrary complete system of two partitions: $[\pi_1=\{\{st0, st1\},\{st2, st3\},\{st4, st5\}\};\pi_2=\{\{st0, st2\},\{st1, st3\},\{st4\};\{st5\}\}]$.

The $M(\pi_i)$ are $M(\pi_1)=\{\{st0\}, \{st1, st2\}, \{st3\},\{st4\},\{st5\}\}$ and $M(\pi_2)=\{\{st0\},\{st1\},\{st2\},\{st3\},\{st4\},\{st5\}\}$. The structure of the network "bbtas" is depicted on the Figure 4-5. The internal binary connections are:

$z^1{}_1=\{\{st0, st1, st2, st3\},\{st4, st5\}\}$;
$z^1{}_2=\{\{st0, st1\}, \{st2, st3, st4, st5\}\}$;
$z^2{}_1=\{\{st0, st2, st4\},\{ st1, st3, st5\}\}$;
$z^2{}_2=\{\{st0, st2, st5\},\{ st1, st3, st4\}\}$.

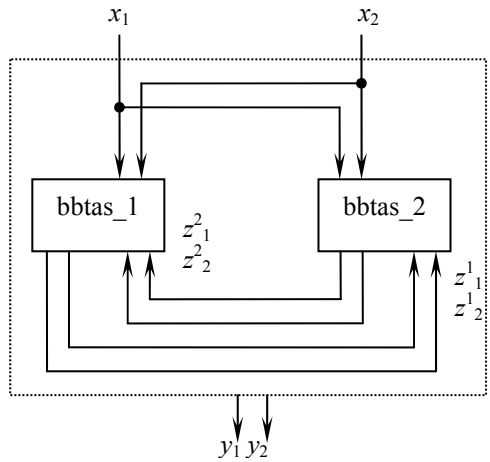

Figure 4-5 Structure of the network "bbtas" before optimization

We have applied the proposed encoding and as a result have received the encoding matrices, Table 4-17. The second matrix in the Table 4-17 presents the codes of the states of the network "bbtas" which are presented by blocks of

partitions from the complete system of partitions. The first three rows of the matrix correspond to blocks of the first partition and next four rows correspond to blocks of the second partition.

Table 4-17 Encoding matrices for decomposable FSM "bbtas"

States	Codes
*st*0	110
*st*1	010
*st*2	101
*st*3	001
*st*4	100
*st*5	000

Constraints	Codes
*c*0	-10
*c*1	-01
*c*2	-00
*c*3	1--
*c*4	0--
*c*5	100
*c*6	000

New encoding enables optimization of the network of machines in terms of number of internal binary connections by generation of new *M*-partitions: $M(\pi_1)$={{*st*0}, {*st*1,*st*2}, {*st*3}, {*st*4},{*st*5}} and $M(\pi_2)$={{*st*0,*st*3,*st*5}, {*st*1}, {*st*2}, {*st*4}}. The new realization of the "bbtas" network is depicted on the Figure 4-6. The internal binary connections are:

z^1_1={{*st*0, *st*1, *st*4, *st*5},{*st*2, *st*3}};
z^1_2={{*st*0, *st*1}, {*st*2, *st*3, *st*4, *st*5}};
z^2_1={{*st*0, *st*2, *st*4},{ *st*1, *st*3, *st*5}}.

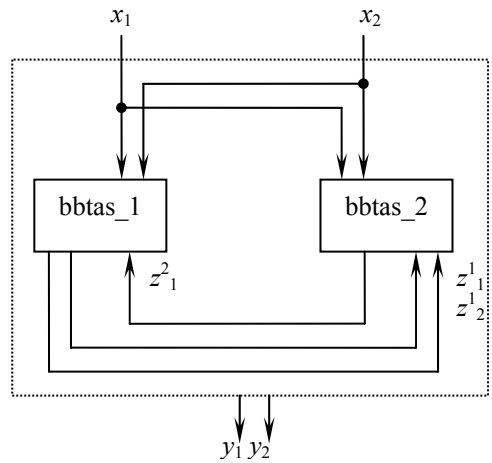

Figure 4-6 Structure of the network "bbtas" after optimization

4.3 Experimental results

The part of the experimental results reported in current subchapter has been presented in the articles mentioned in list of publications. All circuits are from industrial MCNC benchmarks suite [48]. Experiments were run on java applets on various aspects on decomposition and synthesis from the project Decomposition & Synthesis (D&S) [18]. The series of applets of the project D&S is a decomposition software system which is on the one hand a research tool and on the other hand it is an educational system [BEC'02], and [EUROCON'03].

The experiments carried on decomposition software system because of common standard of output files was supplemented with commercial design frames – SYNOPSIS and SIS: A System of Sequential Circuits System [63].

Stochastic investigation of an FSM

We have first report the stochastic investigation of FSM decomposition which has been done using FSMNet Stochastic Explorer Applet (available: http://www.pld.ttu.ee/applets/probability/). Applet represents an FSM or a FSM network stochastic exploration tool, providing user with a possibility to carry out some entropic evaluations.

To ensure that partition entropy is a good indicator of implementation complexity, experiments have been carried on hundreds of machines [35]. They proved that the correlation between of decomposition partition π_i and the complexity of sub-FSM A_i (area) are very high (more than 0,95). What makes entropy especially useful from decomposition point of view is fact, that partitions that are incomparable under the least upper bound and the largest lower bounds in classic lattice usually do have different entropy values [34], so they become comparable from the power consumption point of view.

Two experiments were performed [DDECS'02]:

- We show the difference D* between the maximum switching activity $\max(H^{sw}(A))$ and switching activity for assumption of equiprobability of primary input patterns $H^{sw}(A)$. This is done to emphasize that if we have not real distribution of inputs, the assumption of their eqiprobability (which is sometimes used) is doubtful for estimation.
- The second part is devoted to the two-component decomposition with estimates. The first sub-FSM corresponds to two-block partition of the set of states that has the minimum entropy by comparison with other partitions of the same rank. We can consider this partition of set space as the first step for iterative process of decomposition.

Table 4-18 Stochastic analysis and partition examples

Circuits	$(H^w(A))$	$\max(H^w(A))$	D*	(D*)%	$H^w(A_1)$	$H^w(A_2)$	$H^w(N_1)$
bbtas	2.4025	3.5850	1.1824	61.01	0.5313	1.9770	2.5083
beecount	1.2816	4.5236	3.2420	28.33	0.0543	1.2408	1.2951
dk15	1.7463	3.5850	1.8386	48.71	0.2502	1.5498	1.8000
dk27	2.6737	3.7004	1.0267	72.25	0.2762	2.5425	2.8187
lion	1.1983	3.3219	1.4036	57.75	0.9183	1.0000	1.9183
mc	1.4216	3.0000	0.5784	47.39	0.2285	0.6984	0.9269
s8	2.2566	3.7004	1.4439	60.98	0.8366	1.5219	2.3586

Complexity criteria based on state probability distribution

Table 4-19 summarizes of area estimation and power consumption for the set of selected benchmarks using commercial design frame (SYNOPSIS). These parameters were chosen as complexity criteria for decomposable system.

The complexity of the components depends not only on the number of states [10], but also on the number of inputs. Most sequential components have large state space that cannot be enumerated in a reasonable amount of time. However, the input distribution gives us the possibility to external control of machines behavior. Evidently that if we can mark out from all FSM inputs such that have the most informational content, we will have the techniques for external input division. Accordingly of this assumption, informational laden inputs direct the states which are compose the network of two components.

Table 4-19 Area and power estimation for decomposable machine

Circuits	States	Inputs	Cells	Area combin unit	Area noncombin unit	Power net switch (μW)
log	17	9	71	91	35	1,1147
dvram	35	8	100	136	42	1,1103
nucpwr	29	13	90	120	35	1,0246
sync	52	19	147	223	42	1,3254
planet	48	7	237	356	42	6,5247
ex6	8	5	72	107	21	2,9367
opus	10	5	54	79	28	2,6848
ex4	14	6	48	66	28	1,1102
rie	29	9	98	135	35	1,0383

An experiment for circuits in Table 4-20 illustrates the complexity parameters for first component. Table 4-21 presents the values of these parameters for second component machine.

Table 4-20 Area and power estimation for the first component machine

Circuits	State	Inputs	Cells	Area combin unit	Area noncombin unit	Power net switch (μW)
log	5	3	22	28	11	0,6682
dvram	6	3	21	29	9	0,5788
nucpwr	8	5	28	38	11	0,711
sync	18	6	50	76	15	0,9879
planet	17	2	82	123	15	4,964
ex6	7	2	50	75	15	2,9367
opus	3	1	15	22	8	2,2909
ex4	5	1	15	20	9	0,7084
rie	11	4	39	54	14	1,0383

Table 4-21 Area and power estimation for the second component machine

Circuits	State	Inputs	Cells	Area combin unit	Area noncombin unit	Power net switch (μW)
log	5	5	31	39	15	0,5210
dvram	8	5	31	42	13	0,5700
nucpwr	7	8	33	43	13	0,3899
sync	8	13	44	66	13	0,4134
planet	7	5	52	78	10	1,9657
ex6	2	3	28	42	9	0,0001
opus	4	4	26	37	14	0,7192
ex4	4	5	22	30	13	0,4501
rie	4	5	24	32	9	0,5098

The comparison of decomposition techniques

In the experiment was used tree applets of D&S: Multiplicative Decomposition (available: http://www.pld.ttu.ee/applets/dsa/), Additive Decomposition (available: http://www.pld.ttu.ee/applets/decS/), and Generalized Additive decomposition (available: http://www.pld.ttu.ee/applets/decO/).

Table 4-22 contains the results of comparative experiments of decomposition techniques and approaches. The area estimation was done using the commercial design frame (SYNOPSIS). This parameter was chosen for complexity criteria for decomposition system [EUROCON'03].

Table 4-22 Results of comparison and implementation

Circuits	Total # of states of component machines	Total # of states (alt. approach)	Area combinat ratio
log	12	19	0.75
dvram	14	37	0.50
nucpw	15	31	0.68
sync	26	54	0.67
planet	24	50	0.59
ex6	9	10	1.14
opus	7	12	0.78
ex4	9	16	0.75

The promising application of our technique is low power design of control-dominated discrete systems. The idea of partition for low power is that in behavioral descriptions of hardware, a small set of computation often accounts for most of the computational complexity as well as power dissipation. The decomposition focuses on power dissipation as the main criteria of design optimization. Techniques based on disabling the input/state registers when some input conditions are met have been proposed and shown to be among the most effective in reducing the overall switching activity in sequential circuits.

Our reasoning proceeds from the premise that the solution of the problem of FSM synthesis for low power can be reduced to the FSM decomposition with distributed primary output/input variables and appropriate synthesis of machines network. Results confirmed that it is possible to significantly reduce switching activity of implementation and that significant reduction in power consumption could be achieved without performance degradation.

State encoding of the network of machines

The experimental investigations described in this section present the efficiency of our new encoding algorithm of the network of machines [WsBP'04]. The experiment was done using *Network Encoding* function of Java Applet on Multiplicative Decomposition (available: http://www.pld.ttu.ee/applets/dsa/).

The strategy of the algorithm allows demonstrating the resources by two experiments. The first experiment is oriented to the encoding of the network of machines in general. The second experiment is practical application of the algorithm to the network of machines optimization.

The goal of the first experiment is to illustrate the efficiency of proposed algorithm for encoding of the network of machines. We compare the time of finding the best decision between exact and heuristic algorithm. Our goal is to execute both algorithms of full search with same tests. Tests are containing from 6 to 17 states. As can be seen from the Table 4-23 heuristic algorithms executes the full search about two times faster than algorithm without heuristics.

Table 4-23 Comparison of exact and heuristic algorithms

	Exact			Heuristic			Average	
States	#1	#2	#3	#1	#2	#3	Exact	Heuristic
6	0	0	0	0	0	0	0,00	0,00
7	0	0	0	0	0	0	0,00	0,00
8	3	3	3	0	1	1	3,00	0,67
9	6	4	1	1	1	0	3,67	0,67
10	3	1	6	1	1	1	3,33	1,00
11	4	8	1	2	3	1	4,33	2,00
12	29	29	13	6	10	1	23,67	5,67
13	27	91	10	7	31	1	42,67	13,00
14	27	33	74	7	14	34	44,67	18,33
15	65	6	73	46	2	23	48,00	23,67
16	23	Time out	44	17	212	37	-	88,67
17	58	Time out	Time out	11	112	177	-	100,00

The second experiment considers a problem to show an efficiency of the algorithm. As our goal is to find decision with minimal number of bits, we can compare the first answer found with our heuristics and minimal decision. As can be seen from the Table 4-24 the difference is not essential. However, the difference grows with increase in quantity of states.

Table 4-24 Comparison of the results of the algorithm and the shortest decision

	Found decision			Shortest decision			Difference
States	#1	#2	#3	#1	#2	#3	
6	3	3	3	3	3	3	0
7	3	3	3	3	3	3	0
8	4	4	4	4	4	4	0
9	4	4	4	4	4	4	0
10	4	4	4	4	4	4	0
11	5	5	4	5	5	4	0
12	5	5	5	5	5	4	1
13	6	5	4	5	5	4	1
14	6	6	5	5	5	5	2
15	6	5	5	6	5	5	0
16	6	5	6	6	5	6	0
17	5	6	6	5	5	5	2

Comparison among new encoding method and well-known encodings

Experiment was carried with aim to compare the standard encoding methods using in SYNOPSIS and proposed encoding for the network of machines. We compare three importance parameters: entropy of the network of machines, Table 4-25, area implementation, Table 4-26 (I – first component machine, II – second component machine), and power consumption, Table 4-27.

Table 4-25 Entropy of the network of machines

Circuits	*Binary*	*Gray*	*One Hot*	*New*
bbara	2,844	2,892	4,042	2,778
bbtas	2,878	2,733	3,746	2,602
beecount	1,283	1,770	2,144	1,283
cse	1,542	2,111	1,856	1,226
dk27	2,777	2,964	3,988	2,771
ex4	3,917	3,863	4,940	3,670
mark1	0,988	0,983	0,781	0,976
opus	1,714	2,577	2,825	1,714

Table 4-26 Area results

	Binary		*Gray*		*One Hot*		*New*	
Circuits	I	II	I	II	I	II	I	II
bbara	32	21	35	21	39	42	38	28
bbtas	20	21	31	21	23	42	21	21
beecount	34	21	32	21	32	42	30	21
cse	172	28	180	28	143	112	194	28
dk27	20	21	22	21	19	49	20	21
ex4	66	28	64	28	30	98	67	28
mark1	65	28	61	28	52	98	70	28
opus	79	28	68	28	56	63	79	28

Table 4-27 Power consumption results (μw)

Circuits	*Binary*	*Gray*	*One Hot*	*New*
bbara	0,8505	0,9126	0,9766	0,8116
bbtas	0,6952	0,7068	0,8992	0,6176
beecount	0,9856	1,0394	1,1287	0,9556
cse	3,1439	3,6804	3,7449	2,9881
dk27	0,5751	0,5908	0,5925	0,5489
ex4	1,1102	1,1036	1,0572	0,9025
mark1	1,2315	1,2684	1,5245	1,0618
opus	2,6848	2,2019	2,0264	2,4876

Optimization of the network of machines

Through the experiment was used Java Applet on Multiplicative Decomposition (available: http://www.pld.ttu.ee/applets/dsa/). For each benchmark circuit we generated the system of two partitions with the maximum (or close by maximum) number of constraint with aim to emphasize how the coding length is increased at the decomposition by using the standard coding methods using in SIS. We decompose a prototype N state machine into two interacting component machines with N_1 and N_2 states such that $\lceil logN \rceil < \lceil logN_1 \rceil + \lceil logN_2 \rceil$. The main condition of such partitioning is that the sum of the number of constraints must be less or equal to the number of states of source FSM. Table 4-28 summarizes the results obtained using proposed encoding algorithm.

Table 4-28 Comparison of the results of the algorithm

Circuits	#states	$\#constr_{\Sigma}$	$\#len_{\Sigma}$	$\#len_{new}$	reduc (%)
cse	16	15	7	5	28,6
dk16	27	21	8	6	25,0
donfile	24	23	8	6	25,0
ex1	20	18	6	5	16,7
ex4	14	12	6	4	33,3
keyb	19	18	7	5	28,6
kirkman	16	15	7	5	28,6
planet	48	37	10	7	30,0
pma	24	22	8	6	25,0
s1	20	19	8	6	25,0
s208	18	15	7	6	14,3
s420	18	16	7	5	28,6
s510	47	42	10	7	30,0
s820	25	22	8	6	25,0
s832	25	24	8	6	25,0
sand	32	30	9	6	33,3
tbk	32	31	9	7	22,2

In the Table 4-28, #states is the number of states in FSM, $\#constr_{\Sigma}$ is the total number of constraints in the first and in the second partitions respectively, $\#len_{\Sigma}$ is the length of encoding need to code the sum of generated constraint, $\#len_{new}$ is the code length in the new encoding algorithm and the last column reduc (%) is reduction of coding length in percents. From the table it can be seen that using the encoding algorithm results in average reduction of coding length of 24%. The results are practically good for circuits with larger number of states. Thus, for circuit's planet and s510 the reduction of coding length is up to 30%.

4.4 Summary

Step by step procedure for the construction of the network of component machines is described in detail. The methodology of global and local transformations gives opportunity to divide complicated process of decomposition into two sequential stages.

The stage of global transformations characterizes the mechanism for rough decomposition of the source FMS into the network of component machines. The mechanism of decomposition based on the theory of partition pair algebra determines by a complete system of partitions, a partition, or a cover on the set of states of a decomposable FSM. Using this tool we have proposed three variants of restrictions applied on the network of machines. First is the restriction on inputs separation between component machines in the network. Second is the restriction on computational complexity of component machines according to state probability distribution of the decomposable machine. Third is the restriction on outputs separation between component machines. As well as we add the restriction on the number of component machines in the network – no more than two components in the network. The step of global transformations also determines the structure of the network of component machines.

Local transformations consist of encoding of component machines, optimization of the network by reducing the number of internal binary variables and defining the basis of the networks. In accordance to the structure of the network we have proposed two encodings – composite encoding of the network of machines and independent encoding of each component machines. In case of encoding of the network of machines we complete the procedure of decomposition by stage of optimization of the network.

Multiplicative, additive and generalized additive FSM decomposition techniques are presented. Multiplicative decomposition uses a complete system of partitions can be regarded as a general case of decomposition based on the theory of partition pair algebra. At that time additive and generalized additive decompositions are special cases of general decompositions.

Additive decomposition which is built on a decomposition partition on the set of states of a decomposable machine can be replaced by multiplicative decomposition where a complete system of partitions built in following way: the first block of a partition is replaced in the first partition in which other block consist of states not included in the first block; the second block of a partition is replaced in the second partition and etc. In other words, we replace a partition by a complete system of partitions in which the number of partitions is equal to the number of blocks in a decomposition partition.

The component machines defined by partitions from the complete system of partitions are interconnected and interacted machines that will be work simultaneously or in series. In case of additive or generalized additive decomposition the network will be consist of component machines such that only one of the component machine is active in time period while other component machines wait in idle state.

Decomposition strategies target low power FSM

We have described three power-managed FSM decomposition strategies: decomposition with a separation of inputs, decomposition based on probability distribution and decomposition with a separation of outputs. These strategies are based on the classic decomposition theories. We have extended it and have developed the framework employ it for power optimization.

Decomposition of FSM with separation of inputs of an original FSM allows separating so called "active" and "lazy" inputs [64]. By "active" inputs we mean primary inputs with high probability. The method of separation of inputs of decomposable machine based on the concept of α-partitions allows moving "lazy" inputs away from the selected component machine. It is important to note that the calculation of α-partitions has polynomial computational complexity. The results confirmed that significant reduction in power consumption could be achieved using proposed methodology of primary inputs separation.

The idea of optimizing complex digital systems based on probabilistic analysis of an FSM has been extensively exploited [57]. The main challenge in the implementation of these techniques is to effectively partition a design in such a way that commonly executed computations can follow a highly optimized path without being slowed down by the circuitry needed for dealing with all corner cases [10]. In this work we introduce the technique for identifying the most probable behaviors in FSM and building a dedicated logic block that correctly implements such behaviors. Presented concept can be exploited for power optimization. Decomposition techniques reported sizable power reduction [42]. Using additive decomposition we present an source FSM as a network of component machines such that only one machine is active. When one component machine identifying with most probable states of the source FSM is enables the other machine (or machines) is "frozen", thereby nullifying switching activity.

Separation of outputs by decomposition of an original FSM leads to several important advantages. We consider the decomposition technique for controller and data-path simultaneously. The decomposition procedure that can be applied for an FSM with Data-path (FSMD model) [68] was considered. Output partitioning can be regarded as functional partition approach for low-power synthesis at RT-level. The reasons why FSMD functional partitioning can significantly reduce the switching activities at the registers and the functional modules and only one (subset) of machines is (are) executing a computation at any given time while the other processors will be idle. In addition to reducing power, FSMD functional partitioning also provides solutions to a variety of synthesis problems. Comparative experiments and approaches used in [29], [40], and [53] showed that such architecture need considerably less implementation area. Moreover, decomposition with distribution of outputs among component machines (corresponding to the given cover on the set of states of decomposable FSM) does not possible to achieve using alternative decomposition approaches.

5 CONCLUSION AND FUTURE DIRECTION

5.1 Thesis summary

The research described in the thesis concentrates on the problem of reducing the dynamic power dissipation in synchronous sequential circuits modeled by FSM. The most popular technique to reduce power in an FSM is to modify the state encoding with aim to minimize the Hamming distance of the most probable state transitions. Other idea for a low power FSM is the use of power management. That is, to shutdown the blocks of hardware in these periods where they are not producing useful data. Shutdown can be fulfilled in three ways: by turning off the power supply, by disabling the clock signal, or by "freezing" (blocking) the input data. Under the last category falls FSM decomposition method. The basic idea of decomposition is to disable the inactive part of an FSM. The deactivation is reached either by blocking the inputs or power-down by clock gating the part of the circuit that is not used. This reduces switching activity and hence, the total power dissipation.

The importance of the synthesis of sequential circuits for low power is considered in the first chapter of the thesis. The second chapter consists of main concepts from the machine theory and from the algebraic structure theory of sequential machines. The next two chapters concentrate on the corresponding area of research and describe the results obtained by the author. This chapter summarizes some particular conclusions, introduces the general conclusion of the thesis and outlines some suggestions to improve the presented tool and methodologies.

The main results of the work are:

1. An overview of state encoding methods to reduce the power consumption for sequential circuit modeled by FSM has been presented. The reduction of the average switching activity of the state variables is minimizing the number of bit changes during state transitions. The problem of finding of an appropriate state encoding for a low power FSM is connected with the problem of Minimum Weighted Hamming Distance. Hence, the main result is the development of a new heuristic method of state encoding with aim to minimize weighted hamming distance.
2. A novel technique for FSM state encoding with aim to minimize the number of states variables that change their value when FSM moves between two adjacent states was developed. At the heart of the presented approach lies the strategy of constructing a set of edge cuts for a set of states of an FSM. Each edge cut corresponds to an encoding partition on the set of states of an FSM and to one bit in a binary state encoding matrix.
3. The presented technique was updated to the problem of state encoding for a low power FSM that links to a probabilistic description of an FSM. The switching probability (or transition probability) has a good

approximation to the average switching activity that is proportional to the average power dissipation.

4. Basing on the overview, the subset of state-of-the-art encoding methods was selected to compare with the proposed technique. Experimental results were conducted on a set of MCNC benchmark circuits. For all benchmarks our state encoding produced circuits with lees (or equal in several cases) switching activity that selected methods.
5. An overview of FSM decomposition methods has been described. Recent attempts using decomposition for a low power FSM realization was classified. More general, an original FSM decomposed into a set of state machines interacting with each other and running concurrently. When machines have a self-loop clock and primary inputs are disabled for the respective machine/machines, therefore several of them require primary input disabling AND gates and clock disabling AND gate as well. In this case could be obtained significant power reduction along with area reduction. In other case, an STG is partitioned into several pieces, each piece being implemented as a separate machine with a wait state. Only one of the sub-machines is active and other sub-machines are in the resent state. Therefore, the clock for inactive sub-machines can be gated and primary inputs can be disabled which reduces the switching activity and hence total power dissipation.
6. A new FSM decomposition procedure was presented. The proposed procedure based on the concepts of global and local transformations of an FSM during its partitioning. The step of global transformations allows determining the wishful structure of the network of machines. The step of local transformations gives possibility of component-wise optimization, including state encoding of the network of machines. We would like to emphasize that the current work more focus on the construction of a network of machines for the given decomposition; and less concentrated on the finding of such decompositions. Nevertheless, several variants of decomposition for a low power FSM were proposed.
7. Experimental results have been obtained of a set of MCNC benchmark circuits using java-based applets of the project D&S. Looking at the empirical results; one may deduce that our framework of decomposition for a low power FSM is enough comprehensive tools for the complete investigation of various decomposition styles.

General conclusion of the thesis is that the problem of low power synthesis of an FSM is tightly connected with the classical combinatorial problems – FSM decomposition and FSM state encoding. Therefore, a new heuristic to FSM state encoding has been presented. A novel approach to FSM decomposition has been elaborated.

An estimation of the success of the work can be done by analyzing the practical application of the work. The key to the advantage of this work is a combination of a novel heuristic approach with well-known approved techniques and methods.

5.2 Future work

The thesis addressed the synthesis of FSM targeted low power dissipation. The problem was considered from two aspects. We proposed an FSM state encoding approach aiming at reducing the switching activity and combined it with one of dynamic power management techniques. From experimental analysis of the investigation presented in the thesis follows the practical efficacy. This fact justifies our strategy to solve the former and extend it later to the later.

We want to generalize the existing FSM state encoding approach and algorithm in the following directions:

- Solve the problem of finding the optimal state encoding solution due to insignificant increasing the number of state registers, this is an important practical problem,
- Solve the problem of finding state encoding for the FSM with a large number of states.

The other issue is a proposal for optimization of an approach for hierarchical test generation for FSM after low power state encoding. An ongoing work in this field have been presented in [EWDTW'05b]. We have made an attempt to apply the proposed strategy to test generation that is used on two levels: behavioral level in terms of an FSM and gate level. We hope that the knowledge, which was obtained after low power encoding, will help to enhance test generation of a low-power FSM and decrease test generation time.

Elaboration of decomposition procedures based on quality relationship measures also deserves further investigation. We plan to study both global and local measures.

An interesting direction for future work is to design the Globally Asynchronous Locally Synchronous (CALS) system [89]. Synchronous digital design is approaching a critical point, with clock distribution becoming an increasingly costly and complicated issue, and power consumption rapidly emerging as a major concern. Asynchronous digital design styles promise to liberate digital systems from clock skew problems, offer the potential for low power and high performance, and encourage a modular design philosophy. The preliminary analysis confirms that power consumption could be achieved without essential performance degradation. It is promising that the GALS paradigm could be used for composing blocks specified as FSM and making them communicate asynchronously to avoid the difficult and power consuming task of distributing the global clock to all parts of the circuits. Our future task is to elaborate partitioning techniques of state-based descriptions targeting the network of synchronous units which are interacting asynchronically.

Future work also involves extending the design automation in the educational system using opportunities of education via Internet [79].

In conclusion, numerous challenging problems and open issues pave the road towards system level tools for power and performance optimization, but we believe that this remains the primary research direction for the next few years.

REFERENCES

[1] Abdollahi A., and M. Pedram, "*Low power RT-level synthesis techniques: a tutorial*", to appear in IEEE Proceedings on Computers and Digital Techniques, 2005.

[2] Ashar P., S. Devadas and A. R. Newton, "*Testability Driven Synthesis of Interacting Finite State Machines*", (ICCD'90), IEEE International Conference of Computer Design: VLSI in Computers and Processors. Proc., pp. 273-276, 1990.

[3] Ashar P., S. Devadas and A. R. Newton, "*A Unified Approach to the Decomposition and Re-decomposition of Sequential Machines*", in Proc. of the 27th Design Automation Conference (DAC'90), pp. 601-606, 1990.

[4] Baccheta P., L. Daldoss, D. Sciuto and C. Silvano, "*Lower-Power State Assignment Techniques for Finite State Machines*", IEEE International Symposium on Circuits and Systems (ISCAS'00), pp.II-641-II-644, 2000.

[5] Baranov S. *Logic Synthesis for Control Automata*, Kluwer Academic Publishers, 1994.

[6] Benini L. and G. De Micheli, "*State Assignment for Lower Power Dissipation*", IEEE Journal of Solid State Circuits, vol. 30, no 3, pp. 258-268, 1995.

[7] Benini L., G. De Micheli, E. Macii, M. Poncino, and R. Scarsi, "*Symbolic Synthesis of Clock-gating Logic for Power Optimization of Control-oriented Synchronous Networks*", in Proc. European Design and Test Conf., pp. 514-520, 1997.

[8] Benini L, "*Automatic Synthesis of Sequential Circuits for Low Power Dissipation*", Ph.D. Thesis, Dept. of Electrical Engineering Stanford University, 1997.

[9] Benini L., G. De Micheli, and F. Vermeulen, "*Finite State Machine Partitioning for Low Power*", in Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS'98), pp. 5-8, 1998.

[10] Benini L., G. De Micheli, A. Lioy, E. Macii, G. Odasso, and M. Poncino, "*Synthesis of Power-managed Sequential Components Based on Computational Kernel Extraction*", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transaction on, vol. 20, No. 9, pp. 1118-1130, 2001.

[11] Cheng K.T. and V.D. Agrawal, "*An Entropy Measure for the Complexity of Multi-output Boolean Functions*", Proc. of the 27th Design Automation Conference (DAC'90), pp. 302-305, 1990.

[12] Chow S-H., Y-C. Ho, and T. Hwang, "*Low Power Realization of Finite State Machines – A Decomposition Approach*", Design Automation of Electronic Systems, ACM Transactions on, vol. 1, No. 3, pp 315-340, 1996.

[13] De Micheli G., R. Brayton, A. L. Sangiovanni-Vincentelli "*Optimal State Assignment for Finite State Machines*", IEEE Transactions on Computer-Aided Design, Vol. CAD-4, no.3, pp 269-285, 1985.
[14] De Micheli G., "*Synthesis and Optimization of Digital Circuits*", McGraw-Hill, 1994.
[15] Devadas S., and A.R. Newton, "*Decomposition and Factorization of Sequential Finite State Machines*", IEEE Trans. on Computer-Aided Design, CAD-8, no. 11, pp.1206-1217, 1989.
[16] Devadas S., "*Optimizing Interacting Finite State Machines Using Sequential Don't Cares*", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 10, pp.1473-1484, 1991.
[17] Devadas S., H-K. Ma, A.R. Newton, A. Sangiovanni-Vincentelli, "*MUSTANG: State Assignment of Finite State Machines Targeting Multilevel Logic Implementation*", IEEE Trans. on Computer-Aided Design, CAD-7, no. 12, pp.1290-1300, 1998.
[18] D&S Applets. Available: http://www.pld.ttu.ee/dildis
[19] Douglas B. West, *"Introduction to graph theory"* (2ed). Upper Saddle River: Prentice Hall, 2001.
[20] Du X., G. Hachtel, B. Lin and R. Newton, "*MUSE: A Multilevel Symbolic Encoding Algorithm for State Assignment*", IEEE Trans. on CAD/ICAS, vol. CAD-10, no.4, pp. 28-38, 1991.
[21] Eggermont R, "*PROSA: Profiling-based State Assignment for Low Power Dissipation*", MSc Thesis, Delft University of Technology, Netherlands, 2003.
[22] Geiger M. and T. Müller-Wipperfürth, "*FSM Decomposition Revisited: Algebraic Structure Theory Applied to MCNC Benchmark FSMs*", 28th ACM/IEEE Design Automation Conference (DAC'91), pp. 182-185, 1991.
[23] Goldberg E.I., T. Villa, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "*A Fast and Robust Exact Algorithm for Face Embedding*", Intl. Conf. on Computer-Aided Design (ICCAD'97), pp. 296-303, 1997.
[24] Goldberg E.I., T. Villa, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "*Theory and Algorithms for Face Hypercube Embedding*", IEEE Trans. on Comp., vol. 17, no 6, pp. 472-488, 1998.
[25] Hachtel G., E. Macii, A. Pardo and F. Somenzi, "*Probabilistic Analysis of Large Finite State Machines*", 31st ACM/IEEE Design Automation Conference (DAC'94), pp. 270-275, 1994.
[26] Hachtel G., E. Macii, A. Pardo and F. Somenzi, "*Markovian Analysis of Large Finite State Machines*", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transaction on, vol. 15, No. 12, pp. 1479-1493, 1996.
[27] Hassoun, S., Sasao, T., editors. *Logic Synthesis and Verification*, Kluwer Academic Publishers, 2002.

[28] Hartmanis J. and R. E. Stearns, "*Algebraic Structure Theory of Sequential Machines*", Englewood Cliffs, N. J.: Prentice-Hall, 1966.

[29] Hwang E., F. Vahid, and Y.-C. Hsu, "*FSMD Functional Partitioning for Low Power*", in Proc. DATE Conf., pp.22-28, March 1999.

[30] Jozwiak L., J.C. Kolsteren, "*An Efficient Method for the Sequential General Decomposition of Sequential Machines*", Micro-processing and Microprogramming, North Holland, vol. 32, pp. 657-664, 1991.

[31] Jozwiak L., "*Information Relationships and Measures in Application to Logic Design*", Proc IEEE Int. Symposium on Multiple-Valued Logic, pp. 228-235, 1999.

[32] Jozwiak L. and A. Chojnacki, "*High-quality Sub-function Construction in Functional Decomposition Based on Information Relationship Measures*", Design, Automation and Test in Europe (DATE'01), pp.383-390, 2001.

[33] Jozwiak L. and A. Chojnacki, "*Effective and Efficient FPGA Synthesis through Functional Decomposition Based on Information Relationship Measures*", Proc. Euromicro Symposium on Digital Systems Design (DSD'01), pp. 30-37 2001.

[34] Keevallik A., and T. Lausmaa, "*Informational Operators of Finite Automata*". Transaction of Tallinn Technical University, vol.708, pp. 20-25, 1990.

[35] Keevallik A., M. Kruus, J. Udre, "*Informational Modeling of FSM Networks*", MIXDES, Poznan, pp. 167-172, 1997.

[36] Koegest M., O. Coudert, and ST. Rülke, "*A Generalized Constraint-Driven State Encoding Strategy* ", in Proc. EUROMICRO'99, 1999.

[37] Koegest M., ST. Rülke, H. Sübe, and I. Lemberskis, "*Derivation of Constraints for Lower Power State Encoding*", in Proc. 42nd Riga Technical University Conference, Section on Electronics and Telecommunications (RTUCET'01), pp.107-112, 2001.

[38] Koegest M., S. Rülke, G. Franke and M. J. Avedillo, "*Two Criteria Constraint-Driven FSM State Encoding for Lower Power*", IEEE, pp.94-99, 2001.

[39] Lee W-K., and Tsui C-Y., "*Finite State Machine Partitioning for Low Power*", in Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS'99), pp. 306-309, 1999.

[40] Lee M-H., Hwang T-T., and Huang S-Y., "*Decomposition of Extended Finite State Machine for Low Power Design*", in Proc. of the IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE'09), 2003.

[41] Macii E., "*Sequential Synthesis and Optimization for Lower Power*", Lower Power Design in Deep Submicron Electronics,

NATO ASI Series, Series E: Applied Sciences, vol. 337, pp. 321-353, 1997.

[42] Macii E., M. Pedram and F. Somenzi, "*High-level Power Modeling, Estimation and Optimization*", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transaction on, vol. 17, No. 11, pp. 1061-1079, 1998.

[43] Marculescu D., R. Marculescu and M. Pedram, "*Stochastic Sequential Machine Synthesis Targeting Constrained Sequence Generation*", Proc. of the 33rd ACM/IEEE Design Automation Conference (DAC'96), pp. 696-701, 1996.

[44] Marculescu D., R. Marculescu, and M. Pedram, "*Information Theoretic Measures for Power Analysis*", IEEE Trans. Computer-Aided Design, vol. 15, pp. 599-610, 1996.

[45] Marculescu D., "*Information Theoretic and Probabilistic Measures for Power Analysis of Digital Circuits*", Ph. D. Thesis, Dept of Electrical Engineering University of Sourthen Calofornia, 1998.

[46] Martinez M., M. J. Avedillo, J. M. Quintana, and J. L. Huertas, "*A Dynamic Model for the State Assignment Problem*", DATE'98, pp. 835-839, 1998.

[47] Martinez M., M. J. Avedillo, J. M. Quintana, M. Koegest, S. Rülke, and H. Suesse, "*New Lower Power State Assignment Approach*", Design of Circuits and Integrated Systems Conference (DCIS'00), pp. 181-187, 2000.

[48] McElvain K., LGSynth'93 Benchmark Set: Version 4.0. Available: http://www.cbl.ncsu.edu/benchmarks, 1993.

[49] Monteiro J. C, and S. Devadas, "*Techniques for the Power Estimation of Sequential Logic Circuits under Use-Specified Input Sequence and Programs*", in Proc. Intl. Symp. Lower Power Design, 1995.

[50] Monteiro J. C, Devadas S., and Glosh A., "*Sequential Logic Optimization For Lower Power Using Input-disabling*", IEEE Trans. Computer-Aided Design, vol. 17, no 3, pp. 279-284, 1998.

[51] Monteiro J. C. and A. L. Oliveria, "*Finite State Machine Decomposition for Low Power*", Design Automation Conference (DAC'98) pp. 758 – 763, 1998.

[52] Monteiro J. C. and A. L. Oliveria, "*FSM Decomposition by Direct Circuit Manipulation Applied to Low Power Design*", in Proc. Asia and South Pacific Design Automation Conference (ASP-DAC'00) pp. 351-358, 2000.

[53] Monteiro J. C. and A. L. Oliveria, "*Implicit FSM Decomposition Applied to Lower-Power Design*", IEEE Trans. VLSI Syst., vol 10, pp. 560-565, 2002.

[54] Najm, Goel, and Hajj, "*Power Estimation in Sequential Circuits*", Proc of the 32th DAC, pp. 635-640, 1995.

[55] Nőth W. and R. Kolla, "Spanning Tree Based State Encoding for Lower Power Dissipation", Technical report, Dept. of Computer Science, University of Wűrzburg, 1998.

[56] Olson E. and S. M. Kang, "*Low-Power State Assignment for Finite State Machines*", in Proc. of Int. Workshop on Low Power Design, pp. 63-68, 1994.

[57] Pedram M., "*Power Minimization in IC Design: Principles and Applications*", ACM Trans. Design Automat. Electron. Syst., vol.1, pp. 3-56, 1996.

[58] Perkowski M., "*Digital Design Automation: Finite State Machine Design*", Record of Northcon'86, pp. 11/0.1 - 11/0.14, Seattle, 1986.

[59] Pottosin Yu. V. "*Assembling: A Boolean hypercube: an approach to state assignment of finite state machines*", in Proc. of the Second Int. conference CAD DD'97, Minsk, pp. 54-59, 1997.

[60] Roy S., P. Banerjee, and M. Sarrafzadeh, "*Partitioning Sequential Circuits for Low Power*", VLSI Design, Proc. 8th International Conference, pp. 212 – 217, 1998.

[61] Roy K., and S. Prasad, "*Syclop: Synthesis of CMOS Logic for Low-Power Application*", in Proc. of Int'l Conf. on Computer Design, pp. 464-467, 1992.

[62] Selvaraj H., M. Rawski, and T. Luba, "*FSM Implementation in Embedded Memory Blocks of Programmable Logic Devices Uses Functional Decomposition*", in Proc. Intl. Conf. on Information Technology: Coding and Computing, IEEE Comp. Society, pp. 355-360, 2002.

[63] Sentovich E., K. Singh, et. al. "*SIS: A System of Sequential Circuits Synthesis*", Tech. Rep. M92/41. Electronic Research Laboratory. Colleague of Engineering. University of California, Berkley, 1992.

[64] Shelar R., M. Desai, and H. Naraian, "*Decomposition of Finite State Machines for Area, Delay Minimization*", Proc. IEEE Conf. on Computer Design (ICCD'99), pp. 620-625, 1999.

[65] Shelar R., H. Narayanan, and M. P. Desai, "*Orthogonal Partitioning and Gate Clock Architecture for Lower Power Realizations of FSMs*", IEEE Int. ASIC/SOC Conference, pp. 266-270, 2000.

[66] Silvano C., "*Power Estimation and Optimization Methodologies for Digital Circuits and Systems*", Ph.D. Thesis in Information Engineering, Università degli Studi di Brescia, 2000.

[67] Story J.R., H.J. Harrison, E.A. Reinhard, "*Optimum State Assignment for Synchronous Sequential Circuits*", IEEE Trans. on Comp., vol. c-21, no 12, pp. 1365-1373, 1972.

[68] Sudnitson A., "*Register Transfer Low Power Design Based on Controller Decomposition*", in Proc. IEEE 24th International

Conference on Microelectronics (MIEL 2004), Nis, Yugoslavia, pp.735-738, 2004.

[69] Surti P. and L. F. Chao, "*Lower Power FSM Design Using Huffman-Style Encoding*", IEEE European Design and Test Conference (EDTC-97), pp. 521-525, 1997.

[70] Sutter G., E. Todorovich, S. Lopez-Buedo and E. Boemo, "*FSM Decomposition for Lower Power in FPGA*", Lecture Notes in Computer Science, Vol.24, no.38, pp.350-359, 2002.

[71] Tsui C-Y, M. Pedram and A. M. Despain, "*Efficient Estimation of Dynamic Power Dissipation with an Application*", in Proc. ACM/IEEE Intl. Conference on Computer-Aided Design, pp.224-228, 1993.

[72] Tsui C-Y, M. Pedram and A. M. Despain, "*Lower-Power State Assignment Targeting Two- and Multilevel Implementation*", IEEE Trans. on Computer-Aided Design, vol. 17, no 12, pp.1281-1291, 1998.

[73] Tyagi A., "*Entropic Bounds on FSM Switching*", IEEE Trans. VLSI Syst., vol. 5, pp. 456-464, 1997.

[74] Venkataraman G., Reddy S.M., Pomeranz I., "*GALLOP: Genetic Algorithm Based Lowe Power FSM Synthesis by Simultaneous Partitioning and State Assignment*", 6th Intl. on VLSI Design, 2003.

[75] Villa T., "*Encoding Problems in Logic Synthesis*", PhD Thesis in Electrical Engineering and Computer Science, University of California at Berkley, 1995.

[76] Yuan L-P., C-C. Teng, and S-M. Kang, "*Statistical Estimation of Average Power Dissipation in Sequential Circuits*", 34th Design Automation Conference (DAC'97), pp. 377-382, 1997.

[77] Wu X., M. Pedram, and L. Wang, "*Multi-code state assignment for low power design*", IEEE Proceedings - Circuits, Devices and Systems, Vol. 147, No. 5, pp. 271-275, 2000.

[78] Wu X., and M. Pedram, "*Low power sequential circuit design by using priority encoding and clock gating*", in Proc. of Symp. on Low Power Electronics and Design, pp. 143-148, 2000.

[79] Wuttke H.-D., Henke K., R. Peukert, "*Internet Based Education: An Experimental Environment for Various Educational Purposes*", in Proc. of the IASTED Int. Conf. on Computers and Advanced Technology in Education, IASTED/Acta Press No. 292, pp. 50-54, 1999.

OTHER BIBLIOGRAPHY

The list of additional bibliography that was also used during the preparation of the thesis is presented below.

[80] Ashar, P., Devadas, S., and Newton, A. R. *Sequential Logic Synthesis*. Boston: Kluwer Academic Publishers, 1992.

[81] Brassard G., P. Bratley, "*Fundamentals of Algorithms*", pp. 136-142, 291-292.

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

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

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

[85] Marculescu D., R. Marculescu and M. Pedram, "*Sequence Compaction for Probabilistic Analysis of Finite State Machines*", in Proc. of the 34th ACM/IEEE Design Automation Conference (DAC'97), pp.12-15, 1997.

[86] Martinez M., M. J. Avedillo, J. M. Quintana, and J. L. Huertas, "*An Algorithm for Facet-Constrained Encoding of Symbols Using Minimum Code Length*", DATE'99, pp. 521-525, 1999.

[87] Papoulis A., "*Probability, Random Variables, and Stochastic Processes*", McGraw-Hill Co., 1984.

[88] Ross K. A. and C. R. B. Wright, "*Discrete Mathematics*", Englewood Cliffs, New Jersey, 1992.

[89] J. Sparso, S. Furber, *Principles of Asynchronous Circuit Design*, Kluwer Academic Publishers, 2001.

[90] Villa T., A. Sangiovanni-Vincentelli, "*NOVA: State Assignment of Finite State Machines for Optimal Two-level Logic Implementation*", IEEE Trans. on Computer-Aided Design., CAD-9, no 9, pp. 905-924, 1990.

[91] Wu Q., M. Pedram and X. Wu., "*Clock-gating and its application to low power design of sequential circuits*", IEEE Trans. on Circuits and Systems Part 1, Vol. 47, No. 3, pp. 415-420, 2000.

[92] Закревский А. Д., "*Алгоритмы синтеза дискретных автоматов* (*The Algorithms of Synthesis of Discrete Automata*)", Nauka (in Russian), 1971.

[93] Закревский А. Д., "*Логический синтез каскадных схем* (*Logic Synthesis of Cascade Circuits*)", Nauka (in Russian), 1981.

[94] Закревский А. Д., Потосин Ю. В., Черемисинова Л.Д., "*Основы логического проектирования* (*Logic Design Bases*)", Minsk (in Russian) 2004.

Curriculum Vitae

1. Personal Data

Name	Elena Fomina
Date of birth	07.04.1974
Citizenship	Estonian
Marriage status	single
Children	no

2. Contact Data

Address	Raja str. 15 Tallinn, 12618, Estonia
Phone	+372 620 22 55
E-mail	elfom@staff.ttu.ee

3. Education

1999 entered Ph.D studies in TUT (Tallinn University of Technology), faculty of CE (Computer Engineering)
1996-1999, TUT, faculty of CE, M.Sc
1991-1996, TUT, faculty of CE, engineer

4. Languages

English	advanced
French	elementary
Estonian	good
Russian	native

5. Professional Experience

2004- TUT, Dept of CE, researcher

6. Academic Degree

Master of Science in Computer Engineering, TUT, 1999

7. Research Interests

Synthesis of digital circuits
FSM decomposition and state encoding
Low power FSM realization

Elulookirjeldus

1. Isikuandmed

Nimi	Elena (Jelena) Fomina
Süniaeg	07.04.1974
Kodakondsus	Eesti
Perekonnaseis	valaline
Lapsed	ei ole

2. Kontaktandmed

Aadress	Raja tänav 15 Tallinn, 12618, Eesti
Telefon	+372 620 22 55
E-posti aadress	elfom@staff.ttu.ee

3. Hariduskäik

1999 doktorantuur TTÜs (Tallinna Tehnikaülikool), ATI (arvutitehnika institut)
1996-1999, TTÜ, ATI, tehnikamagister
1991-1996, TTÜ, ATI, enseneer

4. Keelteoskus

Englise	kõrg
Prantsuse	alg
Eesti	hea
Vene	emakeel

5. Teeninduskäik

2004- TTÜ, ATI, teadur

6. Kaitstud Lõpputööd

Tehnikamagister, TTÜ, 1999

7. Teadustöö Põhisuunad

Digitaalskeemide süntees
Lõplike automaatide dekompositsioon ja kodeerimine
Automaatide madala energiatarbega realisatsioonid

Dissertations Defended at Tallinn University of Technology on Informatics and System Engineering