Web-Based Software Implementation of Finite State Machine Decomposition for Design and Education

Sergei Devadze, Margus Kruus, Alexander Sudnitson

Abstract: *This work focuses on particular but comprehensive problem of decomposition of finite state machines (FSMs), which provides a mathematical model for discrete, deterministic computing (control) devices with finite memory. We are concerned with solving complex logical tasks arising from the process of implementation of complicated algorithms onto contemporary hardware basis using computer aided design tools. The educational aim of this work is to provide a basic theoretical background for discrete systems synthesis using opportunities of asynchronous mode of education via Internet.*
Key words: *Sequential Machine Decomposition, Partition Pair Algebra, Design Constraints, Internet Based Teaching.*

INTRODUCTION

Finite state mashine (FSM) is conveniet model for specification, analysis and synthesis of control part of electronic sytems. Decomposition of FSM is essential to many computer-aided design (CAD) applications [3]. A large hardware behavioral description is decomposed into several smaller ones. One goal is to make the synthesis problem more tractable by providing smaller subproblems that can be solved efficiently. Another goal is to create descriptions that can be synthesised into a structure that meets the design constraints. In the past synthesis focused on quality measures based on area and perfomance. The continuing decrease in feature size and increase in chip density in recent years have given rise to consider decomposition theory for low power synthesis.

A substantial part of this work is the description of a user-friendly interactive system developed for World Wide Web (WWW) that assists designers to deepen basic concepts and notions in digital design and helps to synthesize complex control devises. The system uses Java technology that represents a powerful tool for the development of platform-independent interactive software, which can be used on the WWW through Java enabled Web browser. This system provides remote distance interactive learning as an important emerging educational trend. The modern information technologies have enabled education using synchronous and asynchronous tools. In asynchronous mode students can have access to instructional material at any time and from any convenient location. Asynchronous learning networks provide in addition a network of people who can interact with each other using electronic connectivity tools to simulate the interactivity of physical presence.

The outline of this paper is as follows. In the next section, the motivation of development of FSM decomposition methods and their software implementation for design and education and some basic concepts on decomposition are presented. Section 3 describes the interactive system based on Java applets. To illustrate procedure of decomposition example applet is discussed in section 4. Section 5 concludes the paper.

PRELIMINARIES

FSM decomposition has been a classic problem of discrete system theory for many years. Various techniques have been developed to enhance the capability and efficiency of decomposition, and they fall broadly into two categories: those based on the algebraic theory [5] and those based on the factorisation or on the identification in the state transition graph of subroutines or factors [1, 2]. Theoretical background of our system is more general approach that is based on the FSM decomposition theory, which uses partition pair algebra proposed in [5]. The importance of this theory lies in the fact that it provides a direct link between algebraic relationships and physical realizations of machines. The mathematical foundation of this theory rest on an

algebraization of the concept of "information" in a machine and supply the algebraic formalism necessary to study problems about the flow of this information in machines as they operate. The formal techniques are very closely related to modern algebra. It has, we believe after Hartmanis and Stearns [5], 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.

FSM as algebraic system is a quintuple A = $\langle Z, I, O, \delta, \lambda \rangle$, where Z is a finite non-empty set of states, I is a finite non-empty set of inputs and O is a finite set of outputs. δ: $I \times Z \rightarrow Z$ is called transition (or next state function) and λ: $Z \rightarrow O$ is called the output function of A (Moore type FSM).

Informally the essence of the decomposition task could be described as follows [1, 2, 5].

Given a FSM description of a desired terminal behavior, the decomposition problem is to find two or more machines which, when interconnected in a prescribed way, will display that terminal behavior. The individual machines that make up the overall realization are referred to as *component FSMs (submachines).* Each submachine corresponds to a partition on the set of states (a partition π on the set of states, Z, in a machine is a collection of disjoint subsets of states whose set union is Z). All the states belonging to a single block in a submachine are given the same code in that submachine. Therefore, there is no way of distinguishing between two states belonging to a single block in a submachine without recourse to information from other submachines. A block of states in a partition effectively corresponds to a state in the submachine associated with that partition. *The prototype FSM* corresponds to the machine that was used to define the terminal behavior to be realized. The most complex step of decomposition under development is to define some information partitions which are induced on A by a network N that defines (realizes) A. These "associated" partitions on A may be thought of as a global characterization of the information used and computed in a component machine of N. This natural correspondence between global and local properties allows us to approach the structure of machines with partition algebra.

The theory of decomposition is concerned with logical or functional dependence in machines and studies information flow of the machine independently of how the information is represented and how logical functions are implemented. Our decomposition approach can be widely used for investigation of all kinds of internal functional dependencies of a FSM (these are input-state, state-state, input-output and state-output dependencies). It assists in better understanding how input information is transformed into state information, state information to output, and input information to output.

Decomposition design technique allows one to reduce step by step the complexity of design optimization tasks and design cycle consists of sequential steps of transformations. Network components, as the result of decomposition, can be considered as the project independent parts and could be distributed between different designers in order to achieve the highest efficiency.

SOFTWARE BASED ON JAVA APPLETS

To implement the educational system's architecture we should follow four main requirements [7]:

- possibility to ran under various operating systems;
- implementation of new modules without changing the rest of the system;
- realizing a client server architecture;
- using the same source to generate the printed and interactive worksheets to prevent inconsistency after modifications.

These requirements cause the use the applet concept of Java language. Java is the natural programming language of choice on the client side because of its flexibility of GUI design, convenient network programming, and platform independence. The last property is especially significant since it allows the same applet program to run on client computers of different platform. We have possibility to use one of the features of WWW that is the fact that informations in hypermedia format can be easily created and disseminated, combining hypertext and multimedia.

Developed system includes building tutorial of FSM synthesis theory and additional useful information for working with client software. The advantage of the tutorial is interconnectedness among different topics and with other related tutorials, which is easy to implement on the WWW using the hypertext markup language. The tutorial contains many examples to study and compare.

The sequence of applets to understand the essence of decompositions is developed. Next, we discuss main of them from "informational" point of view.

Applet 1 describes how partitions on a set can be "*multiplied*" and "*added*". These operations on partitions play a central role in the structure theory of FSM and form a basic link between machine concepts and algebra. The sum of two partitions π_1 and π_2 is the largest partition (the one with the most blocks) that is refined by both π_1 and π_2. The product of π_1 and π_2 is the smallest partition (the one with the fewest blocks) that refines both π_1 and π_2. A partition on the set of states of the FSM can be considered as a measure of information about the FSM. That it is why the multiplication of all partitions in the set must be zero partition in order to preserve all the information about the source FSM behavior in the network of FSMs defined by the partition set. The functionality of the prototype machine is maintained in the decomposed machine if the partitions associated with the decomposition are such that their product is the zero-partition on *Z* (every block of partition consists exactly of one state).

Applet 2 exhibits formal correspondence to intuitive concept of a "subcomputation". We consider the concept of a homomorphism. Since a machine *A* can be used to realize its homomorphic image *A'*, we can say informally that *A'* does a part or a subcomputation of the computation performed by *A.* From partition algebra point of view the concept of homomorphism relates to partitions with substitution property. We recall that if a partition π on the set of states of a machine *A* has the substitution property, than as long as we know the block of π which contains a given state of *A*, we can compute the block of π to which that state is transformed by any given input sequence. Intuitively we say that the "ignorance" about the given state (as specified by the partition π) does not spread as the machine operates [5].

Applet 3. The concept of partition pairs is more general than substitution property and is introduced to study how "ignorance spreads" or information flows" through a sequential machine when it operates. If (π,π') is a *partition pair* on the FSM *A* than blocks of π are mapped into the blocks of π' by *A* In other words, if we only know the block of π which contains the state of *A*, then we can compute for every input the block of π' to which this state is transferred by A. It is natural that for any partition π we can determine the $M(\pi)$ partition. The operator $M(\pi)$ gives the maximum front partition of partition pair. Informally speaking, 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 the next state (i.e., the block of π which contains the next state of *A*). Thus these partition gives precise meaning to our intuitive concept "how much do we have to know about the present state to compute … about the next state".

Applet 4 performs construction of FSM network that realizes the prototype FSM. We consider FSM network as algebraic system $N = \langle B, I, O, g, F \rangle$ where *B* is a set of component FSMs, *I* is a set of inputs and *O* is a set of outputs, *g* is output function of network, $F \subseteq B \times B$ is a relation of connection of component FSMs of the network. We

represent F as incidence matrix $\| r_{ij} \|$. $r_{ij} = 1$ means i-th component FSM receives information from j-th component FSM. Every FSM from B is in correspondence with chosen partition π_i. If partition $\tau = M(\pi_i)$ is less or equal to multiplication of partitions from $\{\pi_j \mid r_{ij} = 1\}$ than it means that i-th component FSM receives enough information from component FSMs with which it is connected accordingly F to compute the next state.

Our work proceeds from the fact that the principal NP-complete problem of FSM decomposition is searching of a set of partitions on the set of states of prototype FSM. As it was shown in [5], only such a set of partitions may be used for FSM decomposition. The lack of a methodology of searching of these partitions is substantial limitation of application of powerful algebraic decomposition theory in practice. We attempt to surmount this obstacle. Implementation of FSM in a device with the lack of external terminals appears very often in practice and has always been a problem for designers.

Applet 5 is devoted to choice of decomposition partition on the set of states of prototype FSM to meet a requirement on the number of inputs. Here we should emphasize the fact that the machine decomposition problem and the reduction of variable dependence are virtually identical concepts. This is NP-hard problem, and amounts to solving a face hypercube embedding problem [1]. In spite of recent advances, computing a decision of this task remains prohibitive for FSM of practical complexity. In this applet we show how the input-state dependencies can be used to decrease the number of inputs. Theoretical foundation of our approach is based on the new notion of partition with don't care's and its relation to pair algebra.

A partition with Don't Care's (PDC) ρ of a set S is a collection of disjoint nonempty subsets of S. The disjoint subsets are called blocks of ρ and their set union is equal to $S_d \subseteq S$. The set difference $S \setminus S_d$ is Don't Care's area of the PDC and we can consider it as some distinguished (special) block b_c which may be empty. The PDC ρ in reality defines a set of conventional partitions, denoted by $G(\rho)$, generated by distributing the elements of distinguished block over the other blocks of the PDC and over new created blocks in all possible ways. The set of all PDC pairs on A is a pair algebra on $\mathfrak{I} \times \mathfrak{I}$, where $\mathfrak{I}$ is lattice of PDC of S. Thus, all the results which are derived about a pair algebra [5] hold for PDC pairs on A.

The idea of the next two applets is to introduce additional "idle" states into the FSM in the hope to meet design constraints. The network of FSMs consists of components working alternatively in time, i.e. all components except one are suspended in one of extra state (the "wait" state). In [1] similar approach is called factorization of the sequential state machines. This property gives opportunity to apply sleep mode operation (dynamic power management) for saving power consumption. *Applet 6* enables to decompose a prototype FSM into a set connected component FSMs with given constraints on the complexity of component FSMs (a number of inputs, outputs, states and rows in their transition tables) on the base of one partition on the set of states. The number of states of component FSM is equal to the number of states in corresponding block of partition π plus 1 (wait or idle state). *Applet 7* implements a method for FSM decomposition with outputs distributed among the component FSMs. A partition on the set of prototype FSM outputs is taken as primary design requirement.

EXAMPLE APPLET

In practice commercial CAD tools of digital systems generate register-transfer level designs from behavioral specifications. The design consists of a datapath and controller. The controller is usually represented as FSM with binary inputs and binary outputs and they are determined by external requirements [3]. We use the formal notion of transition that is a triplet $< z_i , z_j , \alpha_h >$ where z_i is the present state, z_j is the next state, α_h is the input condition (Boolean function). The search for the next state means

the evaluation of the Boolean functions. It is necessary to evaluate which of these functions has value "true" for a given input binary vector.

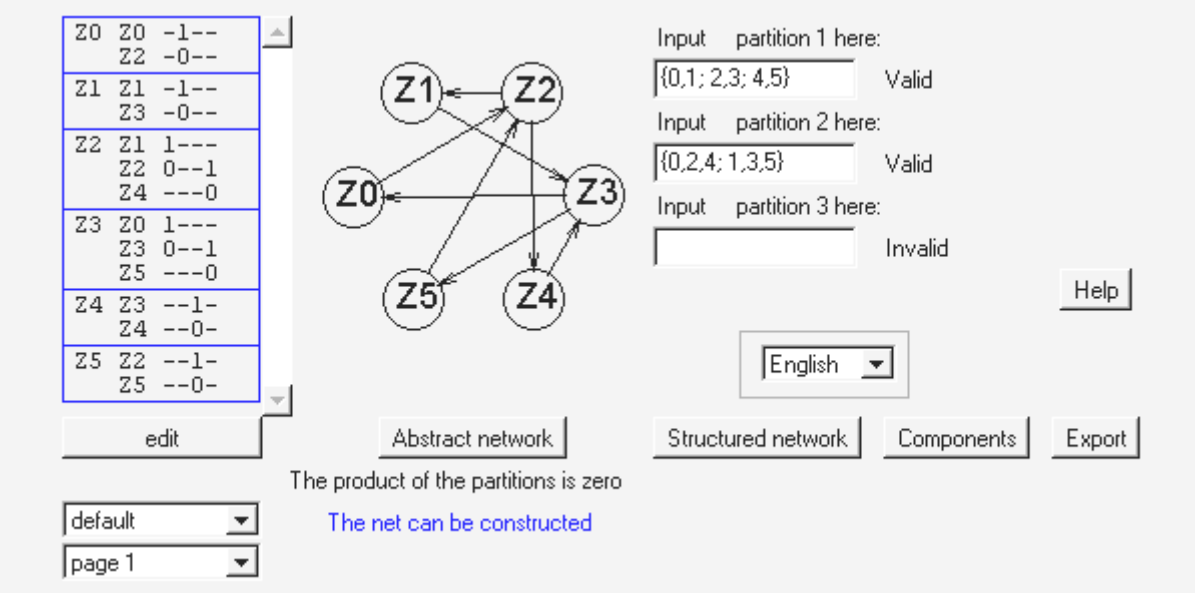

Figure 1: example applet 4

The behavior of a FSM can be described by state transition graph or, equivalently, by presentation by the list of transitions. Here we take into account that each transition depends essentially on relatively few input variables. While we investigate mainly state-state and input-state dependencies, FSM formally is treated as triple $\langle Z, X, \delta \rangle$, where $Z = \{z_1, \ldots, z_n\}$ is a set of internal states, $X = \{x_1, \ldots, x_l\}$ is a set of input variables (channels), $\delta : \{0, 1\}^l \times Z \rightarrow Z$ is a multiple valued next state function of the FSM. The familiarity with representation of Boolean functions with cubes is assumed. We refer to [4] for acquaintance via Internet with such kind of representation of Boolean functions.

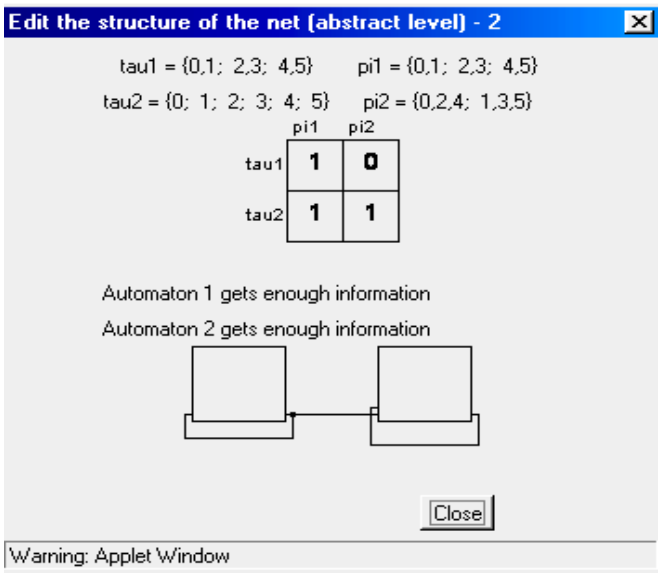

Figure 2: example abstract network

Let us consider the applet of construction of FSM network (available in [6]). The applet under consideration (figure 1) gives possibility to experiment ("play") with decomposition of FSM. At first, we should choose an FSM to "play" with. Either select

one from a combobox in the left bottom corner of the screen, or enter any FSM we like by pushing the "edit" button. Then we should choose the decomposition partitions. We can enter partitions into the second combobox. If everything is ready for decomposition, then we can look at the resulting network. It is possible to examine intermediate steps of the decomposition procedure by pushing "abstract network" (figure 2) and "Structured network" buttons. Now we can see the matrix that represents the relation of connection of the network. In such a way, we can "play" with the network. Then it is possible to find out how the component FSM look like by pushing "Components" button. And finally, it is possible to save the network to a file by pushing Expert button.

CONCLUDING REMARKS

This work focuses on how new Web-based frameworks can enable research and development in synthesis of control-dominated discrete systems. The pedagogical basis on which our system is built is that students learn most quickly when they are presented with material that they can quickly use to solve design problems. Students can decompose a given FSM into a set of connected FSMs with given constraints on the complexity of component FSMs (a number of inputs, outputs, states and rows in their transition tables). The system assists to fulfil projects on design of digital devices.

In future we plan to update the system to carry on an interactive Web-based decomposition synthesis at a higher level that considers both the controller and datapath simultaneously. The synthesis system under development should not be only design automation software but it should be a research tool and educational system we will be able to use for further development of synthesis theory.

ACKNOWLEDGEMENT

This work has been supported partially by the Estonian Science Foundation (under Grant 4876) and by the Ministry of Education in Thüringen, Germany (DILDIS project).

REFERENCES

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

[2] Baranov S. Logic Synthesis for Control Automata. Boston: Kluwer Academic Publishers, 1994.

[3] De Micheli G. Synthesis and Optimization of Digital Circuits. New York: McGraw-Hill, Inc., 1994.

[4] M.Dubin, Neese T., and Moon I. Action Based Learning for Switching and Automata Theory. Available: http://vlsi.colorado.edu/~mooni/N_ABLE/N_ABLE.html.

[5] Hartmanis J., R.E.Stearns. Algebraic Structure Theory of Sequential Machines. Englewood Cliffs, New York: Prentice-Hall, 1966.

[6] Sudnitson A., S. Devadze, A. Levenko. Finite State Machine Decomposition. Available: http://www.pld.ttu.ee/dildis/automata/applets.

[7] Wuttke H-D., Henke K., Peukert R. Interned Based Education – An Experimental Environment for Various Educational Purposes. Proc. of the IASTED International Conference Computers and advanced Technology in Education, May 6-8, Phaladelfia, PA USA, pp. 50-54, 1999.

ABOUT THE AUTHORS

Student Sergei Devadze,
Assoc. Prof. Margus Kruus, PhD,
Assoc. Prof. Alexander Sudnitson, PhD,
Department of Computer Engineering, Tallinn Technical University, Raja 15, 12617 Tallinn, Estonia, Phone: +372 620 2251, E-mail: alsu@cc.ttu.ee