WEB-BASED TOOLS FOR DECOMPOSITION-ORIENTED DIGITAL DESIGN

M. Kruus, H. Lensen, and A. Sudnitson
Tallinn Technical University
Raja 15, 12618 Tallinn, Estonia
alsu@cc.ttu.ee

ABSTRACT

Decomposition of digital systems is essential to many computer-aided design applications. Here we should emphasize the fact that the machine decomposition is the organic part of synthesis process. 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 performance. 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 new dimension of the design process.
A substantial part of this work is the development of a user-friendly interactive system developed for WWW that assists designers to deepen basic concepts and notions in digital design and helps to synthesize control devises. We are concerned with solving complex combinatorial tasks arising from the process of design. An approach to modeling of information flows in networks of finite state machines is considered. A lower power synthesis framework can integrate the proposed techniques as result of the fact that decomposition yields attractive power reduction in the final implementations. 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.

NOMENCLATURE

FSM	Finite State Machine
STG	State Transition Graph
WWW	World Wide Web

1. INTRODUCTION

Synthesis systems typically take a hardware description language model of a design as the initial and then the syntheis path follows several steps: high-level synthesis, state assignment, logic synthesis and library binding. This work targets the very first step in this synthesis flow where the FSM characterizing the control part of the high-level representation is typically described in the form of STG and each state is represented in a symbolic form.

This work focuses on particular but comprehensive problem of FSMs decomposition. Decomposition of FSMs is essential to design optimization in implementation-independent manner.

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 [1] and those based on the factorisation or on the identification in the STG of subroutines or factors [2].

Theoretical background of our system is the algebraic structure theory of sequential machines, which uses partition pair algebra proposed in [1]. The importance of this theory lies in the fact that it provides a direct link between algebraic relationships and physical realizations of finite state 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 problem of estimation different design alternatives is probably the most difficult one at each design stage. The design for low power cannot be achieved without power prediction tools. Some authors presented a methodology for the synthesis of FSMs targeted towards low power dissipation using information theoretic measures [3-5]. Unlike previous works, which focus on the modeling of informational flows of individual FSM, our approach also covers the quantitative distribution of information in FSMs networks.

The outline of this paper is as follows. In the next section, the introduction of the main concepts of FSM decomposition behind our approach is presented. In section 3, we introduce informational measurements for low power synthesis of FSM networks. Section 4 describes the interactive system based on Java applets. Section 5 concludes the paper.

2. DECOMPOSITION BASICS

Formally, the FSM is defined as a quintuple
$< S, X, Y, \delta, \lambda >$ where
$S = \{ s_1, \dots , s_M \}$ is a set of states;
$C = \{x_1, \dots , x_L \}$ is a set of binary input variables;
$U = \{y_1, \dots , y_T\}$ is a set of binary output variables;
d: $D(d)$ ® S is a multiple valued next state function
with domain $D(d) = D_1$ ´ … ´ D_L ´ S and codomain S, $D_i = \{0, 1\}$ represents a set of values each input variable x_i may assume;
l: $D(l)$ ® $R(l)$ is an output function with domain $D(l) = D(d)$
and codomain $R(l) = E_1$ ´ … ´ E_T .
$E_i = \{0, 1\}$ represents a set of values each output variable y_i may assume.

The behavior of a control sequential component can be described by STG or, equivalently, by presentation by the list of transitions.

In this work, we are concerned with the problem of decomposition of FSM. Informally, the essence of the decomposition task could be described as follows. Given a prototype FSM description of a desired terminal behavior, the decomposition problem is to find sub-machines which, when interconnected in a prescribed way, will display that terminal behavior. Our procedure of decomposition is based on the general form of decomposition without the restriction on their interconnection (Fig. 1). Each sub-machine corresponds to a partition on the set of states (a partition π on the set of states, S, in a machine is a collection of disjoint subsets of states whose set union is S).

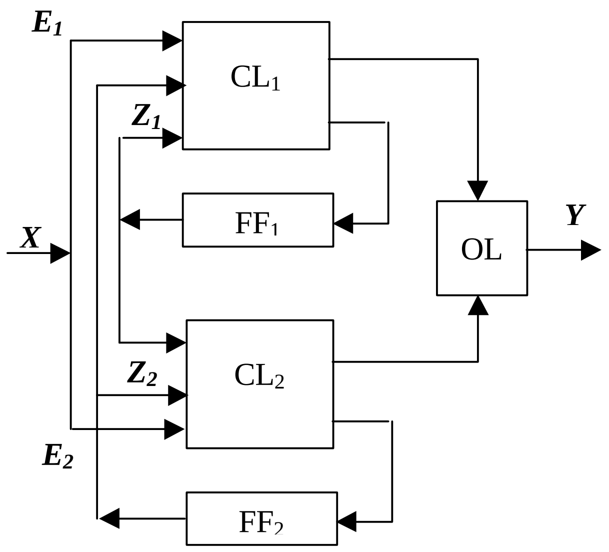


Fig. 1 Structure of decomposed machine

The state behavior of the FSM network [1] forms the basis of decomposition model. The state behavior of prototype machine formally is described by the network of state machines A_i = $< X_i , S_i , d_i >$ where
S_i is the set states elements of which correspond to blocks of partition p_i.
$X_i = Z_i$ È E_i , where Z_i is a set of internal symbolic variables (state variables) and E_i Í X is a set of external inputs. Each of the sub-machine receives, as inputs, not only the primary inputs and its own sate variables, but also the sate variables of the other sub-machine.
d_i: $D(d_i)$ ® S_i, is a transition function.

To describe the network more thoroughly, we use the set of internal symbolic variables of net $Z = \{z_i$ | z_i Î S_i, i Î $I = \{1,\dots,n\}\}$ and representation of relation of connection R as incidence matrix $|| r_{ij} ||$.

3. RELATIONSHIP MEASURES FOR LOW POWER SYNTHESIS

In our reasoning, we proceed from information theoretic concepts, which are rationalized on the basis of algebraic structure theory of sequential machine. In the following, we assume that the state lines of the FSM are modeled as Markov chain characterized by the stochastic matrix $(q_{ij})_{1 \leq i,\ j \leq m}$, where q_{ij} is the conditional probability of the FSM being in j-th state given that it was previously in i-th state. These probabilities, along with the steady state probability vector $(p_i)_{1\ £\ i\ £\ m}$ (we suppose that all states are reachable) can be found using standard techniques for probabilistic analysis of FSMs [3].

Let $E = \{e_1, e_2, \dots e_g\}$ be a complete set of events which may occur with the probabilities $p_1, p_2, \dots ,p_g$ In order to quantify the content of information Shannon introduces the concept of entropy.

Entropy of E (denoted $H(E)$) is given by:

$$H(E) = -\sum_{e \in E} p(e) \cdot \log_2 p(e) \qquad (1)$$

Depending on the specified sense of event, we can define several entropy measures, e.g. the entropy of FSM based on the state of occupation probabilities or based on the state transition probabilities. Reasoning similarly, we define the entropy of partition $\boldsymbol{p}$ as:

$$H(\boldsymbol{p}) = -\sum_{B \in \boldsymbol{p}} p(B) \cdot \log_2 p(B) \qquad (2)$$

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

Entropy of FSM network corresponding to the set of partitions, N, is equal to

$$H(N) = \sum_{\boldsymbol{p} \in N} H(\boldsymbol{p}) \qquad (3)$$

Information theoretic approaches for power estimation depend on information theoretic measures of activity. Power consumption of CMOS circuit is composed mainly from dynamic power [4] due to capacitive charging and discharging when a signal toggles as computed from the equation:

$$Power = 0.5 \cdot V^2 \cdot f \cdot C_{tot} \cdot E_{avg} \qquad (4)$$

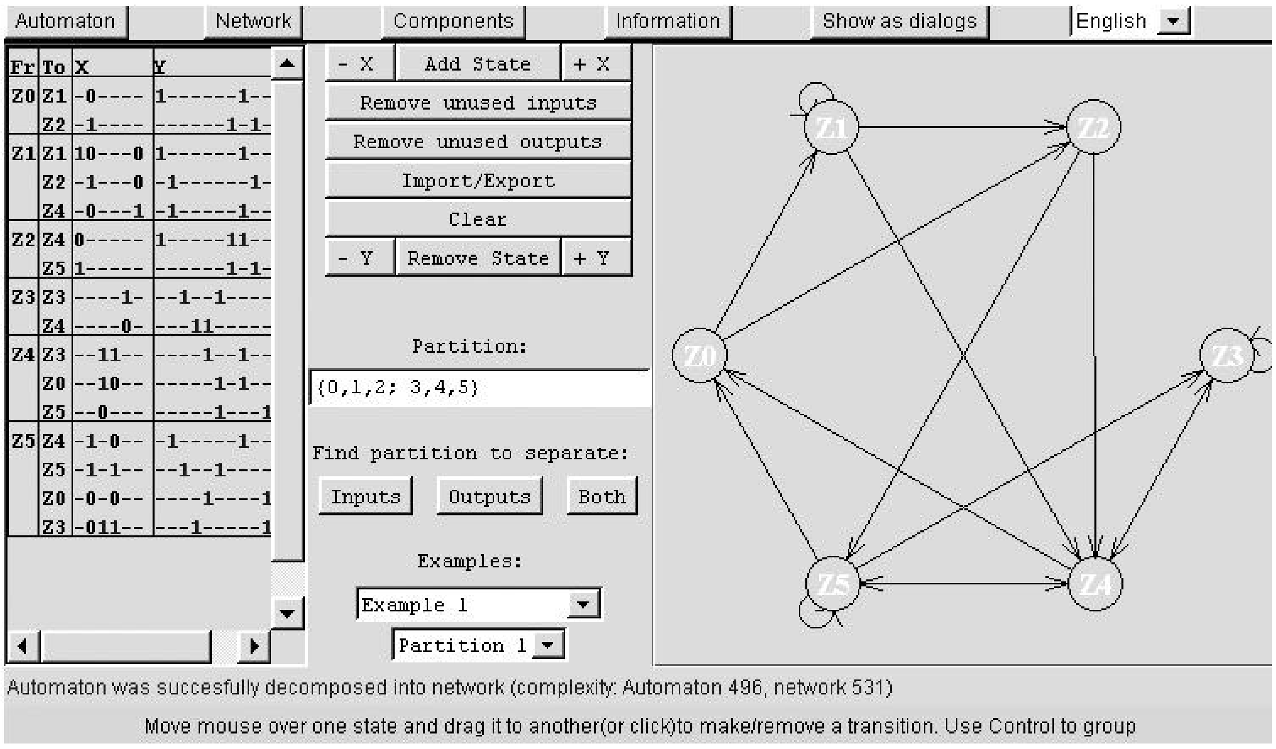
where f is the clock frequency, V is the supply voltage, C_{tot} is the total capacitance of the logic module (or the complexity factor), and E_{avg} is the average switching activity of the unit (or the activity factor). From this equation, we see that power estimation depends on several factors that are known only after hardware assignment, scheduling and placement. Furthermore, the activity factor is known only after executing the design. At the register transfer level, much of the information is not known. Fortunately, for high-level optimization, relative evaluation of different designs is more important than absolute evaluation, and consistency is more important than accuracy.

Entropy is related to switching activity, that is if the signal switching is high, it is likely that entropy is high also [5]. Theoretically confirmed high correlation proves that partition entropy is suitable for estimating corresponding sub-machines, which makes it a good measure for partition choice for appropriate decomposition. We propose measures to enable analysis of the information structure and information flows of a FSM network to control low-power synthesis of a sequential circuit. For estimation of switching activity of FSM as complete set of events, we consider the set of all transitions (corresponding to edges of STG) in the FSM.

4. SOFTWARE SYSTEM

To implement the software system’s architecture we should follow four main requirements [6] :

- 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.



Fr	To	X	Y
Z0	Z1	-0----	1------1--
	Z2	-1----	------1-1-
Z1	Z1	10---0	1------1--
	Z2	-1---0	-1------1-
	Z4	-0---1	-1-----1--
Z2	Z4	0-----	1-----11--
	Z5	1-----	------1-1-
Z3	Z3	----1-	--1--1----
	Z4	----0-	---11-----
Z4	Z3	--11--	----1--1--
	Z0	--10--	-----1-1--
	Z5	--0---	-----1---1
Z5	Z4	-1-0--	-1-----1--
	Z5	-1-1--	--1--1----
	Z0	-0-0--	----1----1
	Z3	-011--	---1-----1

{0,1,2; 3,4,5}

Automaton was succesfully decomposed into network (complexity: Automaton 496, network 531)

Move mouse over one state and drag it to another(or click)to make/remove a transition. Use Control to group

Fig.2 Example applet

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.

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 related tutorials, which is easy to implement on the WWW using the hypertext mark-up language.

The sequence of applets is developed. Next, we discuss main of them from “informational” point of view. Let us break them down into three groups.

Group 1 help us to understand the essence of decomposition problem. The first applet 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 p_1 and p_2 is the smallest partition that is refined by both p_1 and p_2. The product of p_1 and p_2 is the largest partition that refines both p_1 and p_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 *S* (every block of partition consists exactly of one state). The next applet 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 *p* on the set of states of a machine *A* has the substitution property, than as long as we know the block of *p* which contains a given state of *A*, we can compute the block of *p* 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 [1]. 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 *(p,p')* is a *partition pair* on the FSM *A* than blocks of *p* are mapped into the blocks of *p'* by *A* In other words, if we only know the block of *p* which contains the state of *A*, then we can compute for every input the block of *p'* to which this state is transferred by A.

For partition pair $\langle p_i, p_j \rangle$ the conditional entropy is

$$H(p_j,p_i) = H(p_i \cdot p_j) - H(p_i) \qquad (5)$$

The concept of partition pair and its "informational representation" is presented by corresponding applet.

It is natural that for any partition *p* we can determine the *M(p)* partition. The operator *M(p)* gives the maximum front partition of partition pair. Informally speaking, for a given partition *p*, the partition *M(p)* describes the least amount of information we must have about the present state of *A* to the next state (i.e., the block of *p* which contains the next state of *A*). Thus these partitions gives precise meaning to our intuitive concept "how much do we have to know about the present state to compute … about the next state".

To calculate this partition we need to find the symbolic cover of the discrete function F_i: $D(d)$ ® p_i. Given a 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 $\langle b, B', B \rangle$ where *B'* is the set of states (block of partition *M*(*p*)) which transit to the next state contained in the same block *B* of partition *p* under input condition *b*. The number of prime implicates, $|K_i|$, is proportional to number of rows in the transition table of corresponding sub-machine.

Our work proceeds from the fact that the principal NP-hard problem of FSM decomposition is searching of a set of partitions on the set of states of prototype FSM. As it was shown in [1], 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.

Group 2 of applets 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 [2]. 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. One applet 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.

Group 3 of applets performs construction of FSM network that realizes the prototype FSM. We represent a relation of connection of sub-FSM in the network 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 p_i. If partition $t=M(p_i)$ is less or equal to multiplication of partitions from $\{p_j \mid r_{ij} = 1\}$ 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.

The idea of the next two applets is to introduce additional "idle" states into the FSM in the hope to meet power 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 [2], similar approach is called factorisation of the sequential state machines. This property gives opportunity to apply sleep mode operation (dynamic power management) for saving power consumption. Corresponding applet 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).

5. CONCLUSIONS

Incorporating functional partitioning into a synthesis methodology leads to several important advantages. In functional partitioning, we first partition a functional specification into a smaller sub-specifications and then synthesise structure for each, in contrast to the approach of first synthesizing structure for the entire specification and then partitioning that structure. In addition to reducing power, FSM functional partitioning also provides solutions to a variety of synthesis problems. One advantage the improvement in input/output performance and package count, when partitioning among hardware blocks with size and input/output constraints, such as FPGAs or blocks within ASIC. A second advantage is reduction in synthesis runtimes. We describe these improvement advantages, concluding that further research can lead to improved results from synthesis environments. This suggests the need for further investigation and development of automated functional decomposition tools, in order to meet design constraints.

The system uses Java technology that represents a powerful tool for the development of platform-independent interactive software [6], which can be used on the WWW through Java enabled Web browser. The developed synthesis system should not be considered only as particular design automation software, but it should be a research tool we will be able to use to carry out experiments guided to further development of logical synthesis theory. The educational aim of this work is to provide a basic theoretical background for discrete systems design using opportunities of asynchronous mode of education via internet. Asynchronous learning networks provide in addition a network of people who can interact with each other using electronic connectivity tools.

6. REFERENCES

1. Hartmanis J., and Stearns R. E.-Algebraic Structure Theory of Sequential Machines- Englewood Cliffs, N.J.: Prentice-Hall, 1966.
2. P. Ashar, S.Devadas, and A.R.NewtonA. R.- Sequential Logic Synthesis- Kluwer Academic Publishers, Boston, 1992.
3. Hachtel G. D., Macii E., Pardo A., and Somenzi F.- Markovian Analysis of Large Finite State Machines- IEEE Trans. Computer-Aided Design, Vol. 15, 1996, pp.1479-1493.
4. Macii E., Pedram M., and Somenzi F.-High-level Power Modeling, Estimation, and Optimization- IEEE Trans. Computer-Aided Design, Vol. 17, 1998, pp.1061-1079.
5. Marculescu D., Marculescu R., and Pedram M.-Information Theoretic Measures for Power Analysis- IEEE Trans. Computer-Aided Design, Vol. 15, 1996, pp. 599-610.
6. Wuttke H.-D., Henke K., Peukert R.-Internet Based Education : An Experimental Environment for Various Educational Purposes- Proc. of the IASTED Int. Conf. on Computers and Advanced Technology in Education, Philadelphia, PA USA. IASTED/Acta Press No. 292, 1999, pp. 50-54.
7. Sudnitson A., Devadze S., Levenko A.- Finite State Machine Decomposition- Available: http://www.pld.ttu.ee/dildis/automata/applets

ACKNOWLEDGEMENTS

This work is partially supported by the Ministry of Education in Thüringen, Germany (DILDIS project), and by the Estonian Science Foundation (Grant No 4876).