FSM Decomposition for Low Power in FPGA

Gustavo Sutter[1], Elias Todorovich[1], Sergio Lopez-Buedo[2], and Eduardo Boemo[2]

1. INCA, Universidad Nacional del Centro, Tandil, Argentine
{gsutter, etodorov}@exa.unicen.edu.ar
2. School of Computer and Telecommunication Engineering
Universidad Autonoma de Madrid, Spain
{sergio.lopez-buedo, eduardo.boemo}@uam.es

Abstract. In this paper, the realization of low power finite state machines (FSMs) on FPGAs using decomposition techniques is addressed. The original FSM is divided into two submachines using a probabilistic criterion. Only one submachine is active at a time, meanwhile the other is disabled to save power. Different deactivation alternatives and state encoding have been studied. For each option, actual measurements of power consumption have been done using the MCNC and the PREP benchmark circuits. A Xilinx XC4K device has been utilized as technological framework. The proposed technique fits well with big FSM, where power reductions up to 46% are obtained.

I. Introduction

In this work, the problem of optimising FPGA-based finite state machines (FSM) circuits for low power is addressed. Several techniques for state assignment have been proposed in the past for cell-based or gate array technology. The main idea has been to lower the average switching activity in two ways: either by disabling the input data to the FSM, or by blocking the state registers. The cost is an extra hardware to detect certain conditions to stop parts of the machine.

The experiments presented in this paper are based on the ideas proposed in [7][15][6][3][4], adapted or modified to suit well with the technological target: LUT-based FPGAs. The original FSM is divided into two sub-FSMs. Each submachine must to have roughly the same amount of states. A probabilistic approach is utilised to determine an optimal partition that guarantees a minimum interaction between the submachines. The hardware overhead associated with the decomposition technique makes this method neither effective for FSMs with small numbers of states (under 10) nor applicable for circuits whose decomposition has a highly transition probability between submachines. However, for large machines, an improvement in power consumption up to 46% can be obtained.

The paper is organized as follows. Section II reviews the basic definitions and highlights the main aspects of the traditional approaches to FSM decomposition. The FSM architecture proposed in this paper is described in Section III. In the next section, the characteristics of the benchmark circuits are exhibited. Finally, in Section VI, the main experimental results are summarized.

II. Background

A finite state machines is defined by a 6-tuple $M = (\Sigma, \sigma, Q, q_0, \delta, \lambda)$, where Σ is a finite set of input symbols, $\sigma \neq \varnothing$ is a finite set of output symbols, $Q \neq \varnothing$ is a finite set of states, $q_0 \in Q$ is the "reset" state, $\delta(q, a): Q \times \Sigma \to Q$ is the transition function, and $\lambda(q, a): Q \times \Sigma \to \sigma$ is the output function.

The 6-tuple M can be described by a state transition graph (STG). Nodes represent the states, and directed edges (labeled with input and output values), describe the transition relation between states. In hardware materializations, each state corresponds to a binary vector stored in registers. From the current state and input values, the combinational logic computes the next state and the output function.

The decomposition for low-power FSM requires first calculating the transitions probabilities in order to divide the machine. Thus, the activity can be reduced. Then, these submachines must be efficiently mapped in a FPGA, so that the hardware overhead does not compensate the power saving of a lower node activity.

II.a. Calculating Probabilities

In order to decide the submachine partitioning, a probabilistic model [24] must be utilized. To compute the transition probabilities for a given STG, it is first necessary to know the probability distribution for the inputs. Those values can be obtained by a higher-level simulation of FSM in a context close to the actual environment of the design. Then, the transition probability for each edge in the STG can be determined by modeling the STG as a Markov chain. A Markov chain is a stochastic process whose dynamic behavior is such that the probabilistic distribution for its future behavior depends only on the present state, without taking into account how the process arrived in that state.

The steady state probability for a state q_i is defined as the chance of the FSM to remain in q_i. This value is not time dependent. That is, as the time increases, it converges to constant real numbers. Let $\mathbf{P}$ be the conditional transition probability matrix, and $\mathbf{v}$ be the steady state probability vector (whose components are the state probabilities). Then, the steady state probabilities can be compute by solving the following system of $n+1$ equations: $v \cdot P = v$ and $\sum_{i=0}^{n-1} P_i = 1$;

where $v = \begin{bmatrix} P_0 P_1 ... P_{n-1} \end{bmatrix}$ and $P = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n-1,0} & p_{n-1,1} & \cdots & p_{n-1,n-1} \end{bmatrix}$

Here, $\mathbf{P}$ is a stochastic matrix (i.e. all the entries are non-negative and the sum of each row is one) whose entries are the conditional transition probabilities. The total transition probabilities $\mathbf{P_{i,j}}$ can be calculated as: $P_{i,j} = p_{i,j} \cdot P_i$
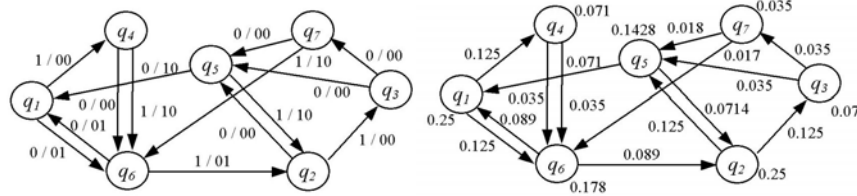
Fig. 1. a) State Transition Graph (STG), b) steady state probabilities and total transition probability

II.b. Low-power design of FSM

The most popular technique to reduce power in FSM is to modify the state encoding [27][4][16][28][14]. These works are focused on the Hamming distance minimization of the most probable state transitions. However, this solution usually increases the required logic to decode the next state. Then, a tradeoff between switching reduction and extra capacitance exists.

In the area of FPGAs, the most utilized state encoding technique is one-hot [25][9]. Nevertheless, empirical measurements indicate that binary state encoding is better for low power [21], [8] in these FSMs that have less than eight states.

Other idea for low-power FSMs 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 finally by "freezing" (blocking) the input data.

Under the last category, fall methods like precomputation, gated clock, selectively clocked systems, and decomposition. In the gated-clock technique [3][5], the clocking of a FSM is stopped when the machine is in self-loops and the outputs do not change. In precomputation [1], a simple combinational block is added to the original circuit. Under certain input conditions, the precomputation logic disables the loading of the input registers. This paper is focused mainly on the decomposition approach, detailed in the following paragraphs.

II.c. Decomposition architectures

Several researchers addressed the decomposition o partitioning of FSM. First, the goal is to reduce the complexity of the combinational block to be mapped in a fixed logic [2][11]. The FSM is divided in two (or more) interacting machines (fig.2), where each submachine knows in which state is the other. This strategy adds an idle state to each sub-FSM.
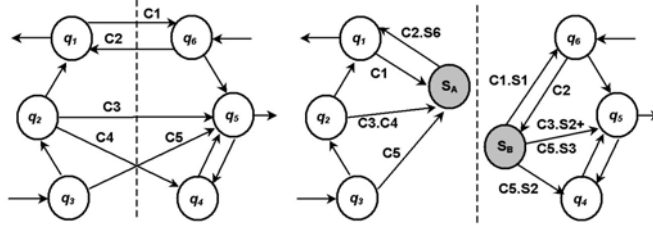
Fig. 2. State diagram of a FSM and the partitioned diagram in the traditional approach.

Other scheme is the orthogonal partition [20]. In this case, the number of states in the partition its not $n_1+n_2 = n$, but its approximately $\sqrt{n}$ in each partition. Let consider Q = $\{q_1,q_2,...,q_n\}$ the original state set. Two partitions $\prod_A = \{A_1, A_2, ..., A_m\}$ and $\prod_B = \{B_1, B_2, ..., B_k\}$ of Q are *orthogonal* if, for $i \le m, j \le k$, either $A_i \cap B_j = \varnothing$ or $A_i \cap B_j = \{q_i\}$. Thus, in order to represent an original q_t state, this method uses a combination of an A_i and a B_j.

II.d. Decomposition techniques for low power

The basic low-power idea of decomposition is to disable the inactive part of a FSM. The deactivation is reached either by blocking the inputs (using latches, ANDs or tri-estates buffers) or power-down the part of the circuit that is not used (by clock gating). In [15][6], the FSM is partitioned into several pieces, that are implemented as a separate machine with an extra wait state (idle state). In this case, only one of the sub-machines is active, meanwhile the others are idle. Therefore, the clock for inactive sub-machines can be gated and primary inputs can be disabled. This reduces switching activity and hence, the total power dissipation. In [15], the STG is partitioned into two unbalanced sub-machines: a small one that is active most of the time, and a large submachine that is usually disabled.

An interesting disjoint encoding schema is proposed in [7]. The resulting partition not follows strictly the standard structure of FSM decomposition. In this method, the STG is partitioned in two (or more) sets of states. All the states of a given set are encoded with the MSB (most significant bit) at 0, meanwhile the states of the other sets are encoded with the MSB at 1. Thus, the combinational logic can be broken into two separate blocks: one that is active when the first state bit is 0, and other that is active when the first state bit is 1. In this way, the power consumption can be potentially reduced.

Other technique is to use an orthogonal partitioning together with gated clock mechanism [19]. An *N* state machine is decomposed into two interacting machines with N_1 and N_2 states, such that $N \le N_1 \times N_2$. In each sub-FSM, the partition tries to maximize the number of self-edges (where the machine remains in the same state after the clock edge). For all the self-edge conditions, the inputs and clock signal are disabled.

III. A decomposition architecture suitable for FPGAs

In this paper, a decomposition architecture based on [7] have been constructed and evaluated in terms of area-time-power. The structure fits well with LUT-based FPGAs. The same codes are utilized in both submachines, but only one is active. To point the active machine, an extra bit called *ActiveFSM* is set. The first architectural option is shown in Fig.3a. The original FSM is decomposed in two combinational circuits (machines A and B) that compute both outputs and next state. Only one submachine is active at a time. The transference of the control between the machines is based on the values of the inputs and actual state. If the next state corresponds to the other machine, the *activeFSM* is asserted. The shaded blocks of the Fig.3 indicate the circuits that "freeze" the inactive machine. A second architectural option has also been implemented (fig 3.b). In this case, two registers are utilized to stop the machine evolution. In this case, two possibilities can be explored to control the registers: via an enabled signal, or by using a gated-clock.

For both architectures, the same design flow must be followed. First, an algorithm to decompose the FSM in two or more sub-machines must be selected. Second, one of the blocking methods must be implemented. Finally, a synthesizable code of the circuit must be written.
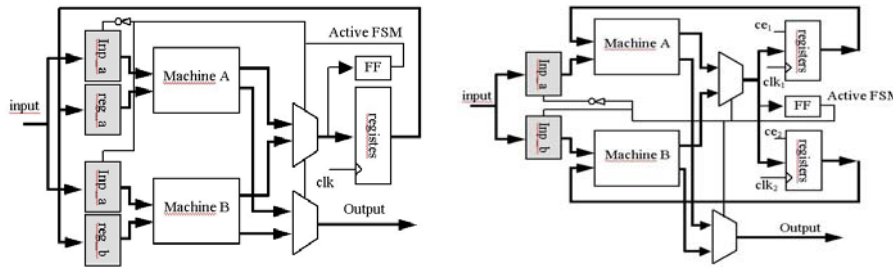

Fig. 3. Two options for decomposition FSM. a) Architecture I, b) Architecture II.

III.a. Partitioning a FSM into submachines

The technique separates the FSM into two or more submachines so that the probabilities of state transitions inside each submachine are maximized, meanwhile the interaction with the other submachines is low.

First, the transition probability over the STG is calculated (as is shown in fig 1.b). Then, a partition with equal cardinality in subFSMs is achieved. For instance, let consider the two partitions $\Pi_A = \{S_{a1}, S_{a2}, \ldots S_{an}\}$ and $\Pi_B = \{S_{b1}, S_{b2}, \ldots S_{bn}\}$, with a transition probability $p(i,j)$ between state S_i and S_j. In this case, the algorithm minimizes sum of transition probabilities between submachines. That is:

$$\min(\sum p(i,j)), \quad \forall i \in \Pi_A, j \in \Pi_B .$$

No greedy algorithm is necessary, because a backtracking technique with an effective prune is fast enough.

III.b Blocking method

In order to eliminate the activity in the idle FSM, the best alternative in Fig.3.a machines is to latch the data (*Blocking Latches*). Other possibilities like the use of ANDs, or buffers have been tested and discarded: they are more expensive in both area and time. For FSMs like the described in Fig.3.b, the alternatives are to implement a gate-clock or use the clock enable signal. In the second alternative the clock lines continue consuming power.

III.c. Synthesis of the final machine

In this work, a tool that automatically generates a set of benchmark FSMs has been developed. The inputs are the FSM description in a KISS2 format [18], and some extra parameters like the blocking scheme, and the state encoding technique. The tool calculates the steady state probability, divides the machine as described in section IIIa, and finally, writes a synthezable VHDL code. The file contains the entity of the machine, and three processes: one for the combinatorial logic, other for the blocking data circuitry, and the last one to incorporate tri-states buffers at the outputs pads to separately measure the off-chip power.

IV. Experiments

In this paper, the benchmark circuits have been implemented in several ways: First, in the original form with both binary and One Hot state encoding. Then, each machine was partitioned in two ways: one corresponding to the *architecture I* and other for the *architecture II* scheme (fig. 3). Once again, binary and One Hot encoding was the applied in each submachine. Additionally, the option named *architecture I*, was tested using different blocking techniques.

All the experiments use the MCNC91 benchmark set [13]. In addition, a large FSMs extracted from the PREP consortium [17] was utilized. Each FSM was first minimized with STAMINA [12]. Number of inputs, outputs, next state rules (arcs in the STG), and number of states of the benchmark machines are summarized in Table 1. Additionally the probability of transition and the number of arcs between submachines is reported.

The resulting VHDL code was compiled into a XC4010EPC84-1 FPGA sample, using the FPGA Express [10] and the Xilinx Foundation tools [26]. This circuit model does not have latches, so they were constructed using LUTs. All circuits has been implemented and tested under identical conditions. That is, all the measurements are related to the same FPGA sample, output pins, tool settings, printed circuit board, input vectors, clock frequency, and logic analyzer probes. Random vectors were utilized to stimulate the circuit. At the output, each pad supported the load of the logic analyzer, lower than 3pf [22].

Each circuit was measured at 100 Hz, 2 MHz, and 4 MHz to extrapolate the static power consumption. All prototypes include a tri-state buffer at the output pads to measure the off-chip power [23].

Example	Original FSM				Partition	
	$|\Sigma|$	$|\sigma|$	$|Q|$	$|\delta|$	Prob	Arcs
Bbsse	7	7	13	208	0,024	52
Cse	7	7	16	91	0,017	36
Dk16	2	3	27	108	0,247	28
Dk512	1	3	15	30	0,175	7
Ex1	9	19	18	233	0,022	53
Ex2	2	2	14	56	0,218	25
Keyb	7	2	19	170	0,004	63
Kirkman	12	6	16	370	0,002	46
Mark1	5	16	12	180	0,037	79
Planet	7	19	48	115	0,052	14
Prep4	8	8	16	78	0,041	9
S386	7	7	13	69	0,024	27
S820	18	19	24	254	0,006	138
S832c	18	19	24	243	0,006	118

Table 1. Original FSM data, number of inputs, outputs, states and arcs. In adition partition information is provided (probability and arcs between partitions)

V. EXPERIMENTAL RESULTS

The slope of the power consumption, expressed in mW/MHz, is depicted in Table II. The first columns show the value for the original FSM coded in One Hot (OH) and binary (bin). Then, the results for the partitioned circuits (encoded in One Hot and binary), are listed for the four different forms: *Architecture 1* (*Arch1*), *Architecture 2* (*Arch2*), *Architecture* 1 without blocking method (*No Blk*), and finally, *Architecture I* with blocking ANDs (*Blk and*). A power improvement factor is defined: it express the relationship between the power consumption of the best original FSM respect the best-partitioned one.

Power Improvement: For most of the FSM, a power saving is obtained. It can be up to the 42,4%. However, in five circuits, no improvement or negative results can be observed. This is caused by a high transition probability between submachines in these circuits (Table 3).

Binary vs. One Hot encoding in submachines: In accordance with previous results related to non-partitioned states machines [sut02], one hot encoding provides better results in FSMs with more than 16 states. On the contrary, for machines equal or smaller than 8 states, binary state encoding is better.

Blocking method: Latches are better in most of the cases. The improvement respect to the AND gates can be up to the 30%). Only in two benchmark FSMs, the blocking AND gates resulted better, because the low activity of the machines.

Sample	Original FSM		Partitioned One Hot Encoded				Partitioned Binary Encoded				Power Improvement
	OH	Bin	Arch1	Arch2	No Blk	Blk and	Arch1	Arch2	No Blk	Blk and	
Bbsse	3,90	4,70	3,80	3,95	4,04	4,34	3,55	3,76	4,23	3,91	9,0%
Cse	3,85	4,10	3,24	3,46	4,29	5,30	3,00	2,88	3,83	3,59	25,3%
Dk16	3,88	10,00	5,80	5,76	5,81	6,34	7,50	7,01	9,09	9,96	-32,8%
Dk512	1,84	2,80	2,46	2,79	2,44	2,14	2,24	2,51	2,16	1,94	-5,2%
Ex1	7,09	8,56	6,73	6,53	8,11	8,16	6,53	6,11	7,90	7,79	13,8%
Ex2	2,51	4,10	3,40	3,09	2,69	3,26	3,09	2,88	3,58	3,46	-6,5%
Keyb	5,50	7,06	4,73	4,31	7,88	7,69	3,66	4,65	5,25	6,81	33,4%
Kirkman	4,50	4,61	4,90	4,66	4,49	4,50	4,80	4,49	4,83	4,80	0,3%
Mark1	2,70	3,30	3,01	3,01	3,31	3,09	2,66	2,78	2,63	2,88	2,8%
Planet	8,04	16,80	9,18	9,29	10,23	10,01	10,88	11,81	15,18	16,99	-12,4%
Prep4	4,66	5,71	5,44	5,38	6,86	7,55	5,11	4,66	6,86	6,44	0,0%
S386	4,23	4,84	4,08	4,45	4,98	4,98	4,21	4,21	5,55	4,59	3,6%
S820	7,84	9,28	5,81	5,44	8,43	7,98	4,51	4,65	8,83	7,30	42,4%
S832c	7,01	10,21	5,08	5,00	7,64	6,60	4,73	5,04	7,55	6,75	32,6%

Table 2. Power consumption expressed in mW / MHz.

Area penalty: Both the synchronization and the partition circuitry add extra logic to the FSM. This overheard depends on the number of inputs, outputs and states. Each input signal requires 2 LUTs to implement the latches, and each output an extra LUT to implement the output multiplexer. Finally, each state add 2 extra LUTs to implement the latches in *architecture I* (*architecture II* not need extra logic to implement the blocking states). In terms of power, *Architecture I* its slightly better than *Architecture II*.

Sample	$|\Sigma|$	$|\sigma|$	$|Q|$	$|\delta|$	Arcs bet. part	% arcs bet. part.	Prob	Power Improv.
Dk16	2	3	27	108	28	26 %	0,247	-32,8 %
Ex2	2	2	14	56	25	45 %	0,218	-6,5 %
Dk512	1	3	15	30	7	23 %	0,175	-5,2 %
Planet	7	19	48	115	14	12 %	0,052	-12,4 %
Prep4	8	8	16	78	9	12 %	0,041	0,0 %

Table 3. Circuits where no improvement its possible due the highly probability of transition between submachines.

Clock period penalty: The synchronization scheme produces speed degradation. Table 4 shows the maximum frequency in MHz in each case. The influence of latches is remarkable nevertheless the partitioned *architecture* with blocking *and gates* shows a better performance.
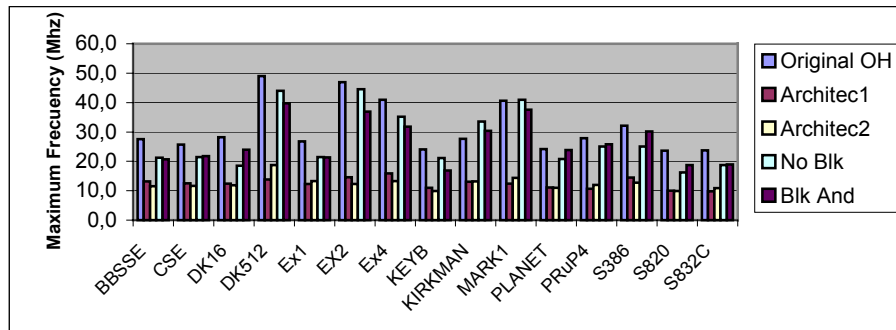
Fig. 4. Frequency chart, where the negative impact of the blocking latches can be observed.

VI. Conclusions

This paper explores partitioning methods to reduce power in FPGA-based FSMs. The main conclusions are that classical decomposition techniques developed for cell-based circuits can be adapted to FPGA. A significant power reduction in big FSMs (up to the 46%) can be obtained. These results should be more significant in devices that have embedded latches (Xilinx XC4000EX or the Virtex and Spartan 2). The state-encoding scheme of each submachine plays an important role: Binary based state encoding works better for small submachines (up to eight), meanwhile one hot is the best option in big FSMs. Finally, the achievement of a low activity between submachines is essential to get full advantage of the technique.

Acknowledgments

References

[1]   M.Alidina, J.Monteiro, S.Devadas, A.Gosh, M.Papaefthymiou, Precomputation-Based sequential logic optimization for Low-Power, *IEEE VLSI, V.2,n°4,pp.426-435*, Dec 1994.

[2]   P.Ashar, S.Devadas, and A.Newton. Optimum and heuristic algorithms for an approach to fsm decomposition. *IEEE Trans.Computer-Aided Design*, 10(3):296-310, March 1991.

[3]   L.Benini P.Siegel and G.De Micheli. Automatic synthesis of low-power gated-clock finite-state machines. *IEEE Trans.on CAD of IC, vol.15, Issue6*, June 1996, pp. 630– 643.

[4]   L.Benini and G. De Micheli. State Assignment for Low Power Dissipation. *IEEE Journ. of Solid State Circuits*, Vol. 30, No. 3, pp. 258-268, March 1995.

[5] L. Benini, and G. De Micheli, Transformation and Synthesis of FSMs for Low Power Gated Clock Implementation, *ISLP'95 International Symposium on Low Power Design, ACM-SIGDA and IEEE-CAS*, April 23-26, 1995.

[6] L. Benini, G. De Micheli, and F. Vermeulen, Finite-state machine partitioning for low power. *In Proc. IEEE Int'l Symposium on Circuits and Systems (ISCAS '98), volume 2, pages 5-8*, Monterey, California, May31-June3, 1998.

[7] S-H.Chow, Y-C.Ho, and T.Hwang. Low Power Realization of Finite State Machines Decomposition Approach. *ACM Trans on Design Aut. Elec. Systems*, 315-340, July 1996.

[8] J.Dunoyer, F.Pétrot, L.Jacomme. Intrinsic limitations of logarithmic encodingsfor low power finite state machines. *Mixed Des of VLSI Circ Conf, p 613-618*, Pologne, 1997.

[9] FPGA Compiler II / FPGA Express VHDL Reference Manual, Version 1999.05, Synopsys, Inc.,May 1999

[10] FPGA Express home page. Synopsis, inc.; http://www.synopsys.com/products/fpga/fpga_express.htm

[11] M.Geiger T.Müller-Wipperfürth, FSM Decomposition Revisited: Algebraic Structure Theory Applied to MCNC Benchmark FSMs. *28th ACM/IEEE DAC Conference*, 1991

[12] G.Hachtel, J.Rho, F.Somenzi, R.Jacoby. Exact and Heuristic Alg. for the Minimization of Incompletely Specified State Machines. *Europ. DAC, pp 184–191,* Amsterdam, feb 1991.

[13] Bob Lisanke. "Logic synthesis and optimization benchmarks". *Technical report, MCNC, Research Triangle Park*, North Carolina, December 1988.

[14] M. Martínez, M. J. Avedillo, J. M. Quintana, M. Koegst, ST. Rulke, and H. Susse: Low Power State Assignment Algorithm, *DCIS'00 conf*, pp. 181-187, 2000.

[15] J. Monteiro, A. Oliviera, Finite State Machine Decomposition for Low Power, *Proceedings 35th Design Automation Conference* , San Franscisco, 1998, pp. 758-763.

[16] Winfried Nöth and Reiner Kolla. Spanning Tree Based State Encoding for Low Power Dissipation". *In Proc of Date99, pp 168-174,* Munich, Germany, March 1999.

[17] PREP Benchmarks (Programmable Elect. Performance Company), http://www.prep.org.

[18] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli. SIS: A System for Seq. Circuit Synthesis. *Tech. Report Mem. No. UCB/ ERL M92/41*, Univ. of California, Berkeley, 1992.

[19] R.Shelar, H.Narayanan, M.Desai, Orthogonal Partitioning and Gated Clock Architecture for Low Power Realization of FSMs, *IEEE Int ASIC/SOC conf*, Sep 2000, pp. 266-270.

[20] R.Shelar, M.P. Desai, H.Narayanan, "Decomposition of Finite State Machines for Area, Delay Minimization", *IEEE ICCD99* Austin, 10-13 Oct. 99, pp. 620-625.

[21] G. Sutter and E. Boemo Low Power Finite state machines in FPGA: Bynary vs One hot encoding. *VIII Workshop Iberchip*, Guadalajara, Mexico, April 2002.

[22] Tektronix inc., *"TLA 700 Series Logic Analyzer User Manual",* http://www.tektronix.com.

[23] E. Todorovich, G. Sutter, N. Acosta, E. Boemo and S. López-Buedo, "End-user low-power alternatives at topological and physical levels. Some examples on FPGAs", *Proc. DCIS'2000*, Montpellier, France, November 2000.

[24] C-Y Tsui, M.Pedram, A. Despain, Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs, *31st Design Autom. Conf.*, pp. 18-23, 1994.

[25] Xilinx software manual, Synth. and Sim. Design Guide: Encoding State. *Xilinx Inc*, 2000

[26] Xilinx Foundation Tools F3.1i, www.xilinx.com/ support/library.htm

[27] C.Tsui, M.Pedram, C.Chen, A.Despain, "Low Power State Assignment Targeting Two- and Multi-level Logic Implement., *ACM/IEEE Inter.Conf. of CAD*, pp. 82-87, Nov.1994

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