Saving Power by Mapping Finite-State Machines into Embedded Memory Blocks in FPGAs

Anurag Tiwari and Karen A. Tomko
Department of ECECS, University of Cincinnati
Cincinnati, OH 45221-0030, USA
{atiwari, ktomko}@ececs.uc.edu

Abstract

Modern FPGAs contain on-chip synchronous embedded memory blocks (SEMBs), these memory blocks can be used to implement control units, when not used as on-chip memory. In this paper, we explore the mapping of Finite State Machines (FSMs) into the SEMBs for power and area minimization. We have shown the SEMB based implementation of the FSMs and compared it with conventional Flip-Flop (FF) based implementation. The proposed implementation of the FSMs consumes less power and has lower area and routing overhead than the FF based approach and it can be clocked at the maximum clock frequency supported by the SEMBs. Experimental results show that the SEMB based FSM consumes 4% to 26% less power than the conventional implementation. In addition it is observed that the power consumption can be further reduced by stopping the clock to the SEMBs during the idle states.

1. Introduction

The first generation FPGAs had less than a few thousand gates and were only able to support designs running upto 30 MHz. Today, FPGAs contain more than a million gates and are able to clock designs at a frequency greater than 200 MHz. Because of the substantial logic resources and higher processing speeds, FPGAs now can be used for some of the applications previously targeted to the Application Specific Integrated Chips (ASICs) and can be found in portable computing devices, mobile and wireless communication equipments. They are also used extensively in space based applications. With the rising FPGA complexity, power drawn by the devices has increased in comparison to the previous generation FPGAs. Furthermore, FPGA chips for these applications are battery-powered, thus power consumed by an FPGA device has became an important consideration to prolong battery-life. The control path of any FPGA design which consists of finite state machines consumes a significant amount of power; thus minimizing power consumed by the FSMs can significantly reduce the total power consumed by a design.

Traditionally, the FSMs in an FPGA are implemented using FFs and programmable Look Up Tables (LUTs). In addition to the programmable LUTs and FFs, the current generation of commercial FPGA contains embedded memory blocks which can be used to create single or duel port RAM, ROM and FIFOs. For example, Xilinx Virtex-II FPGA contains blockRAM [1], Altera Stratix FPGA contains TriMatrix memory [2], and Actel 42MX FPGA contains dual port SRAM modules [3]. The embedded memory blocks are synchronous SRAM modules (with their outputs latched), which can be configured in many different width and depth combinations.

The aforementioned FPGAs provide a large number of embedded memory arrays. In Xilinx Virtex-II FPGA, this number ranges from 4 blockRAMs for the Xilinx-XC2V40 device to 168 blockRAMs for the XC2V8000 device. The FPGA's silicon area is partitioned into these memory arrays and the programmable logic. Since different designs have varying memory requirements some embedded memory arrays may not be utilized in logic-intensive designs. These unutilized memory arrays can be used to implement control units and FSMs, which will unburden the routing resources and reduce power consumption of a design.

In this paper mapping of an FSM into the SEMBs is explored for FPGAs and power consumed by it is compared with the FF based approach. Besides reduced power consumption, mapping of an FSM into the SEMB has the following advantages:

- Quick and easy change in the FSMs functionality by directly changing the SEMBs contents. No design recompilation (synthesis, place and route) necessary for changing the FSM's functionality.
- Fixed timing regardless of the FSM's complexity
- No additional clock gating circuit is required to cut off the clock during idle states; this can be achieved by controlling the enable signal.

The SEMB approach is in contrast to the traditional FF and logic based approach, in which a complex state machine can occupy a large percentage of the logic and routing resources in a device. In the SEMB approach, address lines of the SEMB are connected to the next state

1530-1591/04 $20.00 (c) 2004 IEEE

bits and inputs of an FSM. The memory contents are programmed with next state address location which is formed in conjunction with the inputs to the FSM. If there is sufficient space some of bits of the SEMB output can be used for the FSM's output. The FF and SEMB based FSMs are illustrated using figure 1a and 1b.

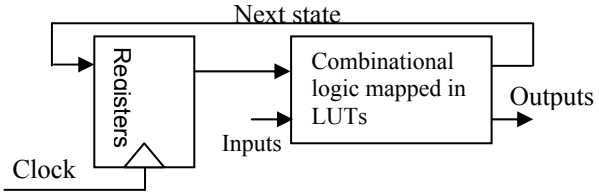

Figure 1a. FF and LUT based FSM in an FPGA

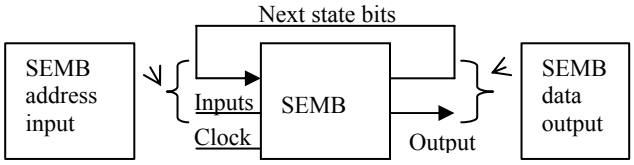

Figure 1b. SEMB based FSM implementation

This paper is organized as follows: Section 2 provides some background on power consumption in FPGAs. Section 3 discusses related research done on low power FSM design for FPGAs and mapping of logic into unused embedded memory arrays. Section 4 gives a detailed description of how to map an FSM into the SEMBs and describes why power consumption by the SEMB approach is lower than the FF based approach. In addition to presenting the SEMB mapping, this section compares the power consumed by both implementations. Section 5 and 7 present experimental results and conclusion respectively. Section 6 shows that power can be further reduced by stopping the clock to the SEMBs during idle states.

2. Power Consumption in FPGAs

In a typical FPGA 60% of power is consumed by the programmable interconnects, 16% is consumed by programmable logic and 14% by the clock distribution network [4]. This distribution is different from ASICs, in which the majority of power is consumed by the clocking network. Programmable interconnects in an FPGA are the dominating power consuming source, because a routed signal may have to pass though a number of programmable switches before reaching its destination. Other power consuming sources are the logic programmed into the FPGA's computing elements and the clock distribution network. Power consumed by an FPGA design is primarily dependent on three factors: *clock frequency, resource utilization,* and *switching activity*. Dynamic power dissipation, which constitutes a major portion of the total power dissipated by an FPGA, is caused by signal transitions, i.e. a change of signal value from logic level '0' to '1' and vice versa. Therefore, a design running at a higher clock frequency will have increased power dissipation due to more frequent signal transitions. Resource utilization is another factor which affects the power dissipation. Unused resources in an FPGA do not consume any dynamic power and as different designs have varying resource utilization, power consumed by a design is dependent on the resources it uses. The switching activity, which is equal to the number of signal transitions in a clock cycle, also contributes to the dynamic power dissipation. The switching activity of a signal depends upon the type of design and on the inputs to the design.

3. Related Work

Extensive research has been done in the area of low power design for FSMs. However, the majority of the previous research work is focused on ASIC implementation. Our work in this paper is different from earlier research as it takes advantage of the architectural features of newer FPGAs. Sutter et al [5] have proposed a decomposition based approach for FPGAs, in which the original FSM is divided into many smaller FSMs. There has been some research done for mapping combinatorial logic into the embedded memory arrays [6][7]. However, techniques presented in [6] and [7] are limited to the asynchronous on-chip memory blocks present in some previous generation FPGAs. Benini et al [8] have presented a clock gating technique which stops clock to the FSM during idle states. The clock stopping work presented in section V is similar to the research in [8], but our work is adapted and modified for reducing power consumed by the embedded memory blocks in FPGAs.

4. Mapping Finite State Machines to on-chip Memory Arrays

An FSM can be described by a six-tuple (I, O, S, r_0, δ, Y) where I is the set of inputs, O is the set of outputs, S is the set of states, and r_0 is the initial (reset) state, δ: I X S = S is the state transition function and Y: I X S = O is the output function for Mealy machines. The six-tuple machine can be depicted as a state transition graph (STG), where nodes represent states and directed edges represent outputs and inputs for the state transition

4.1 FF Based FSM Implementation in FPGA

When an FSM is implemented using FFs, each state is represented by the binary value stored in the FFs. The outputs and the next state values are calculated by the combinatorial logic. Figure 1a shows a FF and LUT based implementation of an FSM. The combinatorial portion of an FSM is implemented by LUTs in the FPGAs, which are connected by programmable interconnect. As the complexity of an FSM increases the number of LUTs and FFs, and programmable interconnect resources utilization increases proportionally. The number of FFs used to implement an FSM depends on the state encoding, such as

sequential, one-hot, grey encoding. The number of LUTs increases with the complexity of the FSM. It depends on the number of states in an FSM, number of transition in the state transition graph, number of inputs and outputs. In a design which utilizes a small percentage of total FPGA resources, LUTs and FFs are placed close to each other so that minimal programmable interconnects are used. On the other hand, in a denser design, due to routing congestion, LUTs and FFs may be spread all across the FPGA chip. This will increase the programmable interconnect utilization and hence the power consumption. Contrary to this the power consumed by the SEMB based FSM does not change with routing congestion, because most of the logic is programmed in the embedded memory itself. In the case where more than one embedded memory is required to implement an FSM, high speed dedicated interconnects between the memories blocks can be used [1].

4.2 FSM Implementation Using SEMBs

In the SEMB FSM implementation, the memory array contents are programmed with the encoded state bits (which along with the FSM's inputs also form the address for the next state memory location), and FSM's output. This approach is explained with the help of Figure 2b, the implementation shown is of a "0101" sequence detector, whose state diagram is shown in Figure 2a. The output of this sequence detector is '0' till the last '1', if the sequence is detected, at which time it becomes '1'. The embedded memory blocks present in the FPGAs have their output latched, which can be set or cleared after the device configuration or on the application of a reset signal. Since the output signals of the memory array are routed to its address inputs, the initial state of the FSM can be programmed at the location addressed by set or cleared outputs, usually it is the first memory location. In the sequencer example shown in Figure 2b, the initial state is the memory location "000", which is programmed with an encoded value of state 'A'. When the sequencer is in state A and if the input to it is '0', memory location "000" is addressed, the contents of which is "010", which is the memory location for the next state, B. Similarly, there is a state transition to other states by the change in address of the memory array. In some cases no separate logic is needed to generate the output of an FSM, instead it can be realized using the memory arrays itself. In the example discussed, the 1-bit output of the FSM is programmed in bit 'D0' of the memory array.

Some FSMs may have more outputs than can be programmed into a single SEMB. For such FSMs multiple SEMBs can be connected in parallel using the same address inputs. For Moore type FSMs, in which the output depends on the current state, the state bits coming out of the SEMBs can be used to implement the output function external to an SEMB. The output function is implemented using the LUTs, this is illustrated in figure. 3. A Mealy machine can be transformed into a Moore machine [12], if the output are to be implemented using the LUTs.

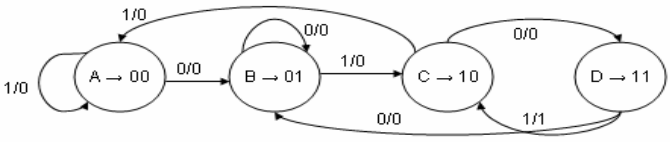

Figure 2a. State diagram for a "0101" detector

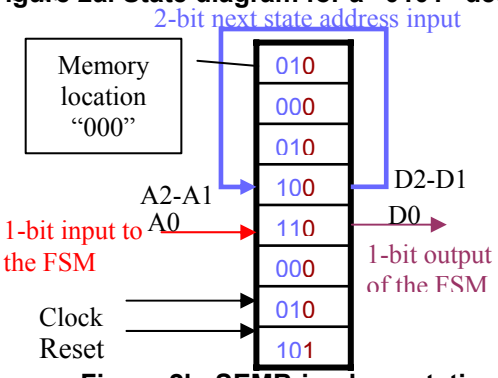

Figure 2b. SEMB implementation

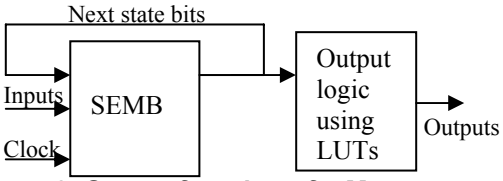

Figure 3. Output function of a Moore type FSM is implemented using LUTs

The input bits in the STG of some FSMs may contain many *don't-care* bits. If these *don't-care* bits are separated from the input bits, fewer input bits will be required to determine the state transition for each state. The *don't care* bits can be removed from the inputs by column compaction. In the STG, if all the rows specific to a state have the *don't care* bits at the same bit position then those bits can be removed from the input provided all the other states can also remove the same number of *don't care* bits. Since the position of the *don't care* bits can differ for different states, an input encoder is needed to select the corresponding inputs for each state. Our column compaction problem and the solution is different from the ones proposed earlier for symbolic inputs and outputs [13,14], because in our technique the inputs are not symbolic. This process is illustrated with figure 4. Column compaction is helpful when the total number of inputs and state bits are more than the number of address lines present in the SEMB. Thus instead of connecting more SEMBs in series to increase the address lines a multiplexer can be used to implement an FSM with fewer SEMB. This is also advantageous for power savings, as instantiating more SEMB increases the power consumption.

Once implemented, an SEMB based FSM has some advantages over a traditionally implemented FSM. The

functionality of an SEMB based FSM can be changed by changing the contents of the SEMB. The changes can be made quickly by re-writing the memory location which needs to be changed. This process of changing SEMB contents is much faster than going through the complete synthesis and placement and routing process. This is helpful for last moment engineering change orders (ECOs) and for unforeseen design changes.

The timing of the SEMB based FSM is predictable since the critical path is from the output of the SEMB to its address inputs. Thus no matter how many state transitions an FSM may have the timing of it does not change.

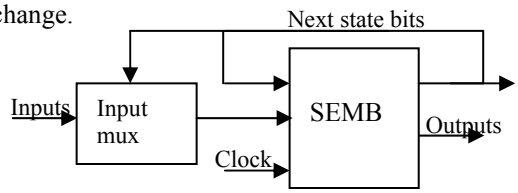

Figure 4. SEMB with an input multiplexer to select inputs for each state.

Although memory arrays have greater power consumption when compared to individual LUTs and FFs, for state machine which uses several FFs, LUTs, and significant routing resources, the SEMB based approach has lower power consumption than the FF based approach. The SEMB approach uses minimal routing resources. For example, if there are N states in a state machine, then $\log_2 N$ bits of the output of the memory array are connected to its address input along with the state machine's inputs. Routing resources are needed only to route a small number of signals from the output of the memory array to its address input, and to connect inputs of the FSM to the address input of the memory array. The state change in this approach only requires a change in address, thus any state machine, no matter how complex, can be implemented using just the memory arrays without any additional logic. Further, the FSMs implemented can be clocked at the maximum clock frequency supported by the memory arrays, which is close to the maximum clock frequency a design can execute on an FPGA. Figure 5 shows the algorithm to map an FSM into the SEMBs.

5. Experimental Results

To compare the traditional and proposed technique, power consumed by a design by both the approaches was measured. The target FPGA for our experiments was Xilinx Virtex-II XC2V250-6fg256. We have used benchmark circuits from the MCNC benchmark set [9], these benchmark circuits represent the STG of FSMs. In addition to the MCNC benchmark circuits we have used an FSM *prep4*, which is part of the prep benchmark suite [10]. The STG of these FSMs were synthesized using SIS [11] and a net-list in blif format was generated. This net-list contains the combinatorial portion of the FSMs and FFs to store the states. The blif net-list was then translated into structural VHDL and was technology mapped for the target FPGA using *synplify_pro* tool from Synplicity. The placement and routing of the mapped design was done using Xilinx *ISE 4.2.03i* design tool suite. The XPower tool, which is part of the ISE design suite, was used to calculate the power dissipation. XPower takes the design information from the placed and routed (.ncd) file and takes as input the clock frequency and switching activity information for all the nets in the design from a .vcd (value change dump) file. For all the benchmark designs, post place and route simulation was done using ModelSim simulator for a large number of random inputs. The switching activities for all the components was then saved in the .vcd file and was input to the Xpower tool to estimate the power consumption of the design.

```
Algorithm Map_FSM_in_SEMBs
    I:  number of inputs to an FSM
    O:  number of outputs of the FSM
1. encode each state in the STG, with total number of state bits
    equal to s
2. if( I+s < number of address lines available at any SEMB
    configuration) then    {
3.  if( O+s < data out width of the SEMB) then
4.          calculate SEMB contents using the STG;
5.  else
6.     while (! (O < data out width of SEMBs in parallel)) do
7.          join 1 SEMB in parallel having the same address
    inputs;
8.     endwhile;
9.     break;              }
10.  else   {
11.       from the STG find out the maximum number of inputs
    i' any state uses excluding don't care bits;
12.       if ( i' + s < number of address lines available at any
    SEMB) then      {
13.      decode input i to i' for each state and connect i' to
    address input of SEMBs;
14.         goto3;      }
15.       else      {
16.          while (! (i' + s < number of address lines available
    at any SEMB)) do
17.                    join 1 SEMB in series;
18.          endwhile
19.          goto 3;}
20.          }
End_Algorithm
```

Figure 5. Algorithm to map an FSM into the SEMBs

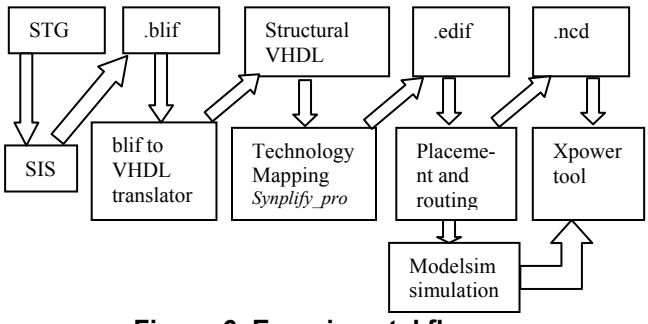

Figure 6. Experimental flow

For the SEMB implementation, the blockRAMs were instantiated in the VHDL code and connection to their address lines and outputs were made. The contents of the blockRAMs were initialized in the VHDL code; we have written a 'C' program to automatically generate the VHDL initialization string for these blockRAMs. The experimental flow to estimate power for the SEMB based FSMs is same as shown in figure 6. But instead of synthesizing the design using SIS, synplify_pro is used to synthesize the VHDL code. Among the benchmark circuits we used, the outputs of prep4 were implemented using the LUTs. The prep4 circuit has 16 states and 8 outputs, 16 states were encoded using 4 output lines of the blockRAM, which were also connected to the inputs of 8 LUTs to generate the FSM's output. Table 1 shows the area occupied by each benchmark circuit. Table 2 shows power (in MW) consumed by each benchmark circuit at different clock frequencies. In the SEMB based implementation only those benchmark circuits which need an input multiplexer require LUTs in addition to the blockRAMs. It is observed that power consumption of the FF based state machines goes up with the increase in complexity of the state machine (more number of transitions) and with increasing number of states. This is because more LUTs, FFs and programmable interconnect resources are needed to implement a complex state machine. For the SEMB based implementation, power consumption goes up with increasing number of inputs, outputs and states. Power consumed by the blockRAM is dependent upon the number of word-lines used, and number of bits in a word-line used. Therefore, an increase in the number of inputs and outputs and the number of states increases the power consumption of a blockRAM.

6. Stopping the Clock during Idle States

The embedded memory block in an FPGA has higher clock capacitance than an LUT and an FF. For this reason more power is consumed in clocking a blockRAM than an FF in a Virtex-II device. An FSM may not change its state and its output on every clock cycle, thus clocking the embedded memory array during these idle states will waste power. Through the STG it is possible to detect states and the corresponding set of inputs for which there will no state and output change. The clock to the blockRAM can be disconnected for these idle states until a transition in the state or outputs must be performed. In the Virtex-II series of FPGAs, the blockRAM provide an *enable port*. This *enable port* enables the read/write operation to the blockRAM. During the idle states of the FSM this port can be disabled, and the blockRAM is not clocked. Unlike the gated clock techniques, this method does not require any external clock gating and thus is glitch free.

Table 1. FPGA device utilization for different benchmarks circuits by both the approaches.

Benchmark	FF/LUT based FSM			SEMB FSM		
	LUT	FF	slice	LUT	slice	block RAM
Prep4	69	4	35	8	4	1
Dk16	114	5	58	0	0	1
Tbk	342	10	174	0	0	1
Keyb	112	5	57	16	8	1
donfile	66	5	33	0	0	1
Sand	263	10	134	20	10	2
Styr	241	8	123	16	8	2
Ex1	144	5	73	15	8	3
Planet	320	12	163	18	9	3

The clock control logic of the blockRAM is implemented using the LUTs. The inputs to the clock control logic for a Moore machine are the current state bits and the inputs to the FSM. For a Mealy machine output of the FSMs are also used to implement the clock control logic, because in a Mealy machine there can be conditions when the state does not change but outputs may change. We have written a program in 'C' which identifies all such idle states from the state transition graph and generates a behavioral VHDL code for the clock control logic.

The clock control technique can be used for the FF based implementation also, and it may appear that the same percentage of power should be saved. However, there is an important difference; the clock control logic will only stop the clock supplied to the FFs. The inputs connected to the combinatorial logic may still switch on each clock cycle even when there is no change in the state and the outputs. Thus the combinatorial portion of the FSM will continue to consume power during the idle states even after clock gating.

The clock control logic uses the state bits, input, and output signals as its input; therefore, it is in the critical path of the design. Similar to the input setup time, the enable signal should be stable before the rising edge of the clock. In other words, the clock frequency of the design will be slower proportional to the delay introduced by the clock control logic.

Table 3 shows the power consumed by each benchmark circuit with the clock control logic and table 4 shows the area overhead to implement the clock control logic. The amount of power savings achieved with the clock control logic is dependent upon the total time an FSM spends in idle states. For an FSM which spends very little time in idle states, there will be very little improvement over the traditional FF based implementation. On the other hand, significant power savings can be seen for an FSM which spends much of the time in idle states. Table 3 shows an average case (with 50% idle states) to make a comparison between the two techniques.

Table 2. Power consumed (in MW) by different benchmark circuits by both the approaches

Benchmark	FF/LUT based FSM			SEMB FSM			Percentage saving of SEMB implementation at 100MHz
	50MHz	85MHz	100MHz	50MHz	85MHz	100MHz	
Prep4	147.39	149.84	158.42	131.51	143.70	152.09	4.0
Dk16	128.42	139.56	144.33	122.19	128.97	131.38	9.0
Tbk	148.35	174.68	177.14	122.28	129.59	132.61	25.19
Keyb	127.18	137.45	141.85	122.22	129.02	131.93	7.0
Donfile	126.75	136.72	141.00	125.00	133.50	137.75	2.30
Sand	148.76	174.14	185.08	137.67	155.28	162.83	12.02
Styr	148.63	173.92	184.76	139.01	157.56	165.51	10.40
Ex1	155.00	184.75	197.50	156.94	188.04	190.37	3.6
Planet	168.45	207.61	224.39	156.18	186.75	199.85	11.0

7. Conclusion

In this paper, we have presented an alternate implementation of FSMs using embedded FPGA memory blocks. A complete procedure starting from an STG to the hardware implementation is outlined. Experimental results have shown that, the SEMB based approach saves power from 4% to 26% compared to the FF based approach. Furthermore, it is shown that clock supply to the embedded memory can be stopped during the idle states of the FSM for additional power savings. The SEMB based approach also has low area-overhead and provides flexibility to change an FSM's functionality without any design recompilation, by changing the memory contents.

Mapping of the FSMs into embedded FPGA memory array can unburden the programmable routing resource and has very low area overhead. Since commercial FPGAs have their own set of programming tools, to be used efficiently, this technique will have to be automated for different proprietary tools. Currently it is automated only for Xilinx Virtex-II series of FPGAs.

Acknowledgement: The synthesis and FPGA tools used in this work were donated by Synplicity and Xilinx respectively. Part of this work was supported by the University of Cincinnati summer graduate student research fellowship.

Table 3. Power consumed (in MW) by the benchmark circuits with clock control logic

Bench mark	SEMB FSM with clock control logic			percentage saving compared to the FF based approach at 100MHz
	50MHz	85MHz	100MHz	
Prep4	130.43	142.61	148.28	6.40
Dk16	120.24	125.65	127.97	11.33
Tbk	121.73	128.10	130.86	26.12
Keyb	120.91	126.79	129.13	9.00
Donfile	123.47	131.15	134.44	4.60
Sand	135.29	151.23	158.07	14.50
Styr	136.30	152.95	159.09	13.89
Ex1	156.14	186.84	188.75	4.44
Planet	151.49	178.77	190.47	15.20

Table 4. Area overhead of clock control logic for different benchmark circuits

Benchmark	Area overhead	
	LUTs	Slices
Prep4	10	5
Dk16	4	2
Tbk	62	32
Keyb	4	3
Donfile	4	2
Sand	47	25
Styr	17	30
Ex1	49	28
Planet	9	13

REFERENCES

1. Xilinx Incorporate, San Jose California, *Virtex-II data sheet*, version 2.3, March 2003

2. Altera Corporation, *Stratix Programmable Logic Device Family Data Sheet*, version 2.0, April 2002

3. Actel Corporation, *42MX FPGA Data Sheet*, ver 5.0, Feb' 01

4. L.Shang et al, "Dynamic Power Consumption in Virtex™–II FPGA family", *Proc. FPGA'03*, ACM Feb 2002

5. G.Sutter et al, "FSM Decomposition for Low Power in FPGA", Proc. FPL'02

6. J.Cong et al, "Technology Mapping for FPGAs with Embedded Memory Blocks", *Proc. FPGA'98*, ACM,

7. S.Wilton, "Heterogeneous Technology Mapping for FPGAs with Dual-Port Embedded Memory Array", *Proc. FPGA'00*, ACM, pp. 67-74

8. L.Benini et al, "Automatic Synthesis of Low-Power Gated-Clock Finite-State Machines", *IEEE Trans. CAD of IC*, vol. 15 No.6 June 1996

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

10. Programmable Electronic Performance Group, "PREP Benchmark Suite #4, Version 1.3," Los Altos, CA, 1994.

11. E. Sentovich, et al, SIS: A System for Seq. Circuit Synthesis. *Tech. Report Mem. No. UCB/ ERL M92/41*, Univ. of California, Berkeley, 1992.

12. Z. Kohavi, *Switching and Finite Automata Theory.* McGraw-Hill New York, 1978

13. Mitra, S. et al., "An Output Encoding Problem and a Solution Technique", *IEEE Trans.CAD*, vol. 18, no. 6, June'99

14. Binger, D. et al., "Encoding Multiple Outputs for Improved Column Compaction", *Proc. ICCAD-91*, 1991, pp. 230-233