A High Performance Low Power Implementation Scheme for FSM

Shuai Li, Ken Choi
VLSI Design and Automation Laboratory
Department of Electrical and Computer Engineering
Illinois Institute of Technology
3301 S Dearborn St., Chicago, IL 60616, USA
sli97@hawk.iit.edu, kchoi@ece.iit.edu

Abstract

Finite state machine (FSM) takes an important part in digital logic system. FSMs partition is one of the effective methods in regards to low power technique. Most of time only one of sub-FSMs need to be clocked, consequently power is saved. In this paper, we propose a high performance algorithm based on state transitions probability and low complexity control logic to implement the partitioned FSMs. The cost from one sub-FSM to other sub-FSMs could be minimum, and the transitions probability in one sub-FSM should be maximum. Based on the low complexity of control logic, we further give an optimized hardware architecture for the partitioned FSM model. Our proposed scheme has been implemented by using tsmc 45 nm technology library. Experimental results show that an average power reduction of 59% has been obtained for a set of standard FSM benchmark circuits.

Keywords-Low power; finite state machine (FSM); partition model; hardware structure

I. Introduction

Power consumption is a critical consideration in the performance of VLSI design. There are many different methods utilized in power reduction [1]. Partitioned finite state machine (FSM) is a useful technique widely used in low power design.

Power reduction is computed using (1):

$$P = P_{dynamic} + P_{short\_circuit} + P_{leakage} \tag{1}$$

The three terms indicate dynamic power, short circuit power and leakage power respectively. To address the low power design problem, a wide range of FSM partition algorithms and architectures have been developed [2-6].

In [2], the author proposed a transformed states transition graph (STG) in decomposed FSM. However, the extra transformed STG led to more area dissipation and control logic. In [3], the author proposed an asynchronous model for FSM partition, however, the control logic takes too much power consumption and the asynchronous sequential circuit is complex.

In this paper, we utilize a high efficient partition algorithm to divide the FSM to two or several sub-FSMs, we optimized the control logic by using the state transition table, and the synchronous structure we proposed is much easier to implement than the asynchronous structure.

The rest of the paper is organized as follows. Section II describes the proposed FSM partition algorithm. In section III, we propose the high efficient control logic and its corresponding structure. The experimental results of our proposed method are given in section IV. Finally, section V concludes this paper.

II. The proposed partition algorithm

A. *Probability prediction*

For a n-state FSM, the transition probability from state S_i to S_j is

$$P_{ij} = P\{X_n = j \mid X_{n-1} = i\} \tag{2}$$

We define each bit of the input signal has the same probability, e.g. p(0)=p(1)=0.5, so we can get the FSM states transition probability matrix P.

If the Markov chain is irreducible and aperiodic, there is a unique stationary $\pi = \{\pi_j, j \geq 0\}$, we have:

$$\sum \pi_j = 1, \pi_j > 0 \tag{3}$$

$$\lim_{n\to\infty} P_{ij}^{(n)} = \pi_j \tag{4}$$

$$\sum_j \pi_j P_{ij} = \pi_j \tag{5}$$

Through (3) to (5), we got the dcistribution π of a FSM.

$$\pi = \lim_{n\to\infty} P^{(n)} = [P_1 \quad P_2 \quad \cdots \quad P_{n-1} \quad P_n] \tag{6}$$

P_i denotes the steady probability of state S_i.

Finally, we got the total transition probability between S_i and S_j as shown in (7):

$$P_{i,j} = P_{ij} \cdot \pi_i + P_{ji} \cdot \pi_j \tag{7}$$

B. *Partition algorithm*

Since we got $P_{i,j}$, The partition flow is shown in Fig.1.

```
Algorithm: partition the monolithic FSM into n sub-FSMs
Input: $P_{i,j}$ and its corresponding states $S_i$ and $S_j$;
Output: sub-FSMs $M_1$, $M_2$, ... , $M_n$
Begin
    Sorting $P_{i,j}$ in decreasing order and marking them as $P_1$, $P_2$, ...,
$P_m$, (here m>n), thus the state pair ($S_i$, $S_j$) also sorted in decreasing order.
for (h=0;h<m;h=h+n) // h is the variable of state pairs
    for (k=1;k<=n;k=k+1) // k is the variable of sub-FSM
        if state pair $S_i$ and $S_j$ are never appeared before
            then $\{P_{k+h} \rightarrow (S_i, S_j)\} \in M_k$;
        else if $S_i$ (or $S_j$) already existed in sub-FSM $M_{sub}$
            then $S_j$ (or $S_i$) should also merged into the sub-FSM $M_{sub}$;
        else if both of $S_i$ and $S_j$ already existed in sub-FSM $M_{sub}$;
            then skip;
        end if
    endfor
endfor
End
```

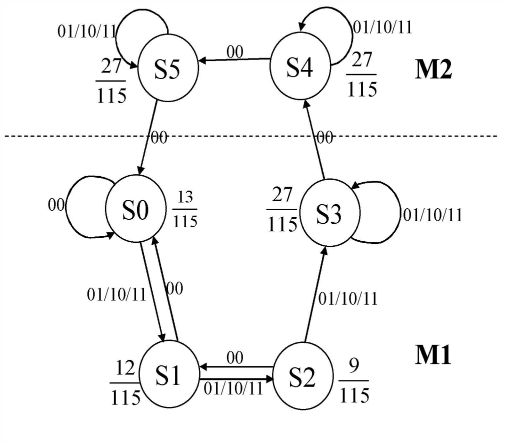
Figure 1. FSM partition algorithm

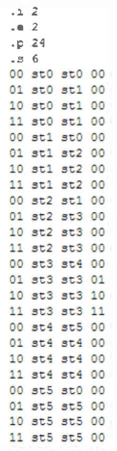
Table 1. Comparison of the results between monolithic FSM and proposed method

Bench mark	**states**	**transitions**	**Monolithic FSM**		**Conventional method**				**Proposed method**			
			Power (μW)	Area (μm^2)	Power (μW)	Area (μm^2)	Power Red.	Area Incr.c	Power (μW)	Area (μm^2)	Power Red.	Area Incr.
dk27	7	14	17.4	127	9.03	129	48%	1.6%	8.21	131	52.8%	3.1%
dk512	15	30	27.08	224.3	19.67	335.5	27.4%	49.6%	16.32	359.48	39.7%	60.3%
bbtas	6	24	13.22	122.01	5.80	155.33	56.1%	27.3%	2.743	136.10	79.2%	11.5%
lion	4	11	7.37	74.15	3.431	108.9	53.4%	46.9%	1.445	99.02	80%	33.5%
dk17	8	32	28.44	212.84	14.80	257.18	48.0%	20.8%	12.46	255.30	56.2%	19.9%
dk16	27	108	56.95	656.08	32.10	954.56	43.6%	45.5%	30.79	978.96	45.9%	49.2%
avg	-	-	-	-	-	-	**46.1%**	**32.0%**	-	-	**59.0%**	**29.6%**

III. Control logic and hardware structure

We take MCNC [7] benchmark "bbtas" as an example to illustrate our control logic as shown in Fig.2.





```
.i 2
.o 2
.p 24
.s 6
00 st0 st0 00
01 st0 st1 00
10 st0 st1 00
11 st0 st1 00
00 st1 st0 00
01 st1 st2 00
10 st1 st2 00
11 st1 st2 00
00 st2 st1 00
01 st2 st3 00
10 st2 st3 00
11 st2 st3 00
00 st3 st4 00
01 st3 st3 01
10 st3 st3 10
11 st3 st3 11
00 st4 st5 00
01 st4 st4 00
10 st4 st4 00
11 st4 st4 00
00 st5 st0 00
01 st5 st5 00
10 st5 st5 00
11 st5 st5 00
```

kiss2 format of "bbtas"

Figure 2. State transition graph of "bbtas"

We utilize the algorithm in section II to divide the FSM into two groups: $M_1:\{S0,S1,S2,S3\}$ and $M_2:\{S4,S5\}$. Through transition state table we can get the control logic. The proposed hardware structure as shown in Fig.3.

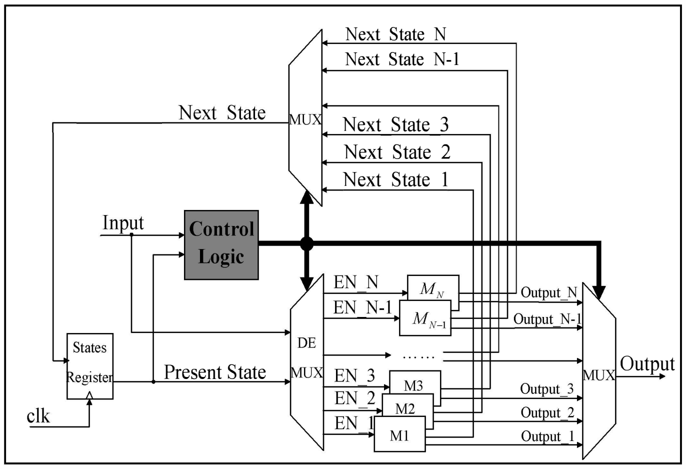


Figure 3. Optimized FSM hardware structure

The implementation model includes: State registers, control logic, sub-FSMs, MUX and DeMUX.

IV. Experimental results

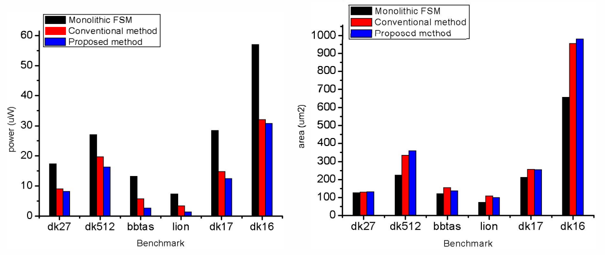


Figure 4. Comparison among monolithic FSM, conventional method and proposed method

As shown in table 1 and Fig.4, compared with orginal monolithic FSM and conventional method, the proposed method has three advantages:

1) Power dissipation is around 59% lower than monolithic.
2) The overall area is smaller than the convertional method.
3) The implementation is less complex.

V. Conclusion

We proposed a low power implementation model for partitioned FSMs and its corresponding structures. The average power reduction is 59% compared with the monolithic FSM, and the area is only increased by 29.6%. The results demonstrate that the proposed method is superior in terms of power dissipation to the anchor.

References

[1] M. Pedram, et al, "Low-power RT-level synthesis techniques: a tutorial," Computer and Digital Techniques, IEE Proceedings, Vol 152, Issue 3, Page(s): 333-343, 6 May 2005.
[2] Cao C, et al, "Synthesis tool for low-power finite state machines with mixed synchronous/asynchronous state memory," Computer and Digital Techniques, IEE Proceedings, Vol 153, Issue 4, Page(s): 243-248, 3 July 2006.
[3] Oelmann B, et al, "Asynchronous control of low-power gated-clock finite-state-machines," Electronics, Circuits and Systems, Proceedings of ICECS, Page(s): 915-918, 1999.
[4] Sue-hong Chow, et al, "Low power realization of finite state machine – a decomposition approach," ACM transactions on Automation of Electronic System, vol. 1, No. 3, July 1996, Pages 315-340.
[5] Monteiro, et al, "Finite state machine decomposition for low power," Design Automation Conference, 1998.
[6] Luca Benini, et al, "Automatic Synthesis of Low-Power Gated-Clock Finite-State Machines," IEEE transactions on computer-aided design on integrated circuits and systems, vol. 15, No. 6, June 1996.
[7] Saeyang Yang, "Logic synthesis and optimization benchmark user guide version 3.0," 1991.