State Re-Encoding for Peak Current Minimization

Shih-Hsu Huang
Department of Electronic Engineering, Chung Yuan Christian University, Chung Li, Taiwan, R.O.C.
shhuang@cycu.edu.tw

Chia-Ming Chang
Department of Electronic Engineering, Chung Yuan Christian University, Chung Li, Taiwan, R.O.C.
changb@vlsi.el.cycu.edu.tw

Yow-Tyng Nieh
SOC Technology Center, Industrial Technology Research Institute, Hsin Chu, Taiwan, R.O.C.
ytnieh@itri.org.tw

ABSTRACT

In a synchronous finite state machine (FSM), huge current peaks are often observed at the moment of state transition. Previous low power state encoding algorithms focus on the reduction of switching activities of state registers (i.e., state bits). However, even though the switching state registers are the same, different combinations of switching directions still result in different peak currents. Based on that observation, in this paper, we propose the first approach to re-encode an FSM by considering the switching directions of state registers in order to minimize the peak current caused by the state transition. Experimental data consistently show that the peak current is reduced with no penalty.

Categories and Subject Descriptors

B.6.3 [**Logic Design**]: Design Aids – *Optimization*.

General Terms

Design, Reliability.

Keywords

Finite state machine, Peak current, Sequential circuit synthesis.

1. INTRODUCTION

A sequential circuit can be thought of as a finite state machine (FSM). In sequential circuit synthesis, state encoding is to assign binary codes to the states in the FSM, and it has been recognized that state encoding greatly influences all quality aspects of the final implementation. Earlier attempts [1,2] at state encoding focused on the minimization of circuit area. In recent years, low power has become an important design issue. Since the power consumption of a CMOS circuit is highly dependent on the number of switching activities, it is natural that the objective of low power state encoding is to minimize the expected number of switching state registers (i.e., state bits) per state transition. Many low power state encoding approaches [3-6] have been proposed to assign binary codes with small Hamming distances to pairs of states that have a high transition probability.

As the process shrinks into the deep sub-micron technology, the power supply level fluctuation is exacerbated since the noise margin is considerably reduced. Since huge current peaks may lead to logic errors due to voltage drop or reliability problems due to electromigration, there is a strong demand to reduce the peak current. For a synchronous FSM, huge current peaks are often observed at the moment of state transition (since all state registers are clocked). Although previous studies [3-6] have aimed to minimize average power consumption caused by the state transition, no attention has been paid to the minimization of peak current caused by it.

Our paper is the first attempt to the minimization of peak current caused by the state transition. In fact, even though the same state registers switch, different combinations of switching directions still result in different peak currents. Based on that observation, in this paper, we propose the first work to re-encode an FSM by considering the switching directions of state registers. We use integer linear programming (ILP) to formally formulate our problem. We also present a polynomial time complexity algorithm to solve the same problem heuristically.

Our approach can be easily integrated into the existing design flow. The following three points are worthy to be addressed:

(1) For each state transition, the switching state registers in our re-encoded FSM are exactly the same as those in the original FSM. Therefore, compared with the original FSM, our re-encoded FSM has no penalty on the circuit area, the critical path delay, and the average power consumption.

(2) Our approach can be applied at different design stages. The designer can use either the ECO (engineering change order) process or the re-synthesis process to implement our re-encoding solution.

(3) Although a sequential circuit can always be represented by an FSM, data-dominated circuits (especially their data-paths) are often not designed from the viewpoint of FSM. If it is time-consuming to derive all states and all state transitions, our approach can be applied to reduce the peak current of functional simulation.

2. PRELIMINARIES

An FSM is conveniently described by a *state transition graph*, where each node represents a state, and each directed edge $S_i \rightarrow S_j$, associated with input and output values, represents a transition from state S_i to state S_j. Take the state transition graph shown in Figure 1 as an example. In state S_0, if the input value is 0, then the output values are 01 and the FSM moves to state S_3, whereas if input value is 1, then the output values are 01 and the FSM moves to state S_6; and so on.

When implemented in hardware, an FSM is generally realized by an architecture such as that shown in Figure 2. Here, each state corresponds to a binary code represented by the state registers.

ICCAD'06, November 5-9, 2006, San Jose, CA

The combinational logic computes the next state and output values based on the current state and input values. The input and output values are generally determined by external requirements, while the state encoding is left to the designer.

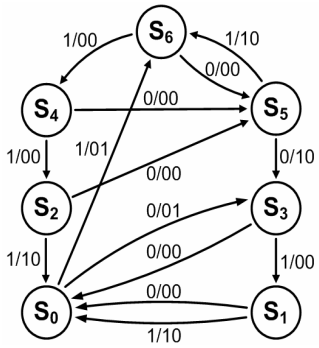

Figure 1: An example of state transition graph.

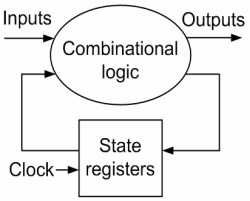

Figure 2: General architecture for FSM hardware implementation.

In a CMOS circuit, power is primarily consumed for charging and discharging activities. Since each state is encoded by the state registers, the power consumption of state transition is proportional to the number of switching state registers; i.e., the power consumption of state transition $S_i \rightarrow S_j$ is proportional to the Hamming distance between the binary code of state S_i and the binary code of state S_j. To reduce the average power consumption, binary codes with smaller Hamming distances should be assigned to pairs of states that have a higher transition probability.

3. MOTIVATIONAL EXAMPLES

In an FSM, a state register often drives many fan-outs with long wire length. Thus, for a state register, its output capacitance is often much larger than its internal capacitances. In other words, for a state register, power is primarily consumed for charging and discharging the output capacitance.

In fact, the switching directions of state register determine whether the output capacitance is charged or discharged. We analyze the following two switching directions.

(1) If the value of state register switches from logic level 0 to logic level 1, the output capacitance is charged. Thus there is a current that flows from the power line V_{DD} to the output capacitance.

(2) If the value of state register switches from logic level 1 to logic level 0, the output capacitance is discharged. Thus there is a current that flows from the output capacitance to the ground line V_{SS}.

Accordingly, we make the following hypothesis. If the value of state register switches from logic level 0 to logic level 1, the peak current occurs in the power line V_{DD}; on the other hand, if the value of state register switches from logic level 1 to logic level 0, the peak current occurs in the ground line V_{SS}.

To verify this hypothesis, we perform an experiment in which D-type flip-flop DFFX2 in TSMC 0.18 µm standard cell library is used as the state register. We assume that the clock transition time is 0.100 ns and the output capacitance is 0.300 pf. Figure 3 gives the HSPICE simulation results. The notation V(CLK) denotes the voltage of clock pin CLK. The notation V(Q) denotes the voltage of output pin Q. Note that the value of output pin Q corresponds to the value of state register. The notation $I(V_{DD})$ denotes the current that flows from the power line V_{DD}. The notation $I(V_{SS})$ denotes the current that flows to the ground line V_{SS}. As shown in Figure 3, the HSPICE simulation results are consistent with our analysis:

(1) If the value of the state register (i.e., the value of output pin Q) switches from logic level 0 to logic level 1, the maximum current of $I(V_{DD})$ is 692 µA, while the maximum current of $I(V_{SS})$ is 348 µA. Therefore, if the value of state register switches from logic level 0 to logic level 1, the peak current occurs in the power line V_{DD}.

(2) If the value of the state register switches from logic level 1 to logic level 0, the maximum current of $I(V_{DD})$ is 303 µA, while the maximum current of $I(V_{SS})$ is 829 µA. Therefore, if the value of state register switches from logic level 1 to logic level 0, the peak current occurs in the ground line V_{SS}.

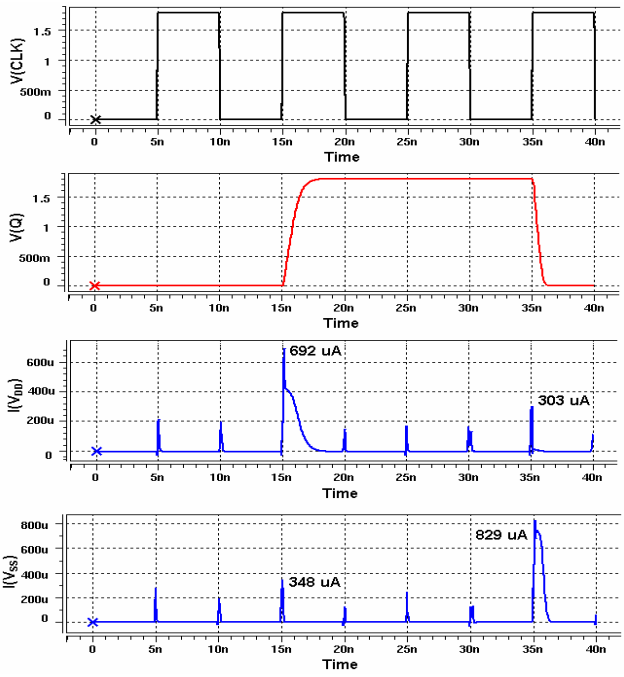

Figure 3: Simulation waveforms of D-type flip-flop.

4. MOTIVATION

Previous low power state encoding algorithms do not consider the switching directions of state registers. However, even though the same state registers switch, different combinations of switching directions still result in different peak currents.

We consider the following two encoding solutions for a state transition $S_i \rightarrow S_j$.

(1) If the binary codes of state S_i and state S_j are 000 and 111, respectively, the number of switching state registers is 3.

(2) If the binary codes of state S_i and state S_j are 001 and 110, respectively, the number of switching state registers is also 3.

Since the numbers of switching state registers are the same, the two encoding solutions are expected to have the same power consumption. Therefore, in previous low power state algorithms, the two encoding solutions are supposed to have exactly the same quality for this state transition.

However, in fact, for a state transition $S_i \rightarrow S_j$, the two encoding solutions correspond to different peak currents. We analyze their peak currents as below.

(1) If the binary codes of state S_i and state S_j are 000 and 111, respectively, all the three state registers switch from logic level 0 to logic level 1. Therefore, the output capacitances of all the three state registers are charged.
(2) If the binary codes of state S_i and state S_j are 001 and 110, respectively, two state registers switch from logic level 0 to logic level 1 and one state register switches from logic level 1 to logic level 0. Therefore, the output capacitances of two state registers are charged and the output capacitance of one state register is discharged.

The peak current of the first encoding solution is caused by the simultaneous charging activities of three state registers, while the peak current of the second encoding solution is caused by the simultaneous charging activities of two state registers. Therefore, the peak current of the second encoding solution is about 2/3 that of the first encoding solution. Obviously, from the viewpoint of peak current, the second encoding solution has better quality.

5. ILP APPROACH

In this section, we use ILP to formulate our problem: re-encoding an FSM to minimize the peak current caused by the state transition. Note that, for each state transition, the switching state registers in our re-encoded FSM must be exactly the same as those in the original FSM. Therefore, the peak current is minimized with no penalty on average power consumption.

For brevity sake, we use the term "peak value" to denote the maximum number of state registers that simultaneously switch in the same direction. Our objective is to minimize the peak value, and our ILP method guarantees obtaining the optimal solution. Suppose that the given encoded FSM has s states, and the corresponding hardware implementation has m state registers, where $2^m \geq s$. Without loss of generality, we assume that these m state registers are R_0, R_1, …, and R_{m-1}, and the state register R_{m-1} denotes the most significant bit of each binary code.

For each state transition $S_i \rightarrow S_j$ in the given encoded FSM, we define the two following functions: switching function $h_{i\rightarrow j}$ and direction function $d_{i\rightarrow j}$. The switching function $h_{i\rightarrow j}$ denotes the switching activities of state registers. If the value of state register R_k has a change when the state transition $S_i \rightarrow S_j$ occurs, then $h_{i\rightarrow j}(k) = 1$; otherwise, $h_{i\rightarrow j}(k) = 0$. The direction function $d_{i\rightarrow j}$ denotes the switching directions of state registers. If the value of state register R_k switches from logic level 0 to logic level 1 when the state transition $S_i \rightarrow S_j$ occurs, then $d_{i\rightarrow j}(k) = 1$; otherwise, $d_{i\rightarrow j}(k) = 0$.

Our approach is to re-encode the given encoded FSM. For each state register R_k, we define a binary variable x_k to denote the difference between our re-encoded FSM and the original given encoded FSM. If the value of the binary variable x_k is 0, then the value of state register R_k remains unchanged. On the other hand, if the value of the binary variable x_k is 1, then the value of state register R_k is complemented.

Let the integer variable *peak* represent the peak value. Then, this problem can be thought of as the problem of minimizing the value of the integer variable *peak* by assigning an appropriate binary value (i.e., 0 or 1) to each binary variable x_k, where k = 0, 1, …, m-1. Note that each state transition $S_i \rightarrow S_j$ imposes the following two constraints on the value of the integer variable *peak*.

(1) If the state register R_k switches from logic level 0 to logic level 1 in the original given encoded FSM and the value of the binary variable x_k is 0, then the state register R_k switches from logic level 0 to logic level 1 in our re-encoded FSM. On the other hand, if the state register R_k switches from logic level 1 to logic level 0 in the original given encoded FSM and the value of the binary variable x_k is 1, then the state register R_k also switches from logic level 0 to logic level 1 in our re-encoded FSM. The number of state registers that simultaneously switch from logic level 0 to logic level 1 gives a lower bound on the peak value. The constraint can be formulated as below:

$$\sum_{k=0}^{m-1} h_{i\rightarrow j}(k)\times d_{i\rightarrow j}(k)\times(1-x_k)+\sum_{k=0}^{m-1} h_{i\rightarrow j}(k)\times(1-d_{i\rightarrow j}(k))\times x_k \leq peak.$$

(2) The number of state registers that simultaneously switch from logic level 1 to logic level 0 gives a lower bound on the peak value. This constraint can be formulated as below:

$$\sum_{k=0}^{m-1} h_{i\rightarrow j}(k)\times(1-d_{i\rightarrow j}(k))\times(1-x_k)+\sum_{k=0}^{m-1} h_{i\rightarrow j}(k)\times d_{i\rightarrow j}(k)\times x_k \leq peak.$$

Therefore, if the given encoded FSM has n state transitions, our ILP formulation has $2n$ constraints.

Take the state transition graph shown in Figure 1 as an example. Suppose that the binary codes of state S_0, S_1, S_2, S_3, S_4, S_5, and S_6 are 000, 010, 101, 011, 110, 100, and 111, respectively, and Table 1 tabulates the binary code of each state. Then, we analyze the peak value of this given encoded FSM. For each state transition, Table 2 describes the switching directions of state registers and their corresponding currents. When the state transition $S_0 \rightarrow S_3$ occurs, two state register switch from logic level 0 to logic level 1 (i.e., the output capacitances of two state register are charged); when the state transition $S_0 \rightarrow S_6$ occurs, three state registers simultaneously switch from logic level 0 to logic level 1; and so on. From Table 2, we find that the peak value of this given encoded FSM is 3 and it is caused by the state transition $S_0 \rightarrow S_6$.

S_0	S_1	S_2	S_3	S_4	S_5	S_6
000	010	101	011	110	100	111

Table 1: Original binary codes of states.

State Transition	State Registers			Corresponding Currents
	R_2	R_1	R_0	
$S_0 \rightarrow S_3$	0→0	0→1	0→1	2 charging
$S_0 \rightarrow S_6$	0→1	0→1	0→1	3 charging
$S_1 \rightarrow S_0$	0→0	1→0	0→0	1 discharging
$S_2 \rightarrow S_0$	1→0	0→0	1→0	2 discharging
$S_2 \rightarrow S_5$	1→1	0→0	1→0	1 discharging
$S_3 \rightarrow S_0$	0→0	1→0	1→0	2 discharging
$S_3 \rightarrow S_1$	0→0	1→1	1→0	1 discharging
$S_4 \rightarrow S_2$	1→1	1→0	0→1	1 charging, 1 discharging
$S_4 \rightarrow S_5$	1→1	1→0	0→0	1 discharging
$S_5 \rightarrow S_3$	1→0	0→1	0→1	2 charging, 1 discharging
$S_5 \rightarrow S_6$	1→1	0→1	0→1	2 charging
$S_6 \rightarrow S_4$	1→1	1→1	1→0	1 discharging
$S_6 \rightarrow S_5$	1→1	1→0	1→0	2 discharging

Table 2: Analysis of the given encoded FSM.

Table 3 tabulates the switching function and direction function of this given encoded FSM. For example, for the state transition $S_0 \rightarrow S_3$, we have $h_{0\rightarrow 3}(2) = 0$, $h_{0\rightarrow 3}(1) = 1$, $h_{0\rightarrow 3}(0) = 1$, $d_{0\rightarrow 3}(2) = 0$, $d_{0\rightarrow 3}(1) = 1$, and $d_{0\rightarrow 3}(0) = 1$. According to Table 3, we have the following ILP formulation:

minimize *peak* subject to

$(1-x_1)+(1-x_0) \leq peak;$ $\quad x_1+x_0 \leq peak;$

$(1-x_2)+(1-x_1)+(1-x_0) \leq peak;$ $\quad x_2+x_1+x_0 \leq peak;$

$x_1 \leq peak;$ $\quad (1-x_1) \leq peak;$

$x_2+x_0 \le peak$; $(1-x_2)+(1-x_0) \le peak$;
$x_0 \le peak$; $(1-x_0) \le peak$;
$x_1+x_0 \le peak$; $(1-x_1)+(1-x_0) \le peak$;
$x_0 \le peak$; $(1-x_0) \le peak$;
$x_1+(1-x_0) \le peak$; $(1-x_1)+x_0 \le peak$;
$x_1 \le peak$; $(1-x_1) \le peak$;
$x_2+(1-x_1)+(1-x_0) \le peak$; $(1-x_2)+x_1+x_0 \le peak$;
$(1-x_1)+(1-x_0) \le peak$; $x_1+x_0 \le peak$;
$x_0 \le peak$; $(1-x_0) \le peak$;
$x_1+x_0 \le peak$; $(1-x_1)+(1-x_0) \le peak$.

After solving the ILP formulation, we have $peak = 2$ under the condition that $x_2 = 0$, $x_1 = 0$, and $x_0 = 1$. Therefore, the values of state registers R_2 and R_1 remain unchanged and the value of state register R_0 is complemented. As a result, we obtain the re-encoding solution shown in Table 4. For each state transition, Table 5 describes the switching directions of state registers and their corresponding currents. Compared with the original given encoded FSM, the peak value is reduced from 3 to 2. Therefore, our approach reduces the peak value about 33.3%, i.e., (3-2)/3.

State Transition	switching function $h_{i\to j}(k)$			direction function $d_{i\to j}(k)$		
	R_2	R_1	R_0	R_2	R_1	R_0
$S_0 \to S_3$	0	1	1	0	1	1
$S_0 \to S_6$	1	1	1	1	1	1
$S_1 \to S_0$	0	1	0	0	0	0
$S_2 \to S_0$	1	0	1	0	0	0
$S_2 \to S_5$	0	0	1	0	0	0
$S_3 \to S_0$	0	1	1	0	0	0
$S_3 \to S_1$	0	0	1	0	0	0
$S_4 \to S_2$	0	1	1	0	0	1
$S_4 \to S_5$	0	1	0	0	0	0
$S_5 \to S_3$	1	1	1	0	1	1
$S_5 \to S_6$	0	1	1	0	1	1
$S_6 \to S_4$	0	0	1	0	0	0
$S_6 \to S_5$	0	1	1	0	0	0

Table 3: Switching function and direction function.

S_0	S_1	S_2	S_3	S_4	S_5	S_6
001	011	100	010	111	101	110

Table 4: Our re-encoding solution.

State Transition	State Registers			Corresponding Currents
	R_2	R_1	R_0	
$S_0 \to S_3$	0→0	0→1	1→0	1 charging, 1 discharging
$S_0 \to S_6$	0→1	0→1	1→0	2 charging, 1 discharging
$S_1 \to S_0$	0→0	1→0	1→1	1 discharging
$S_2 \to S_0$	1→0	0→0	0→1	1 charging, 1 discharging
$S_2 \to S_5$	1→1	0→0	0→1	1 charging
$S_3 \to S_0$	0→0	1→0	0→1	1 charging, 1 discharging
$S_3 \to S_1$	0→0	1→1	0→1	1 charging
$S_4 \to S_2$	1→1	1→0	1→0	2 discharging
$S_4 \to S_5$	1→1	1→0	1→1	1 discharging
$S_5 \to S_3$	1→0	0→1	1→0	1 charging, 2 discharging
$S_5 \to S_6$	1→1	0→1	1→0	1 charging, 1 discharging
$S_6 \to S_4$	1→1	1→1	0→1	1 charging
$S_6 \to S_5$	1→1	1→0	0→1	1 charging, 1 discharging

Table 5: Analysis of our re-encoded FSM.

6. HEURISTIC ALGOROTHM

Our heuristic algorithm is an iteration process of re-encoding. For an encoded FSM *e*, we define the followings: the notation $peak_e$ represents the peak value of encoded FSM *e*, the notation $peak_{e,i\to j}$ represents the peak value of the state transition $S_i \to S_j$, and the notation num(*e*,*a*) represents the number of state transitions whose peak values are *a*.

Figure 4 gives the pseudo code of our algorithm, where the input is an encoded FSM *org*. The notation *sol* denotes our current solution. The function comp(*sol*,R_k) provides the re-encoding solution of encoded FSM *sol* by complementing the value of state register R_k. We say that the cost of encoded FSM *e1* is smaller than the cost of encoded FSM *e2*, if and only if one of the following two conditions is met:

(1) The value of $peak_{e1}$ is smaller than the value of $peak_{e2}$.
(2) Both the value of $peak_{e1}$ and the value of $peak_{e2}$ are equal to *a*, and the value of num(*e1*,*a*) is smaller than the value of num(*e2*,*a*).

A state register R_k is marked if its value in our current solution *sol* is complemented with that in the input *org*. At the beginning of our algorithm, all state registers are unmarked. We say that an unmarked state register R_{best} is the best-improvement state register, if both the following two conditions are met:

(1) The cost of re-encoding solution comp(*sol*,R_{best}) is smaller than the cost of our current solution *sol*.
(2) There does not exist another unmarked state register R_k so that the cost of re-encoding solution comp(*sol*,R_k) is smaller than the cost of re-encoding solution comp(*sol*,R_{best}).

In each loop of the while-do loop iteration, we attempt to find the best-improvement state register R_{best} among all unmarked state registers. If the best-improvement state register R_{best} exists, our solution *sol* becomes comp(*sol*,R_{best}), and the state register R_{best} is marked. If we cannot find the best-improvement state register, then we break from the while-do loop iteration and return our current solution *sol*.

```
Procedure Heuristic(org)
begin
  sol =org;
  all state registers are unmarked;
  while (true) do
  begin
    attempt to find the best-improvement state register R_best
    among all unmarked state registers;
    if (the best-improvement state register R_best exists) then
    begin
      sol = comp(sol,R_best);
      mark the best-improvement state register R_best;
    end
    else return(sol);
  end
end.
```

Figure 4: Pseudo code of our heuristic algorithm.

The number of loops spent in the while-do loop iteration is at most *m*, where *m* is the number of state registers. In each loop of the while-do loop iteration, the time complexity of finding the best-improvement state register R_{best} is O(*mn*), where *n* is the number of state transitions. Therefore, the time complexity of our heuristic algorithm is $O(m^2n)$.

7. APPLICATIONS AND EXTENSIONS

In this section, we study the applications and extensions of our approach. We address the following three points.

First, our approach can be applied at different design stages. Here we illustrate two methods to implement our re-encoding solution:

(1) One method is to apply the ECO process to implement our re-encoding solution based on the gate-level netlist of the original encoding solution. For each state register whose value is complemented with the original encoding solution, a pair of inverters is added around it.

(2) The other method is to apply the re-synthesis process to implement our re-encoding solution. To control the peak current of each state register, we can set constraints on the output of each state register during the re-synthesis process (e.g., the constraints on the maximum output capacitance value and the maximum fan-out number).

Second, since it may be time-consuming to derive the state transition graph of a data-dominated circuit, we propose a simulation-based methodology as below. First, by analyzing the switching directions of registers at each clock cycle of functional simulation, our approach can be used to reduce the peak current of functional simulation. Then, the ECO process (i.e., inserting inverter-pairs around appropriate registers) can be used to implement our solution. Note that, this simulation-based methodology is practical, since in the existing design flow, the designer also uses the functional simulation to validate the logic function and power consumption of a data-dominated circuit.

Third, our objective function is not restricted to minimize the number of state registers that simultaneously switch in the same direction. In fact, if we are given a gate-level netlist, we can have the output capacitance value of each register. Then, it is also reasonable to minimize the summation of output capacitance values that simultaneously switch in the same direction.

8. EXPERIMENTAL RESULTS

We use fourteen circuits to test the effectiveness of our approach. The fourteen circuits are classified into the following two groups:

(1) There are ten FSM designs adopted from MCNC FSM benchmark suite. Initially, these ten FSM designs are encoded by the POW3 algorithm proposed in [4].

(2) There are four data-dominated circuits. The data-dominated circuits IND1, IND2, and IND3 are industrial designs used in wide-band application, and the data-dominated circuit DSP is a digital signal processor.

8.1 Peak Value Minimization

We use Extended LINGO Release 8.0 as the ILP solver. As an alternative, we also use C++ programming language to implement our heuristic algorithm. Our platform is a Window XP personal computer with AMD K8-3GHz CPU and 1 Giga-Bytes RAM.

Table 6 tabulates the experimental results on the ten FSM designs. The column *States* gives the number of states. The column *Trans* gives the number of state transitions. The column *Peak Value* denotes the peak value. The column *Original* gives the peak value obtained by the original encoding solution. The column *Ours* describes the peak values obtained by our ILP approach and our heuristic algorithm, respectively. The column *ILP* denotes our ILP approach. The column *Heuristic* denotes our heuristic approach. We find that, in every FSM design, our ILP approach and our heuristic approach achieve exactly the same peak value. The column *Improve* gives the relative improvement of our re-encoding solution over the original encoding solution, i.e., *100% - Ours / Original*.

Table 7 tabulates the experimental results on the four data-dominated circuits. The column *Gates* gives the number of logic gates. The column *Regs* gives the number of registers. The column *Original* gives the peak value of the original circuit by analyzing the waveforms of functional simulation. Since the problem size is too large (i.e., the simulation patterns includes too many clock cycles), the ILP formulation cannot be solved within 6 hours. On the other hand, in each circuit, the CPU time of our heuristic algorithm is within 6 hours. Therefore, the column *Ours* only gives the peak values obtained by our heuristic algorithm.

Circuit	States	Trans	Peak Value			
			Original	Ours		Improve
				ILP	Heuristic	
BBSSE	16	56	4	3	3	25.0 %
EX1	20	138	4	3	3	25.0 %
S298	218	1096	8	7	7	12.5 %
S386	13	64	3	2	2	33.3 %
S510	47	77	5	3	3	40.0 %
S832	25	245	4	3	3	25.0 %
SAND	32	184	4	3	3	25.0 %
SSE	16	56	4	3	3	25.0 %
STYR	30	166	4	3	3	25.0 %
TBK	32	1569	5	4	4	20.0 %

Table 6: Experimental results on FSM designs.

Circuit	Gates	Regs	Peak Value		
			Original	Ours (Heuristic)	Improve
IND1	15329	1452	561	438	22.0 %
IND2	16863	597	207	171	17.3 %
IND3	5825	428	192	160	16.7 %
DSP	176685	6314	2852	2548	10.7 %

Table 7: Experimental results on data-dominated circuits.

8.2 Implementation Results

The fourteen test circuits are targeted to TSMC 0.18 μm process technology. The logic synthesis tool is Synopsys *Design Compiler*. To measure the peak current, we use Synopsys *PrimePower* as the gate-level power simulator. For each FSM design, in the power simulation process, every combination of state value and input values occurs at least one time. For each data-dominated circuit, the patterns used in power simulation are exactly the same as those used in functional simulation.

First, we apply the ECO process (i.e., inserting inverter-pairs around appropriate registers) to implement the results of our approach. Table 8 tabulates the ECO implementations of the ten FSM designs. Table 9 tabulates the ECO implementations of the four data-dominated circuits. For the convenience of readers, in Table 8 and Table 9, we compare the ECO implementations with the original circuits. The column *ECO* denotes the ECO implementation of our result. The column *Registers Peak Current* gives the peak current of registers. The column *Whole-Circuit Peak Current* gives the whole-circuit peak current. Note that the whole-circuit peak current happens at the moment of state transition. However, since the current waveforms of combinational logic have some overlaps with the current waveforms of registers, the whole-circuit peak current is larger than the peak current of registers. The column *Circuit Area* gives

the circuit area. The column *Over* denotes the relative increase (i.e., overhead) in the circuit area, i.e., *ECO / Original - 100%*.

Next, we apply the re-synthesis process to implement the results of our approach. Note that the re-synthesis process is only applicable to FSM designs. Therefore, we do not have the re-synthesis implementations of data-dominated circuits. For the ten FSM designs, Table 10 tabulates the comparisons of the re-synthesis implementations of our result with the original circuit. The column *Re-Syn* denotes the re-synthesis implementation.

9. CONCLUSIONS

Our paper is the first work for the minimization of peak current caused by the state transition. We show that the peak current of a switching state register is related to the switching direction. Based on the observation, we propose an approach to re-encode an FSM by considering the switching directions of state registers. Note that our approach can be applied at different design stages. Experimental data show that our approach works well in practice.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Council of R.O.C. under grant number NSC 95-2221-E-033-076.

REFERENCES

[1] T. Villa and A. Sangiovanni-Vincentelli, "NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementation," IEEE Trans. on Computer-Aided Design, vol. 9, no. 9, pp. 905—924, 1990.

[2] B.N.V.M. Gupta, H. Narayanan, and M.P. Desai, "A State Assignment Scheme Targeting Performance and Area," Proc. of IEEE International Conference on VLSI Design, pp. 378—383, 1999.

[3] E. Olson and S.M. Kang, "State Assignment for Low-Power FSM synthesis using Genetic Local Search," Proc. of IEEE Custom Integrated Circuits Conference, pp. 140—143, 1994.

[4] L. Benini and G.D. Micheli, "State Assignment of Low Power Dissipation," IEEE Journal of Solid State Circuits, vol. 30, no. 3, pp. 258—268, 1995.

[5] W. Noth and R. Kolla, "Spanning Tree Based State Encoding for Low Power," Proc. of IEEE/ACM Design, Automation, and Test in Eurpoe, pp. 168—174, 1999.

[6] F. Gao and J.P. Hayes, "ILP-Based Optimization of Sequential Circuits for Low Power," Proc. of IEEE International Symposium on Low Power Electronics and Design, pp. 140—145, 2003.

Circuit	Peak Value (bits)			Registers Peak Current (μA)			Whole-Circuit Peak Current (μA)			Circuit Area (μm^2)		
	Original	Ours	Improve	Original	ECO	Improve	Original	ECO	Improve	Original	ECO	Over
BBSSE	4	3	25.0 %	2261	1412	37.5 %	2930	2088	28.7 %	1317	1337	1.5 %
EX1	4	3	25.0 %	1938	1785	7.9 %	3336	3127	6.3 %	2574	2601	1.0 %
S298	8	7	12.5 %	7167	6213	13.3 %	11883	11248	5.3 %	19705	19748	0.2 %
S386	3	2	33.3 %	739	497	32.7 %	1002	835	16.7 %	1423	1440	1.2 %
S510	5	3	40.0 %	2214	1508	31.9 %	2765	2226	19.5 %	2025	2059	1.7 %
S832	4	3	25.0 %	3203	2306	28.0 %	3931	3255	17.2 %	2840	2867	1.0 %
SAND	4	3	25.0 %	947	737	22.2 %	1389	1245	10.4 %	4297	4317	0.5 %
SSE	4	3	25.0 %	2161	1204	44.3 %	2530	1523	39.8 %	1317	1337	1.5 %
STYR	4	3	25.0 %	2947	2398	18.6 %	4389	3845	12.4 %	4038	4071	0.8 %
TBK	5	4	20.0 %	1119	1050	6.2 %	4177	4128	1.2 %	3276	3309	1.0 %

Table 8: ECO implementations of FSM designs.

Circuit	Peak Value (registers)			Registers Peak Current (mA)			Whole-Circuit Peak Current (mA)			Circuit Area (μm^2)		
	Original	Ours	Improve	Original	ECO	Improve	Original	ECO	Improve	Original	ECO	Over
IND1	561	438	22.0 %	143.2	112.4	21.5 %	214.0	187.2	12.5 %	304092	306137	0.7 %
IND2	207	171	17.4 %	51.2	41.4	19.1 %	96.4	81.1	15.9 %	125704	126302	0.5 %
IND3	192	160	16.7 %	36.3	28.7	20.9 %	53.6	43.9	18.1 %	65057	65558	0.8 %
DSP	2852	2548	10.7 %	439.4	398.0	9.4 %	695.3	661.3	4.9 %	2179926	2184783	0.2 %

Table 9: ECO implementations of data-dominated circuits.

Circuit	Peak Value (bits)			Registers Peak Current (μA)			Whole-Circuit Peak Current (μA)			Circuit Area (μm^2)		
	Original	Ours	Improve	Original	Re-Syn	Improve	Original	Re-Syn	Improve	Original	Re-Syn	Over
BBSSE	4	3	25.0 %	2261	1519	32.8 %	2930	2115	26.5 %	1317	1237	-6.1 %
EX1	4	3	25.0 %	1938	1098	43.3 %	3336	2555	23.4 %	2574	2621	1.8 %
S298	8	7	12.5 %	7167	5648	21.2 %	11883	11229	5.5 %	19705	19655	-0.3 %
S386	3	2	33.3 %	739	588	20.4 %	1002	883	11.9 %	1423	1423	0 %
S510	5	3	40.0 %	2214	2076	6.2 %	2765	2644	4.4 %	2025	2029	0.2 %
S832	4	3	25.0 %	3203	2835	11.5 %	3931	3716	5.5 %	2840	2983	1.5 %
SAND	4	3	25.0 %	947	730	22.9 %	1389	1163	16.3 %	4297	4454	3.7 %
SSE	4	3	25.0 %	2161	1549	28.3 %	2530	1854	26.7 %	1317	1237	-6.1 %
STYR	4	3	25.0 %	2947	2461	16.5 %	4389	4077	7.1 %	4038	4028	-0.2 %
TBK	5	4	20.0 %	1119	1097	2.0 %	4177	4150	0.6 %	3276	3379	3.1 %

Table 10: Re-synthesis implementations of FSM designs.