State assignment for Low Power Dissipation

L. Benini, G. De Micheli

Center for Integrated Systems
Stanford University

Abstract

In this paper, we address the problem of reducing the power dissipated by sequential circuits. We propose a novel approach to power reduction based on transition graph description of sequential circuits. We formulate a general theoretic framework for the solution of the state assignment problem, and we propose different solutions trading off computational effort for quality of the results.

A heuristic algorithm has been implemented and been shown to sensibly reduce the switching activity of sequential circuits.

1 Introduction

Recent trends in VLSI system design and performance evaluation have stressed the increasing importance of power dissipation as a limiting factor and a cost parameter, as opposed to more traditional constraints, like area and speed.

The critical importance of low power circuits has attracted the attention of several researchers, in first instance for what concerns modeling and estimation of the average/instantaneous dissipated power, and later addressing the problem of synthesis for low power. Recent results [10, 3, 2] confirm that technology-independent and technology-dependent logic transformations applied to the combinational part of sequential circuits can be quite effective in low power optimization.

In our approach, we want to explore low power optimizations at the signal transition graph level of abstraction. Given a transition graph, a synthesis system has the possibility to decide important details of the implementation, more precisely those linked to the number of feedback variables and the state assignment.

The possibility of choosing a state assignment that minimizes the switching activity of the feedback network allows the synthesis algorithm to use degrees of freedom that are completely lost at the technology mapping phase and gives as a result an encoded state transition table that is a very effective starting point for further technology-dependent optimizations on the combinational part. The algorithms that we have developed target a state assignment that minimize the switching activity between state transitions, in such a way that the combinational part of the machine has lower input transition probability and it is more likely to give small power dissipation when synthesized.

2 Probabilistic models

At the gate level of abstraction, the power consumption in a circuit is proportional to its switching activity [4]. To find the average total power dissipated, we consider the average power dissipated by each gate during one clock cycle (or an arbitrarily defined $\Delta t = T_{cycle}$), we multiply it for the transition probability p_i of the gate output and we sum over all the gates in the network [1].

$$\overline{P_{tot}} = \frac{1}{2} V_{dd}^2 / T_{cycle} \sum_{i=1}^{N_{gates}} C_i \, p_i \qquad (1)$$

Power consumption minimization at the gate level targets the reduction of one or more of the parameters in eq. 1, namely:

- Voltage swing or frequency reduction. These parameters are decided by circuit designers.
- Reduction of $\overline{P_{tot}}$ by restructuring the sequential circuit.

  (1) Resynthetize combinational logic, for example reduce the number of nodes with high switching activity. This can be done in the technology mapping phase [3, 10, 2], or even include some ad-hoc technology independent logic optimizations. Note that combinational resynthesis affects the number of nodes in the network and their load capacitances.

  (2) Resynthetize the entire sequential circuit, by determining both a combinational structure and a register configuration that minimizes $\overline{P_{tot}}$.

We address the last problem, but, since it is a formidable task, we concentrate in this paper on the state assignment problem that determines the register configuration. Note again that state assignment strongly affects the size and the structure of the combinational component too.

In our approach, we consider first power minimization by restricting the cost function to the power dissipated by the registers, and then we investigate the impact of such minimization and its extension to the complete problem. In synthesis, the rationale in our solution can be summarized as follows:

- Determine a state encoding that minimizes code distance between state pairs with most frequent transitions.
- At the same time, bias the encoding algorithm in such a way that the resulting combinational part is minimal according to some area-related cost metrics.

Our starting point will be the state transition graph (STG) description of a sequential machine, defined by a vertex (state) set $S = \{s_1, ..., s_{N_{states}}\}$ and a related edge set representing the transitions.

We interpret the STG from a probabilistic point of view as a Markov chain. Given as initial data an input probability distribution (we assume for simplicity, equiprobability and independence), we can associate with each edge in the

0-7803-1886-2/94 $3.00 © 1994 IEEE

IEEE 1994 CUSTOM INTEGRATED CIRCUITS CONFERENCE

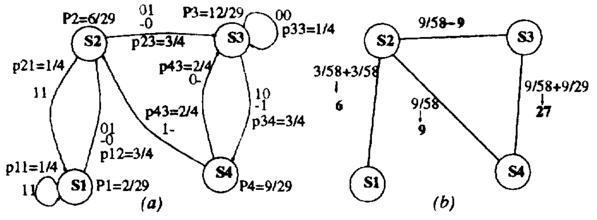

Figure 1: Part (a): STG, transition conditional probabilities (indicated with p_{ij}), input values corresponding to state transitions (indicated with binary digits) and consequent state probabilities (indicated with P_i). Part (b): transformation of the STG and normalization of the total probabilities for the transitions

graph a real value representing the probability that the corresponding transition takes place, given that the present state is the tail of the edge (this value is the conditional transition probability). But what we really need to know is the total probability of every transition in the graph, and this depends on the state probability distribution (in fact the total probability of a transition can be calculated as the product of the conditional probability and the state probability of the node at the tail of the corresponding edge [9]).

The first problem is to show that it is possible to compute the state probabilities and, more importantly, to show that these values are not dependent on the discrete time index, in other words that it is possible to compute a steady state probability vector. If we call "reset state" a state such that there is a transition to it from every other state in the STG, we can state (without proof) the following:

Theorem 1 *A STG with reset state and given conditional transition probabilities is an irreducible aperiodic Markov chain with all the states recurrent non null for which the steady state probability vector exists and is unique.*

An "irreducible Markov chain with all the states recurrent non null" is a chain where every state can be reached from any initial state, and the greatest common divisor of the length of the possible closed paths from every state is one.

In this case the problem of finding the steady state probability vector is reduced to finding the left eigenvector of the transition matrix corresponding to the unit eigenvalue, and normalizing it in order to make the sum of its elements equal to unity [9].

The transformation of the STG that is needed for our purposes can be summarized as follows:

- Calculation of the state stationary probability vector and of the total transition probabilities.
- Removal of the self loops (that do not imply state transitions) and labeling of the other edges with a weight equal to the correspondent transition probability (the weights are normalized to integers for simplicity).
- Collapsing of the possible multiple directed edges between two states with one undirected edge with weight equal to the sum of the directed edges probabilities.

Example 1 The transformation of the STG is illustrated in Fig. 1 (assuming that the input values are equiprobable and independent). Note that an edge with high conditional probability (like $s_1 \rightarrow s_2$) can have total probability equal or even smaller than an edge with small conditional probability (like $s_4 \rightarrow s_2$). Note that this graph is without reset state, but the stationary probability vector exists; this shows that the theorem above gives a sufficient but not necessary condition for the existence of such vector.

3 State assignment for low power

The main idea in our approach to this problem is to find a state assignment that minimizes the number of state variables that change their value when the FSM moves between two adjacent states; if we ideally can guarantee that each possible state transition implies the change of value of only one state variable, we know that nothing better can be done in reducing the switching activity associated with the given STG, as far as the registers are concerned. Once stated the essence of our goal, we now give some examples of possible particular solutions.

Example 2 Given a counter's STG, a state assignment that gives the minimum switching activity in the circuit is a Gray encoding of the states. Gray encoding is a solution only for this particular form of STG, for which the problem is quite trivial (note that in a counter the input is irrelevant, and all the transitions are equiprobable).

Example 3 Given a STG of arbitrary structure, a good performance in term of reduced switching activity is reached with One-Hot encoding of the states. One-hot encoding guarantees the switching of only two state variables for every transition, thus it is not minimum, but it does not require any algorithmic effort.

One hot encoding is not a practical solution because the number of state variables needed is equal to the number of states. Even if we accept such an overhead, our circuit will likely present a total increase of dissipated power, mainly because the combinational part will have a large number of inputs and outputs, with obvious increase in complexity and capacitive load.

An algorithm for low power state encoding must be applicable to arbitrary STGs and the number of state variables used should not be far from the minimum $\lceil log_2 N_{states} \rceil$. Given these intuitions, we can now proceed to a more formal definition of the problem.

We define the Hamming distance $d_{a,b}$ between two Boolean vectors $a, b \in B^n$ as the number of bits in the same position a_i, b_i with different phases: $d_{a,b} = \sum_{i=1}^{n} a_i \oplus b_i$, where the summation is the usual arithmetic summation.

If we describe the encoding using a Boolean matrix with rows corresponding to state codes and columns corresponding to the state variables, encoding for minimum switching activity can be formalized as finding Boolean row vectors $[e_i^1..e_i^{Nvar}]$, where $Nvar$ is the number of state variables used, that are solutions of the following ILP:

$$Min(\sum_{k=1}^{Nedges} w_{i,j}^k \sum_{l=1}^{Nvar} e_i^l \oplus e_j^l) \quad such\ that \tag{2}$$

$$\sum_{l=1}^{Nvar} e_i^l \oplus e_j^l \geq 1 \; \forall s_i, s_j, \; s_i \neq s_j \tag{3}$$

where $w_{i,j}^k$ is the weight on the edge between states s_i, s_j. The inequalities express the fact that no two states can be allowed to have the same code, and the cost function to minimize takes into account the need to give adjacent codes to states with high-probability transitions.

This "theoretical" solution is of little interest, because the size of the problem is formidable: the number of inequalities is proportional to the square of the number of the states, and the solution space to be explored is approximatively proportional to the factorial of the number of states. Nevertheless, we shall see in the next section that this formulation is very useful to give insight on more affordable heuristic solutions.

In summary two more observation are of interest. First, for several problems a solution with distance one between all the connected states is impossible: the presence of an odd cycle in the graph is an example of constraint not satisfiable with any distance-one encoding, but this fact does not mean that the minimum of the cost function cannot be reached. Second, the minimum length constraint can be relaxed, because an increase of the number of state variables can often lead not only to strong reductions of switching activity, but also to more compact realization.

4 Algorithms for state encoding

The problem of state encoding has been extensively studied in the recent past [6, 8, 7]. We propose a column-based approach (this name stems from the fact that the encoding Boolean matrix is found column by column) that can be summarized as follows. We consider one single state variable and we try to assign its value for each state in the graph, in a way that is likely to minimize the switching activity for the complete assignment; then, we proceed to the next variable.

The rationale is the following. The same value of the considered state variable should be given to states that are linked by high weight edges. During this process we must consider that, once all the available state variable have been assigned, no two states are allowed to have the same code (during the assignment procedure, if two variables have the same partial code, they are said to belong to the same indistinguishability class).

If the maximum number of state variables that we are allowed to use is N_{max} and we are assigning the bit codes for the l-th variable, the maximum number of indistinguishable partial state codes after the assignment must be less than $2^{(N_{max}-l)}$, otherwise in the following steps the remaining unassigned variables will not be able to create a code that identifies every single state.

A solution to this problem can be once again formulated as an ILP. Let e^l be the l-th variable (code bit) $1 \leq l \leq N_{max}$ of the state array, we call e^l_i its value (1 or 0) for the state s_i and $C_{k,l}$, $k = 1..Nclass_l$ the classes of states with indistinguishable partial codes after the assignment of the preceding $l-1$ variables:

$$Min(\sum_{h=1}^{Nedges} w^h_{i,j} \, (e^l_i \oplus e^l_j)) \text{ such that} \qquad (4)$$

$$\left\{ \begin{array}{lll} \sum_{s_i \in C_{k,l}} e^l_i & \leq & 2^{Nmax-l} \\ \sum_{s_i \in C_{k,l}} e'^l_i & \leq & 2^{Nmax-l} \end{array} \right. \forall C_{k,l}. \qquad (5)$$

This formalism can be clarified through an example.

Example 4 Consider the weighted graph in Fig. 1 (b). $N_{max} = 2$. At the beginning, no variable is assigned and all the states belong to the same class $C_{1,1}$. We have four states, so $2^{N_{max}-1} = 2$ (this means that we cannot give the same value to three states, because the remaining unassigned variable can distinguish only couples of states); ineq. 5 requires that we assign 0 to a couple of states an 1 to the other two. One assignment that minimize eq. 4 is $e^1_3 = e^1_4 = 1$ and $e^1_1 = e^1_2 = 0$ (Fig. 2 (a)).

The advantage of this approach lies in the reduced size of the problem that we have to solve at each step, the price paid is in lost of optimality for the result, in fact the column-based ILP solution guarantees the optimum assignment for one state variable at a time, but not for the global state array.

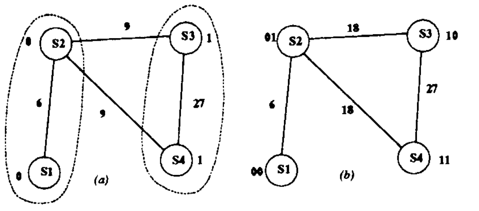

Figure 2: Part (a): First variable assignment. The indisting. classes of states after the first assignment are shown. Part (b): Second variable assignment and termination.

To improve the the final outcome, we want to bias the decision made in solving the l-th partial problem using the results in the preceding assignments: this is done by the dynamic modification of the weights in the cost function that are adjusted after each "column" assignment using the following formula:

$$w^h_{i,j_{new}} = w^h_{i,j_{old}}(d_{i,j} + 1); \qquad (6)$$

Where $d_{i,j}$ is the Hamming distance between the partially assigned codes of states s_i and s_j.

Example 5 In Fig. 2 (b) the new weight set is shown, and the final solution is found assigning the second variable in a way that gives the minimum edge violation and distinguishes all the states.

Even if using the column-based approach an optimal solution can be found for some problems of practical dimension, the ILP is an NP-complete problem. For this reason, in order to provide a faster heuristic, we have proposed a polynomial time solution for the column assignment problem. This is the pseudo-code of the algorithm:

```
assign($S$){
sort edges by weight in decreasing order;
foreach edge {$s_i$,$s_j$} {
  if($s_i$ and $s_j$ not assigned) {
    if(no Class violations) {
      $x$=select_bit($s_i$,$s_j$);
      $e_i = x$ $e_j = x$; }
    else {
      $x$=select_bit($s_i$, $s_j$);
      $e_i = x$; $e_j = x'$; } } }
  else if($s_i$ or $s_j$ not assigned) {
    $s_h$=unassigned($s_i$,$s_j$);
    $s_l$=assigned($s_i$,$s_j$);
    if(no Class violations) {
      $x$=select_bit($e_l$);
      $e_h = x$; }
    else
      $e_h = e'_l$; } }
}
```

The algorithm is based on a greedy choice of the constraint to satisfy, if it is impossible to assign the same code to two states (because an indistinguishability class becomes too big) a different code is assigned. The function "select_bit" makes a choice between two possible assignments based on the already assigned neighbor states.

We are now able to describe a general framework for the solution of the state encoding problem using the column based approach:

```
for l=1 to Nmax {
  adjust Class constraints; (eq.5)
  assign($S$);
  adjust the edge weights; (eq. 6)
}
```

where the procedure "`assign`" can be the exact ILP version or the fast heuristic described above.

If we are not constrained to use the minimum number of state variables, as this is often the case, we can try different solutions for multiple values of N_{max}. Increasing the number of state variable will very likely give a smaller number of constraints violations, but the number of state variables should not be allowed to increase too much from the minimum, in order to avoid an explosion in complexity of the combinational part of the FSM.

At this point, we need to consider the possibility to introduce additional constraints in order to obtain a near area-minimal realization of the combinational part of the network.

In order to tackle this problem, we have adopted a fast heuristic for minimal-area realization targeting multilevel logic implementation of the combinational part. Our algorithms are similar to those proposed in Mustang [7]. The user has the possibility to choose between two different options: a fanout-oriented heuristic, well suited for machines with small number of input and large number of outputs, and a fanin-oriented heuristic that performs better in the opposite situation.

For details on these algorithms, see [7], here only two points are worth some more notes. First, the area constraints are expressed with edge weights exactly like the power constraints, and we are able to give the user the possibility to specify different trade offs in their relative importance according to the design objectives. Second, even if our edge weight calculation for area minimization is similar to the one proposed in Mustang, our state assignment algorithm is column based, and this gives us the possibility to dynamically adjust the weights, allowing a potentially more effective state assignment.

5 Implementation and results

The heuristic algorithm described above has been implemented and applied to some benchmark circuits. The results are illustrated in Tab. 1 and have been obtained as follows. We have used a standard linear algebra package to find the total transition probabilities in the STGs, then we have applied our state assignment algorithm, POW3. We have then obtained a multilevel implementation of the state assigned FSM, using SIS standard script.

The same benchmarks have been processed using an area-oriented state assignment program, JEDI [5], and performing the same optimizations steps with SIS.

The implementations have been simulated with random patterns using MERCURY, a gate level simulator, in order to monitor the transitions in the networks that are directly related to the power consumption of the real circuits.

It is possible to observe that, for all the FSMs considered, our state assignment produces circuits with lower switching activity compared to those produced by JEDI, and the area (linked to the number of literals in the network) penalty paid is almost always small. What is more interesting is that the difference in switching activity seems to increase for bigger circuits, for which low power consumption is even more critical.

Note that we have not mapped the circuits using a technology library, since the algorithm described above is intended to be a preprocessing step in a complete synthesis tool that includes a low-power driven technology mapper, and at present we have only area-driven tools that will very likely reduce the positive impact of our optimizations. Thus our comparisons are based on transition activity, which is a good power

Circuit	Var.	Lit. JEDI / POW3	Tot. tr. JEDI / POW3	St. tr. JEDI / POW3	%
bbara	4	67 / 69	3630 / 3448	327 / 294	12
bbsse	4	126 / 131	8871 / 7970	1033 / 851	21
bbtas	3	25 / 25	2971 / 2690	610 / 456	35
dk14	4	120 / 114	7296 / 7083	1403 / 1104	27
dk17	5	76 / 77	5548 / 5463	1337 / 1081	23
dk512	5	67 / 87	7650 / 4825	2355 / 1538	53
donfile	5	102 / 214	5231 / 4573	1743 / 1378	26
planet	6	697 / 665	27859 / 19771	3204 / 1240	158
planet1	6	708 / 697	25735 / 16306	3205 / 1278	151
s1488	6	742 / 727	14073 / 13123	628 / 341	84

Table 1: Comparison between POW3 and JEDI after multiple level optimization: Circuit name, number of state variables, ratio of total number of transitions, ratio of number of state transitions and percent reduction of state transitions.

estimator for unmapped networks.

The last column in the table shows the reduced switching activity in the state lines, and this is an important information that can be used in future for further minimize the power consumption in the combinational network.

6 Conclusions

A novel state assignment algorithm targeting low-power consumption has been described and implemented. It compares favorably with the existing state assignment tools targeting minimal area implementations, achieving a 20% average reduction of total switching activity, and a 56% average reduction for state variable related activity.

This results confirms the importance of state assignment even for power, it also opens the way to the exploration of new algorithms for the optimization of the FSMs combinational part that take into account the reduced switching activity on the present state inputs (latch outputs).

References

[1] K. Keutzer A. Ghosh, S. Devadas and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proc. of Design Automation Conf.*, pages 253 – 259, June 1992.

[2] S. Devadas A. Shen, A. Ghosh and K. Keutzer. On average power dissipation and random pattern testability. In *Proc. of IEEE Int. Conf. On Computer Aided Design*, pages 402 – 407, November 1992.

[3] M. Pedram C. T. Tsui and A. Despain. Technology decomposition and mapping targeting low power dissipation. In *Proc. of Design Automation Conf.*, pages 68 – 73, June 1993.

[4] M. A. Cirit. Estimating Dynamic Power Consumption of CMOS Circuits. In *Proc. of IEEE Int. Conf. On Computer Aided Design*, pages 534 – 537, November 1987.

[5] B. Lin and A. R. Newton. Synthesis of multiple-level logic from symbolic high-level description languages. In *Proc. of IEEE Int. Conf. On Computer Design*, pages 187 – 196, August 1989.

[6] G. De Micheli. Symbolic Design of Combinational and Sequential Logic Circuits Implemented by Two-Level Logic Macros. *IEEE Transaction on CAD*, 5(4), pages 597 – 616, 1986.

[7] A. R. Newton S. Devadas, Hi-keung Ma and A. Sangiovanni-Vincentelli. MUSTANG: State Assignment of Finite State Machines Targeting Multilevel Logic Implementations. *IEEE Transaction on CAD*, 7(12), pages 1290 – 1300, 1988.

[8] A. Sangiovanni-Vincentelli T. Villa. NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementation. *IEEE Transaction on CAD*, 9(9), pages 905 – 924, 1990.

[9] K. Trivedi. *Probability and statistics with reliability, queuing and computer science applications.* Prentice-Hall, 1982.

[10] S. Malik V. Tiwari, P. Ashar. Technology mapping for low power. In *Proc. of Design Automation Conf.*, pages 74 – 79, June 1993.