Optimal FSMD Functional Partitioning for Low Power

1. Introduction

Partitioning a system has been shown an effective method for power reduction. Sub-circuits that are not needed can be shutdown, thereby reducing the switching activity of the overall system. Previous works have looked at partitioning only the finite state machine [1][2] or only the datapath [3][4]. In [1] and [2], the FSM is partitioned into two or more interacting FSMs and the clock to only one sub-FSM is active at a time. If the output values can be precomputed one cycle before they are needed, then the entire original logic circuit can be turned off in the next cycle as shown in [3]. The guarded evaluation technique in [4] tries to determine which parts of a combinational circuit are computing useful results. Sections that are not needed are then shut off. A FSMD functional partitioning technique where both the finite state machine *and* the datapath are partitioned together was reported in [5]. Results in [5] show that this technique can reduce energy by an additional 18% on average more than partitioning the FSM or datapath alone, with a savings of 41% compared to no partitioning at all. However, a simple partitioning heuristic was used in [5]. In this paper, we introduce an optimal algorithm for FSMD functional partitioning for low power. Our results show that an average energy savings of 49.2% is possible. An important criterion of the algorithm is the ability to estimate power very quickly. We therefore describe an efficient power estimation model and define theoretical energy bounds, which are used by the algorithm.

The rest of the paper is organized as follows. In Section 2, we will give an overview of the FSMD functional partitioning technique. Section 3 describes the power estimation model. Section 4 describes the optimal FSMD functional partitioning algorithm. Experimental results are shown in section 5, and we conclude in Section 6.

2. FSMD Functional Partitioning

The FSMD functional partitioning technique [5] is applied before synthesis. An FSMD, or finite-state machine with datapath, is a computation model commonly used at the register-transfer level, consisting of an FSM extended with datapath operations associated with states and transistions [6]. The original FSMD is first partitioned into several smaller mutually exclusive FSMDs, as shown in Figure 1. Each of these smaller FSMDs is then synthesized to its own custom processor, having its own controller and datapath. A communication bus is added to connect the processors together. The processors together are functionally equivalent to the original unpartitioned FSMD, and no extra cycles are added. Power reduction is accomplished because each processor is smaller than the original one large processor implementing the entire FSMD, and only one processor is executing a computation at any given time while the other processors will be idle. When a processor is idle, we have, in effect, shut down both the controller and the datapath for that processor. However, partitioning introduces extra power consumption for inter-processor communication. Thus, the problem that must be solved is one of partitioning the states such that the reduction in power for computations far outweighs the power increase for communication.

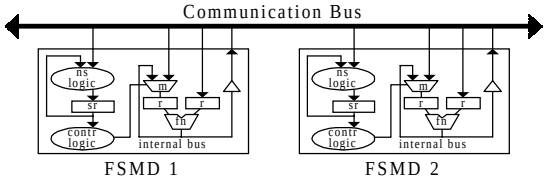

Figure 1. Architectural model.

3. Power Estimation Model

Dynamic power consumption of a general design is given by $P = CV^2fN / 2$ where C is the average capacitance switched per access, V is the supply voltage, f is the clock frequency, and N is the switching frequency of the unit (or the activity factor). From this equation, we see that power estimation depends on several factors that are known only after hardware assignment, scheduling and/or placement. Furthermore, the activity factor is known only after executing the design. At the FSMD level, much of the information is not known. For example, in order to calculate the power consumption of a bus, the bus capacitance must be known. However, the bus capacitance is dependent on the length of the wire and proximity to other wires, and this information is not known until after placement and routing., Much work has been done on performing very accurate power estimation [7], but for our purposes of partitioning optimization we require a very fast estimation technique. Fortunately, for high-level optimization, relative evaluation of different designs is more important than absolute evaluation. Therefore, we developed a fast and simple power estimation model at the FSMD level. The model is divided into two parts: the internal and external energy models. The internal energy is the energy consumed by a single processor while the external energy is the energy consumed by the communication between the processors. The total energy consumed by a partitioned FSMD system is the sum of the internal and the external energy. For the remaining discussion, we will restrict to partitioning the FSMD into two parts. The idea can be easily generalized to more than two parts.

3.1 Internal Energy

We define the internal energy as the total amount of energy consumed by all the states in a part. This excludes the communication energy. The energy for a state is the amount of power consumed by the state multiplied by the amount of time the state spends executing. Let $U = \{s_1, s_2, \ldots, s_u\}$ be the set of u states in the unpartitioned FSMD. Let A and B be two partitions of the FSMD such that $A \cap B = \varnothing$, $A \cup B = U$, and a and b be the number of states in A and

B respectively so that $a+b=u$. Let Es_i = energy consumed by state s_i. From [9], we see that the power consumed by a state can be approximated by the number of functional units and registers, and the amount of time spent executing in a state can be found by profiling. Furthermore, let

$$E_U = \sum_{\forall s_i \in U} Es_i \quad (1)$$

be the sum of the energy of the states in the unpartitioned FSMD,

$$E_A = \sum_{\forall s_i \in A} Es_i \text{ and } E_B = \sum_{\forall s_i \in B} Es_i$$

be the sum of the energy of the states in the partitioned FSMD *A* and *B*. We claim that the total energy for an *n*-state FSMD is equal to

$$\alpha_n \sum_{\forall s_i \in FSMD} Es_i \quad (2)$$

where α_n is determined by the complexity of the *n*-state FSMD. More detail on the complexity is given in section 3.2. Therefore, the total energy usage for the unpartitioned FSMD $E_{unpartitioned}$ is

$$E_{unpartitioned} = \alpha_u E_U , \quad (3)$$

and the total internal energy for the partitioned FSMD $E_{internal}$ is

$$E_{internal} = \alpha_a E_A + \alpha_b E_B . \quad (4)$$

Figure 2 shows the results of an experiment where the energy usage for a FSMD with different numbers of states with identical actions is evaluated. Both lines in the graph have a slope of less than one. This shows that adding the energy for an *n*-state FSMD with an *m*-state FSMD is less than the energy for an *n+m* state FSMD. For example, using the 4 FU line, the energy for the 20-state and 28-state FSMDs (152+241=393) is less than the energy for the 48-state FSMD (467). In other words, when the complexities of two individual states are summed, the result should be less than the complexity of the two states combined. Thus, we have the inequality

$$\alpha_a E_A + \alpha_b E_B < \alpha_u E_U . \quad (5)$$

3.2 FSMD Complexity

The FSMD complexity α addresses the issues of the internal interconnect and the size of the FSMD in terms of energy usage. The internal interconnect deals with the complexity of the datapath, whereas the number of states deals with the complexity of the control unit. It was observed in [8] that smaller capacitance is achieved in smaller designs because there are fewer and/or shorter interconnects, and fewer functional units and registers, which are obstacles during floorplanning and routing, which indirectly influence interconnect capacitance, and therefore, power usage. Thus, the internal interconnect capacitance is dependent on, among other factors, the internal bus length which in turn is dependent on the number of functional units, muxes, registers, etc., that need to be connected together, and the final layout area.

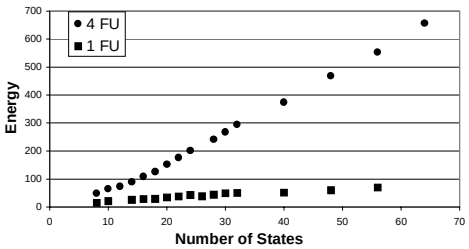

Figure 2. Energy versus the number of identical states for a FSMD.

Figure 2 shows that the energy usage of the FSMD is also related to its size and is approximated by the number of states in the FSMD. A similar relationship was also found in [9]. Figure 2 also shows how the number of states relates to the energy usage for different numbers of functional units. Thus, an approximation of the complexity, α_n, for an *n*-state FSMD is

$$\alpha_n = n \times (FU + mux + reg) \quad (6)$$

where *n* is the number of states in the FSMD, and *FU*, *mux*, and *reg* are the average number of functional units, multiplexers and registers respectively per state. α_1 is the complexity for one state. If α_1 for two different states are not equal, then we use the smaller number.

3.3 External Energy

The total energy of the partitioned system is not just $E_{internal}$. When there are two or more parts, communication must be added between the parts. We define the external energy as the energy consumed by the communication between parts. So the total energy for the partitioned system is

$$E_{partitioned} = \alpha_A E_A + \alpha_B E_B + E_{comm} . \quad (7)$$

The communication energy E_{comm}, is simply the sum of the energy (weights) of all the edges crossing between the parts, E_{xedge}, multiply by their activity factor, β:

$$E_{comm} = \sum_{\forall cross-edge\, i} (\beta_i \times E_{xedge_i}) \quad (8)$$

In our architectural model shown in Figure 1, only one external common bus is used to connect the two parts. All communication between parts occurs over this bus. Thus, the external bus is used every time when there is a transition from state s_i to s_j such that s_i and s_j are in different parts. A major factor that affects the bus energy is its length as reported in [8]. The bus length is approximated by the number of parts being connected together. The bus width is derived from the maximum data size crossing the parts. If the data is multiplexed over the bus, then the smaller data size is multiplied by the number of times required to transmit all the data. The data size is obtained from a dataflow analysis as discussed in [5].

Although in equation (5), we claim that $\alpha_A E_A + \alpha_B E_B < \alpha_U E_U$, however, with the added communication, the claim is not always true. In other words, it is possible that $\alpha_A E_A + \alpha_B E_B + E_{comm} > \alpha_U E_U$. Fortunately as we have found in most situations, there will be a partitioning such that

$$\alpha_A E_A + \alpha_B E_B + E_{comm} < \alpha_U E_U \ . \quad (9)$$

3.4 Finding the Energy Bounds

We will now evaluate lower (best) and upper (worst) bounds for the internal energy $E_{internal}$, the external communication energy E_{comm}, and finally the partitioned energy $E_{partitioned}$. These bounds will be used to perform an optimal partitioning using branch-and-bound.

3.4.1 Internal energy bounds

The following internal energy bounds progressively get tighter. We start with $[0, \alpha_u E_u]$ as the first bound. The lower bound is obvious. The upper bound is for an unpartitioned system. A second, tighter bound is $[\alpha_1 E_U, (\alpha_u - \alpha_1)E_U]$. The reason for this second lower bound is that the minimum energy for a partition is when there is no added complexity when all the states are added into the part. Thus, $E_{internal} = \alpha_1 E_A + \alpha_1 E_B = \alpha_1 E_U$. The upper bound comes from the fact that we need at least one state in one part in order to have a 2-way partition. Thus, we subtract the least energy for one state from the unpartitioned energy.

If some states are already assigned to either of the parts, we can get an even tighter third bound. Given the fact that some states are already assigned, we can calculate the internal energy for the current partitioning (i.e. currently known assigned states) using equation (4). From equation (5), we see that the worst that can happen is to put all states in the same part. Thus, to get the upper bound, we put all the remaining unassigned states together in the same part. The resulting energy will be either $(\alpha_a+\alpha_{rs})(E_A+E_{rs})$, or $(\alpha_b+\alpha_{rs})(E_B+E_{rs})$, where α_{rs} is the complexity of the combined remaining states and E_{rs} is the total internal energy of the remaining states. Since we know that some states are already assigned to another part, therefore, $E_{internal}$ can be either $[(\alpha_a+\alpha_{rs})(E_A+E_{rs}) + \alpha_b E_B]$ or $[(\alpha_b+\alpha_{rs})(E_B+E_{rs}) + \alpha_a E_A]$. We select the one that is the largest. Thus, the upper bound, $ub_{Einternal}$, is

$$ub_{Einternal} = max\begin{Bmatrix}(\alpha_a + \alpha_{rs})(E_A + E_{rs}) + (\alpha_b E_B), \\ (\alpha_b + \alpha_{rs})(E_B + E_{rs}) + (\alpha_a E_A)\end{Bmatrix} \quad (10)$$

In fact, this is the exact maximum for $E_{internal}$ given that some states are already assigned to the parts because $(\alpha_a+\alpha_b+\alpha_{rs})(E_A+E_B+E_{rs}) = \alpha_u E_U$ is the absolute maximum.

For the lower bound, we add to the current partitioning energy the total energy for the remaining unassigned states, E_{rs}, using the least complexity (i.e. α_1). Thus, the lower bound, $lb_{Einternal}$, is $lb_{Einternal} = \alpha_a E_A + \alpha_b E_B + \alpha_1 E_{rs}$.

To get an even tighter lower bound, we note that all the remaining states must be assigned to either of the two parts, thus, the complexity for these remaining states must be at least $\min(\alpha_a, \alpha_b)$. Thus,

$$lb_{Einternal} = \alpha_a E_A + \alpha_b E_B + (\min[\alpha_a, \alpha_b]) \cdot E_{rs} \ .$$

Furthermore, since at least one of the remaining states must be added to one part, the complexity of that part must at least be increased by α_1. Thus, an even tighter lower bound is

$$lb_{Einternal} = \alpha_a E_A + \alpha_b E_B + (\min[\alpha_a, \alpha_b] + \alpha_1) \cdot E_{rs} \quad (11)$$

3.4.2 External energy bounds

We will now provide bounds for the external communication energy, E_{comm}. Recall from equation (8) that E_{comm} is the sum of the energy (weights) of all the edges crossing between the parts. Thus, the first lower and upper bounds are when no edges and all edges respectively cross between parts. However, given the fact that some states are already assigned to either of the parts, therefore, some edges are already determined as to whether they cross between parts or not. Thus, knowing the current communication energy, E_{cc}, the upper bound for the communication energy, ub_{Ecomm}, is when all the remaining edges, E_{rc}, will cross between parts:

$$ub_{Ecomm} = E_{cc} + \sum E_{rc} \quad (12)$$

The lower bound for the communication energy, lb_{Ecomm}, is the sum of the currently known communication energy, E_{cc}, plus the minimum of all the remaining communication edges, E_{rc},

$$lb_{Ecomm} = E_{cc} + \min(E_{rc}) \quad (13)$$

3.4.3 Partitioned energy bounds

The bounds for the partitioned FSMD are simply the sum of the internal and communication energy bounds. Thus, the lower bound for the partitioned energy, $lb_{Epartitioned}$, is

$$\begin{aligned} lb_{Epartitioned} &= lb_{Einternal} + lb_{Ecomm} \\ &= [\alpha_a E_A + \alpha_b E_B + (\min[\alpha_a, \alpha_b] + \alpha_1) \cdot E_{rs}] \\ &\quad + [E_{cc} + \min(E_{rc})] \end{aligned} \quad (14)$$

and the upper bound for the partitioned energy, $ub_{Epartitioned}$, is

$$\begin{aligned} ub_{Epartitioned} &= ub_{Einternal} + ub_{Ecomm} \\ &= \left[max\begin{Bmatrix}(\alpha_a + \alpha_{rs})(E_A + E_{rs}) + (\alpha_b E_B), \\ (\alpha_b + \alpha_{rs})(E_B + E_{rs}) + (\alpha_a E_A)\end{Bmatrix} \right] \\ &\quad + [E_{cc} + \sum E_{rc}] \end{aligned} \quad (15)$$

4. Partitioning for Low Power

We will now describe an optimal algorithm for functional partitioning the FSMD for low power. Our objective is to find a partitioning among the FSMD states such that the total energy for the partitioned system is minimized as elaborated in [5]. The cost function used by this algorithm is based on the power estimation model described in the previous section.

Our optimal algorithm is based on a branch-and-bound technique. In this algorithm, a binary tree structure is used. Nodes that are promising are kept for further processing and those that are guaranteed to be worst are pruned. We start with the root having only one state. At each successive level, we add a new state and assign to the nodes in that level all possible combinations of the states for the two parts. Using equations (14) and (15), an upper and lower bound is calculated for each node. Depending on the bounds, a node is either kept or pruned. The most promising

node for a particular level is the one with the minimum energy for that level. Nodes that satisfy one of the following two conditions are guaranteed to be inferior and are therefore pruned:

Condition 1: if $LBn_i > UBmin$ then prune n_i.
Condition 2: if $LBn_i > Emin$ then prune n_i.

where LBn_i is the lower bound for the node n_i, *UBmin* is the current minimum upper bound, and *Emin* is the current minimum energy seen so far.

5. Experimental Results

We implemented the branch-and-bound optimal algorithm and the results are shown in Figure 3. The first column shows the examples used. The *States* column shows the number of states for the examples. The *Unpart* and *B&B* columns show the energy for the unpartitioned and the partitioned system. The *time* column shows the execution time in seconds for the algorithm, and the last column shows the percent energy savings obtained from the partitioning. There is no result for the MP example because the CPU run time took more than 14 hours. An average of 49.2% energy reduction was achieved using this algorithm. The external energy used was 10 times the internal energy. This compares favorably with the 41% average energy savings reported in [5].

Figure 4 compares the effect of different internal and external energy ratios. Columns two to nine show the percent energy reduction achieve for the external to internal energy ratio of 10, 50, 100, 200, and 500 respectively. The percentages show the energy reduction from the unpartitioned FSMD. Depending on the external to internal energy ratio, the average energy reduction can range from 10.9% to as much as 49.2%.

Examples	States	Unpart (uJ)	B&B (uJ)	Time (s)	% Savings
FAC	18	17,604	6,132	2	58.5%
Chinese	41	72,816	22,493	10hrs	66.6%
Diffeq	57	20,577	9,517	10hrs	48.5%
Volsyn	15	1,965	1,029	1	33.7%
NLoops	11	34,826	8,914	1	73.9%
MP	100	96,000	-	14+hrs	-
DSP	12	588	309	1	13.8%
Average					49.2%

Figure 3. Results from B&B partitioning.

Example	E/I=10	E/I=50	E/I=100	E/I=200	E/I=500
FAC	58.5%	43.2%	30.5%	15.9%	0.0%
Chinese	66.6%	60.8%	57.7%	52.2%	19.5%
Diffeq	48.5%	41.9%	0.0%	0.0%	0.0%
Volsyn	33.7%	0.0%	0.0%	0.0%	0.0%
NLoops	73.9%	71.6%	68.7%	63.0%	45.7%
DSP	13.8%	0.0%	0.0%	0.0%	0.0%
Average	49.2%	36.2%	26.1%	21.8%	10.9%

Figure 4. Effects of different external to internal energy ratios.

Figure 5 shows some statistics for the algorithm. The first and second columns show the number of states and the total number of nodes in the binary search tree respectively. The third column shows the percentage of nodes pruned. The *Branches Pruned* columns show the number of branches pruned as a result of satisfying conditions one and two respectively as discussed in section 4. Finally, the *Time* column shows the execution time for the algorithm. Although more than 99.9% of the nodes are pruned for the 50 state example, the execution time is still quite long.

6. Conclusions

We have presented an optimal branch-and-bound algorithm for performing FSMD functional partitioning for low power. The algorithm makes use of the power estimation model and the theoretical energy bounds for functional partitioning presented in the paper. An average of 49.2% energy reduction was achieved using this algorithm. Work is being done on comparing this optimal algorithm with a faster functional partitioning heuristic, and on further tightening the bounds for better performance.

References

[1] L. Benini, P. Vuillod, G. De Micheli & C. Coelho, "Synthesis of Low-Power Selectively-Clocked Systems from High-Level Specification," *International Symposium on System Synthesis*, pp. 57-63, Nov. 1996.

[2] J. Monteiro & A Oliveira, "Finite State Machine Decomposition For Low Power," *Proceedings of the Design Automation Conference*, pp. 758-763, 1998.

[3] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, & M. Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power, *IEEE Transactions on VLSI Systems*, 2(4):426-436, December 1994.

[4] Vivek Tiwari, Sharad Malik, & Pranav Ashar, "Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design," *International Symposium on Low Power Design*, 1995.

[5] Enoch Hwang, Frank Vahid, & Yu-Chin Hsu, "FSMD Functional Partitioning for Low Power," *Proceedings of the Design, Automation and Test in Europe*, pp. 22-28, March 1999.

[6] D. Gajski, N. Dutt, A. Wu, & S. Lin, *High-Level Synthesis Introduction to Chip and System Design*, Kluwer Academic Publisher, Boston, 1992.

[7] F. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on VLSI Systems*, 2(4):446-455, December 1994.

[8] A. Chandrakasan, M. Potkonjak, J. Rabaey, & R. Brodersen, "Hyper-LP: A System for Power Minimization Using Architectural Transformation," *Intl. Conf. on CAD*, pp. 300-303, 1992.

[9] R. Mehra & J. Rabaey, "Behavioral Level Power Estimation and Exploration," First International Workshop on Low Power Design, pp. 197-202, April 1994.

States	Total Nodes	% Nodes Pruned	Branches Pruned		Time (sec)
			Cond 1	Cond 2	
20	1.0×10^6	99.7%	0	21	1
25	3.3×10^7	99.8%	81	28,551	281
50	1.1×10^{15}	99.968%	1,137	495,915	36,719

Figure 5. Branch-and-bound statistics.