FINITE STATE MACHINES WITH DATAPATH PARTITIONING FOR LOW POWER SYNTHESIS

A. Sudnitson
Tallinn Technical University, ESTONIA

KEYWORDS: Dynamic power management, finite state machine with datapath, decomposition

Abstract: Resent investigations have shown the very good results of digital systems and circuits optimization using integration of dynamic power management in the design flow. This approach proceed from detection periods of time during which parts of the circuit are not doing useful work and shut them down by either turning off the power supply or the clock signal. In this work, we take this approach to design at register transfer level. We consider the partition technique for controller and datapath simultaneously and develop a decomposition procedure for the finite state machines with datapath (FSMD) model. The proposed techniques leads to a general low power design methodology based on functional partitioning of FSMD.

INTRODUCTION

With increasing sizes of designs and the need for low power applications, power is another optimization constraint that has become critical in addition to timing and area for very large scale integration circuits. The drive towards system on a chip (SoC) has accelerated the significance of a low power design methodology. In the last ten years, research on techniques for low power at various levels of design have intensified and much work has been done in the area of power consumption estimation and optimization, as surveyed in [1].

At the elementary transistor gate level (CMOS technology), we can formulate total power dissipation as the sum of three major components: switching loss, leakage, and short-circuit loss.

$$PW_{device} =(1/2)\ C\ V_{dd}^{2}\ a\ f + I_{leakage}\ V_{dd} + I_{sc}\ V_{dd}$$

Here, C is the output capacitance, V_{dd} is the supply voltage, f is the chip clock frequency, and a is the activity factor ($0 \le a \le 1$) that determines the device switching frequency. $I_{leakage}$ is the leakag current, and I_{sc} is the average short-circuit current. Also for current ranges of V_{dd} (say, 1volts to 3 volts) switching loss or the power consumed in charging and discharging the load capacitance of a gate (dynamic power dissipation) $(1/2)\ C\ V_{dd}^{2} a$ f remains the dominant component. So as a first-order approximation for the whole chip we may formulate the power dissipation as

$$PW_{chip} = =(1/2)[\sum C_i\ V_i^2\ a_i\ f_i]$$

C_i, V_i, a_i, and f_i are ***i***-th unit or block-specific average values. The summation is taken over all blocks or units *i*, at the microarchitecture level [2].

A wide range of techniques has already been proposed for the optimization of circuits for low power. Current research in low-power design focuses on techniques to reduce dynamic power dissipation of the circuit. The work presented in this paper exploits a fundamental and important source of power reduction – shutting down useless parts of a circuit. This idea is known as power management. Power management can be applied on different levels of the design process of application specific integrated circuits. In this work we consider techniques for register-transfer level (RTL) power optimization. After behavioral synthesis each design consists of at least one control unit (or controller) and one datapath (or operative part of design). The main difference from system level power management is that the shutdown of hardware is decided on every clock cycle, hence the name *dynamic* power management [1].

Organizing the target architecture as a datapath/controller model is an invariant part for most behavioral synthesis tools. Functional partition technique at RTL targeted for low power has been recently proposed by Hwang et al. [3]. It was shown that power savings would increase appreciably if both the controller and the datapath were and if the techniques were applied on the complete circuit, rather than on individual blocks. In addition to reducing power, FSMD functional partitioning also provides solutions to a variety of synthesis problems. As distinct from previous work [3,4] in this article conceptually more general theoretical background for partition [5] is considered and procedure of partition is elaborated.

BACKGROUND

To begin with, we outline three techniques could be used for inserting dynamic power management mechanisms into RT-level designs.

Precomputation relies on the idea of duplicating part of the logic with the purpose of precomputing the circuit output values one clock cycle before they are required, and then uses these values to reduce the total amount of switching in the circuit during the next clock cycle. In fact, knowing the output values one clock in advance allows the original logic to be turned off during the next time frame, thus eliminating any charging and discharging of the internal capcitances. It is shown [6] that it is important to resort to partial, rather than global, shutdown, i.e., to select for power management only a (possible small) subset of the circuit inputs.

Another approach to RT-level dynamic power management, known as gated clocks, provides a way to selectively stop the clock, and thus force the original circuit to make no transitions, whenever the computation to be carried out at the next clock cycle is useless. In other words, the clock signal is disabled in accordance to the idle conditions of the logic network.

Guarded evaluation [7] is the third popular RT-level shutdown technique. The distinctive feature of this solution is that, unlike precomputation and gated clocks, it does not require one to synthesise additional logic to implement the shutdown mechanism; rather it exploits existing signals in the original circuit. The approach is based on placing some guard logic, consisting of transparent latches with an enable signal, at the inputs of each block of the circuit that needs to be power managed. When the block must execute some useful computation in a clock cycle, the enable signal makes the latches transparent. Otherwise, the latches retain their previous state, thus blocking any transition within the logic block. In consequence of analysis of different techniques for dynamic power management at RT-level, our work proceeds from the fact that substantial problem is detection on a per-clock-cycle basis which parts of design is idle and integrate it in the synthesis procedures.

Typically, the synthesized design from high level steps of synthesis (scheduling and allocation) consists of two modules (Figure 1): a control part (or controller) and an operative part (or datapath). The formal description of control unit is a Mealy state machine which generates control signals to activate the different operations in specific clock cycles. The datapath unit consists of instantiation of datapath components such as multipliers, adders, incrementers and multiplexors.

Several different ways to specify register-transfer designs, including popular algorithmic-state machine (ASM) charts, and techniques for converting such an ASM chart into an design implementation consisting of a control unit and a datapath are presented in [5, 8].

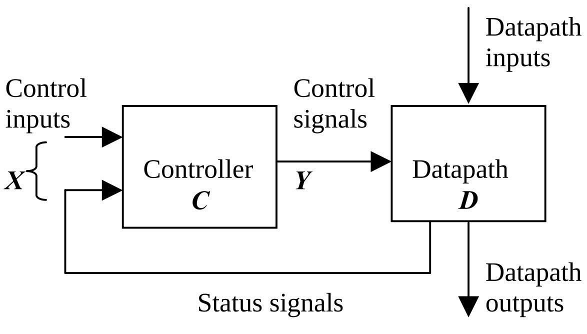

Figure 1: *High-level block diagram*

To synthesize designs at RT-level the model of an FSMD is introduced by Gajski in [8]. The FSMD computes new values for variables stored in the data path and produces outputs.

Our approach is based on the decomposition of FSMD. Informally the essence of the decomposition task could be described as follows.

Given a *prototype* FSMD description of a desired terminal behavior, the decomposition problem is to find two or more machines which, when interconnected in a prescribed way, will display that terminal behavior. The individual machines that make up the overall realization are referred to as *component* FSMDs. Each submachine corresponds to a subset of the set of states of source FSMD. An FSMD is partitioned into the set of interconnected FSMDs targeting optimization by criteria of power consumption. Each of these component FSMDs is then synthesized to its own custom processor, having its own controller and datapath. The objective is to investigate decomposition techniques for reduction of power consumption using *dynamic* power management without increasing appreciable design effort.

FSMD PARTITION

FSMD model

FSMD is an universal model that represents hardware design. The FSMD adds a datapath including variables, operators on communication to the classic FSM. To define FSMD formally, we must extend the definition of an FSM by introducing sets of datapath variables, inputs, and outputs that will complement the sets of FSM states, inputs and outputs.

An FSMD is formulated as a quintuple:
$< S, I \times SS, O \times AS, \delta, \lambda >$, where

- S is the set of states of the FSMD
- $I \times SS$ is the set of inputs of the FSMD. Inputs extended with status expressions
- $O \times AS$ is the set of outputs of the FSMD. Outputs extended with variable assignments
- δ is the next state function, mapping $S \times (I \times SS) \rightarrow S$
- λ is the output function, mapping $S \times (I \times SS) \rightarrow (O \times AS)$

The controller implements the FSM. It computes the next state and the signals controlling the transfers in datapath according to primary control input lines, status lines and the present state. The extracted FSM is described as unit with the set of binary inputs (channels) $X = \{x_1, \ldots , x_L\}$ and the set of binary outputs (channels) $Y = \{y_1, \ldots , y_T\}$ (Figure 1).

Let a microoperation be an elementary indivisible step of data processing in the datapath and let Y be a set of microoperations. Microoperations are induced by the binary signals $y_1, \ldots ,y_T$ from a controller. To perform the microoperation y_i $(i = 1, \ldots, T)$ the signal $y_i = 1$ has to appear at the output y_i . A set of microoperations executed concurrently in the datapath is called a microinstruction. Thus if $\beta_h = \{y^h_1, \ldots , y^h_t\}$ is microinstruction, then β_h is represented as subset of Y and the microoperations $y^h_1, \ldots , y^h_t$ are executed at the same clock period of an FSMD. The Y_t could be empty and we denote such an empty microinstruction Y_0 (“-“).

The variables $x_1, \ldots, x_L$ are the input variables of the controller and they may be changed during the microinstruction implementation. We can consider X as the set of coding variables of the set $(I \times SS)$ that is the set of inputs of FSMD. Similarly $Y = \{y_1, \ldots, y_T\}$ is the set of coding variables of the set $(O \times AS)$ that is the set of outputs of FSMD. The set of states of controller is equal to the set of states of source FSMD. The controller is usually represented as FSM with binary inputs and outputs. Formally, the FSM is defined as a quintuple $< S, X, Y, \delta, \lambda >$, where

$S = \{ s_1, \ldots, s_M \}$ is a set of states.

$X = \{x_1, \ldots, x_L\}$ is a set of binary input variables (channels).

$Y = \{y_1, \ldots, y_T\}$ is a set of binary output variables (channels).

$\delta: D(\delta) \rightarrow S$ is a multiple valued next state function with domain $D(\delta) = D_1 \times \ldots \times D_L \times S$ and codomain S. $D_i = \{0, 1\}$ represents a set of values (symbols) each input variable x_i may assume.

$\lambda: D(\lambda) \rightarrow R(\lambda)$ is an output function with domain $D(\lambda) = D(\delta)$ and codomain $R(\lambda) = E_1 \times \ldots \times E_T$.

$E_i = \{0, 1\}$ represents a set of values each output variable y_i may assume.

The behavior of a controller can be described by state transition graph or, equivalently, by presentation by the list of transitions.

For example of prototype FSMD control part behavior representation we refer to table 1.

Present state s_i	Next state s_j	Input condition α_h	Output signals β_h	h
s_1	s_1	x_1	y_7	1
	s_3	$\wedge x_1$	-	2
s_2	s_3	1	$y_2\ y_{10}$	3
s_3	s_6	x_7	$y_2\ y_{10}$	4
	s_8	$\wedge x_6\ \&\ x_7$	-	5
	s_2	$\wedge x_6\ \&\ \wedge x_7$	$y_2\ y_5\ y_{10}$	6
s_4	s_1	$x_1\ \&\ x_2$	$y_3\ y_4$	7
	s_3	$x_1\ \&\ \wedge x_2$	$y_1\ y_3\ y_4$	8
	s_5	$\wedge x_1\ \&\ x_4$	$y_6\ y_9$	9
	s_8	$\wedge x_1\ \&\ \wedge x_4$	$y_6\ y_8\ y_9$	10
s_5	s_4	$x_3\ \&\ x_4$	$y_6\ y_9$	11
	s_5	$x_3\ \&\ \wedge x_4$	$y_6\ y_9$	12
	s_8	$\wedge x_3$	$y_6\ y_8$	13
s_6	s_2	x_5	$y_2\ y_{10}$	14
	s_3	$\wedge x_5\ \&\ x_7$	y_5	15
	s_8	$\wedge x_5\ \&\ \wedge x_7$	$y_2\ y_{10}$	16
s_7	s_5	1	y_1	17
s_8	s_8	x_3	-	18
	s_5	$\wedge x_3$	$y_8\ y_9$	19

Table 1: *Transition table of illustrative example*

We use the formal notion of generalized transition (g-transition).

g-transition is quartuple $< s_i, s_j, \alpha_h, \beta_h >$ where s_i is the present state, s_j is the next state, α_h is the input condition (Boolean function), β_h is the output (microinstruction) of transition.

In our example, every row of Table 1 defines one g-transition from a source state to a destination state along with certain output microinstruction according to a certain input condition (term).

If s_p is the present state and $\{ s_r / r \in R_m \subseteq \{1,...,M\} \}$ is the set of the next ones where transitions from s_m are possible then there is the set of functions $\{\alpha_{pr} / r \in R_m\}$.

The search for the next state means the evaluation of the Boolean functions. It is necessary to evaluate which of these functions has value "true" for a given input combination **z** from $\{0, 1\}^L$.

Controller Decomposition

Decomposition model. A collection φ of nonempty subsets of a set S whose union is S (such that if $B_i, B_j \in \varphi$, then $B_i \subseteq B_j$ implies $i=j$) is called a *cover* on S. The notion of cover s generalization of a partition, that is a collection of *disjoint* subsets of S whose set union is S. We refer to these subsets as blocks of the cover (partition).

Let $\pi = \{Y^1, \ldots Y^n\}$ be the partition on the set of output variables Y.

Let $G = \{g_1, \ldots g_H\}$ be the set of g-transitions in the transition table (in our example every g-transition corresponds to row in table 1),

$X(g_h)$ and $Y(g_h)$ be the sets of essential input and output variables (microoperations) in the g-transition g_h $(h = 1, \ldots H)$,

$X(s_i)$ and $Y(s_i)$ be the sets of input and output variables at the transitions from the state s_i.

On the set G we define relation ξ such that $g_i\ \xi\ g_j$ if there is at least one common variable in the sets $Y(g_i)$ and $Y(g_j)$:

$g_i\ \xi\ g_j \Leftrightarrow Y(g_i) \cap Y(g_j) \neq \varnothing$

This relation is symmetric and reflexive and induces the cover μ on the set Y.

For every block Y^p from partition π we put in accordance component FSM A^P in the network N.

Let us put the cover φ on the set of states S and the cover ψ on the set of g-transitions G of the transition table in accordance to the pair $< C, \pi >$:

$\varphi = \{B^1, \ldots, B^n\}; \quad B^p \subseteq S, \quad s \in B^p \Leftrightarrow Y(s) \cap Y^p \neq \varnothing;$

$\psi = \{G^1, \ldots, G^n\}; \quad G^p \subseteq G,$

$g \in G^p \Leftrightarrow Y(m) \cap Y^p \neq \varnothing.$

From above it follows that the state s will be in the block B^p of the cover φ, there is at least one output variable from the block Y^p of the partition π at the transitions from this state s. It is also evident that the state s may be in several blocks of φ, for example, in B^p and B^r, if $Y(s) \cap Y^p \neq \varnothing$ and $Y(s) \cap Y^r \neq \varnothing$, i. e. the output variables from Y^p and Y^r are produced at the transitions from state s.

In exactly the same way, it follows that the g-transition g_h will be in the block G^p of the cover ψ, if at least one output variable from the block Y^p of the partition π is written in the row of transition table corresponding transition g_h. Just as for φ, the same m will be in several blocks of ψ, for example, in G^p and G^r, if

$Y(m) \cap Y^p \neq \varnothing$ and $Y(h) \cap Y^r \neq \varnothing$,
i.e. the output variables from Y^p and Y^r are written in the row m.
In our example (Table 1),
μ={ {y_7}, {y_2, y_5, y_{10}}, {y_1, y_3, y_4}, {y_6, y_8, y_9} }
ψ={{1, 2, 7, 8, 17}, {9, 10, 11, 12, 13, 18, 19}, {3, 4, 5, 6, 14,15,16} }

Affirmation 1. Given FSMD *A* and the partition $\pi = \{Y^p \mid, i \in \{1, \dots n\}\}$ on the set of output variables Y. Then there exists a network *N* of FSMDs with alternatively active datapath (datapath of only one component FSMD is active every clock period) that realizes *A* if and only if $\pi \geq \mu$.

The choice is very important step of low power design and should be fulfilled with allocation at high level synthesis. It is not considered in this work.
As an example, we will take partition
π={ {y_1, y_3, y_4, y_7}, {y_6, y_8, y_9 }, {y_2, y_5, y_{10}} },
that satisfies the condition $\pi \geq \mu$..
As an outcome of it, the datapath shutdown techniques could be applied, portions of the combinational logic in the datapath can be shut down for some cycles when those results are either precomputed or are not required.

Affirmation 2. If decompositional partition π on *Y* is such that for all states s_i of FSMD A exists block Y^j such that $Y(s_i) \subseteq Y^j$ than constructed network consists of multiple-exclusive communicating component FSMDs (only one pair controller/datapath is active).

In the last case we are able to apply shutdown technique that considers both the controller and datapath simultaneously. We partition a digital system into multiple simpler communicating processors, and then shut down the inactive processors (i.e. the inactive controller/datapath pairs).
The following procedure of decomposition show the evidence of affirmations.

Decomposition procedure. Let us put the network *N* with n component FSMD
$A^p = < S^p, X^p, V^p, \delta^p, \lambda^p >, \quad p = 1, \dots n,$
in accordance to triplet $< A, \pi, \varphi, \psi >$. The number of component machines is equal to the number of blocks in the partition π (or the cover φ and ψ).
Further the steps of decomposition procedure are presented in formal way.

1. $S^p = B^p \cup \{b_p\}$ is the set of states in the component controller C^p, where B^p is the *p*-th block of the cover φ, and b_p is additional state in C^p.
2. $X^p = X(G^p) \cup Z_x^{\ p}$ is the set of input variables in the component controller C^p.
   Here $X(G^p) = \cup_{m \in G^p} X(g_h)$
   $X(g_h)$ is the set of essential input variables in the g-transition g_h of the controller *C* of the prototype FSMD;
   G^p is the p-th block of the cover ψ.
   $Z_x^{\ p} = \{z_t \,/\, \delta(s_j\,,\,\alpha_h) = s_t\,;\, s_t \in S^p\,,\, s_j \notin S^p\}$.
3. $V^p = Y^p \cup Z_y^{\ p}$
   $Z_y^{\ p} = \{z_i \,/\, \delta(s_t\,,\,\alpha_h) = s_j\,;\, s_t \in S^p\,,\, s_j \in S^k,\, s_t \notin S^k\}$.
4. Assume that there is a g-transition $< s_j, s_t, \alpha_h, \beta_h >$ in of prototype FSMD (the transition from s_j to s_t with the input condition α_h and the output β_h in the controller *C*):
   $\delta(s_j\,,\,\alpha_h) = s_t;\ \lambda(s_j\,,\,\alpha_h) = \beta_h$.

   Define the corresponding transitions in component controller.
   Ω_j be the set of component controllers with the state s_j,
   Ω_t be the set of component controllers with the state s_t,
   $\Omega_{jt} = \Omega_j \cap \Omega_t$ be the set of component controllers with the states s_j and s_t
   If $C^k \in \Omega_{jt}$, then in C^k
   $\delta^k(s_j\,,\,\alpha_h) = s_t$.

   If $A^p \in \Omega_j \setminus \Omega_{jt}$, ($s_i$ is the state of C^p and s_t is not the state of C^p), then in C^p
   $\delta^p(s_j\,,\alpha_h) = b_p$.
   The output controlling datapath of q[th] component FSMD for controller $C^q \in \Omega_j$ (s_i is the state of C^q and it is also possible that the next state s_t is the state of C^q) is equal to $\beta_h \cap Y^q$. In addition to controlling outputs one and only one controller from Ω_j (say C^r), must generate the output signal which forces each controller from $C^u \in \Omega_t \setminus \Omega_{jt}$ (if this set is not empty) to transit from the additional (idle) state b_u to s_t and in C^q
   $\lambda^r\,(s_j, \alpha_h) = \beta_h \cap Y^u \cup \{z_t\}$.
   If $C^u \in \Omega_t \setminus \Omega_{jt}$ (s_t is the state of C^u and s_j is not the state of C^u), then in C^u
   $\delta^u(b_u\,,\,z_t) = s_t$
   $\lambda^u\,(b_u, z_t) = Y_0$.
5. The initial state of the component controller is equal to s_0 , if $C^p \in \Omega_1$ and b_p otherwise.

ILLUSTRATIVE EXAMPLE

Let us examine the decomposition procedure.
The number of states of component FSM is equal to the number of states in corresponding block of cover φ plus 1 (wait or idle state).
The additional output variable z_i in the controller C^p in accordance to each $s_t \notin S^p$, if two conditions are satisfied:

- there is transition from the state s_t included in S^p to the state next s_j not included in S^p in the controller *C* or s_j is included in S^p but $Y(s_j)$ is not subset of Y^p;
- there is at least one block S^k ($k \neq p$) such that next state s_j is included in S^k and s_t is not included in S^k among the blocks of the cover φ.

We put the additional input variable z_t in the controller C^p in accordance to each $s_t \in S^p$, if there is at least one transition to this state from the state s_j not included in S^p in the controller *C*.
If the controller *C* is in the state s_j, the corresponding states in component controllers of a network are following: all controllers from the set Ω_j are in the state

s_j , all others are in b-states. If controller C transits from the state s_j to the state s_t with the input condition α_h and the output β_h , each controller $C^k \in \Omega_{jt}$ also transits from s_j to s_t .Each controller $C^p \in \Omega_j \setminus \Omega_{jt}$ (the next state s_t is not in C^p) transits from s_j to b_p , and the output signal at the corresponding transition from s_j is equal to $\beta_h \cap Y^q$ in each controller $C^q \in \Omega_j$ (it does not matter whether the state s_t is in C^q or is not in C^q). The output signal z_t is produced only in one of the component controller of the set Ω_j . Each component controller C^m of the set $\Omega_t \setminus \Omega_{jt}$, with the state s_t and without the state s_j , transits to this state s_t from the state b_m with the input z_t (this signal z_t is the output in only one controller of the set Ω_j). The output at this transition from b_m , to s_t is zero (Y_0).

Let us put the network in accordance to the example Table 1 and partition

π={ {y_1, y_3, y_4, y_7}, {y_6, y_8, y_9 }, {y_2, y_5, y_{10}} }

The cover on the set of states of our prototype FSMD (this set is represented by the set of states of controller) corresponding to the given partition π is $\varphi=\{\{s_1, s_4, s_7\}, \{s_4, s_5, s_8\}, \{s_2, s_3, s_6\}\}$ and the cover on the set of g-transition is $\psi =\{\{1, 2, 7, 8, 17\}, \{9, 10, 11, 12, 13, 18, 19\}, \{3, 4, 5, 6, 14, 15, 16\}$

The number of component machines is equal to the number of blocks in the partition π or in the corresponding covers φ and ψ.

In our example, the network consists of three component machines A^1, A^2, and A^3. Their g-transitions sets represented in the transition tables 2, 3, 4.

Present state s_i	Next state sj	Input condition α_h	Output signals β_h	h
s_1	s_1	x_1	y_7	1
	b_1	^x_1	z_3	2_a
s_4	s_1	x_1 & x_2	y_3 y_4	7
	b_1	x_1 & ^x_2	y_1 y_3 y_4 z_3	8_a
	b_1	^x_1	-	g_1
s_7	b_1	1	y_1 z_5	17_a
b_1	s_4	z_4	-	11_b
	b_1	^z_4	-	d_1

Table 2: *The first component transition table*

The set of states of the first component is the states from the first block of cover φ plus b_1, $S^1 = \{s_1, s_4, s_7, b_1\}$.

The set of outputs consists of variables from Y which are in g-transitions from the first block of cover ψ plus additional output variable z_3 because there is transition from s_4 to s_3 in the original transition table, but s_3 is the state of the second component C^2.

The set of inputs consists of variables which are in g-transitions from the first block of cover ψ plus additional input variable z_4 because exists g-transition (11) not included in the first block of cover ψ with next state s_4.

Note, that $\Omega_4 = \{ C^1, C^2 \}$ and $\Omega_3 = \{ C^3 \}$, but only C^1 generates output z_3.

The sketch of the of components network is presented in Figure2.

Present state s_p	Next state s_n	Input condition α_h	Output signals β_h	h
s_5	s_4	x_3 & x_4	y_6 y_9 z_4	11
	s_5	x_3 & ^x_4	y_6 y_9	12
	s_8	^x_3	y_6 y_8	13
s_8	s_8	x_3	-	18
	s_5	^x_3	y_8 y_9	19
s_4	s_5	^x_1 & x_4	y_6 y_9	9
	s_8	^x_1 & ^x_4	y_6 y_8 y_9	10
	b_3	x_1	-	g_2
b_3	s_8	z_8	-	$5_b/16_b$
	s_5	z_5	-	17_b
	b_3	^z_5 & ^z_8	-	d_2

Table 3: *The second component transition table*

Note, that when there is transition to the state s_4 both controllers C^1 and C^2 are activated but only one of them will generate outputs controlling own datapath. The datapath of other machine will be silent (it depends on the value of input variable x_1). This is the case, because the condition presented in affirmation 2 is not satisfied in our example.

Present state s_i	Next state s_j	Input condition α_h	Output signals β_h	h
s_2	s_3	1	y_2 y_{10}	3
s_3	s_6	x_7	y_2 y_{10}	4
	b_2	^x_6 & x_7	z_8	5_a
	s_2	^x_6 & ^x_7	y_2 y_5 y_{10}	6
s_6	s_2	x_5	y_2 y_{10}	14
	s_3	^x_5 & x_7	y_5	15
	b_2	^x_5 & ^x_7	y_2 y_{10} z_8	16_a
b_2	s_3	z_3	-	$2_b/8_b$
	b_2	^z_3	-	d_3

Table 4: *The third component transition table*

Thus, when there is a transition between two components, the caller FSMD will go into its idle state while at the same time, callee FSMD goes from its idle state into its next state. The transitions to and from respective idle states for two component FSMDs happen concurrently, thus no extra clock cycle is needed. But overall execution time can be longer or shorter than the in the case of prototype FSMD as shown in [3].

In Figure 1 shutdown mechanism is not presented. For instance, gated-clock technique [1]could be used. In this case, an decomposed FSMD consists of a number of component FSMD and an equally large number of clock control blocks with nand-gates for gating the local clocks. In particular, handshake protocol between components of decomposition could be used.

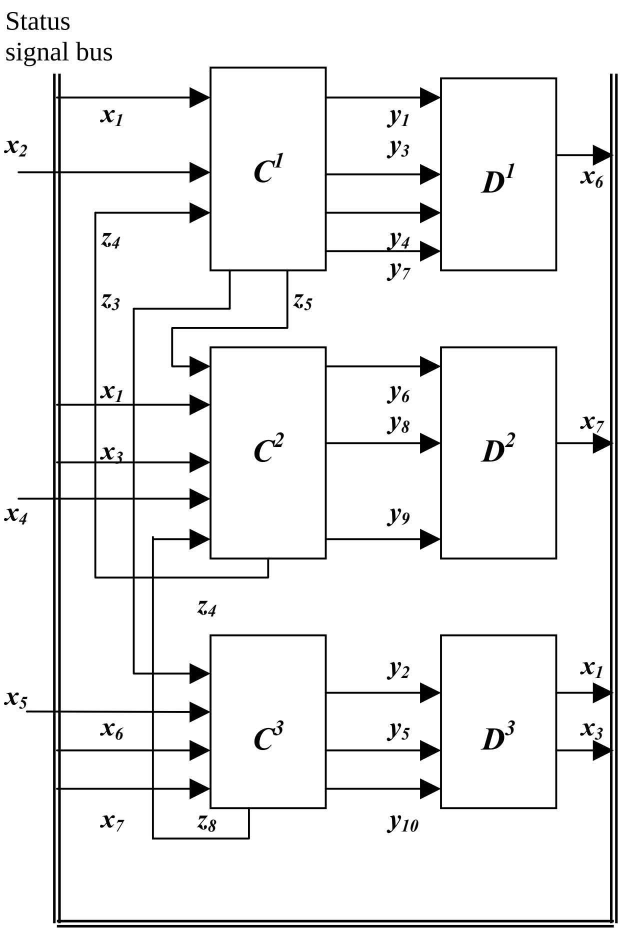

Figure 2: *example FSMD network*

CONCLUSIONS

This paper elaborates functional partition approach for low-power synthesis at RT-level. FSMD functional partitioning technique is applied before logic level of design process. The original FSMD is first partitioned into several smaller FSMDs.

We use technique of dynamic power management to accomplish the task of preventing logic computations in modules when the results will not be used. The reason why FSMD functional partitioning can significantly reduce the switching activities at the registers and the functional modules and only one (subset) of machines is (are) executing a computation at any given time while the other processors will be idle. Here we should emphasize the fact that the machine decomposition is the organic part of synthesis process. In addition to reducing power, FSMD functional partitioning also provides solutions to a variety of synthesis problems.

The solution of problem is reduced to the controller decomposition. In our decomposition procedure, we proceed from assumed partition on the set of outputs of FSM controlling the transfers in datapath. The data path of idle controller is not consuming power because the inputs are not changing. The overhead in this technique is the communication and possible duplication of registers.

Experiments have been carried out on the wide range of random machines and on the set of well-known FSM benchmarks. The results confirmed that it is possible to significantly reduce switching activity of implementation and that significant reduction in power consumption could be achieved without essential performance degradation. Results are much more significant for machines with large number of inputs and data dominated designs with large number of microoperations (in particular, for the most important case for real practice projects when $|R_m| << M$ and when controller has large amount of state-loops).

The method concerns the technique of partition search on the set of FSMD microoperations. Analysis of power dissipation in datapath for partition search is beyond of this work and is the subject of further investigations.

THE AUTHOR

Dr. Alexander Sudnitson is with the Department of Computer Engineering of Tallinn Technical University, Raja 15, 12617 Tallinn, Estonia.

E-mail: alsu@cc.ttu.ee

Acknowledgement – This work has been supported by the Estonian Science Foundation (under Grant G4876).

REFERENCES

[1] E.Macii, M.Pedram, and F.Somenzi, “High-level Power Modeling, Estimation, and Optimization”, IEEE Trans. Computer-Aided Design, vol. 17, no.11, 1998, pp.1061-1079

[2] D.M.Brooks, P.Boose, et al., “Power Aware Microarchitecture: Deign and Modeling Challenges for Next-Generation Microprocessors”, IEEE MICRO, Nov. 2000, pp. 26-43.

[3] E.Hwang, F.Vahid, and Y.-C.Hsu, “FSMD Functional Partitioning for Low Power”, Proceedings of the D A T E International Conference, March 1999, pp.22-28

[4] S.-H.Chow, Y.-C.Ho, T.Hwang, and C.L.Liu, “Low Power Realization of Finite State Machines – A Decomposition Approach”, ACM Trans. on Deign Automation, vol.1, no.3, July 1996, pp. 315-340

[5] S.Baranov “Logic Synthesis for Control Automata”. Kluwer Academic Publishers, 1994

[6] M.Alidina, J.Monteiro, et al., "Pre-computation-Based Sequential Logic Optimization for Lower Power”. Proceedings of the International Conference on Computer Design, October 1994, pp.74-81

[7] V.Tiwari, S.Malik and P.Ashar, "Guarded Evaluation: Pushing power management to logic synthesis/design". IEEE Trans. Computer-Aided Design, vol. 17, no.10, 1998, pp. 1051-1060

[8] D.D.Gajski, F.Vahid, S.Narayan, and J.Gong, “Specification and Design of Embedded Systems”, Englewood Cliffs, N.J.: Prentice-Hall, 1994