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: Recent 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.

\[ P_W = \frac{1}{2} C V_{dd}^2 a f + I_{leakage} V_{dd} + I_{sc} V_{dd} \]

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 ≤ a ≤ 1) that determines the device switching frequency. \( I_{leakage} \) is the leakage 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) \( \frac{1}{2} C V_{dd}^2 a f \) remains the dominant component. So as a first-order approximation for the whole chip we may formulate the power dissipation as

\[ P_W = \frac{1}{2} \sum C_i V_i^2 a_i f_i \]

\( C_i, V_i, a_i, \) 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 capacitances. 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 multiplexers. 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].

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
- \(\delta\) is the next state function, mapping \(S \times (I \times SS) \rightarrow S\)
- \(\lambda\) 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) \(X = \{x_1, \ldots, x_f\}\) and the set of binary outputs (channels) \(Y = \{y_1, \ldots, y_r\}\) (Figure 1).

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.

![Figure 1: High-level block diagram](image-url)
The variables \( x_1, \ldots, x_4 \) 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 \( Y = \{y_1, \ldots, y_7\} \) 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 \( S, \alpha, \delta, \lambda \) where \( S = \{s_1, \ldots, s_m\} \) is a set of states.

\[ X = \{x_1, \ldots, x_4\} \] is a set of binary input variables (channels).

\[ Y = \{y_1, \ldots, y_7\} \] is a set of binary output variables (channels).

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

\[ \lambda(\lambda) \rightarrow \Sigma \] is a set of output variables.

We use the formal notion of generalized transition in microinstruction implementation. We can consider \( \alpha \) and \( \delta \) as generalized transition table (in our example every g-transition corresponds to row in table 1).

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Input condition</th>
<th>Output signals</th>
<th>( h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>( s_1 )</td>
<td>( x_1 )</td>
<td>( y_7 )</td>
<td>1</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>( s_4 )</td>
<td>( x_1, x_2 )</td>
<td>( y_6, y_9 )</td>
<td>9</td>
</tr>
<tr>
<td>( s_5 )</td>
<td>( s_5 )</td>
<td>( x_3 )</td>
<td>( y_6 )</td>
<td>11</td>
</tr>
<tr>
<td>( s_6 )</td>
<td>( s_6 )</td>
<td>( x_3, x_4 )</td>
<td>( y_6, y_9 )</td>
<td>12</td>
</tr>
<tr>
<td>( s_7 )</td>
<td>( s_7 )</td>
<td>( x_4 )</td>
<td>( y_7 )</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 1: Transition table of illustrative example

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

\( g \)-transition is quartuple \( < s_i, s_j, \alpha_0, \beta_0, h > \) where \( s_i \) is the present state, \( s_j \) is the next state, \( \alpha_0 \) is the input condition (Boolean function), \( \beta_0 \) 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 \( \{ s_r / r \in R_w \subseteq \{1, \ldots, M\} \} \) is the set of the next ones where transitions from \( s_p \) are possible then there is the set of functions \( \{\alpha_m / r \in R_w\} \).

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 \( \{0, 1\}^k \).

**Controller Decomposition**

**Decomposition model.** A collection \( \varphi \) 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 union is \( S \). We refer to these subsets as blocks of the cover (partition).

Let \( \pi = \{Y_1, \ldots, Y^k\} \) 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), \( X(g_i) \) and \( Y(g_i) \) be the sets of essential input and output variables (microoperations) in the \( g \)-transition \( g_i \) (\( h = 1, \ldots, H \)), \( X(s_i) \) 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 \( \xi \) such that \( g_i \sim g_j \) if there is at least one common variable in the sets \( Y(g_i) \) and \( Y(g_j) \):

\[ g_i \sim g_j \Leftrightarrow Y(g_i) \cap Y(g_j) \neq \emptyset \]

This relation is symmetric and reflexive and induces the cover \( \mu \) on the set \( Y \).

For every block \( Y^k \) from partition \( \pi \) we put in accordance component FSM \( A^\pi \) in the network \( N \).

Let us put the cover \( \varphi \) on the set of states \( S \) and the cover \( \psi \) on the set of \( g \)-transitions \( G \) of the transition table in accordance to the pair \( \prec, \succ \):

\[ \varphi = \{B_1, \ldots, B^h\}; \quad B^h \subseteq S, \quad s \in B^h \Leftrightarrow Y(s) \cap Y^h \neq \emptyset; \quad \psi = \{G^1, \ldots, G^h\}; \quad G^h \subseteq G, \quad g \in G^h \Leftrightarrow Y(m) \cap Y^h \neq \emptyset. \]

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

In exactly the same way, it follows that the \( g \)-transition \( g_i \) will be in the block \( G^\psi \) of the cover \( \psi \), if at least one output variable from the block \( Y^\psi \) of the partition \( \pi \) is written in the row of transition table corresponding transition \( g_i \). Just as for \( \varphi \), the same \( m \) will be in several blocks of \( \psi \), for example, in \( G^\psi \) and \( G^\psi \), if
Y(m) \cap Y^o \neq \emptyset \text{ and } Y(h) \cap Y^o \neq \emptyset

i.e. the output variables from Y^o and Y^o are written in the row m.

In our example (Table 1),
\[\mu = \{ \{ y_1, y_2, y_3, y_4 \}, \{ y_5, y_6, y_7 \}, \{ y_8, y_9, y_{10} \} \}
\]
\[\nu = \{ 1, 2, 7, 8, 17 \}, \{ 9, 10, 11, 12, 13, 18, 19 \},\]
\{ 3, 4, 5, 6, 14, 15, 16 \}\]

**Affirmation 1.** Given FSMD A and the partition \( \pi = \{ Y^o \}, i \in \{ 1, \ldots, n \} \) 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
\[\mu = \{ \{ y_1, y_2, y_3, y_4 \}, \{ y_5, y_6, y_7 \}, \{ y_8, y_9, y_{10} \} \}, \]
that satisfies the condition \( \mu \geq \pi \).

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 \( \pi \) on Y is such that for all states \( s_j \) of FSMD A exists block \( Y^o \) such that \( Y(s_j) \subseteq Y^o \) such 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
\[A^p = (A, \pi, \varphi, \psi)\]

in accordance to triplet \( (A, \pi, \varphi, \psi) \). The number of component machines is equal to the number of blocks in the partition \( \pi \) (or the cover \( \varphi \) and \( \psi \)).

Further the steps of decomposition procedure are presented in formal way.

1. \( S^o = B^o \cup \{ b_p \} \) is the set of states in the component controller \( C^o \), where \( B^o \) is the p-th block of the cover \( \varphi \) and \( b_p \) is additional state in \( C^o \).
2. \( X^p = X(G^o) \cup Z^p \) is the set of input variables in the component controller \( C^o \).
   Here \( X(G^o) = \cup_{g \in \Delta} X(g) \)
   \( X(g) \) is the set of essential input variables in the \( g \)-transition \( g \) of the controller \( C \) of the prototype FSMD:
   \( G^o \) is the p-th block of the cover \( \psi \).
   \( Z^p = \{ z_i | \delta(s_j, \alpha) = s_i, s_j \in S^o, s_i \in S^o \} \).
3. \( Y^o = Y^o \cup Z^o \)
   \( Z^o = \{ z_i | \delta(s_j, \alpha) = s_i, s_j \in S^o, s_i \in S^o \} \).
4. Assume that there is a g-transition \(< s_j, s_k, \alpha, \beta > \in \text{ of prototype FSMD} \) (the transition from \( s_j \) to \( s_k \) with the input condition \( \alpha \) and the output \( \beta \)).

Define the corresponding transitions in component controller.

\( \Omega_j \) be the set of component controllers with the state \( s_j \).
\( \Omega i \) be the set of component controllers with the state \( s_i \).

\( \Omega_j \cap \Omega_i \) be the set of component controllers with the states \( s_j \) and \( s_i \).

If \( C^o \in \Omega_j \cap \Omega_i \), then in \( C^o \)
\[\delta(s_j, \alpha) = s_i.\]

If \( A^p \in \Omega_j \cap \Omega_i \), \( (s_j \text{ is the state of } C^o \text{ and } s_i \text{ is not the state of } C^o) \), then in \( C^o \)
\[\delta(s_j, \alpha) = b_p.\]

The output controlling datapath of \( p \)-th component FSMD for controller \( C^o \in \Omega_j \) \( (s_j \text{ is the state of } C^o \text{ and it is also possible that the next state } s_i \text{ is the state of } C^o) \) is equal to \( b_i \cap Y^o \). In addition to controlling outputs one and only one controller from \( \Omega_j \) (say \( C^o \)), must generate the output signal which forces each controller from \( C^o \in \Omega_j \cap \Omega_i \) (if this set is not empty) to transit from the additional (idle) state \( b_i \) to \( s_i \) and in \( C^o \)
\[\delta(b_i, z_i) = \beta.\]

If \( C^o \in \Omega_j \cap \Omega_i \), \( (s_j \text{ is the state of } C^o \text{ and } s_i \text{ is not the state of } C^o) \), then in \( C^o \)
\[\delta(b_i, z_i) = s_i.\]

5. The initial state of the component controller is equal to \( s_0 \), if \( C^o \in \Omega_i \) and \( b_i \) 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 \( \varphi \) plus 1 (wait or idle state).

The additional output variable \( z_i \) in the controller \( C^o \) in accordance to each \( s_i \ \notin S^o \), if two conditions are satisfied:

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

We put the additional input variable \( z_i \) in the controller \( C^o \) in accordance to each \( s_i \ \in S^o \), if there is at least one transition to this state from the state \( s_j \) not included in \( S^o \) 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 \( \Omega_j \) are in the state

-
The set of states of the first component is the states from $s_1$ to the state $s_5$ with the input condition $\alpha_0$ and the output $\beta_0$ of controller $C^1 \in \Omega_1$ also transits from $s_1$ to $s_5$. Each controller $C^i \in \Omega_i \setminus \Omega_j$ (the next state $s_6$ is not in $C^i$) transits from $s_1$ to $b_1$, and the output signal at the corresponding transition from $s_1$ is equal to $\beta_0 \cap Y^i$ in each controller $C^i \in \Omega_i$ (it does not matter whether the state $s_i$ in $C^i$ or is not in $C^i$). The output signal $z_i$ is produced only in one of the component controller of the set $\Omega_i$. Each component controller $C^i$ of the set $\Omega_i \setminus \Omega_j$, with the state $s_i$ and without the state $s_j$, transits to this state $s_i$ from the state $b_m$ with the input $z_i$ (this signal $z_i$ is the output in only one controller of the set $\Omega_i$). The output at this transition from $b_m$ to $s_i$ is zero ($Y_0$).

Let us put the network in accordance to the example Table 1 and partition $\pi = \{ \{y_1, y_3, y_4, y_7\}, \{y_6, y_8, y_9\}, \{y_5, y_{10}\}\}$. 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 $\pi$ is $\varphi = \{s_1, s_4, s_5\}, \{s_2, s_3, s_6\}, \{s_8, s_7, s_9\}$ and the cover on the set of $\psi$-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 $\pi$ or in the corresponding covers $\varphi$ and $\psi$. 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.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Input condition $\alpha_0$</th>
<th>Output signals $\beta_0$</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>$s_1$</td>
<td>$y_7$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$b_1$</td>
<td>$x_1$</td>
<td>$z_1$</td>
<td>2a</td>
<td></td>
</tr>
<tr>
<td>$s_4$</td>
<td>$s_4$</td>
<td>$x_1$ &amp; $x_2$</td>
<td>$y_1$ &amp; $y_4$</td>
<td>7</td>
</tr>
<tr>
<td>$b_1$</td>
<td>$x_1$</td>
<td>$z_1$</td>
<td>$y_1$ &amp; $y_4$ &amp; $y_2$</td>
<td>8u</td>
</tr>
<tr>
<td>$s_7$</td>
<td>$b_1$</td>
<td>$y_1$</td>
<td>$z_1$</td>
<td>17g</td>
</tr>
<tr>
<td>$b_1$</td>
<td>$z_1$</td>
<td></td>
<td>$y_1$</td>
<td>11h</td>
</tr>
</tbody>
</table>

Table 2: The first component transition table

The sketch of the of components network is presented in Figure 2.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Input condition $\alpha_0$</th>
<th>Output signals $\beta_0$</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_2$</td>
<td>$s_2$</td>
<td>$y_2$</td>
<td>$y_10$</td>
<td>3</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_3$</td>
<td>$z_5$</td>
<td>$y_10$</td>
<td>4</td>
</tr>
<tr>
<td>$s_5$</td>
<td>$s_5$</td>
<td>$z_7$</td>
<td>$y_10$</td>
<td>6</td>
</tr>
<tr>
<td>$s_6$</td>
<td>$s_6$</td>
<td>$z_8$</td>
<td>$y_10$</td>
<td>14</td>
</tr>
<tr>
<td>$b_2$</td>
<td>$s_2$</td>
<td>$z_1$</td>
<td>$y_10$</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 3: The second 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.
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 \( R_o/<<M \) 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 Sudnitsin 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