Multi-Criterial State Assignment for Low Power FSM Design

Manfred Koegst[1)], Günter Franke[2)], Steffen Rülke[1)], Klaus Feske[1)]

1) Fraunhofer-Institut Integrierte Schaltungen, EAS Dresden
Zeunerstr. 38, D-01069 Dresden, e-mail: koegst@eas.iis.fhg.de
2) Hochschule für Technik und Wirtschaft Dresden (FH), FB Elektrotechnik,
Friedrich-List-Platz 1, D-01069 Dresden, e-mail: frankeg@et.htw-dresden.de

Abstract

Reducing power consumption has become one of the biggest challenges in VLSI design. In control-flow intensive designs like networking and controller applications the largest fraction of power consumption in CMOS is caused by signal switches. The presented approach addresses the reduction of switching activity. This is basically done by state assignment concerning a given user-specified input pattern sequence. Besides reducing register switchings, power can be saved by e.g. a partial deactivation of the circuit or a parts of it. There is a strong dependency between the used deactivation method, the selection of circuit parts for deactivation, the additional deactivation logic and the state assignment. The aim of this paper is to consider these dependencies in a multi-criterial optimization approach. This is done by an unified specification of the additional deactivation costs as constraints for state encoding. The result of our investigation is a multicriterial assignment procedure which enables us to specify an optimal code concerning different criteria.

1 Introduction

The necessity for low power has caused a considerable paradigm shift in the field of VLSI design. The power dissipation is today as important as circuit speed and area. This is driven by a variety of requirements such as prolonging battery life in portable devices for multimedia and communication, and reducing chip packaging costs. Designers are challenged to come up with innovative methods to reduce power, while trying to meet all the other constraints by multicriterial optimization.

Thus, design optimization in combination with low power techniques are of specific importance. As a rule, the higher the abstraction level to achieve power optimization the greater its efficiency. Good overviews of low power techniques can be found e.g. in [AEH-96, Pig-96, RaP-96].

In control-flow intensive applications (e.g. networking and controllers especially for reactive systems) the largest fraction of power consumption in CMOS technology is caused by signal switches.

Besides reducing register switching rate, power can be saved by e.g. a partial deactivation of the circuit. In addition to that the glitching power is reduced. There is a strong correlation between the used deactivation method, the selection of circuit parts for deactivation, the additional deactivation logic and the state assignment. But the deactivation logic takes additional power.

For saving power different state assignment procedures are used in published design approaches:

- state assignment and re-encoding [Hac-94, BeM-95],
- sharing primary output with next state functions [KFF-93],
- reducing glitching activity [For-95, RDJ-97],
- deactivation of selected self-loops by clock gating [BSM-94, BDM-97],
- register gating by internal signals [CPL-96, SCT-97],
- disabling the next state and output logic in self-loops [BeM-96, KFR- 97],
- state controlled multiplexing of selected input signals [Gra-78, Bar-94, KFF- 98].

The aim of this paper is to combine encoding procedures of different low power design approaches by an unified specification of the encoding constraints .

For that we propose a novel approach for state assignment which integrates the above discussed features (deactivation method, partial deactivation, optimization of the deactivation logic) in a multi-criterial optimization method. This is done via an unified specification of the additional deactivation conditions and costs as constraints for state encoding. The presented approach addresses the reduction of switch-

1089-6503/98 $10.00

ing activity of synchronous finite state machines (FSM). The state assignment method takes a given user-specified input pattern sequence into account which results from FSM profiling applying a given user-specified input pattern sequence [MoD-95, KFR- 97].

The approach starts at the register transfer level of the design specification and integrates the encoding conditions and constraints of the different techniques to decrease switching activities. The state assignment algorithms is based on simulated annealing [RoP-93] concerning the real frequency of the state transitions. Conventional encoding methods are not able to consider such heterogeneous encoding constraints.

The paper is structured as follows. Section 2 introduces the model of the FSM and defines the task for state assignment. In the next section three approaches for power saving are outlined. Each approach contributes encoding constraints to the multicriterial state assignment method. This integrating common state assignment method is developed in section 4.

2 Preliminaries and definitions

2.1 Finite state machine

The control task description can be specified as a finite state machine (FSM) which is defined by a 5-tuple $M = (X,Y,S,f,g)$, where X, Y, and S are the finite sets of inputs, outputs, and states, respectively. The unique mapping $f: X \times S \rightarrow S$ and $g: X \times S \rightarrow Y$ defines the so-called state transition function and output function. For the domain D_f and D_g of f and g relation $D_g \subseteq D_f$ is supposed. For each state s of S subset

$$D_f(s) = \{(x',s') | (x',s') \in D_f \wedge f(x',s') = s\}$$

of D_f characterize all transitions ending in state s. State s of S is called Moore state if for all pairs (x',s') and (x'',s'') from $D_f(s)$ the corresponding outputs are identical: $g(x',s') = g(x'',s'')$.

State transition graph (STG) and state transition table (STT) are two equivalent representations of a FSM. A STG is defined by a vertex set S and a related set of edges with elements (s,s') weighted by a condition x for the transition from s to s' = f(x,s) and the corresponding output y = g(x,s). A STT is given by a set of all defined 4-tuples (x, s, s', y). A transition (x,s) is called self-loop if its present state and next state are identical, i.e. f(x,s) = s.

V(s) characterizes the subset of input variables which are essential to select a transition of state s uniquely.
We denote by *card*(W) the number of elements of a set W and by $\lceil b \rceil$ the smallest natural number greater than or equal b.
With respect to all states s of S $n_s = \max\{card(V(s)) \mid s \in S\}$ denotes the greatest number of necessary input variables.

Moreover, with regard to our generalized state assignment method each subset G_a of the state set S is named block, consequently each state is a block too. A set G of blocks is complete if at first set S is an element of G and at second G contains all states of S. Furthermore, G is comparable if for all pairs (G_a,G_b) of elements of G with $G_a \cap G_b \neq \emptyset$ holds $G_a \subseteq G_b$ or $G_b \subseteq G_a$.

2.2 Register switching rate

Our approach is based on a given user-specified input pattern sequence [MoD-95] which can be received by recording the inputs of a FSM in the true environment or in the simulation process. In a profiling process the input pattern sequence supplies the FSM M. The recorded results of profiling are exploited to control the power optimization approach.

In the result of profiling we obtain for each row of the STT of M characterized by pair (x,s) of input x and present state s the number N(x,s) of activations caused by the pattern of the given sequence. Analysing the frequency of these numbers N(x,s), we derive further information for minimizing the register switching activity, especially the probability p(s,s') for the transition between the states s and s'. For that we define the register switching rate of a state code c by

$$\delta(c) = \sum_{s \neq s'} p(s,s') \cdot H(c(s),c(s'))$$

where H is the Hamming distance of the codes of s and s' [KFF-96].

The encoding task is to find an assignment c with a minimal switching rate $\delta(c)$ considering the given encoding constraints which are characterized by subsets of states (so-called blocks). Due to the complex nature of weight δ, we modified the simulated annealing procedure [RoP-93] for solving this problem.

3 Approaches for power reduction

Three approaches for saving switching activities are outlined in this section. Each contributes different types of encoding constraint to the integrating state assignment in section 4. The sharing method in section 3.1 hands over a partial encoding. The clock gating approach delivers block encoding conditions. The precomputation method gives hierarchical encoding constraints (two levels). In section 3.1 and 3.2 the FSM M of Table 1a is used. The explanation in

section 3.3 and 4 are based on the FSM s410 of MCNC benchmark set.

3.1 Sharing of outputs with next state functions

Especially, with the goal to reduce glitching activity [RDJ-97], to create glitch-free outputs [For-95], and to minimize the power dissipation it is useful to share outputs with next state functions [KFF-93], s. Figure 1. Only the so called Moore outputs are sharable. A Moore output depends on states but not on inputs [For-95]. The remaining (not sharable) variables may depend on the inputs.

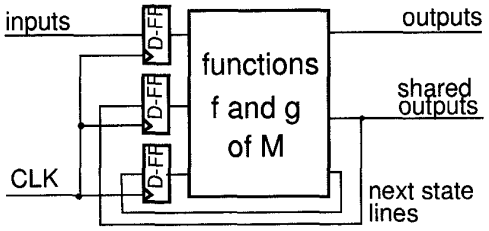

Fig. 1: Principle of sharing

In the example FSM M (Table 1a) both output functions (y_1,y_2) are Moore outputs. Consequently, there is such a state code that the outputs may be expressed by next state (ns) functions. In Table 2 for each state all corresponding outputs are depicted (second column), and finally a partial state code is derived (third column). The real code has to consist of at least three code variables.

x_1x_2	ps	ns	y_1y_2	h
00	**s_0**	**s_0**	**0-**	**1**
01	s_0	s_3	1-	0
10	s_0	s_1	--	0
11	**s_0**	**s_0**	**00**	**1**
00	**s_1**	**s_1**	**0-**	**1**
01	s_1	s_0	-0	0
10	s_1	s_2	10	0
11	**s_1**	**s_1**	**0-**	**1**
- 0	s_2	s_2	10	0
- 1	s_2	s_4	-1	0
0 -	s_3	s_1	0-	0
1 -	s_3	s_4	-1	0
0 -	s_4	s_2	10	0
10	s_4	s_4	--	0
11	s_4	s_0	00	0

Table 1: a) Example FSM M; b) Gate functions h(x_1,x_2,ps) for section 3.2

Concerning the partial code of Table 2 the states of M are encoded in such a way that the outputs are very simple:

$y_1 = z_1$ and $y_2 = z_2$.

ns	outputs y_1,y_2	partial code z_1,z_2
s_0	0-, 00, -0	00
s_1	--, 0-	0-
s_2	10	10
s_3	1-	1-
s_4	-1	-1

Table 2: Derivation of partial code

[KFF-93] describes the procedure in detail. In our context only the first part of the method - the partial code generation - is used. The method in section 4 uses this partial code.

3.2 Gated clock logic

Further encoding constraints follow from clock gating [BeM-96, BDM-97] outlined in this section. By means of clock gating we avoid switching activities caused by input transitions in such situations if states and outputs do not change. For that all primary inputs are buffered (Figure 2). The clock GCLK of the related input registers (D-FF) and the state registers is gated by a special logic h (Figure 2). This additional logic h is sensitive to primary inputs, next state signals and clock CLK. The logic h is ungated and dissipates extra power.

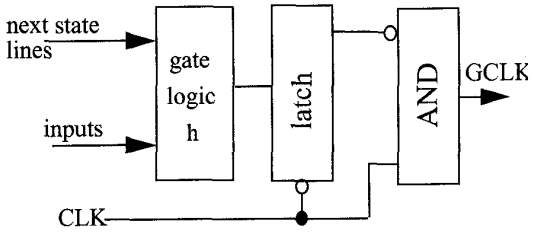

Fig. 2: Gated clock logic

There are two principles for GCLK generation: First the so-called combinational gate logic [BeM-96] where only self-loops in Moore states can be disabled, and second its generalization [KFR-97] so-called sequential gate logic. In both cases we have to define the logic h in an optimal manner. The extra power of h has to be taken into consideration if we draw up the balance sheet of the overall power consumption.

As an example we discuss the first case for the FSM M Table 1a. We have to define the gate function

h: $D_f \rightarrow \{0,1\}$.

"1" in the column of the gate function h in Table 1b means the releted self-loop in states s_0 and s_1 will be gated.

h	$z_1z_2z_3$
s_0	100
s_1	110
s_2	011
s_3	001
s_4	000

Table 3: State code for gate function h

The additional gate circuitry can be reduced when the states of the block $G = \{s_0, s_1\}$ of the corresponding gate function h is encoded as a block c(G) = 1-0 (Table 3).

$$h = \bar{x}_1\bar{x}_2c(s_0) \vee x_1x_2c(s_0) \vee \bar{x}_1\bar{x}_2c(s_1) \vee x_1x_2c(s_1)$$

$$h = \bar{x}_1\bar{x}_2c(G) \vee x_1x_2c(G) \qquad \text{with} \qquad G = \{s_0, s_1\}$$

$$h = (\bar{x}_1\bar{x}_2 \vee x_1x_2)z_1\bar{z}_3 \qquad \text{with} \qquad c(G) = 1-0$$

The integrating approach in section 4 uses the block code c(G) = 1-0 for the specification of gate function h. In our example the necessary number of products in the sum of product form is halfed.

3.3 Multiplexing of input signals

A third contribution to the encoding constraints arises from precomputation [Gra-78, AMD-94, Bar-94]. With respect to separate states the accompanying transitions and outputs depend in control applications on a small number of the primary input variables.

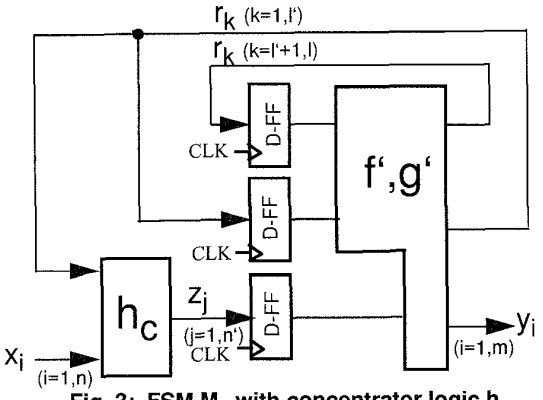

Fig. 3: FSM M_p with concentrator logic h_c

Figure 3 shows a FSM with state controled concentrator logic h_c. The complexity (and switching activities) of the logic h_c can be reduced if h_c depends only on a subset of state variables. For quantifying the reduction of power dissipation by such a logic h_c we denote the original FSM by $M = M(n,l,n_s)$ and the restructured FSM by $M_p = M_p(n,l,n',l')$, where n, l, n' and l' are the numbers of inputs x_i, state variables r_k, intermediate variables z_j and state variables necessary for h_c, respectively. The condition for restructuring is based on the principle of balance registers [LRS-83].

The optimization task consists in finding a suitable partition G as a set of blocks of $M_p = M_p(n,l,n',l')$ with the conditions:

- the number n' of intermediate signals is minimal,
- the number l' of state variables necessary for controlling h_c is minimal, and
- block encoding allows the Boolean optimization of logic h_c.

The principle is demonstrated for the FSM s410 of the MCNC FSM benchmark set with n = 8 inputs, 18 states and at least l = 5 state variables.

Table 4 shows the set $V(s_i)$ of significant variables of the states s_i. The maximum number of significant variables is 4.

s_i	$V(s_i)$
s_0	{1,18,19}
s_1	{1,2,19}
s_2	{1,2,18,19}
s_3	{1,2,17,19}
s_4	{1,2,18,19}
s_5	{1,2,16,19}
s_6	{1,2,18,19}
s_7	{1,2,17,19}
s_8	{1,2,18,19}
s_9	{1,2,15,19}
s_{10}	{1,2,18,19}
s_{11}	{1,2,17,19}
s_{12}	{1,2,18,19}
s_{13}	{1,2,16,19}
s_{14}	{1,2,18,19}
s_{15}	{1,2,17,19}
s_{16}	{1,2,18,19}
s_{17}	{1,2,14,19}

Table 4: Set $V(s_i)$ of significant variables of the states s_i

As a result from these sets we construct the partition $G = \{G_1,...,G_5\}$ depicted in Table 5, where $n' = 4$.

From the number of blocks $p = 5$ follows the minimum number of state variables l', from which the concentrator logic depends: $l' = \lceil \log_2(p) \rceil = 3$.

block G_i	$V(s_i)$
$G_1=\{s_{17}\}$	{1,2,14,19}
$G_2=\{s_9\}$	{1,2,15,19}
$G_3=\{s_5,s_{13}\}$	{1,2,16,19}
$G_4=\{s_3,s_7,s_{11},s_{15}\}$	{1,2,17,19}
$G_5=\{s_0,s_1,s_2,s_4,s_6,s_8,s_{10},s_{12},s_{14},s_{16}\}$	{1,2,18,19}

Table 5: Partitioning of set S in blocks

If we assign the primary input x_1 to the intermediate variable z_1, x_2 to z_2, x_{19} to z_4, and finally x_i for $i \in \{14,15,16,17,18\}$ to the intermediate variable z_3 we get the following specification of the concentrator logic h_c (Figure 4):

$z_1 = x_1$,

$z_2 = x_2$,

$z_3 = h_c(x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, c(s))$,

$z_4 = x_{19}$.

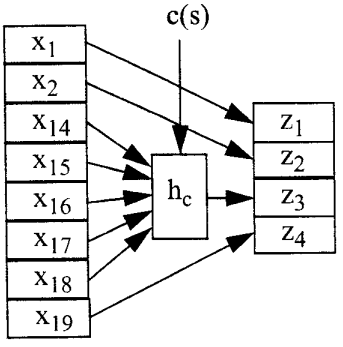

Fig. 4: Assignment of inputs to intermediate signals

The concentrator logic h_c of the intermediate signal z_3 can be described in a sum of product form of 18 products.

$$z_3 = x_{14}c(s_{17}) \vee x_{15}c(s_9) \vee x_{16}[c(s_5) \vee c(s_{13})] \vee x_{17}[c(s_3) \vee c(s_7) \vee c(s_{11}) \vee c(s_{15})] \vee x_{18}[c(s_0) \vee c(s_1) \vee c(s_2) \vee c(s_4) \vee c(s_6) \vee c(s_8) \vee c(s_{10}) \vee c(s_{12}) \vee c(s_{14}) \vee c(s_{16})]$$

To get a simple logic we have to encode the blocks of these expressions with the minimum number of variables.

$$z_3 = x_{14}c(s_{17}) \vee x_{15}c(s_9) \vee x_{16}c(G_3) \vee x_{17}c(G_4) \vee x_{18}c(G_5)$$

Using the block codes of Table 6 we get the minimized concentrator logic h_c of the intermediate signal z_3 which cost is reduced from 18 to five products.

$$z_3 = x_{14}\bar{r}_1\bar{r}_2\bar{r}_3 \vee x_{15}\bar{r}_1\bar{r}_2 r_3 \vee x_{16}\bar{r}_1 r_2\bar{r}_3 \vee x_{17}\bar{r}_1 r_2 r_3 \vee x_{18}r_1$$

This section delivers encoding conditions for the multi-criterial state assignment in section 4.

4 Multi-criterial state assignment

The aim of this section is to combine the different state assignment conditions of the former sections given as a partial code and a set of blocks. This results in a hierarchical state assignment technique. The technique takes into account the state transition frequency resulting from profiling in section 2.2 and encodes neighboring states of a high frequency with a small Hamming distance.

Our encoding procedure takes encoding constraints in consideration which result form different and independent subtasks (section 3). Therefore the integrating view is to interprete all conditions by one set G of blocks. G is supposed to be comparable and complete (s. section 2.1).

The assignment task for the set G consists in encoding all states of S in such a way that the codes of states in each block G_a contain a common part, the so-called block code, which is orthogonal to the codes of all other states of S. For handling the hierarchy we define levels of hierarchy in G, and specify for each element G_a of G a measure $v(G_a)$ of encoding space.

Inductive definition of the hierarchy levels in G:

1. highest level $G^{(m)} = \{S\}$
   (i.e. $G^{(m)}$ contains one block consisting of all states),
2. i-th level $G^{(i)} = \max(G \setminus (\bigcup_{j=i+1}^{m} G^{(i)} \setminus S^*))$,
3. first level $G^{(1)} = S^*$,

where $S^* = \{\{s_0\},...,\{s_n\}\}$ and max(G) is the set of all maximal elements of G concerning the binary relation $\subseteq$.

For our example s410 the hierarchy levels are illustrated

in Figure 5. There are three levels whereby the third level consists of the state set S only which is the maximum element of G.

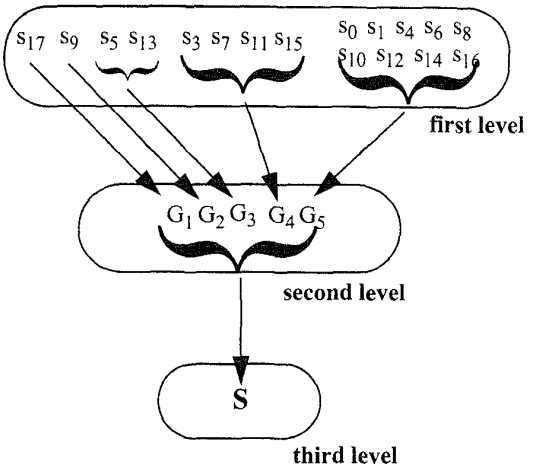

Fig. 5: Block hierarchy for state encoding

Inductive definition of the encoding space $v(G_a)$ for the elements G_a of G:

1. for all elements G_a of the first level, $G_a \in G^{(1)}$, we define $v(G_a) = 1$,
2. for all elements G_a of the i-th level, $G_a \in G^{(i)}$, it is defined $v(G_a) = \sum_{G_b \in G^{(i-1)} \wedge G_b \subseteq G_a} 2^{\lceil \log_2 v(G_b) \rceil}$

Then the minimal width for encoding the blocks of set G is specified by $d(G) = \lceil \log_2 v(S) \rceil$.

Our technique for encoding a set G of blocks consists of two parts. Firstly all blocks are encoded by completing a given partial code concerning special fixed positions, so-called masks (section 4.1). Secondly we specify at each level the corresponding masks (section 4.2).

On the basis of the block hierarchy, we encode the blocks iteratively from the highest to the first level. For that at each level two steps are to perform, first the creation of a code concerning a mask and second the updating of the mask.

As illustrated in Figure 6 mask $m^{(i)}(G_a)$ is a function of the partial code $c^{(i+1)}(G_a)$ of the next higher level and the encoding width $d(G_a)$ of block G_a. Then the code $c^{(i)}(G_a)$ of block G_a is an extension of the partial code $c^{(i+1)}(G_a)$ concerning the above specified mask $m^{(i)}(G_a)$. Our procedure starts with the code "-...-" of block S at the highest level. In our example the code of block S is fixed by "-----". After finishing the encoding at level i the encoding at the next lower level i-1 can be performed. The procedure stops if all single states at the first level are encoded.

level	block	space	mask	code
i+1				$c^{(i+1)}(G_a)$
i	G_a	$d(G_a)$	$m^{(i)}(G_a)$	$c^{(i)}(G_a)$

Fig. 6: Steps of hierarchical block encoding

4.1 Block encoding concerning a mask

In this section we complete given partial codes of blocks by fixed masks. The preconditions for this generalized encoding for a set G of blocks are:

- a code width l equal or greater d(G),
- a mapping a: $G \rightarrow \{0,1,-\}^l$ for describing the partial code of a block,
- a mapping m: $G \rightarrow \{x,-\}^l$ for describing the mask, and
- a mapping p: $G \times G \rightarrow [0, 1]$ which characterizes the probability $p(G_a,G_b)$ for the transition between the two blocks G_a and G_b. This probability can be derived from the transition probability $p(s_i,s_j)$ in the following way $p(G_a,G_b) = \sum_{s \in G_a \wedge s' \in G_b} p(s,s')$.

Then the generalized encoding task is to assign for all blocks G_a of G a code $c(G_a)$ with the property, that

1. code $c(G_a)$ and the given partial code $a(G_a)$ are identical in all positions in which there is sign "x" in the corresponding mask $m(G_a)$, and additionally,
2. weight $w(c) = \sum_{G_a \neq G_e} p(G_a,G_b) \cdot H(c(G_a),c(G_b))$ is mimal where H is the Hamming distance of the corresponding block pair.

For solving this task we use an adapted simulated annealing procedure [RoP-93]. The resulting code is a completion of its partial code by this assignment.

This partial assignment is done with respect to the reduction of the power dissipation. Therefore we create a block code c with a minimal weight resp. register switching rate w(c). For an unique encoding of a block the sign "-" in the corresponding mask can be replaced by sign "0" or "1" and moreover all signs "x" are to replace by "-".

For better understanding we illustrate the assign steps for the blocks G_1,..., G_5 of the second level in Table 6. For encoding e.g. of block G_4 is the related mask "---xx", and the corresponding partial code at the next higher level level is "-----". By means of our adapted simulated annealing procedure the partial code will be qualified in positions in which the mask contains sign "-", and in all other positions the partial codes has to be unchanged. In our example the code "011--" of block G_4 is the result of replacing sign "-" in the first position by sign "0" and in the next two positions by sign "1". The resulting block codes are depicted in the last column of Table 6.

For encoding e.g. the state s_3 at the next lower level we need the block code of G_4 which is interpreted as partial code and additionally the mask "xxx--" of s_3. The resulting code "01100" for state s_3 is depicted in Table 7.

block G_i	partial code	width $d(G_i)$	mask $a(G_i)$	code $c(G_i)$
G_1		0	-----	000--
G_2		0	-----	001--
G_3	-----	1	----x	010--
G_4		2	---xx	011--
G_5		4	-xxxx	1----

Table 6: Block encoding at second level

In the second step the blocks are to assign concerning their masks. For an unique encoding of a block the sign "-" in the corresponding mask can be replaced by sign "0" or "1" and moreover all signs "x" are to replace by "-". But with respect to the reduction of the power dissipation it is to create a block code c with a minimal weight resp. register switching rate w(c).

For our example such a block code is depicted in the last columne of Table 6.

4.2 Mask updating

The necessary generation of masks already in section 4.1 is outlined in this section.

The masks of a level are defined iteratively by the corresponding block codes of the next higher level. For that we need for our construction procedure some parameters for specifying the mask $m(G_a)$ for a block G_a at i-th level:

- width $d^{(i)} = \lceil \log_2 \text{card}(G^{(i)}) \rceil$ for encoding the blocks at the i-th level,
- width $d(G_a) = \lceil \log_2 v(G_a) \rceil$) for encoding the states of block G_a, and
- the corresponding block code $b(G_a)$ at the next higher level as partial code.

The mask $a(G_a)$ is constructed by replacing all sign "0" and "1" and moreover $\max(d - d^{(i)}, d(G_a))$ signs "-" in the block code $b(G_a)$ by sign "x". The masks can be defined iteratively starting with the highest level. After encoding the blocks at level i the mask for all blocks at level (i-1) can be specified.

In our example it is $d = 5$, $d^{(3)} = 0$ (only one block), $d^{(2)} = 3$ (5 blocks), and $d^{(1)} = 5$ (18 state resp. blocks, s. Table 7). E.g., for block G_4 at second level $v(G_4) = 4$ implies $d(G_1) = 2$. Consequently, the mask for G_4 contains two signs "x". For block resp. state s_3 at first level $v(s_3) = 1$ implies $d(s_3) = 0$. Consequently, the mask for s_3 contains three sign "x". caused by the partial code 011-- and but because $d(s_3) = 0$ no additionally sign "x" (Table 7).

state s_i	partial code	width $d(s_i)$	mask $a(s_i)$	code $c(s_i)$
s_{17}	000--	0	xxx--	00000
s_9	001--	0	xxx--	00100
s_5	010--	0	xxx--	01000
s_{13}		0		01010
s_3	011--	0	xxx--	01100
s_7		0		01101
s_{11}		0		01110
s_{15}		0		01111
s_0	1----	0	x----	10000
s_1		0		10001
s_2		0		10010
s_4		0		10011
s_6		0		10100
s_8		0		10110
s_{10}		0		11000
s_{12}		0		11010
s_{14}		0		11100
s_{16}		0		11110

Table 7: State encoding at first level

5 Summary

In the paper we describe a novel multi-criterial state assignment approach, which is intended to integrate different procedures of low power design. The internal encoding procedure fulfills the related encoding constraints via simulated annealing with respect to decrease the switching rate of the registers. For that we use the real frequency of the state transitions resulting from profiling the FSM applying a given user-specified input pattern sequence.

Our approach allows the transformation of different optimization criteria and techniques for low power to a common encoding task. In this way e.g. power and area can be optimized concurrently.

In section 3.1 power and output logic is reduced. The efficiency of this view is demonstrated by some examples in [KFF-93]. Section 3.2 is focused on the reduction of power and gating logic. Related experimental results are given in [KFR-97]. The optimization of power and concentrator logic for inputs is the focal point of section 3.3.

The aim of the future work is to take in consideration besides power and area further optimization criteria of the design.

References

[AEH-96] Arslan,T.; Erdogan,A.T.; Horrocks,D.H.: Low power design for DSP: methodologies and techniques. Microelectronics Journal 27 (1996), 731 - 744.

[AMD-94] Alidina,M.; Monteiro,J.; Devadas,S.; Ghosh,A.; Papaefthymiou,M.: Precomputation-Based Sequential Logic Optimization for Low Power. IEEE Transactions on VLSI Systems, Dec. 1994, pp. 426-436.

[Bar-94] Baranov, Samary: Logic Synthesis for Control Automata. Kluwer Academic Publishers, Dordrecht/ Boston/ London, 1994.

[BDM-97] Benini,L.; De Micheli,G.; Macii,E.; Poncino,M.; Scarsi,R.: Symbolic Synthesis of Clock-Gating Logic for Power Optimization of Control-Oriented Synchronous Networks. EDTC '97, pp.514-520.

[BeM-95] Benini,L.; De Micheli,G.: State Assignment for Low Power Dissipation. IEEE Journal for Solid-State Circuits, Vol. 30, No. 3, March 95, pp. 32-40.

[BeM-96] Benini,L.; De Micheli,G.: Automatic Synthesis of Low Power Gated-Clock Finite-State Machines. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.15, No.6, June 1996, pp. 630-642.

[BSM-94] Benini,L.; Siegel,P.; de Micheli,G.: Saving Power by Synthesizing Gated Clocks for Sequential Circuits. IEEE Design &Test of Computers, 1994, pp. 32-94.

[Cou-98] Coudert,O.: A New Paradigm for Dichotomy-Based Constrained Encoding. IEEE Design, Automation and Test in Europe Conference (DATE'98), February 23-26, 1998, Paris, France, pp. 830-834.

[CPL-96] Chen,X.; Pan,P.; Liu,C.L.: Desensitization for Power Reduction in Sequential Circuits. DAC '96.

[For-95] Forrest,J.: ODE: Output Direct State Machine Encoding. EURO-DAC '95, pp. 600-605.

[Gra-78] Grass,W.: Steuerwerke, Entwurf von Schaltwerken mit Festwertspeichern. Springer-Verlag, Berlin Heidelberg New York 1978.

[Hac-94] Hachtel,G. et al. : Re-encoding sequential circuits to reduce power dissipation. Int. Workshop on Low-Power Design, Napa, April 1994, pp. 69-73.

[KFF-93] Koegst, M.; Feske,K.; Franke,G.: Iterative FSM Structuring by Partial State Assignment. Workshop on Design Methodologies for Microelectronics and Signal Processing. Gliwice - Cracow, Poland, 20 - 23 Oct. 1993, pp. 28-33.

[KFF-96] Koegst, M.; Franke,G.; Feske,K.: State Assignment for FSM Low Power Design. EURO-DAC'96, Sept. 1996, Geneva, Switzerland, pp. 127-134.

[KFR-97] Koegst,M.; Franke,G.; Rülke,St.;Feske,K.: A Strategy for Low Power FSM-Design by Reducing Switching Activity. PATMOS'97, September 8-10, 1997, Louvain-la-Neuve, Belgium, pp. 28-33.

[LRS-83] Leiserson,C.; Rose,F.; Saxe,J.: Optimizing Synchronous Circuitry. Proc. 3th Caltech Conf. on VLSI, Computer Science Press 1983, pp. 87-116.

[MAQ-98] Martinez,M.; Avedillo,M.J.; Quintana,J.M.; Huertas,J.L.: A Dynamic Model for the State Assignment Problem. IEEE Design, Automation and Test in Europe Conference (DATE'98), February 23-26, 1998, Paris, France, pp. 835-839.

[MoD-95] Monteiro,J.; Devadas,S.: Techniques for the Power Estimation of Sequential Logic Circuits Under User-Specified Input Sequences and Programs. Int. Symp. on Low Power Design, Laguna Beach, California, Apr. 1995.

[Pig-96] Piguet,C.: Low-Power Design of Finite State Machines. PATMOS'96, Sixth International Workshop, Bologna, Italy, 23-25 September 1996, pp. 25-34.

[RaP-96] Rabaey,J.M.; Pedram,M.: Low Power Design Methodologies. Kluwer Academic Publishers, Boston/Dortrecht/London, 1996.

[RDJ-97] Raghunathan,A.; Dey,S.; Jha,N.K.; Wakabayashi,K.: Power Managment Techniques for Control-Flow Intensive Designs. DAC '97, Anaheim, California, pp.429-434.

[RoP-93] Roy,K.; Prasad,S.C.: Circuit Activity Based Logic Synthesis for Low Power Reliable Operations.IEEE Transactions on VLSI Systems, Vol.1, No.4, Dec. 1993, pp. 503-513

[SCT-97] Surti,P.; Chao,L.F.; Tyagi,A.: Low Power FSM Design using Huffman-Style Encoding. EDTC '97, pp. 521-525.