Reducing and Smoothing Power Consumption of ROM-based Controller Implementations

Bertrand Le Gal, Aurélien Ribon, Lilian Bossuet, Dominique Dallet
IMS Laboratory, UMR CNRS 5218 – University of Bordeaux
351, cours de la Libération, 33405 Talence, France
{firstname.lastname}@ims-bordeaux.fr

ABSTRACT
Interest in automated methodologies increased last decades due to the ever-growing processing complexity and time-to-market constraints. CAD tools prove their efficiency in power consumption management, which is nowadays a major constraint for embedded systems. Efficient low power techniques for Finite State Machine (FSM) design have been proposed for logic-based controllers. In this paper, we explore the circuit power consumption reduction when the FSM is mapped in ROM blocks. The described methodology achieves power reduction of ROM-based controllers through the transformation of don’t care values in the decoder part of the design. This methodology allows a reduction of the number of resource commutations and smoothes them over the processing execution, limiting current spikes. Experiments show that the number of commutation can be decreased from 64% compared to an area-optimized ROM implementation.

Categories and Subject Descriptors
B.6.1 [Hardware]: Logic Design – Design styles.

General Terms
Algorithms, Performance, Design.

Keywords
FSM implementation, ROM-based design, Low power, CAD.

1. INTRODUCTION
Time to market pressure coupled to application complexity requires designers to use more and more Computer Aided Design (CAD) tools to speed-up their developments. Automatic algorithmic-to-hardware design generation under constraint may be realized using High-Level Synthesis (HLS) methodologies [1-2]. These CAD tools are nowadays required to cope the time to market pressure for digital systems [3-4].

To improve designer constraint handling, researches on HLS methodologies proposed solutions to novel design issues like: reliability [5], multi-mode design (single core with multiple functionalities) [6], power consumption [7], etc. Unfortunately, generated circuits obtained using these methodologies are more complex: the datapath requires more resources (arithmetic, logical, register and multiplexer elements) than hand-optimized ones. As a result, circuit controller complexity explodes – with a high number of resources and states. This complexity increase impacts the architecture performances: a large number of states produces more complex equations, introducing critical path delay issues in circuits.

Controller part of circuits may become the limiting factor for the circuit clock frequency and throughput. It has been demonstrated that implementing the controller in a ROM-based design provides constant delay disregarding the number of states and resources to control [8], compared to LUT/logic-based design [9-10]. Moreover ROM-based implementations are efficient on ASIC as well as FPGA technologies, since ROM resources are also available in low cost FPGA [11].

An important part of custom hardware accelerators implemented targeting FPGA or ASIC technologies is integrated in battery powered embedded systems. Power efficiency of circuits is important while designing embedded systems where limited power dissipation and long battery lifetime are crucial requirements. Controllers based on logic or ROM implementations may be responsible for a significant amount of power consumption [12]. Minimizing power consumed by the FSM implementations can significantly reduce the total power.

In this paper, we present a power optimization methodology for digital controllers implemented using ROM based designs. Solved problem is different from literatures approaches as we (1) do not consider that the next state computation part of the decoder is the most complex one (2) reduce the consumption spikes over the controller execution. Proposed technique helps in reducing the number of controller output commutations, and smoothing commutations spikes over controller transitions.

This article is structured as follows: section 2 presents the literature approaches for controller power optimization, and more precisely on ROM-based implementation of these controllers. Motivations are presented in section 3. Section 4 details the low power approach used to minimize the controller output commutations and to smooth them over the controller transitions. Experimental results validating our technique are reported in Section 5.

2. RELATED WORKS
Works reducing power consumption in FSM circuits have been proposed in [9, 12]. These approaches mainly focus on controller state minimization and state encoding problems to reduce the controller area, delay and power consumption. However, most these works were developed considering:

1. Logic-based implementation of controllers
2. Control intensive applications.
Implementation of controllers using combinatory logic has been proven as delay inefficient for complex controllers (important number of states and output signals) [8]. A way to cope the critical path relation linking the number of states and the signal command values is to implement the controller using ROM-based architectures. In these architectures, the critical paths do not depend on the number of control states; it depends on ROM element characteristics and logical synthesis process. ROM-based design methodology has been studied in [11, 14, 15]. However these works only focus on the area optimization of the next state part of the controller.

The main drawback of ROM-based implementations for controllers comes from the ROM area requirements which may be huge. This issue is addressed in [16, 17]. Described methodologies try to reduce ROM size using redundant row and column information (through don’t care value assignment). However these works ignores the power consumption drawback of their optimization choices. Only [12] takes into account power consumption issue in ROM-based controllers. However, this technique only considers the address modifier part of the controller design; the output decoding part is ignored.

3. WORK MOTIVATIONS

Area saving techniques [16, 17] use don’t cares to minimize ROM size requirements. Replacing don’t care values using real command values to save area impacts on ROM output signals: decoder output signals may commutate more than required. These commutations generate (1) spurious power consumption from a ROM-output and ROM-internal point of views (2) spurious activity for resources connected to the controller. In complex circuit like custom processor controllers or for Built In Self Test (BIST) pattern generators:

1. The number of output signals can be huge like the power repercussion of spurious commutation of the ROM outputs.
2. The address modifier part of the controller is less complex that decoder one. Efficient implementation schemes for the controller part are described in literature works. In this paper we have considered a counter-based design; however, other implementations can be used without approach modification.

To illustrate the power consumption problem, let’s consider Figure 1a and 1b which are respectively: the initial decoder matrix and the initial controller matrix. Consider for simplicity a linear execution path [S0, S1, S2, S3, S4, S0]. In this case, area optimized solution generate 10 output commutations while “power aware” solution (Figure 1c) produces only 9 commutations. The commutation difference is low in this teaching example. However for real ROM decoder components, differences are greater as demonstrated in Section 5.

Power consumption is a bottleneck in most embedded systems. The proposed methodology objectives are:

1. Minimizing the number of controller output commutations over the controller execution.
2. Smoothing the controller output commutations over the controller transitions to avoid power consumption spikes.

These objectives are achieved replacing efficiently don’t cares values from decoder.

4. LOW POWER APPROACH

In this section, we first propose a basic way to replace don’t cares to minimize the number of controller output commutations over its execution. Then we upgrade this proposed approach to smooth controller output commutation spikes.

4.1 Managing don’t care constraints

Controllers implemented using a ROM based design are used in many electronic systems like i.e. to control datapath resources in processing circuits like to generate patterns for Built In Self-Test (BIST), etc.

Depending on the application domain, it may be useful to handle constraints associated to controller output bits. This mean that designers may provide specific replacement rules for don’t care values i.e. while an output signal is used to control a clock gated register, don’t cares must be replaced as fast as possible using 0 value (to reduce register power consumption of the controlled resources).

To manage such constraints on controller output signals we define a symbolic subset \( \{V_0, V_1, S\} \) which is used to specify the default behavior of controller outputs. Nature of output signals is used during the optimization procedure to transform don’t care values to real ones. The three possible constraints are considered:

- \( V_0 \): replace don’t cares as fast as possible using the 0 value,
- \( V_1 \): replace don’t cares as fast as possible using the 1 value,
- \( S \): manage automatically don’t cares to minimize the number of commutations.

4.2 Basic commutation reduction technique

To minimize the controller output commutations, don’t care sets must be replaced efficiently. Let’s consider an example: controller output command is [0, X, X, 1] for cycles [1, 4]. Modifying this command signals to [0, 1, 0, 1] maintain the design behavior, but increases output commutation compared to the sequence [0, 0, 0, 1]. Such spurious commutations:

1. Increase the dynamic power consumption of the circuit
2. Produce spurious consumption on controlled resources.

Column contiguous don’t care sets must be fully replaced using only one value corresponding to the previous or the following one. This approach minimizes the total number of commutations over a processing execution (\( S_0 \rightarrow S_1 \rightarrow S_0 \)). This replacement rule is different for \( \{V_0, V_1\} \) constrained output bits where don’t cares are replaced respectively using the \{0, 1\} value.

The don’t care basic approach consists in:

1. First, replacing the constrained don’t cares using associated value if any.
2. Second, for other don’t cares, the value switching is realized As Soon As Possible (ASAP) or As Late As Possible (ALAP)
This basic technique reduces the total number of ROM output commutations over controller transitions. Finally, ROM area optimization techniques merging rows and columns can be used to reduce ROM size requirements. Resulting ROM components would have an identical number of output commutations that the area un-optimized ones. This approach reduces drastically the number of decoder output commutations compared to ROM area-optimized approach. An illustrative example, comparing results obtained using these two techniques, is provided in Figure 2. Curves indicate the number of output signal commutations depending on transitions. The controller (decoder) data used in this example comes from a controller design driven by control signals to 2D-DCT 8x8 datapath.

### 4.3 Smoothing the power consumption profile

Energy consumption profiles are a real issue for embedded devices. Power consumption spikes impact on battery lifetime and circuit power dissipation. Basic approach reduces the number of output commutations and commutation spike intensity compared to area-optimized solutions. However, it exists in both commutation profiles any commutation spikes. Commutation spikes happen when a large number of output bits switch during the same controller transition. In efficiently replacing don’t cares, it became possible to smooth the controller output signal activity (minimizing the number of parallel commutations).

The main idea of the smoothing process is to use don’t care timing mobility to sparse the output commutations on the overall transitions. An illustrative example of such objective is presented in Figure 3: curves represents the number of ROM output commutations obtained for the same 2D-DCT 8x8 controller example using (1) basic commutation reduction method (2) an ideal commutation repartition over transitions (3) a more realistic one.

Irrespective of the considered solution, the number of controller output commutations is equal. The difference is located in the commutation spreading over transitions. In ideal solution we consider that for each transition we have an equal number of commutations. This ideal solution remove the overall commutation spikes but is unreachable in practice due to irregular commutations and don’t care value repartition over transitions.

To optimize the power profile, don’t care replacement algorithm must takes into account:

1. The number of ROM output commutations for each possible controller transitions.
2. The way to apportion commutations over other controller transitions using don’t care signal mobility.

**Number of ROM output commutations:** the number of ROM output commutations happening on transition from state $S_i$ to state $S_j$ named $\beta(S_i \rightarrow S_j)$ is defined in Equation 1.

$$\beta(S_i \rightarrow S_j) = \sum_{c=1}^{N} (\Gamma_{Si}(c) \otimes \Gamma_{Sj}(c))$$

$\Gamma_{Si}$ represents the decoder output values associated to state $S_i$, $\Gamma_{Sj}(c)$ the value of the $c^{th}$ output signal at state $S_i$ and $N$ the number of controller output signals.

**Number of path commutations:** the number of ROM output commutations for the overall processing named $\Delta \rightarrow \Re$ is defined in Equation 2. This value provides the number of output commutations considering the overall existing execution paths.

$$\Delta = \sum_{a \in \{(S_i \rightarrow S_j)\}} \beta(S_i \rightarrow S_j)$$

**Maximum resource commutations:** in order to measure the resource commutations spikes we defined $\psi \rightarrow \Re$. This function defined in Equation 3 compute the maximum number of parallel commutations in the controller execution paths.

$$\psi = \max_{a \in \{(S_i \rightarrow S_j)\}} \{\beta(y,z)\}$$

Power optimization problem can be formulated as follow: how to replace don’t cares minimizing $\Delta$ and $\psi$ parameters.

### 4.4 Commutation smoothing algorithm

To smooth the controller output commutations over transitions (removing spikes), an algorithm has been developed. The algorithm detailed below is based on the following definitions.

**Don’t care start cycle:** the don’t care starting state named $\alpha(S_y, c) \rightarrow S$ for a specific don’t care signal in $\Gamma_{S_y}$ at column $c$ corresponds to state $S_y$ such that $p \in [S_y, S_j]$ and $\Gamma_{Sj}(c)_{p \rightarrow y} = X$ and $\Gamma_{Sy}(c)_{p \rightarrow y} \neq X$. State $S_y$ is the first state for which the $c^{th}$ output signal has don’t care value, assuming that don’t care value is maintained from state $S_y$ to state $S_j$.

![Fig. 2. # of controller output commutations – 2D DCT controller](image)

![Fig. 3. # of controller output commutations – 2D DCT controller](image)
Don’t care ending cycle: the don’t care ending state named \( \omega(S_y, c) \) corresponds to state \( S_q \) such that \( \Gamma(S_0, S_n) \) and \( \Gamma(S_q, c) \neq X \). State \( p \) is the last state for which the \( c \) th column has a don’t care value, assuming that don’t care is maintained from state \( S_y \) to \( S_p \).

Don’t care mobility: don’t care mobility function, named \( \delta(S_y, c) \), computes the number of consecutive states in which the \( c \) th output signal is maintained to \( X \). This value, computed using Equation 4, measures the number of states during which the commutation from the previous command value to the next value can be performed.

\[
\delta(S_y, c) = \omega(S_y, c) - \alpha(S_y, c) + 1
\]

Let’s illustrate these definitions using Figure 5a. The don’t care element located in state \( S_6 \) on 9th column (in bold), named \( \Gamma(S_6, 9) \) has vertical mobility: it exists don’t cares before and after it in the same column. The starting cycle for the don’t care value is \( \alpha(S_6, 9) = S_5 \) and its ending cycle is \( \omega(S_6, 9) = S_9 \). Starting cycle value cannot be state \( S_3 \) because don’t care values are not contiguous on \([3, 6]\) interval. The mobility \( \delta(S_6, 9) = 5 \) cycles were the transition from value \( \Gamma(S_5, 9) = 1 \) to value \( \Gamma(S_9, 9) = 0 \) can be performed.

To realize the commutation smoothing, we first construct a controller activity graph (directed cyclic graph - DCG) to model the controller possible transitions. This controller activity graph is composed of nodes \( n_i \) \( N \) which represents the different controller states. Each node is weighted using its associated instruction. Directed edges named \( e_{ij} \) links couple of nodes \( (n_i, n_j) \). Edges represent the authorized controller transitions. Each edge is weighted using:

1. the decoder switching activity associated to state transition,
2. the transition probability while available, otherwise we consider identical probability transitions.

Optimization algorithm based on a greedy approach is composed of three steps.

**STEP 1: Replacing constrained don’t cares**

ROM columns, which are not fully specified, may have forced behavior. Depending on the user constraints we first replace don’t cares using the constrained values. This step is illustrated in Figures 4a and 4b. Figure 4a presents the initial ROM contents and Figure 4b the processing results. Designer specified constraints are provided at the top of the columns. Columns tagged using \{V_0, V_1\} are value forced and \{S\} ones are constraint free.

**STEP 2: Removing false don’t cares**

The second step is about false don’t cares. A false don’t care value located in \( \Gamma(S_i) \) is defined as a don’t care which links identical values independently of the considered path \( S_i \rightarrow S_j \). Formally, false don’t cares are such that \( \Gamma(S_i, c+1) = \Gamma(S_i, c) \).

To avoid ROM output commutations, false don’t cares are replaced their previous \( \Gamma(S_i, c+1) \) values. Let’s consider an example: in Figure 5a the \( \Gamma(S_6, 10) \) and \( \Gamma(S_6, 12) \) values are false don’t cares because they link \( \Gamma(S_6, 10) = \Gamma(S_6, 13) = 0 \). In this don’t care sequence \([0, X, X, 0]\), the two don’t cares are replaced using \( 0 \) value. This modification does not generate novel controller output commutation. This optimization step is illustrated by Figure 4c: false don’t cares from Figure 4b were replaced using their previous values (equals to their following ones).

**STEP 3: Replacing remaining don’t care values**

Edge having the maximum \( \beta(S_i, S_j) \) weight in the directed cyclic graph is selected. \( \Gamma(S_i) \) and \( \Gamma(S_j) \) output words are analyzed to find don’t care values. If there is almost one don’t care value in \( \Gamma(S_i) \) and \( \Gamma(S_j) \) output words do not have any don’t care value, we select the transition having the highest \( \beta(S_i, S_j) \) weight. However, if \( \Gamma(S_i) \) and \( \Gamma(S_j) \) output words do not have any don’t care value, we select the transition having the highest \( \beta(S_i, S_j) \) weight.

**Fig. 4. Example of truth table transformation for low output decoder commutation**

<table>
<thead>
<tr>
<th>Controller State</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_0 )</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

(a) Initial truth table (ROM content)
experimentations corresponds to the custom processor circuit methodologies [6, 19]. ROM-based decoder implemented in these optimizations for area and power managing complex implemented our techniques, already has the capability to do GraphLab, the high-level synthesis tool, on top of which we have techniques which are implemented as described in Section 4. In this section, we present experimental results for the proposed 5.

EXPERIMENTAL RESULTS

In section 2 we have listed works proposing to compact ROM-based controller design. These methods factorize redundant rows and columns. Replacing don’t cares to minimize the number of output commutations impacts on the area optimization opportunities: truth table columns and rows must now be fully equals to be merged. As a consequence, solutions first optimized using power aware techniques required larger ROM components containing a don’t care value, we compute the don’t care value

The transition \( S_i \to S_j \) \( \{S_i(S_j, c), S_j(S_i, c)\} \) having the lowest number of output commutations is chosen to implement one output commutation. Output signal commutation from old output value to the new one must be performed in state \( S_i \). Don’t care elements for the \( p \)th column in the interval \( \{a(S_i,S_j, p), a(S_j,S_i, p)\} \) are replaced as follow:

\[
\Gamma(S_i,p) = \begin{cases} 
\Gamma(a(S_i,p)+1,p) & \text{if } S_i < S_r \\
\Gamma(a(S_i,p)-1,p) & \text{otherwise}
\end{cases}
\]

This transformation step is illustrated by Figure 4d: don’t cares are replaced using previous or following values depending on the number of existing commutations.

Step 3 is repeated until don’t care exists in the decoder matrix. This method provides:

- a decoder implementation free of don’t care,
- a decoder having the lowest number of parallel output commutations.

4.5 Area optimization impact

In section 2 we have listed works proposing to compact ROM-based controller design. These methods factorize redundant rows and columns. Replacing don’t cares to minimize the number of output commutations impacts on the area optimization opportunities: truth table columns and rows must now be fully equals to be merged. As a consequence, solutions first optimized using power aware techniques required larger ROM components that area only optimized ones.

Table 1 focuses on ROM output commutations and ROM area characteristics. Results have been measured considering a 64 taps FFT circuit controller. These two commutation profiles show the efficiency of proposed technique.

Table 1. ROM decoder optimization results (output commutations and area) for different controller design

<table>
<thead>
<tr>
<th>Controller parameters</th>
<th>Decoder output commutation</th>
<th>Area results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Output commut.</td>
<td>Saving vs EXP.</td>
</tr>
<tr>
<td></td>
<td>EXP 1</td>
<td>EXP 2</td>
</tr>
<tr>
<td>2D-DCT 8x8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>10122</td>
<td>5244</td>
</tr>
<tr>
<td>120</td>
<td>12384</td>
<td>5387</td>
</tr>
<tr>
<td>inverse JPEG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>16209</td>
<td>6509</td>
</tr>
<tr>
<td>FFT 64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>16934</td>
<td>16934</td>
</tr>
<tr>
<td>274</td>
<td>24468</td>
<td>24468</td>
</tr>
</tbody>
</table>

Table 1 focuses on ROM output commutations and ROM area characteristics. Results have been measured considering a complete processing execution. Resource commutations column indirectly provides results on dynamic power consumption of controller. Proposed methodologies (EXP 1 and EXP 2) reduce the number of resource commutations compared to area-reduced solution. Moreover, approach (EXP 2) allows important saving in ROM output commutation spikes from 57% up to 78% (average=64%) compared to area-optimized solution. Total number of output commutations is equal for EXP 1 and EXP 2, however EXP 2 reduces commutation spike up to 19% (average=17%) compared to EXP 2.

Table 1 focuses on ROM output commutations and ROM area characteristics. Results have been measured considering a complete processing execution. Resource commutations column indirectly provides results on dynamic power consumption of controller. Proposed methodologies (EXP 1 and EXP 2) reduce the number of resource commutations compared to area-reduced solution. Moreover, approach (EXP 2) allows important saving in ROM output commutation spikes from 57% up to 78% (average=64%) compared to area-optimized solution. Total number of output commutations is equal for EXP 1 and EXP 2, however EXP 2 reduces commutation spike up to 19% (average=17%) compared to EXP 2.

Table 1 focuses on ROM output commutations and ROM area characteristics. Results have been measured considering a complete processing execution. Resource commutations column indirectly provides results on dynamic power consumption of controller. Proposed methodologies (EXP 1 and EXP 2) reduce the number of resource commutations compared to area-reduced solution. Moreover, approach (EXP 2) allows important saving in ROM output commutation spikes from 57% up to 78% (average=64%) compared to area-optimized solution. Total number of output commutations is equal for EXP 1 and EXP 2, however EXP 2 reduces commutation spike up to 19% (average=17%) compared to EXP 2.

Table 1 focuses on ROM output commutations and ROM area characteristics. Results have been measured considering a complete processing execution. Resource commutations column indirectly provides results on dynamic power consumption of controller. Proposed methodologies (EXP 1 and EXP 2) reduce the number of resource commutations compared to area-reduced solution. Moreover, approach (EXP 2) allows important saving in ROM output commutation spikes from 57% up to 78% (average=64%) compared to area-optimized solution. Total number of output commutations is equal for EXP 1 and EXP 2, however EXP 2 reduces commutation spike up to 19% (average=17%) compared to EXP 2.
power optimization is 1% bigger than area only optimized one. In the inverse-JPEG circuit, the complete design area obtained after increase is negligible compared to complete circuit area: for the compared to area optimized solutions. However, ROM area lower. In the experiments set, ROM areas are about twice larger replaced before area optimizations, the signal redundancy is and instruction merging efficiency: as don’t cares have been replaced before area optimizations, the signal redundancy is lower. In the experiments set, ROM areas are about twice larger compared to area optimized solutions. However, ROM area increase is negligible compared to complete circuit area: for the inverse-JPEG circuit, the complete design area obtained after power optimization is 1% bigger than area only optimized one.

6. CONCLUSION

In this paper, we have presented a low power optimization technique to minimize the number of controller output commutations. This approach targets controllers implemented using a ROM-based design and more precisely their output decoder part (look-up table). Proposed methodology is efficient to reduce: the total number of controller output commutations over processing execution and the commutation spike strengths. These objectives are achieved by selecting and replacing efficiently don’t care signals in the decoder activity matrix. The power optimization procedure was composed of two major steps. First, we analyzed the activity matrix to remove constraints and false don’t cares. Secondly don’t cares are analyzed and replaced to minimize the commutation spikes according to their timing mobility. Commutation reduction assesses the effectiveness of the methodology. Controller output commutations saving is about 64% compared to area optimized solutions.

7. REFERENCES