A Hierarchical Approach for Power Reduction in VLSI chips
Prakash Arunachalam
Intel Corporation
Hillsboro, OR 97124-5961

Jacob Abraham
Computer Engineering Research Center
The University of Texas at Austin
Austin, TX 78712-1084

Manuel d’Abreu
Design Consultant
El Dorado Hills, CA 95762

Abstract
This paper presents a new mechanism for power analysis and reduction that exploits the hierarchical nature of circuits. A number of mechanisms have been proposed for power reduction, but they do not offer solutions in all cases. An activity-based reduction technique is presented where the clock is turned off for entire modules or sub-modules hierarchically when that portion of the circuit is not in use. Behavioral constraints are used to determine when a portion of the circuit is in use. The method shown is a top-down approach independent of the technology used during fabrication of the chip. Experimental results indicate that this method will result in a considerable reduction in power.

1. Introduction

System performance has been associated with circuit speed. The cost depends upon the implementation strategy. For circuits, there is a reasonable correspondence between the area and the cost. So, the designer usually explores the Area-Time space to strike a balance between the performance and the cost. Until recently, these were the only criteria for the designer, and power consumption was of lesser importance. Power is an important resource that needs to be conserved for the following reasons:

- Higher power consumption leads to reduced reliability in the chip.
- Higher power makes a computer less portable.
- The cost associated with packaging and cooling of high power consuming chips is prohibitive.

Our mechanism is based on the reduction of the circuit activity by turning off the clock in portions of the circuit not in use. Other designs have turned off the clock for conserving power, but these are based on ad hoc techniques that rely upon the intuition of the designer. The proposed mechanism studies the behavior of the circuit to find the conditions to be satisfied at the inputs of the circuit for the circuit to behave in a particular manner. This behavioral knowledge helps us to decide when the circuit is in use and thus allows us to turn off its clock when not in use.

A number of methods of power usage have checked switching activity [5]. These are simulation based and do not attempt to make a hierarchical analysis that helps us target the portions of the chip that consume power. Various levels of abstraction may be studied for analysis of power. They include Process Technology, Physical Design, Signal levels, Circuit styles, Chip Architecture and Logic Design. None of these consider the idea that we are proposing here.

Many mechanisms have been suggested in the past for countering the power due to increasing number of transistors. Two of them are scaling of device feature size and development of high-density and low parasitic packaging such as multi-chip modules [1] [2] [3]. This will later be seen to be useful in designing the power mechanism that we have developed. For computationally intensive functions the best approach was to parallelize the computation. This yields significant reduction in the per-chip power dissipation, but is a complicated procedure.

Section 2 provides an overview of the new mechanism for power reduction. Section 3 gives a summary of the knowledge extraction mechanism. Section 4 presents the results obtained from this new mechanism. Section 5 presents our conclusions.

2. Hierarchical Power Reduction Mechanism

A typical microprocessor consists of a number of modules that are interconnected in a particular man-
ner. Due to the nature of the interconnection, a set of constraints can be obtained for each of these modules to be accessed.

Once the constraints are obtained from any tool that extracts the constraints from the circuit, the circuit is simulated for each instruction to get the number of toggles when the constraints are satisfied and when they are not. This is kept in a table. This must be done over a large number of simulations in order to get the best average. This requires the toggles of all the input and internal nodes of the circuit. A structural description of the circuit is required for this purpose.

Each module has two specifications of power utilization, one when it is used, and one when it is idle. On average, each module has an idle power approximately 9-10% of the active power. The actual power savings is dependent on the program being checked.

Our algorithm shows the mechanism of the power reduction of the module as well as submodule level. The submodules are checked, because a normal processor has large modules in which many submodules are not used when the module is accessed. The modules are chosen and checked if they are accessed or not. If not accessed their clock is turned off. If the module is accessed, its submodules are checked recursively. If the submodules are not accessed, their clocks are turned off. This checking is done on a per instruction basis that results in a good power reduction. The actual turning off of the clock does not involve much switching, as it can be achieved by pass-transistor logic.

This mechanism is orthogonal to low-level power reduction mechanisms stated earlier, and if used in conjunction with them gives a better power reduction. Most of the previous work done on the optimization of power using transformations has been at a low level. This new mechanism employs an optimization of power using transformations at a high level.

3. Mechanism for knowledge extraction

The Automatic Test Knowledge Extractor (ATKET)[9][10] is a software package that automatically extracts the high level information useful for test generation. The high-level knowledge about the behavior of a system (available from the specifications) can be used to combine module tests into high quality tests for complex circuits. An important part of this package was the design and implementation of a procedural interface that provides easy access to the test knowledge database. It should be mentioned here that our method requires the generation of the constraints. While we are using ATKET at this point, other mechanisms could also be used for the generation of the constraints.

ATKET can be used to show the constraints that are to be satisfied at various inputs at different time frames for a specific mode. Thus, if there are toggles in the other inputs it does not affect the output. This is an important behavior that we use for our power reduction mechanism. Some other power-down schemes need all the inputs to be without toggles, which is a much stricter condition.

3.1. Information extraction

ATKET uses a Module Operation Tree (MOT) to capture the behavior of the modules in a design [9]. Knowledge is generated using the MOTs and is represented by the modes. A mode contains a set of constraints on either module inputs or primary outputs that need to be satisfied in each time frame to produce some effect on the outputs of a module. Each constraint in a mode is represented by a tuple (E, B) where E is the VHDL conditional expression and B is the boolean value that indicates the validity of E.

The basic modes are generated from the leaves of the MOT. Usually, the default modes for VHDL operators like '&&', '|', and so on, are used. These modes are combined from the leaf of the MOT up to the root. The information used to combine the modes includes structural connectivity and feedback loops. If any two leaves have conflicting modes, the modes are marked unrealizable and are removed from the generation list.

The knowledge extraction process for sequential circuits can be very complex, time consuming and memory intensive. The different pieces of knowledge generated using the MOT are

1. *Initialization Modes* that enable initialization of any module output to zero or to constant values.
2. *Propagation Modes* that correspond to identity modes of the module.
3. *Hold Modes* that correspond to any module output retaining its previous state.
4. *State Modes* in which the new state of any module output depends on its previous state.

Of these modes the Hold mode is of use to us. When the hold mode is satisfied, the output of the module does not change. Thus, we can turn off the clock of the module when this condition is satisfied.

4. Results

The ATKET package was installed on a SUN SPARC system. The ATKET application was modi-
fied to use memory distribution and then run on various modules from the model of the 8085 microprocessor circuit created by Alec Miczo. Each processor had a main memory of 32Mbytes. The g++ compiler version 2.6.3 was used with default optimizations enabled ("-O"). All experiments were performed when the machines were in multi-user mode. The activity factor was obtained by simulating the model using the Viewlogic Powerview VHDL system and using a C++ program to analyze the toggles.

The implementation is not fully hierarchical, as the modules are small, and do not yield higher improvements when hierarchically checked. This is due to the small size of the modules in 8085 model where the sub-modules do not have sequential elements.

The new 8085 with the power reduction mechanism was synthesized using some clever schemes to yield a large reduction in power consumption. As was previously stated, the power reduction mechanism depends on the instructions being used. Thus, the reduction varies with the application using the processor. A set of benchmark programs of numerical and non-numerical nature was used. The Intel 8085 microprocessor was chosen for the power analysis. The benchmark programs were run through the Borland C++ compiler to get the assembly language code. This was run through the Simulator. The various modules in the 8085 model are:

- Instruction Register - IR
- Register Control - RC
- Data Address - DA
- Arithmetic and Logic Unit - ALU
- Register Pad - RP
- Control Logic 1 - CL1
- Control Logic 2 - CL2
- Interrupt - IN

The activity factor values of the various units are given in Table 1. These were obtained using the Viewlogic VHDL simulator using the constraints obtained by the ATKET mechanism. This gives a feel of the amount of power each module consumes in the active and idle modes.

The percentage improvement in power consumption when the programs were tested is presented in Table 2. These have been broken down into the improvement due to the various modules. These were obtained for the individual modules by the fraction of the total number of toggles on that module to that of the entire unit.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Active Power</th>
<th>Idle Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR</td>
<td>0.0945</td>
<td>0.0103</td>
</tr>
<tr>
<td>DA</td>
<td>0.0710</td>
<td>0.0082</td>
</tr>
<tr>
<td>RP</td>
<td>0.0793</td>
<td>0.0071</td>
</tr>
<tr>
<td>RC</td>
<td>0.3005</td>
<td>0.0382</td>
</tr>
<tr>
<td>ALU</td>
<td>0.3643</td>
<td>0.0379</td>
</tr>
<tr>
<td>IN</td>
<td>0.1365</td>
<td>0.0128</td>
</tr>
<tr>
<td>CL1</td>
<td>0.0831</td>
<td>0.0114</td>
</tr>
<tr>
<td>CL2</td>
<td>0.1212</td>
<td>0.0110</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unit</th>
<th>Numerical</th>
<th>Non-numerical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>19.448</td>
<td>20.106</td>
</tr>
<tr>
<td>IR</td>
<td>1.218</td>
<td>1.233</td>
</tr>
<tr>
<td>DA</td>
<td>1.223</td>
<td>1.382</td>
</tr>
<tr>
<td>RP</td>
<td>2.702</td>
<td>2.736</td>
</tr>
<tr>
<td>RC</td>
<td>4.728</td>
<td>4.661</td>
</tr>
<tr>
<td>ALU</td>
<td>3.284</td>
<td>3.392</td>
</tr>
<tr>
<td>IN</td>
<td>2.126</td>
<td>2.217</td>
</tr>
<tr>
<td>CL1</td>
<td>1.812</td>
<td>1.920</td>
</tr>
<tr>
<td>CL2</td>
<td>2.355</td>
<td>2.565</td>
</tr>
</tbody>
</table>

Table 1. Activity Factor Distribution among modules

Table 2. Percentage reduction in Power

Our mechanism is better than the mechanism that turns off the clock after a certain period of inactivity of the module, as ours is on a per-instruction basis. For instance, some of the designs have the clock turned off if that portion of the circuit is not used for a period of time. If that is applied to the model of the 8085 in the module level, there would be lower power saving, as all the modules of 8085 are used in most instructions. We assumed that we would give a module 5 clocks before we would turn off the clock. Using this, we synthesized a new 8085 that incorporated this mechanism. This resulted in an overall gate count increase of 183, which is modest. The percentage reduction in the alternate mechanism was about 8% for both numerical and non-numerical benchmarks.

The total number of gates added to install our mechanism was 252. The new 8085 with the power reduction mechanism was synthesized and its timing was compared to the previous 8085. A number of programs were used to check the time taken. It was found that the new 8085 yielded exactly the same results as that of the original circuit. It was seen that our mechanism
had a better reduction in power than the one which turns off the clock after a period of inactivity.

The results from the timing information show that this mechanism adds a delay of 3.97% in the case of the non-numerical programs. This is a reasonable cost for a 20% reduction in the power consumption. The overall increase in size, 2.3%, is also negligible as the original 8085 had 3866 gates.

5. Conclusions

Our experiment shows that we get an approximate 10% reduction in power consumption due to this method. We predict that with a fully hierarchical implementation of our method there will be a better reduction in power.

In a large chip, we can predict a larger improvement in power for two reasons:

1. It will have a hierarchical power reduction that will yield a better power reduction. This is because most instructions use only some parts of each module.

2. There is a high degree of locality of the instructions in a large chip. For instance, the floating point unit instructions in a chip will not use the integer modules often.

Our mechanism requires a good knowledge of the behavior of each module or submodule. As there are good means to automatically extract this behavior, we can have a good way of implementing this mechanism. Tradeoffs can easily be realized by discarding those modes that are difficult to implement, i.e., they result in an increase in timing or area that is unacceptable. So the modes that have a good tradeoff between power reduction and increase in timing can be implemented.

We also showed that our mechanism is better another that turns off the clock when there have been no toggles in a particular module for a specific period of time. In fact, the power saving due to this alternative approach will always be less than our method, as our method will always turn off the clock when the other method would.

It can be seen that the top-down approach is a simple, elegant and efficient way of saving power. The drawback of this mechanism is that the clock has to be delayed to include the time taken for power reduction circuitry to complete its operation. This, however, appears to be a small factor. As our mechanism is orthogonal to most other mechanisms for power reduction suggested earlier, it can be used in conjunction with any other method.

6. Acknowledgments

We would like to thank Praveen Vishakantiah for his code of ATKET and for useful technical discussions, without which this work would not have been possible. We would also like to thank Ronald Brashear and Prof. Craig Chase for the help they gave us during the design of the distributed mechanism.

References