A Design Approach for Fine-grained Run-Time Power Gating using Locally Extracted Sleep Signals

Kimiyoshi Usami and Naoaki Ohkubo
Shibaura Institute of Technology
3-7-5 Toyosu, Kohtoh-ku, Tokyo 135-8548, Japan
{usami, m105021}@sic.shibaura-it.ac.jp

Abstract— **Leakage power dissipation becomes a dominant component in operation power in nanometer devices. This paper describes a design methodology to implement runtime power gating in a fine-grained manner. We propose an approach to use sleep signals that are not off-chip but are extracted locally within the design. By utilizing enable signals in a gated clock design, we automatically partition the design into domains. We then choose the domains that will achieve the gain in energy savings by considering dynamic energy overhead due to turning on/off power switches. To help this decision we derive analytical formulas that estimate the break-even point. For the domains chosen, we create power gating structure by adding power switches and generating control logic to the switches. We experimentally built a design flow and evaluated with a synthesizable RTL code for a microprocessor and a 90nm CMOS device model both used in industry. Results from applying to a datapath showed that the break-even point that achieves the gain exists in the number of enables controlling the power switch. By applying the domains controlled by up to 3 enables achieved the active leakage savings by 83% at the area penalty by 20%.**

Index Terms— **Leakage currents, Integrated circuit design, Design methodology, Microprocessors**

I. Introduction

As the scaling of MOS transistors proceeds, leakage power of LSI chips increases dramatically. So far, leakage power has been a major concern in portable devices because it wastes energy at standby mode and leads to shortening the battery life. One of the effective techniques to reduce standby leakage current is "power gating" [1], in which a power switch is inserted between logic circuits and the ground. In the standby mode, the power switch is turned off to electrically disconnect the logic circuits from the ground, resulting in cutting off the leakage. By using a power switch with high-Vth and thicker tox, both subthreshold leakage and gate leakage are reduced. In further scaled devices, leakage is a problem not only in standby mode but also in operation mode because it becomes a visible component in power consumption. In [2], the authors report that the leakage power at room temperature becomes comparable to dynamic power at 20nm node, while at 100C the leakage becomes comparable to dynamic power at 50nm node. Thus, to reduce operation-time power dissipation in nanometer devices, minimizing active leakage power is required in addition to minimizing dynamic power.

One of the techniques to minimize active leakage power is Run-Time Power Gating (RTPG). In LSI chips, all the circuit components are not always required to be active even in the operation mode. RTPG is a technique to detect the idle periods of circuit components in run time and to dynamically turn on/off the power switches for the components. Papers on RTPG techniques at various design levels have been published. Hu, et al [3] studied RTPG at the architecture level and proposed a course-grained technique to put the execution units of a microprocessor into sleep. The execution units are put into sleep after observing a predetermined number of idle cycles. They also proposed an approach to put the execution units into sleep when a branch misprediction is detected. Tschanz, et al [4] and Miyazaki, et al [5] discussed circuit-level techniques to dynamically control the power switches for adders with fast time constants for entering and exiting the idle mode. In contrast, in paper [6] the authors proposed logic-level RTPG for finite state machine (FSM) circuits. When state transitions do not occur, the state flip-flops keep the data and combinational logic gates to load data to the state flip-flops do not need to be active. By dynamically detecting this condition, the power switch for the combinational logic gates is turned off. Experimental results on active leakage power savings with MCNC benchmark circuits have been also reported in [6].

However, these papers do not address a methodology to perform an RTPG design from RTL to layout. Again, an implementation technique to apply fine-grained RTPG to a real block in a microprocessor has not been reported either.

In this paper, we present a top-down design methodology to implement fine-grained RTPG. The primary contributions of this work are two-fold: a proposal for a design flow to use locally extracted sleep signals for fine-grained RTPG, and an analytical model to estimate the break-even point for applying the RTPG to fine-grained domains. The rest of this paper is organized as follows: Section II presents the structure for the fine-grained RTPG and an algorithm to build the structure. Section III presents an analytical model for application of

power gating. Section IV presents the implementation methodology and Section V discusses the results.

II. Fine-grained Run-time Power Gating Structure and Generation

A. Exploiting Enable Signals of Gated Clock

Gated clock is a technique to reduce dynamic power of clock network. When data stored in flip-flops (F/F's) are not updated, clock toggling to the F/F's is stopped to reduce dynamic power. During this period, combinational logic gates located at the transitive fan-in of the F/F's are not required to compute new data to the F/F's. If outputs of the combinational logic gates are not used at anywhere else, the logic gates are considered as "idle". By detecting this idle period, we turn off the power switch provided to the combinational logic gates. This results in reducing active leakage power of the combinational logic gates. Figure 1 shows the basic structure that we use for fine-grained RTPG. We fully exploit the enable signals of gated clock design to control both power switches provided to the combinational logic gates and holders. The holder is composed of low leakage transistors (e.g. high-Vth and thicker gate oxide) and inserted between power-gated and non-power-gated circuits. When the enable signal is 0, the power switch is turned off and active leakage current is cut off at the power-gated logic circuits. The holders keep the input voltage of the non-power-gated circuits to avoid signal floating. When the enable signal is 1, the power switch is turned on and updated data are loaded into the F/F.

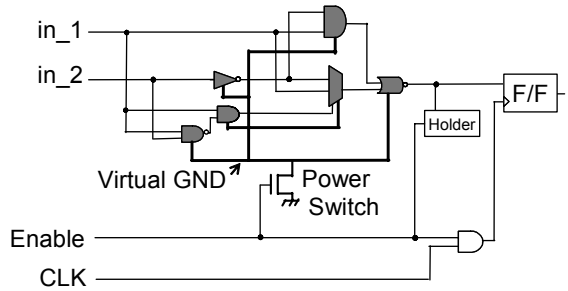

Fig. 1. Basic structure used for fine-grained Run-Time Power Gating.

B. Power Gating Domain

In actual clock-gated designs, it is likely that more than one enable signals exist. To perform fine-grained RTPG for these designs, we propose an idea of "power gating domain" (PG-domain). The PG-domain is defined as a group of circuits that are power gated with a unique enable signal. We describe the PG-domain by using an example shown in Fig. 2. In this circuit there are two enable signals EN_A and EN_B, controlling clock-gating for multi-bit registers regA and regB, respectively. Combinational logic gates enclosed with a dotted line and indicated as "Group_A" perform computation only for the register regA. In other words, the logic gates in Group_A become idle if regA is not updated. This allows us to power gate the combinational logic gates in Group_A with the enable signal EN_A. Hence, we refer to Group_A as the "PG-domain A". Similarly, logic gates indicated as "Group_B" can be power-gated using the enable signal EN_B. We hence refer to Group_B as the "PG-domain B".

In contrast, combinational logic gates indicated as "Group_X" influence not only regA but also regB. These logic gates become idle only when neither regA nor regB are updated. Therefore, we refer to Group_X as the "PG-domain AB" and power gate the domain using both EN_A and EN_B.

Logic gates indicated as "Group_Y" are not power gated because their transitive fanouts are connected to the output pins. Data at the output pins may be used outside of this circuit, and hence should be kept updated. Due to this, we do not power gate the logic gates in Group_Y. They do not belong to any PG-domain. As an extension, if this scheme is applied to the coarse-grained RTPG where the entire circuit is put into sleep, we put the gates in Group_Y into an independent PG-domain. The PG-domain is controlled by a power switch which is turned off only when the entire circuit becomes idle.

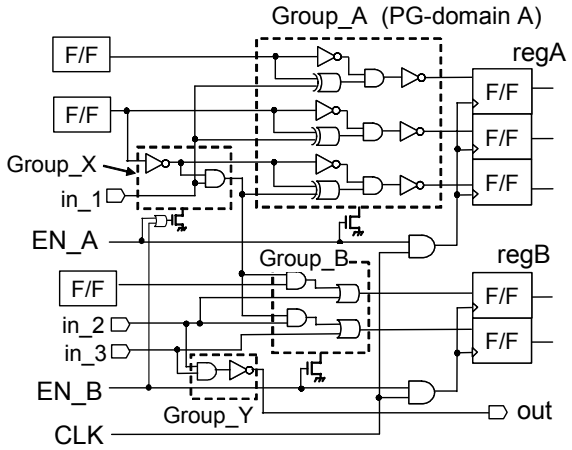

Fig. 2. Power Gating domain.

C. Algorithm to Partition into Power Gating Domains

We describe an algorithm to partition into PG-domains for a given circuit. Let us assume a circuit depicted in Fig. 3 is given. First, we focus on an F/F and find an enable signal controlling the F/F. In Fig. 3, the flip-flop FF1 is controlled by the enable signal EN_A. Next, from the data-input terminal of the F/F we traverse combinational logic network backward until reaching input pins of the given circuit or an output terminal of an F/F. We put a label "A" to all the combinational logic gates that we meet during the traversal. Thus we extract combinational logic gates located at transitive fan-in of FF1. Then we move to the next flip-flop FF2 and find an enable signal of the flip-flop. In this case, the enable signal is identified as EN_A again. Hence, the label "A" is also put to logic gates located at transitive fan-in of FF2.

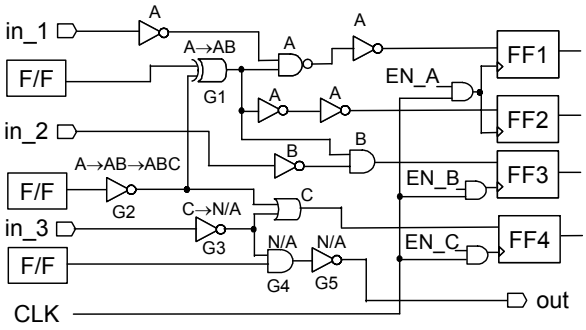

Fig. 3. Algorithm to partition into Power Gating domains.

Since the flip-flop FF3 is controlled by the enable signal EN_B, a label "B" is put to logic gates located at transitive fan-in of FF3 if unlabeled. It should be noted that we do not put the label "B" to the gates G1 and G2 since they are already labeled "A". Instead, we put a new label "AB" to G1 and G2 by ripping off the old label. Next, we focus on FF4 controlled by EN_C and put a label "C" to the extracted logic gates in the same way. Because a label "AB" is already put to the gate G2, we update the label into "ABC" at this gate. After we finish labeling logic gates located at transitive fan-in of all the F/F's, we focus on the output pins of the circuit and perform a similar backward traversal. We put a label "N/A" to the extracted logic gates because they are not power gated in the fine-grained RTPG. If the extracted gate is already labeled, we updated the label into "N/A".

After we complete labeling all the combinational logic gates, we create PG-domains according to the labels. Logic gates labeled "A" are put into the PG-domain A, while those labeled "AB" are put into the PG-domain AB. To each PG-domain, a power switch is connected. Logic gates labeled "N/A" are not put into any PG-domain because they are not power gated.

D. Generation of Control Logic for Power Switches

Since the PG-domains are built based on the labels, each power switch connected to the PG-domain is controlled by the enable signal corresponding to the label. For example, the power switch to the PG-domain A is controlled by the enable signal EN_A. In contrast, the power switch connected to the PG-domain AB has to be controlled by both EN_A and EN_B. Since logic gates in the PG-domain AB are idle when EN_A and EN_B are both "Low", EN_A and EN_B are OR-ed and used to control the power switch. For the power switch to the PG-domain ABC, a 3-input OR gate whose inputs are EN_A, EN_B and EN_C is added. The output of the OR gate is connected to the power switch.

III. Analytical Model for Power Gating

For each PG-domain a decision is required whether we really apply power gating. This is because the power gating comes with dynamic energy overhead due to turning on and off power switches. We extend the equations presented in [3] to a new analytical model enabling to estimate leakage energy savings for the PG-domain controlled by more than one enable signal. We also derive an analytical formula to give the break-even point at which the leakage energy savings equal to the dynamic energy overhead for the power switches.

First we model leakage energy savings in a PG-domain controlled by n-enables. In this model we assume that there is no leakage flowing through the power switch while it is off. However, after turning off the power switch, leakage still continues to flow through logic transistors to charge up the capacitance at the virtual ground C_{VGND} and the capacitance at internal nodes C_{int} that are in logical "zero" state. After these capacitances are fully charged, the leakage stops. Let us assume that the power switch is turned off at *t0*, the capacitances C_{VGND} and C_{int} are charged up at *t1* and the power switch is turned on at *t2*. There are two intervals we need to consider: the number of cycles Nsleep between *t0* and *t2*, and the number of cycles *m* between *t0* and *t1*. For *Nsleep* and *m*, two potential cases exist: (i) $Nsleep \geq m$ and (ii) $Nsleep < m$. First we describe an analytical model for $Nsleep \geq m$. At each sleep event, during the first *m* cycles the leakage gradually decreases and during the remaining (*Nsleep-m*) cycles no leakage flows. Leakage energy savings for a PG-domain per a sleep event is expressed as

$$E^L_{sav,per\ sleepevent} = E^L_{sav,m} + E^L_{sav,Nsleep-m} \quad (1)$$

where $E^L_{sav,m}$ and $E^L_{sav,Nsleep}$ are the leakage energy savings during the first *m* cycles and the remaining (*Nsleep-m*) cycles, respectively. The paper [3] reports that $E^L_{sav,m}$ can be expressed as

$$E^L_{sav,m} = Am^2 E^L_{cyc} \quad (2)$$

$$\text{where } A = \frac{DIBL}{m'V_t}\frac{\alpha L V_{dd}}{4\left(\frac{1}{2}+\frac{C_{VGND}}{C_S}\right)} \quad (3)$$

and E^L_{cyc} is the average leakage energy per cycle. In (3), *DIBL* is the drain-induced barrier lowering factor which is typically close to the value of 0.1, $Vt = kT/q \approx 25$mV is the thermal voltage, and $m' \approx 1.3$. Also, α is the average switching activity of logic gates in the PG-domain, L is a ratio of the average leakage and switching energy dissipated per cycle, and C_S is the total switching capacitance of logic gates in the PG-domain. Assuming the typically quoted values for general microprocessors [3]: $\alpha = 0.1$, $Vdd = 1$V, $L = 0.5$, $C_{VGND}/C_S = 0.5$, we estimate A in (3) to be 0.04.

On the other hand, since $E^L_{sav,Nsleep-m}$ can be expressed as $E^L_{sav,Nsleep-m} = E^L_{cyc}(N_{sleep} - m)$, (1) can be written as

$$E^L_{sav,per\ sleepevent} = E^L_{cyc}(Am^2 + N_{sleep} - m). \quad (4)$$

We assume that the power-switch control signal (*PSC*) connecting to the gate of the power switch transitions from '1' to '0' at the switching rate $\alpha_{PSC,1\to0}$. Within *N* cycles, since the

sleep events occur $N\alpha_{PSC,1\to0}$ times, the total leakage energy savings over N cycles can be expressed as

$$E^L_{sav} = E^L_{dom}(Am^2 + N_{sleep} - m)\alpha_{PSC,1\to0} \quad (5)$$

where is the total leakage energy of the PG-domain over N cycles. We define the signal probability for *PSC* when the signal is in '0' state as $p0_{PSC}$. Since the total sleep cycles within N cycles are $p0_{PSC}N$ and the sleep events occur $N\alpha_{PSC,1\to0}$ times, the average sleep cycles per sleep event can be written as $Nsleep,ave = p0_{PSC} / \alpha_{PSC,1\to0}$. Assuming that the duration in which *PSC* is in '0' state is randomly distributed, we obtain $Nsleep = Nsleep,ave$. From (5) we obtain

$$E^L_{sav} = E^L_{dom}\left\{p0_{PSC} + (Am^2 - m)\alpha_{PSC,1\to0}\right\}. \quad (6)$$

The average switching energy dissipated at logic gates in the PG-domain is expressed as

$$E^S_{dom} = C_S V^2_{dd}\alpha \quad (7)$$

For the following analysis we introduce the *leakage ratio* R_L as a ratio of the leakage energy and switching energy of the PG-domain, $R_L = E^L_{dom} / E^S_{dom}$. Applying this and (7) to (6), we obtain

$$E^L_{sav} = R_L C_S V^2_{dd}\alpha\left\{p0_{PSC} + (Am^2 - m)\alpha_{PSC,1\to0}\right\}. \quad (8)$$

The energy overhead that comes with the power gating is the switching energy dissipated at turning on and off the power switch, given by

$$E^S_{overhead} = \frac{1}{2}C_{PSC}V^2_{dd}\alpha_{PSC,0\to1} + \frac{1}{2}C_{PSC}V^2_{dd}\alpha_{PSC,1\to0}$$

where C_{PSC} is the switching capacitance containing the gate capacitance of the power switch, wire capacitance of the *PSC* line, and the switching capacitances of the OR gate to generate the *PSC* and buffers. Since $\alpha_{PSC,0\to1} = \alpha_{PSC,1\to0}$, $E^S_{overhead}$ can be written as

$$E^S_{overhead} = C_{PSC}V^2_{dd}\alpha_{PSC,1\to0} \quad (9)$$

We introduce the *gain function* G given by $G = E^L_{sav} / E^S_{overhead}$. From (8) and (9), G is expressed as

$$G = R_L\alpha\frac{C_S}{C_{PSC}}\left\{\frac{p0_{PSC}}{\alpha_{PSC,1\to0}} + Am^2 - m\right\}. \quad (10)$$

As described in Section II, the power switch control signal is the output of an n-input OR gate when the PG-domain is controlled by n enables. Hence $p0_{PSC}$ is given by

$$p0_{PSC} = \prod_{i=1}^{n} p0_i \quad (11)$$

where $p0_i$ is the signal probability of the enable signal EN_i when the signal is in '0' state. For simplicity we assume $p0_1 = p0_2 = ... = p0_n = p_0$.

From (11) we obtain

$$p0_{PSC} = {p_0}^n. \quad (12)$$

According to the relationship between the signal probability and the switching activity [7],

$$\alpha_{PSC,1\to0} = (1 - p0_{PSC})\,p0_{PSC} = (1 - {p_0}^n)\,{p_0}^n. \quad (13)$$

Applying (12) and (13) to (10), G is expressed as

$$G = R_L\alpha\frac{C_S}{C_{PSC}}\left\{\frac{1}{1 - {p_o}^n} + Am^2 - m\right\}. \quad (14)$$

Notice that increase of n reduces G since $p0<1$. By setting $G = 1$ in (14), the number of enables that gives the break-even point is derived as

$$n_{break-even} = \log_{p_0}\left(1 - \frac{1}{\frac{1}{R_L\alpha}\frac{C_{PSC}}{C_S} + m - Am^2}\right). \quad (15)$$

Since the number of enables is an integer, the maximum number of enables that achieves the gain is given by the floor function $\lfloor n_{break-even} \rfloor$.

Next we describe an analytical model for $Nsleep < m$. In this case the sleep event ends before the virtual-ground capacitance is fully charged up. Hence the leakage energy savings for a PG-domain per a sleep event is expressed as

$$E^L_{sav,per\,sleepevent} = A\,{N_{sleep}}^2 E^L_{cyc}.$$

The leakage energy savings over N cycles is written as

$$E^L_{sav} = R_L C_S V^2_{dd}\alpha\,A\,{N_{sleep}}^2\alpha_{PSC,1\to0}.$$

The gain function is written as

$$G = \frac{E^L_{sav}}{E^S_{overhead}} = R_L\alpha\frac{C_S}{C_{PSC}}A\,{N_{sleep}}^2. \quad (16)$$

Since $Nsleep$ can be expressed using p_0 in the same manner as described above, G can be finally given as

$$G = R_L\alpha\frac{C_S}{C_{PSC}}A\frac{1}{(1 - {p_0}^n)^2} \quad (17)$$

By setting $G = 1$, the number of enables that gives the break-even point is derived as

$$n_{break-even} = \log_{p_0}\left(1 - \sqrt{R_L\alpha\frac{C_S}{C_{PSC}}A}\right) \quad (18)$$

IV. Implementation Methodology

A. Local Virtual Ground Scheme

To implement the fine-grained RTPG the conventional global virtual-ground rail is not effective because partitioning the global rail is extremely difficult. Instead we use a local virtual ground scheme in which logic cells and power switch cells within a PG-domain are connected with a local virtual ground line [8]. To implement this scheme, we modified the existing technology library for logic cells such that within the cell the source of the nMOS transistor is disconnected from the real ground rail and instead is connected to a newly created virtual ground pin. Power switch cells with varieties of sizes are also provided in the library. The power switch cell contains an NMOS power switch transistor whose drain and gate are connected to a virtual ground pin and an enable pin, respectively. The virtual ground pin of logic cells are connected to those of power switch cells through a local virtual ground

line, which is routed as an inter-cell wire at the routing stage. We describe a design flow from RTL down to layout utilizing this scheme to implement the fine-grained RTPG.

B. Design Flow

From RTL description, we synthesize the gate-level Verilog netlist using the conventional low-Vth standard cell library. Gated clock design is performed in this synthesis step. For the clock-gated netlist, we build a fine-grained RTPG structure by using a technique described in Section II. The clock-gated design is partitioned into PG domains based on the enable signals. Then decision for the application of power gating is made by considering the break-even point. For the PG domains that achieve the gain, power switches are inserted and control logic for power switches is generated by adding OR gates. Thus, a fine-grained power gated netlist is generated. The netlist is fed to a placement tool and initial placement is performed. The placement result is given to the power switch optimization engine where power switch sizing is performed. This task is executed by *CoolPower* [9]. Power switches are sized such that voltage bounce at each virtual ground line may not exceed the user-specified upper limit. Holder-cell insertion to avoid signal floating is also performed at this step. The result of this step is sent to a router and the final layout is generated.

V. Experimental Results

A. Setup for Experiments

We experimentally built the design flow presented in the previous section by using commercial synthesis and P&R tools. As a test bench, we used a Verilog RTL model of an SH3-DSP microprocessor. This processor is a 32-bit RISC embedded CPU with a scalar pipeline. The RTL model and a Verilog simulation platform were provided by Renesas Technology through the VLSI Design and Education center (VDEC), the University of Tokyo. Fine-grained RTPG was applied to the datapath module of the microprocessor. We conducted synthesis, P&R, power-switch optimization, and power analysis by using Toshiba 90nm device models.

B. Partitioning Results

We synthesized netlist for the entire datapath module from the RTL code. Then we partitioned the entire module into 66 PG-domains. We investigated the cell count of each PG-domain and the number of enable signals controlling the domain. Results are shown in Fig. 4.

Domains are controlled by 1 to 9 enable signals. The largest PG-domain containing 1085 cells is controlled by three enable signals. Looking at next largest PG-domains with more than 200 cells, each of them is controlled only by one or two enable signals. In contrast, several smaller PG-domains are controlled by eight or nine enable signals.

C. Break-even Point Analysis

Based on the analytical formulas described in Section III, we analyzed the break-even point for the number of enable signals. Among the parameters affecting $n_{break\text{-}even}$, p_0 is the strong

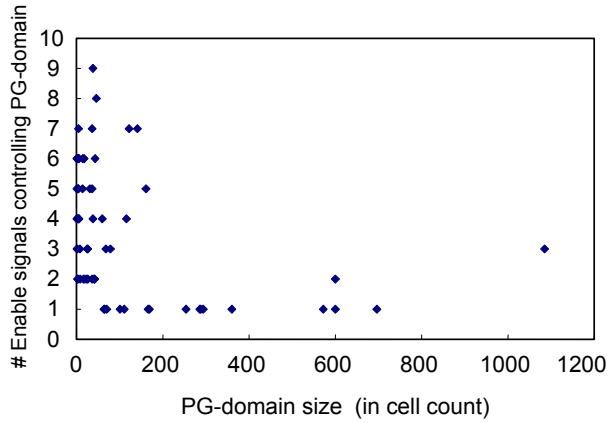

Fig. 4. Results on partitioning into PG-domains.

function of internal architecture of the processor, and m is strongly dependent upon physical implementation and the operating frequency. Assuming that R_L=0.5, α=0.1, C_{PSC}/C_S=0.1 and A=0.04, we estimated $n_{break\text{-}even}$ values against p_0 and m. We show the results based on the equation (15) first and will discuss the results based on (18) next. Figure 5 shows the results for $n_{break\text{-}even}$ based on (15). For m=1, when p_0=0.9 the $n_{break\text{-}even}$ value is approximately 3.9. This means that the PG-domains controlled by less than or equal to 3 enables will achieve the gain. As p_0 gets smaller, the $n_{break\text{-}even}$ value reduces. When $p_0 \leq 0.6$, any PG-domain cannot achieve the gain. Furthermore, as m increases the $n_{break\text{-}even}$ value reduces.

In order to decide the number of enables that achieves the gain in a design, we need to actually know the p_0 and m values.

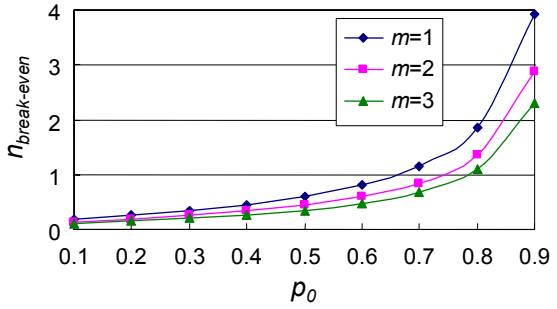

Fig. 5. Results on break-even point analysis.

We conducted a gate-level simulation for the netlist to capture the signal probability for the enable signals. This is because they are the enable signals for clock gating and hence they appear on the gate-level netlist. As a test-bench program we created a bubble sort program in C, compiled and linked with the Verilog simulation environment for the entire processor core. From the simulation results we extracted the cycles for the sorting operation by excluding the initial setting cycles. Results

showed that the p0 value was from 0.9 to 0.97 depending on the enable signals. Hence we use p_0 =0.9 in this experiment.

Since the m value strongly depends on the physical implementation, we conducted an experiment to implement the layout by assuming that we apply the power gating to all PG-domains. We extracted R and C for the virtual ground from the layout data and conducted SPICE simulations. Results showed that the time required to charge up the virtual ground to the voltage of 0.9Vdd is less than one cycle at 200MHz, which is the typical operating frequency for the embedded processor we assumed in this experiment. In our implementation methodology, after the logical partitioning into PG-domains, it is allowed to physically partition the PG-domain into further smaller sub-clusters. Logic cells that are placed closely together and share the same virtual ground are connected to the nearest power switch cell with a local virtual ground line. This enables to minimize the virtual ground capacitance, resulting in minimizing the m value. We decided the maximum number of enables as 3 based on p_0 =0.9 and m=1.

For larger m, there may be a case that $Nsleep < m$. We also estimated the $n_{break\text{-}even}$ value based on the equation (18). Results showed that when p_0=0.9 the $n_{break\text{-}even}$ value is 1.5. This means that the PG-domains controlled by only one enable can achieve the gain. When $p_0 \leq 0.8$, no PG-domain can achieve the gain.

D. Power Savings and Area Penalty

We analyzed active leakage power for the datapath with fine-grained RTPG and compared with the non-power-gated design. This analysis was performed in two steps. First, we conducted a Verilog simulation for the entire microprocessor as in the case of obtaining the signal probability for the power switch control signal. Then we traced and captured state values for input and output pins of each logic gate in the datapath. Next, we conducted leakage power analysis for the non-power-gated design by using captured state values. The analysis was performed using PowerCompiler and a library with state-dependent leakage information. The state-dependent leakage power was pre-characterized at the best process/voltage corner and high temperature (85C).

In this experiment we applied power gating to the PG-domains that are controlled by up to 3 enables. As the result, we applied the power gating to 34 domains (containing 62K cells in total) from among 66 domains (containing 72K cells). Results from the analysis leakage analysis showed that the active leakage power is reduced by 83%.

As stated in the previous section, depending on the p_0 and m values, there may be a case that $n_{break\text{-}even}$ value is reduced to 2 or 1 instead of 3. We estimated the active leakage reduction for these cases. Results are shown in Fig. 6.

When the maximum number of enables for application of power gating is 2, the active leakage power is reduced by 70%. In contrast, when the PG-domains controlled by only one enable signal can achieve the gain, the active leakage reduction decreases to 54%.

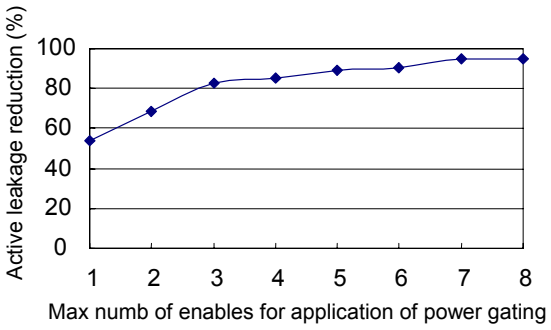

Fig. 6. Results on leakage power analysis.

E. Area Overhead

We investigated the area penalty for the layout in which the power gating was applied to the domains controlled by up to 3 enables. The layout result for the datapath with the fine-grained RTPG is shown in Fig. 7. Power switches are highlighted in purple in the figure (darker portions in the gray-scale figure). The cell area for the fine-grained RTPG was increased over the non-power-gated design by 20%.

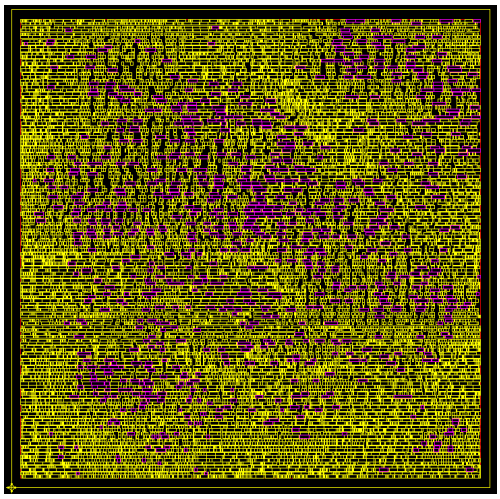

Fig. 7. Layout for datapath with fine-grained RTPG.

VI. Conclusions

We proposed a top-down design methodology to implement fine-grained Run-Time Power Gating. Based on existing enable signals in a gated clock design, we partition the design into Power-Gating domains. By using the analytical formulas we derived, decision to which Power-Gating domains we should apply the power gating was made. Power switches added to the

Power-Gating domains are dynamically controlled by the enable signals to reduce active leakage power. Results from applying this scheme to a datapath of a microprocessor showed that active leakage power was saved effectively at the reasonable area penalty.

ACKNOWLEDGMENT

The authors would like to thank M. Murakata and T. Kitahara at Toshiba for their support. They also thank J. Nishimoto at Renesas Technology for his technical advice. They are grateful to Sequence Design, Inc. for allowing us to access their tools. This work was supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc and Cadence Design Systems, Inc.

REFERENCES

[1] S.Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, J. Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS", IEEE J. Solid-State Circuits, vol.30, no.8, pp.847- 854, Aug. 1995.

[2] D. E. Lackey, P. S. Zuchowski, J. Koehl, "Designing Mega-ASICs in Nanogate Technologies", Proc. DAC'03, pp.770-775, June 2-6, 2003.

[3] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units", Proc. ISLPED'04, pp.32-37, 2004.

[4] J. Tschanz, S. Narendra, Y. Ye, B. Bloechel, S. Borkar, V. De, "Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors", IEEE J. Solid-State Circuits, vol. 38, no. 11, pp. 1838-1845, Nov. 2003.

[5] T. Miyazaki, T.Q. Canh, H. Kawaguchi, T. Sakurai, "Observation of one-fifth-of-a-clock wake-up time of power-gated circuit", Proc. CICC'04, pp.87-90, 2004.

[6] K. Usami, H. Yoshioka, "A Scheme to Reduce Active Leakage Power by Detecting State Transitions", IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), pp. I493-I496, 2004.

[7] J. Rabaey, A.Chandrakasan, B. Nikolic, "Digital Integrated Circuits", 2nd ed. pp.257-259, Pearson Education, Inc.

[8] T. Kitahara, N. Kawabe, F. Minami, K. Seta, T. Furusawa, "Area-efficient Selective Multi-Threshold CMOS design methodology for standby leakage power reduction," Proc. DATE'05, 2005.

[9] http://www.sequencedesign.com/