An Architectural Support for Reduction of In-rush Current in Systems with Instruction Controlled Power Gating

Sumanta Pyne
Deparment of Computer Science and Engineering
National Institute of Technology, Rourkela
Odisha - 769008, INDIA
pynes@nitrkl.ac.in

ABSTRACT

The present work introduces a hardware based technique for reduction of in-rush current in processors with power gating (PG) facility. A PG instruction has been introduced which is responsible in turning on multiple components from sleep to active mode at overlapped time intervals. The supporting hardware for the proposed PG instruction allows overlapped wake-up as long as the resultant in-rush current is tolerable by the system. The efficacy of the proposed method is evaluated on **MiBench** and **MediaBench** benchmark programs. The proposed method reduces in-rush current by an average of 35% with average performance loss of 5%.

KEYWORDS

Leakage power; power gating; wakeup; in-rush current; architectural support

ACM Reference Format:
Sumanta Pyne. 2018. An Architectural Support for Reduction of In-rush Current in Systems with Instruction Controlled Power Gating. In *Proceedings of 2018 Great Lakes Symposium on VLSI (GLSVLSI '18).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3194554.3194645

1 INTRODUCTION

The emergence of deep submicron process technology with the decrease in dimensions of the transistor has increased the transistor count and speed of operation at the cost of greater device leakage currents. Power gating (PG) is a technique used to reduce power consumption of VLSI chips, by shutting off the blocks that are not in use, thus reducing stand-by or leakage power. When a power gated block is switched on from sleep mode to active mode it draws a huge amount of in-rush current due to simultaneous charging of its internal capacitors. In-rush current is several times higher than the actual current required by the block to function in active mode. The flow of in-rush current may cause permanent damage to the circuit and also lead to higher power consumption. It can reduce the battery life for battery-operated systems due to rise in load current. The present work proposes a hardware support for PG instructions that simultaneously activates multiple components. It allows overlapped wake-up with guaranteed tolerable in-rush current. The existing works on management of in-rush current in PG systems are discussed in Sec. 2. The proposed method is explained in Sec. 3. Section 4 covers explaination of the experimental setup with analysis of the results. Finally, Sec. 5 concludes the present work with future directions.

GLSVLSI '18, May 23–25, 2018, Chicago, IL, USA

ACM ISBN 978-1-4503-5724-1/18/05…$15.00
https://doi.org/10.1145/3194554.3194645

2 RELATED WORK

This section discusses the existing research and developmen works on reduction of in-rush current in systems with PG. In [1] the authors proposed PG structures in which sleep transistors are turned on in a non-uniform stepwise manner to reduce the magnitude of peak current. Such a long daisy chain can cause long propagation delay and the slowly rising voltage can introduce other problems such as hot electron effects [2]. In [4] at wake-up the weak transistors are turned on first so as to slowly turn on the rush currents. When the design is discharged (charged) to a voltage close to zero (V_{dd}), the strong transistor pass is turned on ready for normal operation. A tool named CoolTime [2] guides the designer in setting power switch structure and sequence for controlling in-rush current and wake-up time. An in-rush current limiter circuit in [3] can sense the increased load current and produces sense current having a load current - sense current ratio of 1000:1, hence reducing the in-rush current. Kiong et al. introduced the in-rush current optimization power up flow analysis with PFET removal algorithm [5] to improve the in-rush current. In [6] the wake-up procedure the PG scheme implements a small transistor to control the sleep transistor in two stages to limit in-rush current and reduce wake-up time. In [7] the authors proposed model memory access power gating ($MAPG$), a low-overhead technique to enable power gating of an active core when it stalls during a long memory access. A novel framework for generating a proper power-up sequence of the switches to control the in-rush current of a power-gated domain has been introduced in [8]. The authors in [9] explain in-rush current reduction techniques like soft-start with the help of voltage regulators to increase rise time. In [10] Kim et al. discussed the reduction of in-rush current by turning on each switch cell at different times. They showed that in-rush

current can be reduced even more if signal transition time to switch each cell is adjusted.

At present there is a scope for architecture level in-rush current management in systems with instruction controlled *PG*.

3 PRESENT WORK

An arrangement for instruction controlled *PG* is shown in Fig. 1. It has n *PG* components $C_0, C_1, \cdots, C_{n-1}$. *PG* is done with the help of the header p-MOS transistors having higher threshold voltage (V_T). The header switches are controlled by an n-bit power gating control register (PGCR) placed in the power gating controller (PGC).

The bits $0, 1, \cdots, n-1$ are the *PG* bits of $C_0, C_1, \cdots, C_{n-1}$, respectively. If any of these bits $\alpha \in \{0, 1 \cdots, n-1\}$ is '0', then the component C_α is in active mode, otherwise C_α is in sleep mode. Let there be two *PG* instructions *switch_off* and *switch_on* each consuming three clock cycles - one cycle in each of instruction fetch (*IF*), instruction decode (*ID*) and execution (*EX*) stages of the instruction pipeline. To put C_α in sleep (or power gated) mode the instruction *switch_off*(C_α) is used to set the value of α^{th} bit of PGCR. C_α in sleep mode can be put to active mode with the help of the instruction *switch_on*(C_α) which resets the value of α^{th} bit of PGCR. Hence, a program can use this *PG* facility.

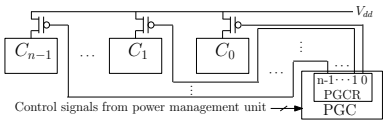


Figure 1: Instruction controlled *PG* system

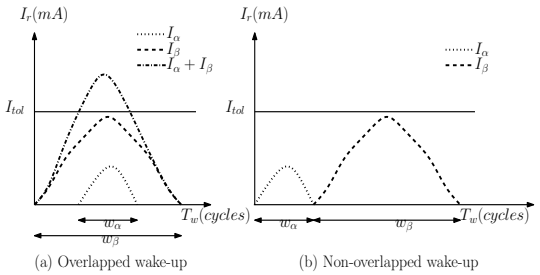


Figure 2: In-rush current for different wake-up

When C_α in sleep mode is switched on using *switch_on*(C_α) it draws in-rush current $I_\alpha \in \mathbb{Z}_{\geq 0}$ for a period of w_α cycles where w_α is the wake-up time (T_w) of C_α. It is considered that $I_\alpha \leq I_{tol}$ for wake-up of any individual *PG* component C_α, where I_{tol} is the maximum tolerable in-rush current for a given system. The problem of intolerable in-rush current may arise during wake-up of multiple components during an overlapped time interval. Fig. 2(a) shows in-rush current (I_r) in milliampere (mA) for overlapped wake-up of two components C_α and C_β where $\beta \in \{0, 1, \cdots, n-1\}$ and $\beta \neq \alpha$. w_β and I_β are the wake-up time and in-rush current of C_β, respectively. The resultant in-rush current is $I_\alpha + I_\beta$. Simultaneous overlapped wake-up of several components can lead to higher flow of in-rush current resulting higher peak power dissipation and reduction of chip reliability. Hence, it is better to have non-overlapped wake-up as shown in Fig. 2(b).

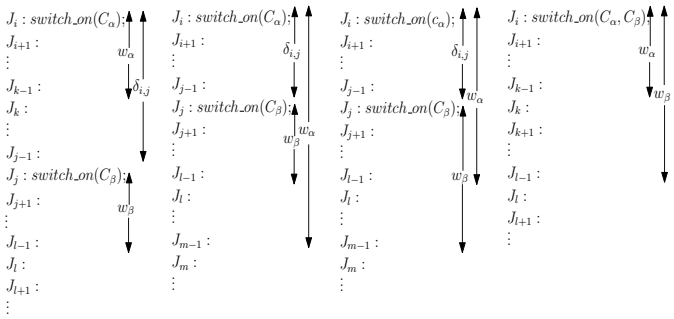


Figure 3: Overlapped wake-up

A program with *switch_on* instructions may cause overlapped wake-up of *PG* components. Considering an assembly language program consuming $N \in \mathbb{N}$ clock cycles. The instructions J_i and J_j are in *EX* stage at i^{th} and j^{th} cycles, respectively. Where J_i is *switch_on*(C_α), J_j is *switch_on*(C_β) and $1 < i \leq j < N$. J_j is executed after $\delta_{i,j} = j - i$ cycles of J_i. J_i and J_j will result non-overlapped wake-up of C_α and C_β if $i < j$ and $w_\alpha \leq \delta_{i,j}$ as shown in Fig. 3(a). Overlapped wake-up occurs if $w_\alpha > \delta_{i,j}$ as shown in figures 3(b), 3(c) and 3(d) where $1 < i \leq j \leq k \leq l \leq m < N$. The total in-rush current due to overlapped wake-up may exceed I_{tol}. To prevent this a hardware arrangement for *switch_on* is proposed that allows overlapped wake-up with guaranteed tolerable in-rush current.

3.1 In-rush aware switch_on instruction

Let *switch_on*(*#components*,*component_list*) be the format of *switch_on* instruction for the proposed approach, where *#components* $\in \{1, 2, \cdots, n\}$ is the number of *PG* components to be activated and *component_list* is the list of *PG* components where the p^{th} element *component_list*$[p] \in \{0, 1, \cdots, n-1\}$ and $p \in \{0, 1, \cdots, \#components - 1\}$. The hardware support for the proposed *switch_on* instruction is shown in Fig. 4 where for each *PG* component C_α an in-rush current table (IT_α) is maintained. The tuple $t \in \{1, 2, \cdots, w_\alpha\}$ of IT_α denoted by $IT_\alpha[t]$ stores I_α^t where I_α^t is the value of I_α during t^{th} cycle of wake-up of C_α. I_α is minimum during cycles 1 and w_α. $I_\alpha = I_\alpha^{pk}$ during cycle $\lceil \frac{w_\alpha}{2} \rceil$ where I_α^{pk} is the maximum or peak value of I_α.

Let $b = \lceil \mathbf{log_2 max}\{I_0^{pk}, I_1^{pk}, \cdots, I_{n-1}^{pk}\} \rceil$ be the number of bits required to represent $IT_\alpha[t]$. I_{tot} is a table having w_{max} tuples, where $w_{max} = \mathbf{max}\{w_0, w_1, \cdots, w_{n-1}\}$. $I_{tot}[t] \in \mathbb{Z}_{\geq 0}$ is the total in-rush current due to overlapped wake-up during cycle $i + t - 1$ where $i \in \{1, 2, \cdots, N\}$ and $t \in \{1, 2, \cdots, w_{max}\}$. $I_{tot}[t] \leq I_{tol} \forall t : t \in \{1, 2, \cdots, w_{max}\}$. $\lceil \mathbf{log_2} I_{tol} \rceil + 1$ bits are used to represent $I_{tot}[t]$. There are w_{max} $(\lceil \mathbf{log_2} I_{tol} \rceil + 1)$-bit adder-subtractors for performing $I_{tot}[t] \leftarrow I_{tot}[t] + IT_\alpha[t]$ and $I_{tot}[t] \leftarrow I_{tot}[t] - IT_\alpha[t]$ operations $\forall t : t \in \{1, 2, \cdots w_{max}\}$ whenever their corresponding control lines ADD and SUB are high. An $\mathcal{O}(w_{max} \times b)$

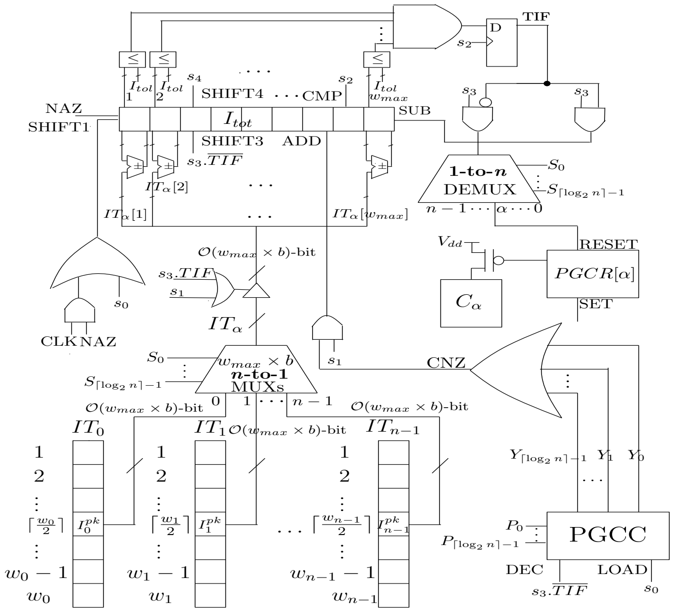


Figure 4: Hardware for in-rush aware *switch_on*

bit bus to carry IT_α. This helps in performing $I_{tot}[t] \leftarrow I_{tot}[t] \pm I_\alpha[t] \forall t : t \in \{1, 2, \cdots, w_{max}\}$ parallely within one clock cycle. $w_{max} \times b$ ***n*-to-1** bus multiplexers are used to select IT_α, where $S_0, S_1, \cdots, S_{\lceil \log_2 n \rceil - 1}$ are select lines representing α in binary form. Each bit of $IT_\alpha[t]$ is connected to the input line α of the corresponding multiplexer. There are w_{max} $(\lceil \log_2 I_{tol} \rceil + 1)$-bit comparators. They help to compare the values of I_{tot} with I_{tol} when the control line CMP is high. The **tolerable in-rush flag** (TIF) is set if $I_{tot}[t] \leq I_{tol} \forall t : t \in \{1, 2, \cdots, w_{max}\}$. The control lines SHIFT1, SHIFT3 and SHIFT4 shifts the contents of I_{tot} by one, three and four positions, respectively. The **not all zero flag** (NAZ) indicates the requirement of SHIFT1 operation after completion of a *switch_on* instruction. For NAZ=1 a SHIFT1 operation is performed in every cycle until NAZ=0 when all the elements of I_{tot} are zero. In Fig. 4 the output line α of the **1-to-*n*** demultiplexer is connected to the RESET line of α^{th} bit of PGCR (PGCR[α]). If RESET=1 then PGCR[α]$\leftarrow$ 0. This arrangement helps to turn on the p-MOS transistor that acts as header switch of the *PG* component C_α. The $\lceil \log_2 n \rceil$-bit **power gating component counter** (PGCC) stores the number of *PG* components to be turned on. It is loaded with *#components* field of a *switch_on* instruction through the input lines $P_0, P_1, \cdots, P_{\lceil \log_2 n \rceil - 1}$. The control line LOAD is high during this operation. The content of PGCR is decreased by one after reset of each PGCR[α]. The control line DEC is high during this operation. The output line **count not zero** (CNZ) is low when PGCC reaches zero. This indicates the completion of *switch_on*.

The micro-operations for the proposed *switch_on* are shown in Fig. 5. They are initiated when a *switch_on* enters the *EX* stage and performed in five sequences s_0, s_1, s_2, s_3 and s_4 each consuming a clock cycle. The micro-operations belonging to a particular sequence are performed in parallel. A control signal s_σ in the proposed hardware needs to be high to carry

s_0:
  LOAD: $PGCC \leftarrow \#components$
  SHIFT1: $I_{tot}[t] \leftarrow I_{tot}[t+1], \ \forall t : t \in \{1, 2, \cdots, w_{max} - 1\}$
    $I_{tot}[w_{max}] \leftarrow 0$
$s_1 : \alpha \leftarrow$ *component_list*[*#components* $- PGCC$]
  ADD: $I_{tot}[t] \leftarrow I_{tot}[t] + IT_\alpha[t], \ \forall t : t \in \{1, 2, \cdots, w_\alpha\}$
s_2 :
  CMP: **if** $I_{tot}[t] \leq I_{tol} \forall t : t \in \{1, 2, \cdots, w_{max}\}$ **then** $TIF \leftarrow 1$ **else** $TIF \leftarrow 0$
s_3 : **if** $TIF = 1$ **then**
    RESET: $PGCR[\alpha] \leftarrow 0$
    SHIFT3: $I_{tot}[t] \leftarrow I_{tot}[t+3], \ \forall t : t \in \{1, 2, \cdots, w_{max} - 3\}$
      $I_{tot}[t] \leftarrow 0, \ \forall t : t \in \{w_{max} - 2, w_{max} - 1, w_{max}\}$
    DEC: $PGCC \leftarrow PGCC - 1$
    if $CNZ = 1$ **then** Goto s_1 **else** Goto s_0 and Stop
  else
    SUB: $I_{tot}[t] \leftarrow I_{tot}[t] - IT_\alpha[t], \ \forall t : t \in \{1, 2, \cdots, w_{max}\}$
s_4 :
  SHIFT4: $I_{tot}[t] \leftarrow I_{tot}[t+4], \ \forall t : t \in \{1, 2, \cdots, w_{max} - 4\}$
    $I_{tot}[t] \leftarrow 0, \ \forall t : t \in \{w_{max} - 3, w_{max} - 2, w_{max} - 1, w_{max}\}$
  Goto s_1

Figure 5: Micro-operations for proposed *switch_on*

out micro-operations belonging to the sequence s_σ, where $\sigma \in \{0, 1, 2, 3, 4\}$.

Initially at s_0, PGCC is loaded with *#components* and SHIFT1 is performed on I_{tot} followed by a transition from s_0 to s_1 ($s_0 \rightarrow s_1$). In s_1, α is obtained from *component_list*. The corresponding elements of I_{tot} and IT_α are added. The sums are stored in I_{tot} followed by $s_1 \rightarrow s_2$. In s_2, the elements of I_{tot} are compared with I_{tol}. TIF is set if none of the elements in I_{tot} exceed I_{tol}, otherwise it is reset. This is followed by $s_2 \rightarrow s_3$. In s_3, if TIF=1, then PGCR[α] is reset followed by SHIFT3 and DEC operations on I_{tot} and PGCC, respectively. If CNZ=1, then $s_3 \rightarrow s_1$ takes place. Otherwise, $s_3 \rightarrow s_0$ occurs, indicating the completion of *switch_on*. In s_3 if TIF=0, then IT_α is subtracted from I_{tot} to restore I_{tot} prior to the most recent addition. After the completion of SUB $s_3 \rightarrow s_4$ occurs. In s_4, a SHIFT4 operation is performed on I_{tot} followed by $s_4 \rightarrow s_1$.

4 EXPERIMENT AND RESULTS

To establish the efficacy of the proposed approach, simulations are carried out on **gem5** [13] architecture simulator. McPAT [15] is used for obtaining power values. **gem5** is configured with the instruction set and functional units (FUs) of the ARM Cortex-M4F processor [14]. The processor has seven FUs. Integer ALU (*ialu*) is not power gated because it is used in majority of the instructions. The bits 0, 1, 2, 3, 4 and 5 of PGCR are the *PG* bits of Floating Point Divider (*fpdiv*), Floating Point Multiplier (*fpmul*), Floating Point Adder (*fpadd*), Integer Divider (*idiv*), Integer Multiplier (*imul*), and Barrel Shifter (*bshf*), respectively as shown in Fig. 6. The size of the instruction cache is considered to be 32 KB. The architectural support for the in-rush current aware *switch_on* is incorporated within the simulation environment.

McPAT is configured with the power model of ARM Cortex-M4F based on 32nm process technology, where the leakage power dissipation is almost 70% of the total power consumption. Here, the processor clock frequency $f_{clk} = 1.0$ GHz, and power supply voltage $V_{dd} = 0.9$ V. V_T of processor's n-MOS and p-MOS transistors are $V_{tn} = 0.18$ V and $V_{tp} = -0.18$ V, respectively. V_T of p-MOS transistors which act as header

switches are -0.45 V. $I_{tol} = 200$ mA. Table 1 show the values of load capacitance ($c_l^{(\alpha)}$), maximum operating current ($I_{op}^{(\alpha)}$), w_α and peak I_α (I_α^{pk}) for each C_α belonging to ARM Cortex-M4F with *PG*.

Table 1: Values of $c_l^{(\alpha)}, I_{op}^{(\alpha)}, w_\alpha$ and I_α^{pk}

C_α	*fpdiv*	*fpmul*	*fpadd*	*idiv*	*imul*	*bshf*
$c_l^{(\alpha)}$(in nF)	6.58	5.9	3.89	2.24	1.81	0.8
$I_{op}^{(\alpha)}$(in mA)	17.24	15.47	12.84	9.63	8.12	4.72
w_α(in $cycles$)	32	30	24	18	16	10
I_α^{pk}(in mA)	185	177	146	112	102	72

The features leading to generation of naive *PG* (using [11, 12]) and in-rush current aware *PG* (*IAPG*) codes are also added to the GCC compiler for ARM Cortex-M4F.

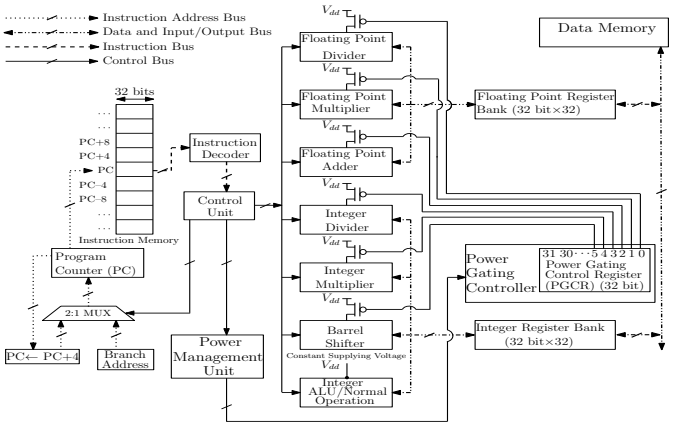
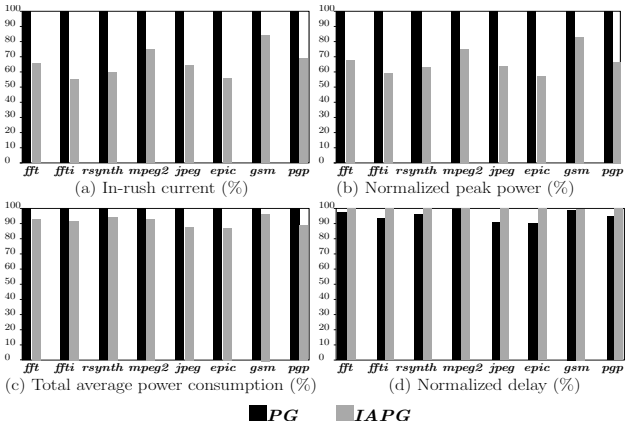


Figure 6: A machine architecture model with *PG*

Table 2: Benchmark description

Program	*fft*	*fftti*	*rsynth*	*mpeg2*	*jpeg*	*epic*	*gsm*	*pgp*
Bench	MiBench	MiBench	MiBench	Media	Media	Media	Media	Media
Category	Telecomm	Telecomm	Office	Video	Image	Image	Speech	Crypto
#cfow	15	8	15	18	14	7	23	9

The proposed technique is tested on MiBench [17] and MediaBench [18] benchmark programs as shown in Table 2, where #cfow is the number of code fragments with overlapped wake-up. The benchmark programs are compiled using updated GCC compiler. The generated target code is executed on gem5 behaving as ARM Cortex-M4F processor. The performance values are generated by gem5 which is used by McPAT to produce values of peak, average, dynamic and leakage power. The value of in-rush current is obtained from the peak power values. The experimental results are shown in Fig. 7. *PG* and *IAPG* are compared with respect to the normalized values of power and delay. Leakage power savings achieved by *IAPG* is similar to that of *PG*. Peak power dissipation $\propto$ in-rush current $\propto$ number of overlapped wake-up (*#overlapped_wake_up*). For *IAPG* *#overlapped_wake_up* is lesser than that of *PG*. Hence, peak power and in-rush current for *IAPG* are lesser than *PG*. Reduction of in-rush current and peak power dissipation experienced by *IAPG* lie within 16-47% and 18-45%, respectively. This enables *IAPG* to achieve 4-14% savings in total average power consumption. Delay $\propto$ number of *switch_on* instructions $\propto \#cfow$. The proposed *switch_on* consumes $\Omega(\#components)$ cycles whereas the naive *switch_on* takes $\Omega(1)$ cycles. The loss in performance experienced by *IAPG* lies within 0.7-10%.



(a) In-rush current (%) (b) Normalized peak power (%)

(c) Total average power consumption (%) (d) Normalized delay (%)

Figure 7: Comparison of experimental results

5 CONCLUSION

The present work introduces a hardware based technique for reduction of in-rush current in *PG* sytems. The proposed method *IAPG* reduces in-rush current by allowing ovelapped wake-up within the limitations of tolerable in-rush current. *IAPG* is evaluated on standard benchmark programs. *IAPG* reduces considerable amount of in-rush at the cost of increase in delay, design space and design cost. The future work will investigate to address these issues. Thus making them fit for real-time and safety-critical embedded systems.

REFERENCES

[1] S. Kim, S. V. Kosonocky and D. R. Knebel. Understanding and minimizing ground bounce during mode transition of power gating structures. In *Proc. of Int. Symp. on Low Power Electronics and Design (ISLPED '03)*, August 2003, 22-25.
[2] K. Choi and J. Frenkil. An Analysis Methodology for Dynamic Power Gating. *Sequence Design Inc.*
[3] A. Ball. Integrated in-rush current limiter circuit and method. *US patent US20040090726 A1*, May 2004.
[4] P. Royannez, H. Mair, F. Dahan and U. Ko. 90nm Low Leakage SoC Design Techniques for Wireless Applications. In *Proc. of IEEE Int. Solid-State Circuits Conf.*, February 2005, 138 - 139.
[5] T. S. Kiong and U. C. Kong. Power Gate Optimization Method for In-Rush Current and Power Up Time. *Intel Corporation.*
[6] K. He, R. Luo and Y. Wang. A power gating scheme for ground bounce reduction during mode transition. In *Proc. of 25th Int. Conf. on Computer Design (ICCD 2007)*, October 2007, 388-394.
[7] K. Jeong, A. B. Kahng, S. Kang, T. S. Rosing and R. D. Strong. MAPG: Memory access power gating. In *Proc. of Design, Automation, Test & Exhibition in Europe Conf. (DATE 2012)*, March 2012, 1054-1059.
[8] S. H. Chen, Y. L. Lin and M. C. T. Chao. Power-Up Sequence Control for MTCMOS Designs. *IEEE Trans. on Very Large Scale Integration (VLSI) Syst.* 21,3 (March 2013), 413-423.
[9] A. Kaknevicius and A. Hoover. Managing Inrush Current. *Application Report SLVA670A, Texas Instruments*, May 2015. www.ti.com/lit/an/slva670a/slva670a.pdf
[10] S. Kim, S. Paik, S. Kang and Y. Shin. Wake-up scheduling and its buffered tree synthesis for power gating circuits. *Integration, the VLSI journal, Elsevier* 53 (March 2016), 157-170.
[11] Y. P. You, C. Lee and J. K. Lee. Compilers for Leakage Power Reduction. *ACM Trans. on Design Automation of Elect. Syst. (TODAES)*, 11, 1 (January 2006), 147-164.
[12] S. Roy, S. Katkoori and N. Ranganathan. A Framework for Power-Gating Functional Units in Embedded Microprocessors. *IEEE Trans. on Very Large Scale Integration (VLSI) Syst.* 17, 11 (November 2009), 1640-1649.
[13] http://gem5.org/Main_Page
[14] http://infocenter.arm.com
[15] http://www.hpl.hp.com/research/mcpat/
[16] https://developer.arm.com/open-source/gnu-toolchain/gnu-rm
[17] http://vhosts.eecs.umich.edu/mibench/
[18] http://mathstat.slu.edu/~fritts/mediabench/