An Automated Design Approach of Dependable VLSI Using Improved Canary FF

Ken Yano[1]
Fukuoka University
Fukuoka, Japan
yano0828@fukuoka-u.ac.jp

Takahito Yoshiki
Fukuoka University
Fukuoka, Japan
td102017@cis.fukuoka-u.ac.jp

Takanori Hayashida
Fukuoka University
Fukuoka, Japan
thayashida@fukuoka-u.ac.jp

Toshinori Sato[1]
Fukuoka University
Fukuoka, Japan
toshinori.sato@computer.org

Abstract— **The demand of power saving and highly efficient LSI has increased by the miniaturization of semiconductor technology and the spread of portable device such as a mobile phone. We propose an automated design approach of dependable VLSI that address the timing error caused by the variation in the element characteristic in a deep submicron domain, aging and soft error. The improved canary FF described in this paper reduces about 8% of power consumption compared with the original canary FF. Using the existing standard cell library, the canary FF is mapped automatically to gate cells and its influence on chip area and power consumption is investigated.**

Keywords-***Flip-flops, timing error, dependable system, design automation***

I. INTRODUCTION

Due to the miniaturization of semiconductor technology and the spread of portable devices such as a mobile phone, further improvement of speed and power-saving LSI has come to be called for. The design method which takes the worst case scenario makes the design margin too large because of the parameter variation of the elements in the deep submicron domain has bad influence for performance and power consumption. Moreover aging and soft error would cause the timing error which is not assumed in the design phase and it has become one of the main factors of malfunction of an integrated circuit.

In this paper, the design technique of dependable VLSI which uses canary FF (CFF) by concentrating on the typical case is proposed. First, the technique of limiting the positions where to replace conventional DFF with CFF by considering the timing error information acquired from the worst case design is described. The proposed method is evaluated on two sample microprocessors. We also propose improved canary FF (iCFF) which is power saving and requires smaller area is introduced by optimizing transistor level circuit design. It is evaluated that the improved canary FF can decrease power consumption by about 8% that of canary FF. This paper further examines area and power overhead by canary FF by introducing a novel cell mapping technique to implement canary FF. Note that the improved canary FF is not used for the analysis of are and power overhead since the cell layout is under development. Canary FF can be used for reducing power dissipation in combination with DVS (Dynamic Voltage Scaling) [2] or for timing error detection like Razor FF[4] or for soft error protection. It is studied that in nanoscale CMOS domain, soft error will just not impact SRAMs but latches/flip-flops and combinational logic as well [7].

After introducing the related research of timing error detecting FF in Section II, Section III describes the transistor level circuit structure of improved canary FF. Detailed implementation method of canary FF is described in Section IV and in Section V the area and power overhead by canary FF are examined. Lastly further works and direction of possible studies are described.

In the following discussion, we use "CFF" for canary FF and "iCFF" for improved canary FF when the meaning is not ambiguous from the context.

II. TIMING ERROR DETECTING FLIP FLOP

Many researches have been done for detecting a timing error of integrated circuit. There are mainly two methods in order to improve the reliability of a circuit, one uses spatial redundancy and the other uses time redundancy. This paper describes a design approach for detection of timing error using CFF which adopts spatial redundancy and the method to integrate dependable LSI by utilizing it. We have so far reported the technique of reducing design margins by the variation of the element on LSI by using canary FF[2]. In recent years the degree of complexity of semiconductor process is increased and highly efficient and low power consumption LSI is demanded. The problem of variation in the device characteristic on a chip is emerging and the

[1] CREST, Japan Science and Technology Agency

TABLE I
Rohm0.18 micron standard cell library

Vdd	Temp.	Process
1.6V	85	Max
1.8V	25	Typ
2.0V	-40	Min

design method which takes the variation into consideration is indispensable. The change of the device characteristic by aged deterioration which is difficult to measure in the design phase and soft error are also serious issues. Flip-flops which detect a timing error such as Razor Flip-Flops (RazorFF) [4],

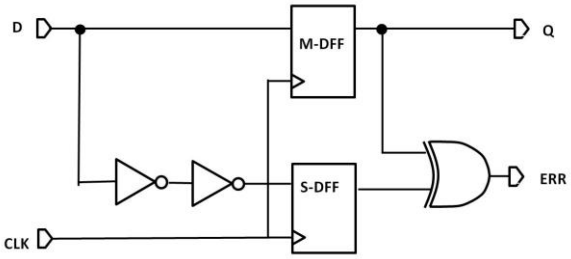


Fig. 1. Canary FF

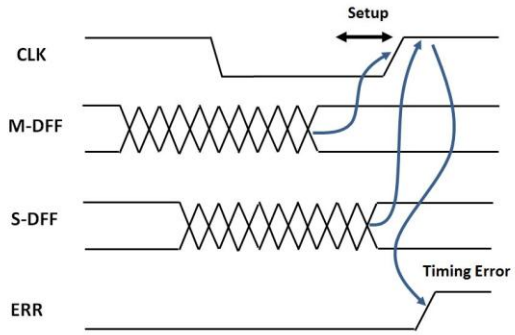


Fig. 2. Conceptual timing diagrams showing timing error detection

RazorII [3], Phase-adjustable Error Detection Flip-Flop (PEDFF), and Delay-Compensation Flip-Flops (DCFF) have been proposed. Moreover against soft error, redundant FF such as Built-In Soft Error Resilience (BISER) [7] and Bistable Cross-coupled Dual Modular Redundancy [5] are proposed. There are problems when using redundant FF, such as the increase of the power consumption and the cell area. It has been reported that the timing error detecting FF requires two to three times cell area compared with the conventional FF because it consists of shadow FF(Latch), a delay element and an error judging circuit in addition to main FF. On the other hand, there is a proposal to reduce the power consumption of the whole chip by utilizing the timing error detecting FF in combination with DVS[2].

III. IMPROVED CANARY FLIP FLOP

Canary FF has been proposed for detecting timing error and its circuit block is shown in Fig.1. In this paper, in order to consider implementation of LSI using canary FF, the

TABLE II
Power analysis of DFF,CFF and iCFF
(Clock cycle: 20ns)

	DFF	CFF	iCFF
Avg.PWR[mW]	0.025	0.068	0.063
Max.PWR[mW]	5.0	5.3	4.9

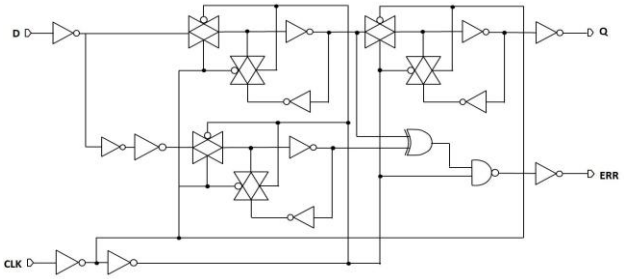


Fig. 3. Improved Canary FF circuit

transistor level circuit of CFF and cell mapping of CFF are examined. As evaluation, ROHM 0.18μm Kyoto University version standard cell library [1] is used. The cell library generated using three design corners by varying temperature and supply voltage: Max(also called worst), Typ(also called typical), and Min(also called best). The three different process corners are shown in Table I. The propagation delay of cell is longest at "Max" condition and shortest at Min "condition".

Fig.1 shows the circuit level schematic of CFF. CFF consists of main FF and shadow FF and delay buffer and XOR gate to detect a timing error. Phase synchronized clock is provided to the main and shadow FFs. Delay buffer is inserted at the input of shadow FF, hence the timing requirement of the shadow FF is severer than the main FF. Timing error is detected by comparing the outputs of two FFs. If two values are equal, system is safe and can scale down supply voltage or scale up frequency. If the two values are different, system is unsafe and about to fail to meet timing condition, so error signal is asserted to alert the error. Then the system might scale up supply voltage or scale down frequency. Fig.2 describes the conceptual timing diagrams of CFF showing timing error detection. From Fig.1, it is expected easily that it requires large cell area and the power consumption will be more than doubled since two equivalent FFs are needed in addition to the delay buffer and error detection logic. In order to reduce the area and power overhead, we try to optimize its transistor-level circuit design. When a timing error occurs, it is possible to detector the error at the timing when the data has been latched by master latch of the master and the shadow FF, hence the slave latch of shadow FF can be removed. The proposed circuit of iCFF is shown in Fig. 3.

Power analysis by HPICE simulation of three FFs:

TABLE III
# of FF at the end of timing error path

RTL	# of FF	# of FF Timing err.	Per.
MeP	3732	60	1.6%
miniMips	1967	228	11.6%

DFF,CFF and iCFF, is shown in Table.II. As expected, the average power of CFF is more than doubled than DFF. It is confirmed that the iCFF can save 7.8% of average power consumption and 7.6% of maximum power consumption of CFF.

A. Strategy of replacing with improved canary FF

If only high reliability is pursued, it is possible to replace all the FFs with iCFF, but the conventional microprocessors use thousands to tens of thousands number of FFs, it is not a practical technique when considering the area and power overhead by iCFF. Moreover, it would become serious issue how to collect timing error signal reported from all iCFFs. Hence, we have proposed that only a small number of DFF which has a small timing margin should be replaced with CFF [8]. We follow almost same approach for the selective replacement method described in [8]. Selective replacing method also used in [3] to replace DFFs of critical paths to Razor FF, however the detailed criteria and its implementation are not described in both studies. Hence in this paper we try to show our replacing method and its implementation in more detail. For the evaluation of the proposed method, we use RTL of Toshiba MeP processor [9] and miniMips processor [10].

The detail of selective replacement method is as follows: First, the RTL description is synthesized into the structured netlist by using Synopsys Design Compiler (D-2010.03-SP5). Then the delays of critical paths are analyzed. The minimum clock cycle is measured by using "Typ" condition of cell library. The minimum clock cycle is obtained by varying the clock length so that under such clock cycle no paths barely reports timing errors.

It must be confirmed that when setting the clock cycle to the measured minimum value with the process condition of cell library either with "Typ" or with "Min", no timing errors are ported. On the other hand, setting the same clock cycle length and using the library of process condition with "Max", some of the circuit paths must be ported as timing errors. This is because the "Max" process condition is the worst case of the three and the propagation delay of each cell is longest.

In Table. III, the number of DFF reported as timing error is shown. It turns out that 1.6% out of the whole DFF for the MeP and 11.6% out of the whole DFF for the miniMips are estimated to cause timing errors. Although the number of path reported as timing error is dependent on the clock cycle length, we set the smallest clock cycle under which there is no timing error is chosen when the process condition is "Typ" as described before, and select the DFFs at the end of paths reported error for replacement when using library with process condition "Max". That means, under that minimum cycle length timing errors will not occur in normal operating condition, however the possibility that timing errors will occur rise in change of environmental conditions such as sudden voltage drop, aging deterioration etc. By carrying out this strategy, it becomes possible to limit the number of DFFs which need to be replaced with CFF significantly.

B. Replacement Automation of Canary FF

The detailed procedure of replacement of DFF to iCFF is

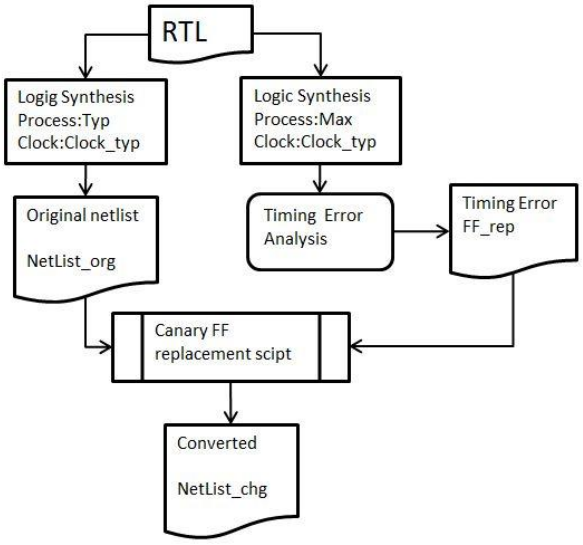


Fig. 4. Canary FF replacement procedure

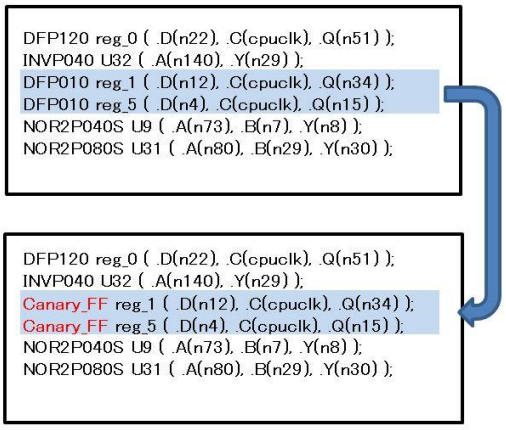


Fig. 5. Example of original netlist(top) and netlist(bottom) after some DFFs are replaced with iCFF.

shown in Fig.4. First the logic synthesis is performed by setting process condition to "Typ" and decides the smallest clock cycle length {clk_typ} from the critical path. Note

that this is an iterative process until the smallest clock cycle {clk_typ} is detected under which no timing errors are reported.

The generated netlist is saved as {Netlist_org}. Next the logic synthesis is performed again by setting process condition to "Max" and uses {clk_type} as the clock cycle. By analyzing the report from the synthesis tool, the timing error paths are extracted and the FF at the end of each path is saved in {FF_rep} for later process. After the analysis is done, the FF replacement script reads {Netlist_org} and {FF_rep} as input files and then replaces the DFF cell to iCFF cell when the instance of DFF is found in {FF_rep}. When all DFFs which might cause timing errors are replaced to iCFF, the converted netlist is outputted as {Netlist_chg}.

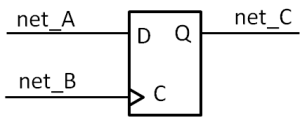
Fig.5 shows the example of original netlist and modified netlist after some DFFs are replaced with CFF. In this example, the DFF instances reg_1 and reg_5 are replaced to the iCFF since these DFFs are registered in {FF_rep}. As for the selective replacement method just proposed, we assume that iCFF is registered in the standard cell library as a custom cell and is ready for synthesis and placement & routing. However, this is not the case when the cell library is provided from third party and it is not permitted to modify or customize the library. In such cases, the iCFF has to be implemented by utilizing existing standard cells. Moreover the implementation of iCFF still underway, we propose our implementation method of CFF using existing standard cells in the next section.

IV. IMPLEMENTAION OF CANARY FF

The previous section explained the method of transforming a netlist on the assumption that we already have the cell library of iCFF. Since it is under development and the cell library is not always modifiable, we describe the implement method of CFF using existing standard cells. The same technique can be using when the iCFF is build using existing standard cells. Here the CFF is implemented using the existing standard cell and placement and routing are performed.

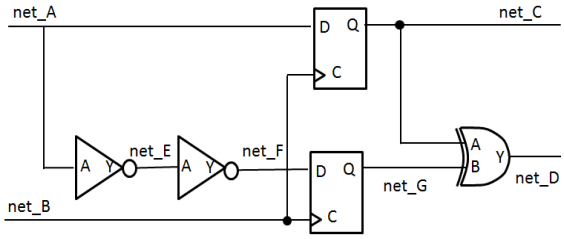
A. Cell mapping of canary FF

To help understanding following discussion, original circuit




```
ROHM18DFP010  reg_1  ( .D(net_A), .C(net_B), .Q(net_C) );
```

Fig. 6. DFF and netlist
(Before replacing to canary FF)




```
ROHM18DFP010  reg_1  ( .D(net_A), .C(net_B), .Q(net_C) );
ROHM18DFP010  UU_1  ( .D(net_F), .C(net_B), .Q(net_G) );
ROHM18INV010  UU_2  ( .A(net_A), .Y(net_E) );
ROHM18INV010  UU_3  ( .A(net_E), .Y(net_F) );
ROHM18XOR2P010  UU_4  ( .A(net_C), .B(net_G), .Y(net_D) );
```

Fig. 7. canary FF and netlist
(After replacing to canary FF)

block and netlist are shown in Fig.6. Here we consider changing reg_1 which is an instance of DFF into CFF. The instance name and the signal name are changed into the intelligible name for explanation, and as for net_a, net_b,

```
Netlist after 1st parsing

module  module_A(a, b, c);
 input    a;
 input    b;
 output   c;
 wire      net_A,   net_B,  net_C;
 @wire_decl@
   .
   DFF  reg_1 ( .D(net_A), .C(net_B), .Q(net_C));
   DFF  UU_1 ( .D(net_F, .C(net_B), .Q(net_G));
   INV  UU_2 ( .A(net_A),  .Y(net_E) );
   INV  UU_3 ( .A(net_E),  .Y(net_F) );
   XOR  UU_4 ( .A(net_C),  .B(net_G), .Y(net_D) );
   .
end module
```

```
Netlist after 2st parsing

module  module_A(a, b, c);
 input    a;
 input    b;
 output   c;
 wire      net_A,   net_B,  net_C;
 wire      net_D,   net_E,  net_F,  net_G;
   .
   DFF  reg_1 ( .D(net_A), .C(net_B), .Q(net_C));
   DFF  UU_1 ( .D(net_F, .C(net_B), .Q(net_G));
   INV  UU_2 ( .A(net_A),  .Y(net_E) );
   INV  UU_3 ( .A(net_E),  .Y(net_F) );
   XOR  UU_4 ( .A(net_C),  .B(net_G), .Y(net_D) );
   .
end module
```

Fig. 8. Netlist after 1st parsing(top) and after 2nd parsing(bottom)

net_C, it is necessary to extract the actual signal name used by each FF which needs to be replaced to CFF in the FF replacement script. Although FF replacement script to be used is almost the same as that of what was explained in Section 3, it is not simply to replace DFF to CFF, but to replace it by the cell group which constitutes the CFF. The details are stated following.

The circuit block and netlist after transformation is shown in Fig. 7. Since INV and EXOR cells are already registered into the standard cell library currently used, each gate is mapped to those cells. All the cells to be used adopt those of the minimum drive capability. UU_1, UU_2, UU_3, and UU_4 are the added instance name of the generated cells and net_D, net_E, net_F, and net_G are the added signal names. In order to avoid the duplication of name which are used by the original netlist, it is necessary to create these unique added names by combining a suitable prefix name and consecutive numbers. Since these names must be unique only within each module definition, consecutive numbers are reset when FF replacement script analyzes the syntax of the start part of a module definition, and they are incremented whenever a new name is generated. About the newly added signal name, it needs to be declared in the definition part of the module for which it is used. However, in the stage in which FF replacement script parses the module definition part, since it is not clear which signal names should be declared, it is decided to make a signal declaration by using a double parsing system. By the first parsing, while performing the processing which replaces applicable DFF to a CFF, the place holder {@wire_decl@} is described as a mark into the portion which makes a signal declaration in the head portion of module declaration as explained in Fig.8. The place holder is replaced by the declaration of the added signal names by the 2nd parsing. The added signal names are managed using the associative array referred to from the module name to which it is scheduled to be declared by the first parsing.

V. Power and Area Overhead by Canary FF

In this section, power and area overhead by CFF is investigated. Placement and routing (P&R) are performed using Synopsys IC Compiler (D-2010.03-ICC-SP2-1). Note that the layout of iCFF is not implemented yet, so the original CFF circuit is used and the netlist is generated by the method described in section IV. We use four different sets of configuration to estimate the overhead by CFF. Each configuration is described as follows,

(1) T : P&R is performed using cell library (“Typ” case) and no DFFs are replaced by CFFs
(2) TC: P&R is performed using cell library (“Typ” case) and some DFFs are replaced by CFFs using selective replacement method
(3) M : P&R is performed using cell library (“Max” case) and no DFFs are replaced by CFFs,
(4) TCA : P&R is performed using cell library (“Typ” case) and all DFFs are replaced by CFFs

Cell area and power estimates are obtained from the result of P&R by IC Compiler. Fig.9 and Fig.10 show power and area overhead for minimips and MeP processor respectively. Area is normalized based on config. (M) for both cases. For minimips, power of config. (TC) is 19.11[mW], which is increased by 26% from config. (M) and by 9.8% from config. (T). It turns out that power overhead by CFF is relatively large. In case of config. (TCA), power is estimated to 34.18[mW] which is 2.25 times larger than that of config. (M). Hence the selective replacement of CFF is deemed an effective method to decrease the power overhead by CFF. The power of config.(T) is larger than that of config.(M). This might be because in cell library “Typ”, the supply voltage is defined as 1.8V and in cell library “Max”, it is defined as 1.6V, hence the net total power consumption of config.(T) is slightly larger than that of config.(M).

On the other hand, the area of config.(TC) is decreased by 23% from config. (M) and increased by 3.6% from config. (T). This means that area overhead by CFF is much less than the cell area estimated by considering worst case “Max”

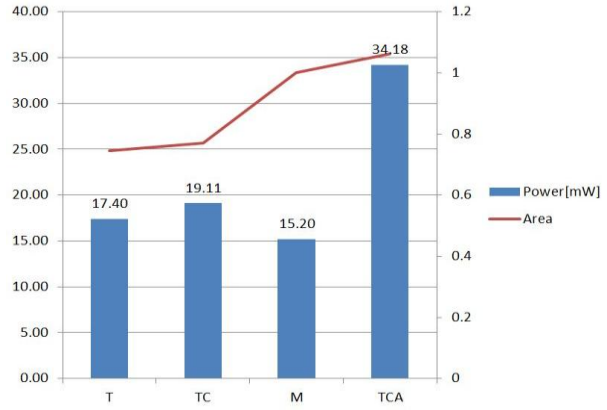


Fig. 9. miniMips
(Power and area overhead by CFF)

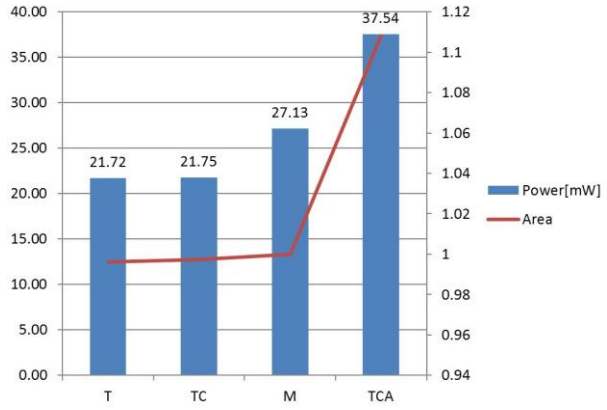


Fig. 10. MeP
(Power and area overhead by CFF)

condition. The cell area of config.(M) is relatively larger than that of config.(T). This is because in case of config.(M), gate cells with bigger drive strength, so larger cell area, are selected to compensate for longer propagation delay. Hence the area overhead by canary FF is deemed very small.

As for MeP, area overhead by CFF are very small since the chip area is mostly occupied by instruction and data caches. Instruction and data caches are implemented as hard macros using library with "Typ" condition. Actually the area of config.(TC) is decreased by 0.3% from that of config.(M) and increased by 0.1% from that of config.(T). Increase rate of area of config.(TCA) is relatively larger compared with minimips case. This might be because the number of DFFs is much larger than minimips. The power of config.(TC) is decreased by 20% from config.(M) and is increased by 0.1 % from config.(T). The power of config.(M) is much larger than config.(T). This might be since the chip area of MeP is occupied with large cache area, so the power consumed by cache is not negligible.

The difference of the two results comes from different architecture of the two microprocessors. MeP is an off the shelf commercial processor and uses state-of-art technology and contains large cache area to increase the performance (IPC); however the minimips is an open architecture processor and contains no cache. The chip area of Mep is about 7 times larger than minimips and the power consumption is about 1.25 times larger than minimips.

CFF can be used not only for microprocessor but also any sequential circuits which require timing error detection mechanism. The area and power overhead by CFF can be suppressed low by selecting DFFs for replacement by analyzing the critical paths under system timing requirement.

VI. CONCLUSION AND FUTURE WORKS

By miniaturization of semiconductor technology, the timing error caused by process variation of within-die or intra-die and aging deterioration is considered serious issue especially in deep submicron domain. The importance of the technique avoiding the defect of LSI during operation is increasing. In this paper, an automatic design method of reliable LSIs with canary FF is proposed which concentrates on typical case to ease the design margins incurred by worst case analysis. We show that by selectively replacing DFFs with canary FFs, the area and power overhead by canary FFs can be suppressed very small. The selection of DFFs is done by analyzing critical paths from worst case based on the results of typical case.

The future remaining studies regarding canary FF are as follows. First, we have to build the cell library of the improved canary FF. Since the CMOS circuit is complicated, it will be designed as double height cell. When the library is built, power and area overhead by iCFF can be measured in more detail and the comparison with CFF can be discussed. It also necessary to consider the method of collecting error signals when timing error is detected from CFFs and utilization of that signal. If there is N CFFs and when the collection of error signals is constituted from OR gates of two ports, an error signal will travel $\log_2 N$ piece of OR gate. Then the wiring delay of error signal cannot be disregarded. The error signal could be used to trigger DVS or DVFS to control the supply voltage and the clock frequency. Moreover, it is also necessary to examine the amount of delay buffer inserted at the shadow latch. When the amount of delay buffer estimates to be large excessively, timing error information occurs more than needed, and it will affect the performance. Conversely, when it is estimated too small, the possibility of overlooking timing errors becomes high and system reliability falls down. Furthermore, testing and verification of proposed method are not yet done and needs to be addressed more.

ACKNOWLEDGMENT

This study is supported in part by CREST project "Fundamental technologies for dependable VLSI system" of Japan Science and Technology Agency. The cell library used on this research was developed by Tamaru/Onodera laboratory, Kyoto University, and is released by Prof. Kobayashi of Kyoto Institute of Technology. This work is supported by VLSI Design & Education Center (VDEC), the University of Tokyo [11] in collaboration with Synopsys, Inc. and Cadence Design Systems, Inc.

REFERENCES

[1] H. Onodera, A. Hirata, T. Kitamura, K. Tamaru, "P2Lib: Process Portable Library and Its Generation System", IPSJ Journal, Vol.40, No.4, pp.1660-1669, 1999.

[2] T. Sato, Y. Kunitake, "Canary: A Variation Resilient FF to Eliminate Design Margin for Energy Reduction", IPSJ Journal, Vol.49, No.6, pp.2029-2042, 2008.

[3] D. Blaauw, S. kalaiselvan, K. Lai, et al., "RazorII:In Situ Error Detection and Correction for PVT and SER Tolerance", International Solid-State Circuits Conference, pp.400-622, 2008.

[4] D. Ernst, N. S. Kim,S. S. Das, et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation", 36th International Symposium Microarchitecture,pp.7-18,2003.

[5] J. Furuta, C. Hamanaka, K. Kobayashi, and H. Onodera, "A 65nm Bistable Cross-coupled Dual Modular Redundancy Flip-Flop capable of protecting soft errors on the C-element", IEEE Symposium on VLSI Circuits, 2010.

[6] K. Hirose, Y. Manzaw, M. Goshima, and S. Sakai, "Delay-Compensation Flip-Flops for Timing-Error Tolerant Circuit Design", International Conference on Solid State Device and Materials,pp.440-481,2007.

[7] S. Mitra, N. Seifert, M. Zhang, Q. Shi, K. S. Kim, "Robust System Design with Buit-In Soft-Error Resilience", IEEE Computer, pp. 43-52, February, 2005.

[8] Y. Kunitake,T. Sato,S. Yamaguchi,H. Yasuura, "Insertion-Point Selection of Canary FF for Timing Error Prediction", IEICE Technical Report, DC2008-42, 2008.

[9] Toshiba MeP Processor, http://www.semicon.toshiba.co.jp/

[10] miniMips processor, http://opencores.org/project,minimips

[11] VDEC, the University of Tokyo, http://www.vdec.u-tokyo