Minimization of Register Area Cost for Soft-Error Correction in Low Energy DMR Design

Kazuhito Ito Takumi Negishi

Graduate School of Science and Engineering
Saitama University
255 Shimookubo, Sakura-ku, Saitama, 338-8570, Japan
{kazuhito,negishi}@elc.ees.saitama-u.ac.jp

Abstract— Double modular redundancy (DMR) is to execute an operation twice and detect soft-error by comparing the operation results. The soft-error is corrected by executing necessary operations again to obtain correct results. Such re-executing operations requires thier input data and many registers are needed to store the necessary data. In this paper, a method to minimize the area cost of registers is proposed while the minimization of operation energy consumption is considered with respect to the give constraints of time, resource, and delay penalty for error correction. The experimental results show about 20% of register cost is reduced on average.

I. Introduction

As LSI chips integrate more transistors and other components and the operating power supply voltage decreases, LSI chips are becoming more vulnerable to soft error caused by single event upsets (SEU) in registers and memories [1, 2, 3] and single event transients (SET) in logic circuits [4, 5, 6].

There are several approaches to achieve the sufficient level of reliability of LSIs against the SEU and SET. The first is the fault-tolerant approach where the error is corrected at the system level. For example, with the triple modular redundancy (TMR), the same operation is executed three times, the results are compared, and the majority is adopted as the correct result [7, 8, 9, 10]. The drawback of this approach is that many functional units or long operation time is necessary to execute the operation three times. The second is the circuit hardening where each elementary circuit is designed so that it is immune to SEUs [4, 5]. Although the hardening works effectively to the moderate charge deposited by an SEU, there can be an error when the charge more than the predetermined threshold is deposited. The third is the error detection where an error is detected by executing an operation twice and comparing the results [11, 12, 13]. If the results are not identical, it is known that an error has occurred. This scheme is called double modular redundancy (DMR). In [11, 13] an error is detected but not corrected. The method in [12] mitigates both SEU and SET by introducing sufficient length of delay. The drawback of the method is the operation speed is half the circuit without DMR.

A soft-error in DMR is corrected by executing necessary operations again to produce the correct data. Such re-executing operations requires their input data and many registers are needed to store those necessary data. Thus the minimization of the area cost of registers is an important task in DMR design. In [13], a delay penalty was introduced to minimize the energy consumption of operations by appropriately assigning power supply voltages to operations. In this paper, a method to minimize the area cost of registers is proposed while the minimization of energy consumption is considered with respect to the given constraints of time, resource, and delay penalty for error correction.

II. Error collection and register usage in DMR

A. Error model and DMR

To detect soft-error in DMR, an operation execution and the input data to the operation are duplicated, and the duplicated operation results are compared. Error is modeled so that no more than one error occurs at the same time in the input data, duplicated operation execution, and the comparison, all of which are related to an operation. When an error is detected for an operation, the error exists in one of the duplicated input data, in the operation executions, or in the comparison. In the case that an error is detected, the operation is executed in a duplicated manner and the results are compared again. The re-execution is done within a sufficiently short time interval from the error and thus no error occurs during the re-execution.

B. Error correction by replay

When an error is detected in DMR, the error can be mitigated by executing necessary operations. Figure 1 shows an example of the error mitigation. Assume there are two operations '1' and '2' and there exists data dependency that '2' uses the result of '1' as shown in Fig. 1(a). Let '2' be said a *child* of '1'. With DMR, each operation is executed twice and these are called respectively the *primary* and *secondary* executions of the operation. They are denoted as '1_P' and '1_S' for operation 1. When both 1_P and 1_S are executed and the results are stored in registers, they are compared to check an error. Figure 1(b) is a time chart showing the schedule of operation executions and comparisons, where 'E' denotes a comparison. Let $D_{i,m}$ denote

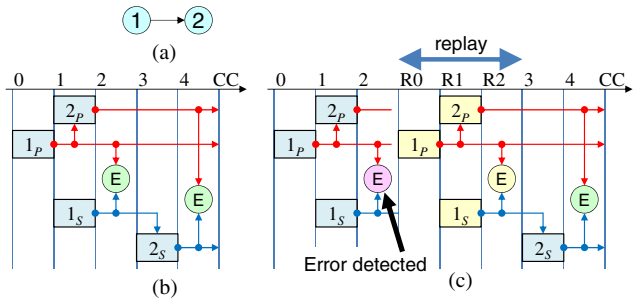

Fig. 1. An example for error correction by Replay in DMR. (a) a DFG. (b) schedule for operation execution. (c) replaying the schedule when an error is detected.

the result produced by i_m for operation i ($m = P$ or $m = S$). For example, 1_P and 1_S start at clock cycle (CC) 0 and 1, respectively. Both 1_P and 1_S take one CC and $D_{1,P}$ and $D_{1,S}$ are stored in registers at the end of CC 0 and CC 1, respectively. An arrow in Fig. 1(b) represents the lifetime of data stored in a register. The comparison between $D_{1,P}$ and $D_{1,S}$ is done at CC 2. If the comparison is affirmative, i.e., the compared data are equal, any of 1_P, 1_S, and $D_{1,P}$ and $D_{1,S}$ stored in registers does not contain an error. If the comparison is negative, either 1_P or 1_S, or one of the data stored in registers, or even the comparison itself contains an error. Since it is not possible to identify which is erroneous and which are correct, both 1_P and 1_S are executed again, and $D_{1,P}$ and $D_{1,S}$ are stored in registers again. According to the error model where at most one error occurs within a short time period, re-executed 1_P and 1_S, and $D_{1,P}$ and $D_{1,S}$ stored in registers are error-free, and thus the error is mitigated.

When an error occurred with operation 1, the child (children) of operation 1 which has executed before re-execution of operation 1 might have used erroneous $D_{1,P}$ and $D_{1,S}$. Thus the child (children) must be re-executed when operation 1 is re-executed. Such re-execution of a child (children) can be easily achieved by repeating the schedule from the start of 1_P. Let this scheme be called *replay*. Figure 1(c) shows an example of replay when an error occurred for operation 1. Since 1_P is scheduled at CC 0 and the error occurred for operation 1 is detected at CC 3, the part of the schedule from CC 0 to CC 3 is executed again in the replay. The replayed CC is denoted as 'Rt' for the original CC t. By re-executing 1_P, the error-free $D_{i,P}$ is stored in a register, and 2_P is also re-executed using the error-free $D_{i,P}$. After the replay, the original schedule resumes.

C. *Register requirement for replay*

Figure 2(a) shows another possible schedule of operations in the DFG in Fig. 1(a). When either $D_{1,P}$ or $D_{1,S}$ or both are used by operation 2, $D_{1,P}$ and $D_{1,S}$ are compared to check whether $D_{1,P}$ or $D_{1,S}$ contains an error. The origin of the error may be either (1) an SET in the functional unit (FU) to execute 1_P or 1_S produces an erroneous $D_{1,P}$ or $D_{1,S}$ and it is stored in a register, or (2) the correct $D_{1,P}$ and $D_{1,S}$ are stored in registers but an SEU in the register causes the error. In the schedule of Fig. 2(a), $D_{1,P}$ and $D_{1,S}$ are compared at CC 3. If the comparison is negative, replay is initiated to re-execute 1_P and 1_S. By the replay, the execution of operations is delayed. In real-time applications, such delay penalty must be limited within a predetermined value. Let p_E denote the allowed maximum delay penalty caused by the replay. Assume p_E is specified as 4 CCs. In the case of the schedule shown in Fig. 2(a), the delay penalty by replaying operation 1 is 4 CCs and it is within p_E. In the case of a schedule shown in Fig. 2(b), an error in registers storing $D_{1,P}$ and $D_{1,S}$ are detected at CC 6 when 2_S is executed and replaying 1_P causes the delay penalty of 7 CCs and it violates p_E.

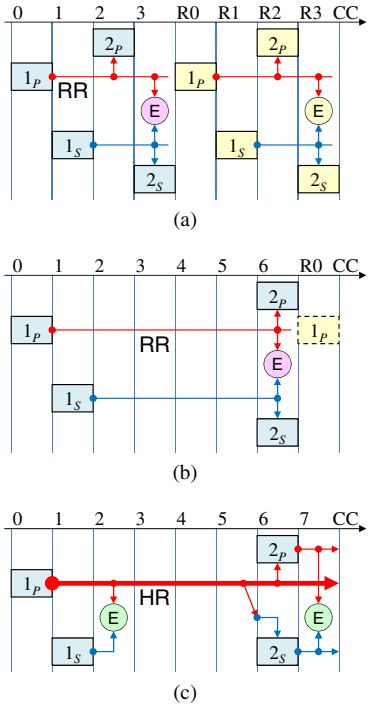

Fig. 2. Register mode selection for replay delay penalty and register cost. (a) operation 1 is in RR mode satisfying $p_E = 4$. (b) delay penalty by replaying operation 1 is larger than $p_E = 4$. (c) operation 1 is in HR mode.

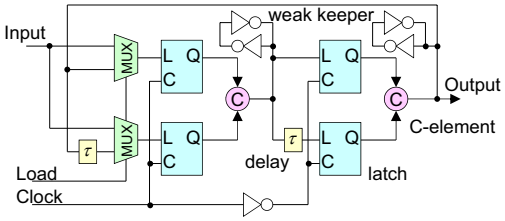

Fig. 3. Hardened register (HR).

Figure 3 shows the hardened register (HR) introduced in [12]. It is a DMR-style register and has the advantage that an SEU in one of the latches in an HR does not cause an error at the output and the SEU is corrected as the clock signal transitions. The drawback is that even when a given input data is wrong, it is loaded in the register. Therefore the correctness of the data given to an HR register must be checked outside the register.

An HR can be used instead of a regular register (RR) to store

$D_{1,P}$ if the replay delay penalty violates the specified p_E. In the schedule shown in Fig. 2(c), $D_{1,P}$ is stored in an HR and its correctness is checked by comparing it with $D_{1,S}$ at CC 2. Once the correctness of the data is ensured, it is not necessary to further check the data for error detection since HRs are SEU-free. Because the comparison between $D_{1,P}$ and $D_{1,S}$ is eliminated, the replay for operation 1 is not necessary and no delay penalty is caused. Let this situation be said "operation i is in HR mode" where $D_{i,P}$ is stored in an HR. On the other hand, if an RR can be used to store $D_{i,P}$ without violating the delay penalty (i.e., Fig. 2(a)), it is said "operation i is in RR mode."

The area cost of an HR is larger than the cost of two RRs because an HR consists of 4 latches and additional circuitry of C-elements and weak keepers. Therefore RR mode is preferable to minimize the area cost of registers and HR mode should be used only when necessary.

Assume that operation j is a child of operation i. If operation i is in HR mode, $D_{i,S}$ is used to check whether $D_{i,P}$ stored in an HR is correct. Once the correctness of $D_{i,P}$ in the HR is ensured, $D_{i,S}$ becomes unnecessary. Although HRs have self-error correction capability, an SET in either the C-element or the weak keeper at the output of an HR cause an error in the output as can be seen from the circuit of the HR shown in Fig. 3. It suggests that j_P and j_S must not refer $D_{i,P}$ in an HR at the same time. To prevent j_P and j_S from referring $D_{i,P}$ at the same time, $D_{i,P}$ is read from the HR before j_P referring the data as the input and it is stored in an RR to be used as the input to j_S. This is illustrated in Fig. 2(c). In minimizing the area cost of registers, the use of RRs as described above must be taken into account.

III. Problem Formulation as MIP model

The problem of minimizing energy consumption and the area cost of registers is formalized as a mixed integer programming (MIP) model. By solving the MIP model, the execution start time, voltage assignment, and the register mode for each operation are determined so that the total area cost of registers is minimized as well as the energy consumption of operations is minimized with respect to the given time constraint and FU configuration constraint. The following variables and parameters are employed. The objective is to minimize the energy consumption in Eq. (18) with respect to the constraints in Eqs. (1)–(17).

The range of possible execution start time of an operation is called a *scheduling range* [14]. The scheduling range is determined by the 'as soon as possible' (ASAP) and the 'as late as possible' (ALAP) schedules. The ASAP schedule is obtained by solving the longest path problem on the DFG describing the given processing algorithm and is known to satisfy all the precedence constraints [15]. The ALAP schedule is obtained similarly as the ASAP schedule [15].

- DFG (N,E): a given processing algorithm is denoted as a data-flow graph (DFG) with the set of nodes N representing operations and the set of edges E representing data dependencies among operations. d_{ij} denotes the delay count on the edge $(i,j) \in E$.
- PP_i: the subset of N and consists of the nodes which share the child (children) with i in the DFG. In other words, for each node j where the edge $(i,j) \in E$, the node k is included in PP_i if an edge $(k,j) \in E$.
- V: the set of available supply voltages.
- F: the set of operation types.
- $T_{i,m}$: the scheduling range of the operation execution i_m.
- $X_{i,m}^{t,v}$: a binary variable and becomes 1 when i_m starts at time t and is assigned the voltage v.
- $MR_{i,m}^{t}$: a binary variable and becomes 1 when an RR is required at time t to store a data produced by i_m. $MR_{i,S}^{t}$ becomes 1 also when the result of i_P is in the HR mode and it is copied and stored in an RR at t to be used as the input data by the secondary of the children of i.
- $MH_{i,P}^{t}$: a binary variable and becomes 1 when an HR is required at time t to store a data produced by i_P.
- WH_i: a binary variable and becomes 1 when the result of i_P is in the HR mode.
- $RegH$, $RegR$: integer variables representing the required number of HRs and RRs, respectively.
- q_i^v: a constant indicating the duration of the operation execution i_P or i_S when it is assigned the voltage v.
- u_i^v: a constant indicating the duration for which an execution of an operation i requires its input data when it is assigned the voltage v.
- K_f^v: the maximum number of FUs of the operation type f and the voltage v.
- N_f: the set of operations of the operation type f.
- L_f^v: a constant indicating the duration for which an execution of an operation occupies an FU of the operation type f and the voltage v.
- Tr: a constant indicating the iteration period of the given processing algorithm.
- $c(t)$: a function to obtain the time class of time t. It is the remainder of the integer division t/Tr.

The following notations are used to simplify the desciption.

$$t_{i,m}^{S} = \sum_{v\in V}\sum_{t\in T_{i,m}} tX_{i,m}^{t,v} \quad (1)$$

$$t_{i,m}^{E} = \sum_{v\in V}\sum_{t\in T_{i,m}} (t+q_i^v-1)X_{i,m}^{t,v} \quad (2)$$

$t_{i,m}^{S}$ is the executing start time of i_m and $t_{i,m}^{E}$ is the last time of executing i_m for $i \in N$ and $m = \{P,S\}$. Let $D_{i,m}$ denote the data produced by the operation execution i_m.

Operation execution

$$\sum_{v\in V}\sum_{t\in T_{i,m}} X_{i,m}^{t,v} = 1 \qquad \forall i \in N, m = \{P,S\} \quad (3)$$

i_m is executed exactly once at some time at one of the voltages.

Constraints on the primary and secondary executions

$$t^S_{i,P} \leq t^S_{i,S} \quad \forall i \in N \tag{4}$$
$$t^E_{i,P} \leq t^E_{i,S} \quad \forall i \in N \tag{5}$$
$$t^E_{i,S}+1 < t^S_{i,P}+p_E \quad \forall i \in N \tag{6}$$

Eq. (4) constrains that i_S starts no earlier than i_P. Eq. (5) constrains that i_S ends no earlier than i_P. The data $D_{i,S}$ becomes available at $t^E_{i,S}+1$ and is compared with the data $D_{i,P}$ at or after $t^E_{i,S}+1$. Eq. (6) constrains that the penalty for replaying $i_{i,P}$ (also $i_{i,S}$ and other operations) is within p_E.

Precedence constraint

$$t^S_{j,P} \geq t^E_{i,P}+1-d_{ij}Tr \quad \forall (i,j) \in E \tag{7}$$

If there exists data dependency from an operation i to an operation j, then the precedence relation $t^S_{j,m} \geq t^E_{i,P}+1-d_{ij}Tr$ must be satisfied for every edge $(i,j) \in E$ and for $m = \{P,S\}$. Since $t^S_{j,S} \geq t^S_{j,P}$ with Eq. (4), the precedence relation between i_P and j_S is automatically satisfied. Thus only Eq. (7) is sufficient for the precedence constraint.

Judge register mode of the primary result

$$t^E_{j,S}+1 < t^S_{i,P}+p_E+B\times WH_i \quad \forall (i,j) \in E \tag{8}$$
$$t^E_{k,S}+1 < t^S_{i,P}+p_E+B\times WH_i \quad \forall i \in N, k \in \mathrm{PP}_i \tag{9}$$

B is a sufficiently large positive constant. Assume that an operation j is a child of an operation i (an edge (i,j) exists in E). The data $D_{j,S}$ becomes available and is compared with the data $D_{j,P}$ at $t^E_{j,S}+1$. If $D_{j,P}$ and $D_{j,S}$ differ, i.e., either j_P or j_S contains an error, j is replayed using $D_{i,P}$ as the input data. To prepare for the replay, $D_{i,P}$ must be held in a register until $t^E_{j,S}+1$. In the case that $t^E_{j,S}+1 \geq t^S_{i,P}+p_E$, WH_i is set to 1 by Eq. (8) to indicate that an HR must be used to hold $D_{i,P}$.

Assume that j is also a child of another operation k ($k \in \mathrm{PP}_i$). There can be the case that $t^E_{k,S}+1 > t^E_{j,S}+1$, that is, $D_{k,P}$ and $D_{k,S}$ are checked after $t^E_{j,S}+1$ as illustrated in Fig. 4. If k_P or k_S contains an error and k is replayed, j is also replayed and $D_{i,P}$ is required as the input data of j. Therefore $D_{i,P}$ must be held in a register until $t^E_{k,S}+1$ to prepare for the replay. Eq. (9) constrains that WH_i is set to 1 to indicate that an HR must be used to hold $D_{i,P}$ when $t^E_{k,S}+1 \geq t^S_{i,P}+p_E$.

Note that in Fig. 4, both j_P and j_S refer $D_{k,P}$ at CC 3 as the input. If an error occurred in the register storing $D_{k,P}$ at CC 3, both j_P and j_S use the erroneous data, but the error cannot be detected by comparing the results of j_P and j_S. $D_{k,S}$ is compared with $D_{k,P}$ and the error in $D_{k,P}$ is detected at CC 5. In that case, j_P and j_S are re-executed properly by the replay.

Register requirement for the primary result

$$\sum_{v\in V}\sum_{\substack{t\in T_{i,P}\\ t<\tau-q^v_i}} X^{t,v}_{i,P} + \sum_{v\in V}\sum_{\substack{t\in T_{j,S}\\ t+q^v_j\geq\tau-d_{ij}Tr}} X^{t,v}_{j,S} \leq MR^\tau_{i,P}+MH^\tau_{i,P}+1 \tag{10}$$

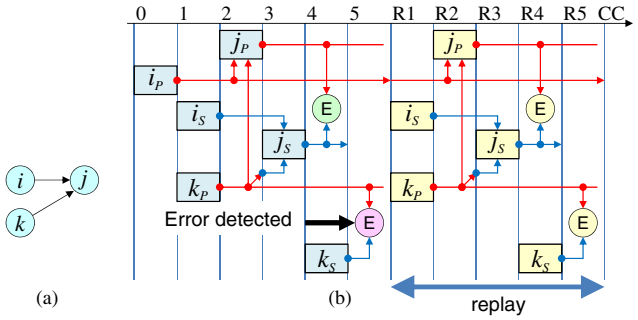

Fig. 4. Register requirement for the primary result. (a) a DFG. (b) $D_{i,P}$ must be stored until $k \in PP_i$ is checked.

$$\sum_{v\in V}\sum_{\substack{t\in T_{i,P}\\ t<\tau-q^v_i}} X^{t,v}_{i,P} + \sum_{v\in V}\sum_{\substack{t\in T_{k,S}\\ t+q^v_k\geq\tau}} X^{t,v}_{k,S} \leq MR^\tau_{i,P}+MH^\tau_{i,P}+1 \tag{11}$$
$$\forall (i,j) \in E, \tau \in OR(i,j)$$
$$\forall k \in PP_i, \tau \in OR(i,k)$$

When $D_{i,P}$ is produced before time τ and held at or after τ (the left hand side of Eq. (10) and/or Eq. (11) becomes 2), a register, eigher RR or HR, is required at τ to store $D_{i,P}$. This is checked for each τ in the range during which $D_{i,P}$ may be stored in a register. The range $OR(i,j)$ begins from the lower bound of $t^E_{i,P}+1$ at which $D_{i,P}$ is produced and ends at the upper bound of $t^E_{j,S}+1$ for $(i,j) \in E$. The range $OR(i,k)$ begins at the same time as $OR(i,j)$ and ends at the upper bound of $t^E_{k,S}+1$ for $k \in PP_i$.

Register requirement for the secondary result

$$\sum_{v\in V} X^{\tau-q^v_i,v}_{i,S} \leq MR^\tau_{i,S} \quad \forall i \in N, \tau-q^v_i \in T_{i,S} \tag{12}$$

$$\sum_{v\in V}\sum_{\substack{t'=0\\ \tau-t'\in T_{i,S}}}^{u^v_i-1} X^{\tau-t',v}_{i,S} \leq MR^\tau_{i,S} \quad \forall i \in N, \tau-q^v_i \in T_{i,S} \tag{13}$$

$$\sum_{v\in V}\sum_{\substack{t\in T_{i,S}\\ t<\tau-q^v_i}} X^{t,v}_{i,S} + \sum_{v\in V}\sum_{\substack{t\in T_{j,S}\\ t+u^v_j-1\geq\tau-d_{ij}Tr}} X^{t,v}_{j,S} \leq MR^\tau_{i,S}+WH_i+1$$
$$\forall (i,j) \in E, \tau \in OR(i,j) \tag{14}$$

When i_S is completed, $D_{i,S}$ is stored in an RR to be compared with $D_{i,P}$. Thus an RR is used at time τ if i_S starts at $\tau-q^v_i$ with the voltage v. This is constrained by Eq. (12). If an operation j is a child of i ($(i,j) \in E$ exists), executing j_S requires the result of operation i as the input and it is stored in an RR which is different from the register holding $D_{i,P}$. Hence an RR is necessary from the start of j_S for the duration u^v_j. This is constrained by by Eq. (13).

In the case that i_P is in HR mode, the input data of j_S mentioned above is copied from an HR holding $D_{i,P}$ just before executing j_S. In the case that i_P is in RR mode, on the other hand, $D_{i,S}$ produced by i_S is stored in an RR and held there until it is used by j_S. Eq. (14) constrains that, if i_P is in RR mode ($\mathrm{WH}_i = 0$), an RR is used to store $D_{i,S}$ from the time it is produced until the time it is last used.

Register count

$$RegH \geq \sum_{i \in N} \sum_{\substack{t \\ c(t)=tc}} MH^t_{i,P} \quad (15)$$

$$RegR \geq \sum_{i \in N} \sum_{\substack{t \\ c(t)=tc}} MR^t_{i,P} + MR^t_{i,S} \quad (16)$$

$$\forall tc = \{0, 1, \ldots, Tr-1\}$$

The required number of registers is obtained as the largest register usage among all the time classes.

FU constraint

$$\sum_{i \in N_f} \sum_{m=\{P,S\}} \sum_{t'=0}^{L^v_f - 1} \sum_{\substack{t-t' \in T_{i,m} \\ c(t-t')=tc}} X^{t-t',v}_{i,m} \leq K^v_f \quad (17)$$

$$\forall tc = \{0, 1, \ldots, Tr-1\}, f \in F, v \in V$$

For each time class tc, operation type f, and voltage v, the number of operations executed simultaneously in tc should not exceed the specified constraint K^v_f.

Objective The objective is to minimize the *Cost* calculated as follows.

$$Cost = \sum_{f \in F} \sum_{v \in V} ECQ^v_f \sum_{i \in N_f} \sum_{m=\{P,S\}} \sum_{t \in T_{i,m}} X^{t,v}_{i,m} + \alpha \left(ER^H RegH + ER^R RegR \right) \quad (18)$$

ECQ^v_f, ER^H, and ER^R are respectively the energy consumption of one execution of an operation of type f at voltage v, the area cost of an HR, and the area cost of an RR. With a sufficiently small positive weighting factor α, the area cost of registers is minimized as long as the energy consumption of operation executions is minimized.

IV. Experimental Results

An MIP solver IBM ILOG CPLEX 12.6.0.0 [16] was used to solve MIP models. MIP models were generated by a program implemented using C++ programming language. The CPU time for model generation is less than 1 s for each model. All the experiments were done on a PC with a 2.2 GHz microprocessor running 8 threads on 4 physical cores and 8 GB of main memory. The processing algorithms used were the 5th order wave elliptic filter (WEF) [14] consisting of 8 multiplications and 26 additions, the WEF unfolded [18] by factor 3 (WEF3, 24 multiplications and 78 additions), and 8-point 1D DCT [17] (DCT) consisting of 11 multiplications and 29 additions.

Assuming a 0.18μm CMOS process as the target, the characteristics of the 16-bit functional units are summarized in Table I. In the experiments, only two levels of supply voltages were employed. The higher voltage is 1.8 V and the lower voltage is 1.2 V. Shown in Table I are from left to right, the name of the functional unit, the operation type (f), the power supply voltage, the operation duration in clock cycle (q), the duration for which an execution of the operation occupy an FU (L), and the average energy consumption in pJ. The operation duration of the addition AH is 1 clock cycle. The addition at the lower voltage requires 2 clock cycles to execute and occupies an FU for 2 clock cycles. The multipliers are pipelined. The operation duration of MH is 2 clock cycles but the same FU is used for the next execution after 1 clock cycle. The multiplication ML is pipelined into 3 clock cycles. Thus the operation duration is 3 clock cycles and an FU (a multiplier at the lower voltage) is occupied for 1 clock cycle.

TABLE I
Specification of Resources

ID	f	Vdd [V]	q	L	ECQ [pJ]
AH	addition	1.8(H)	1	1	3.9626
AL	addition	1.2(L)	2	2	1.7611
MH	multiplication	1.8(H)	2	1	44.949
ML	multiplication	1.2(L)	3	1	19.977

TABLE II
Results for WEF

DFG	T	FU	p_E	HR	RR	RC	EC	CPU
WEF	16	4,2,2,0	3	11	12	−3.1%	967.4	<1
		4,2,2,0	3	13	8			<1
	22	3,2,2,0	3	9	10	−12.8%	950.3	17
		3,2,2,0	3	12	7			9
	28	2,2,2,0	3	10	7	−4.8%	936.5	232
		2,2,2,0	3	11	6			199
WEF	16	4,2,1,1	4	9	13	−20.7%	593.3	<1
		4,2,1,1	4	14	8			<1
	22	3,2,1,1	4	13	11	−9.9%	602.3	29
		3,2,1,1	4	16	8			20
	28	2,2,1,1	4	11	8	−10.4%	634.8	(180)
		2,2,1,1	4	13	7			(179)
WEF	16	4,2,1,1	5	9	13	−21.3%	590.1	<1
		4,2,1,1	5	15	7			<1
	22	3,2,1,1	5	10	10	−15.4%	509.3	21
		3,2,1,1	5	14	6			10
	28	2,2,1,1	5	10	8	−12.6%	487.5	168
		2,2,1,1	5	13	5			38

The values of ER^H, ER^R, and α in Eq. (18) were set to 2.67, 1.00, and 0.01, respectively, in the experiments.

The results by solving the MIP models are summarized in Tables II, III, and IV. The tables show the processing algorithm, the iteration period Tr specified as the time constraint, the resource constraint for FUs, the limit of replay delay penalty p_E, the required numbers of HRs and RRs, comparison of the area cost of registers, the minimized energy consumption in pJ, and the CPU time in second. The resource constraint for FUs indicates the specified limit of the numbers of AH, AL, MH, and ML. For each combination of DFG and Tr, the upper row is the result for optimized selection between HR and RR modes, and the lower row is the result where only HR mode is used. RC shows the reduction of the area cost of registers by the register mode selection.

In the case of $p_E = 3$, ML cannot be used because $q = 3$ for ML and using ML always cause the replay delay penalty larger than p_E. As p_E gets larger, more operations are in RR mode

TABLE III
RESULTS BY WEF3

DFG	Tr	FU	p_E	HR	RR	RC	EC	CPU
WEF3	48	4,2,2,0	3	5	13	−17.7%	2892	< 1
		4,2,2,0	3	9	8			< 1
	51	3,3,2,0	3	7	9	−13.5%	2877	350
		3,3,2,0	3	9	8			152
WEF3	48	4,2,1,1	4	3	15	−30.3%	1771	2
		4,2,1,1	4	9	9			< 1
	51	3,3,1,1	4	6	13	−20.9%	1678	19
		3,3,1,1	4	10	10			8
WEF3	48	4,2,1,1	5	4	14	−26.8%	1762	5
		4,2,1,1	5	10	7			1
	51	3,3,1,1	5	5	14	−28.7%	1595	54
		3,3,1,1	5	11	9			17

TABLE IV
RESULTS FOR DCT

DFG	Tr	FU	p_E	HR	RR	RC	EC	CPU
DCT	8	6,5,4,0	3	6	21	−15.9%	1267	< 1
		6,5,4,0	3	12	12			< 1
	12	4,4,2,0	3	4	15	−16.3%	1236	26
		4,4,2,0	3	7	12			13
	14	3,3,2,0	3	5	10	−20.4%	1223	87
		3,3,2,0	3	8	8			1610
DCT	8	6,5,3,1	4	5	19	−28.1%	993.3	< 1
		6,5,3,1	4	12	13			< 1
	12	4,4,1,2	4	3	13	−26.7%	620.9	53
		4,4,1,2	4	7	10			45
	14	3,3,2,1	4	4	13	−20.2%	633.1	159
		3,3,2,1	4	7	11			46
DCT	8	6,5,3,1	5	3	25	−29.8%	940.5	< 1
		6,5,3,1	5	12	15			< 1
	12	4,4,2,2	5	4	15	−24.5%	557.0	14
		4,4,2,2	5	9	10			7
	14	3,3,2,1	5	5	13	−22.5%	592.0	35
		3,3,2,1	5	9	10			18

and the requirement for using HRs is reduced. Hence the area cost of registers becomes smaller for larger p_E. The results show that the area cost of registers can be minimized without degrading the minimization of energy consumption with respect to the given time and resource constraints.

V. CONCLUSIONS

In this paper, a method to minimize the area cost of registers is proposed while the minimization of energy consumption is considered with respect to the give constraints of time, resource, and delay penalty for error correction. The area cost of registers is reduced by about 20% on average without degrading the minimization of energy consumption.

Considering the energy consumption of registers as well as other components such as multiplexors and voltage level converters, and developing a heuristic algorithm for the problem remain as future work.

REFERENCES

[1] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," IEEE Trans. Nucl. Sci., vol.47, no.6, pp.2586–2594, 2000.

[2] P. Hazucha and C. Svensson, "Cosmic ray neutron multiple-upset measurements in a 0.6-μm CMOS process," IEEE Trans. Nucl. Sci., vol.47, no.6, pp.2995–2602, 2000.

[3] J.A. Maestro and P. Reviriego, "Study of the effects of MBUs on the reliability of a 150 nm SRAM device," Proc. DAC 2008, pp.930–935, 2008.

[4] M.P. Baze, S.P. Buchner, and D. McMorrow, "A digital CMOS design technique for SEU hardening," IEEE Trans. Nucl. Sci., vol.47, pp.2603–2608, 2000.

[5] R. Garg and N. Jayakumar, "A design approach for radiation-hard digital electronics," Proc. DAC 2006, pp.773–778, 2006.

[6] R. Garg, C. Nagpal, and S.P. Khatri, "A fast, analytical estimator for the SEU-induced pulse width in combinational designs," Proc. DAC 2008, pp.918–923, 2008.

[7] S.W. Kwak and B.K. Kim, "Task-scheduling strategies for reliable TMR controllers using task grouping and assignment," IEEE Trans. Reliability, vol.49, pp.355–362, 2000.

[8] B.J. LaMeres and C. Gauer, "A power-efficient design approach to radiation hardened digital circuitry using dynamically selectable triple modulo redundancy," Military and Aerospace Programmable Logic Devices (MAPLD) Conference, pp.1–5, 2008.

[9] S. Golshan and E. Bozorgzadeh, "SEU-aware resource binding for modular redundancy based designs on FPGAs," Proc. DATE 2009, pp.1124–119, 2009.

[10] D. Tsuruta, M. Wakizaka, Y. Hara-Azumi, and S. Yamashita, "A TMR-based soft error mitigation technique with less area overhead in high-level synthesis," Proc. SASIMI 2012, pp.396–401, 2012.

[11] S. Matsuzaka, K. Inoue, "A Dependable Processor Architecture with Data-Path Partitioning," IPSJ Tech. Report, vol. 2004-SLDM-117, pp. 7–11, 2004.

[12] S. Mitra, M. Zhang, S. Waqas, N. Seifert, B. Gill, and K. S. Kim, "Combinational Logic Soft Error Correction," Proc. IEEE Int. Test Conf., pp. 824–832, 2006.

[13] Y. Suda and K. Ito, "A Method of Power Supply Voltage Assignment and Scheduling of Operations to Reduce Energy Consumption of Error Detectable Computations," Proc. SASIMI 2012.

[14] S.M. Heemstra de Groot, S.H. Gerez, and O.E. Herrmann, "Range-chart-guided iterative data-flow graph scheduling," IEEE Trans. Circuits Syst.-I: Fund. Theory & Appl., vol.39, pp.351–364, 1992.

[15] G. D. Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, New York, 1994.

[16] IBM ILOG CPLEX, http://www.ilog.com/.

[17] C. Loeffler, A. Ligtenberg, and G.S. Moschytz, "Practical fast 1-d dct algorithms with 11 multiplications," Proc. IEEE ICASSP '89, pp.988–991, 1989.

[18] K.K. Parhi and D.G. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," IEEE Trans. Computers, vol.40, pp.178–195, 1991.