# Minimization of Energy Consumption of Double Modular Redundancy Design of Conditional Processing by Common Condition Dependency

Kazuhito Ito

Graduate School of Science and Engineering
Saitama University
Saitama 338-8570, Japan
kazuhito@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 error is corrected by executing necessary operations again. The DMR design for conditional processing is considered in this work. A method is proposed which makes the secondary executions of the duplicated operations be dependent on the primary execution of the condition operation, thereby widening the schedule solution space and allowing better results to be derived. The minimization of energy consumption with the proposed method is formulated as ILP models and the optimum solution is obtained by using an ILP solver.**

## I. Introduction

As large-scale integrated circuits (LSI) chips integrate more transistors and other components, and the operating power supply voltage decreases, LSI chips are becoming more vulnerable to the soft error caused by an incident of a neutron induced from cosmic rays and so on [1, 2].

As a countermeasure against soft error, the redundancy technique is examined [3, 4, 5]. Double modular redundancy (DMR) executes the same operations in double, the obtained data are respectively retained in registers or memories, and the data are compared. When the data are not identical, a soft error is detected, and the error is corrected by re-executing necessary operations. DMR requires less resources such as functional units (FUs) and registers and consumes less energy for processing execution than triple modular redundancy (TMR), Thus DMR is more attractive in design optimization for redundant systems, and some design automation approaches have been proposed [6, 7].

In conditional processing, an operation to be executed later is selected depending on the execution result of a certain operation. While there exists non-conditional processing, such as digital filters and FFT, many digital systems consist of general conditional processing.

In conditional processing, two or more operations are exclusively executed according to the truth or falsity of a condition, and those operations can share a single FU at the same time. In addition, the processing execution time can be shortened by speculative execution in which an operation depending on the condition is executed before the truth or falsity of the condition is proven [8, 9]. As described above, it is important to optimize the implementation of conditional processing by operation scheduling which determines the execution time of the operation.

The DMR is effective for the conditional processing, but the DMR technique for the conditional processing known so far simply duplicates the non-redundancy processing schedule.

In this paper, we propose a method to obtain an optimum solution for minimizing energy consumption of DMR conditional processing design with the constraints of execution time and resource usage. The optimization problem is formulated as an integer linear programming (ILP) model and is solve by an ILP solver.

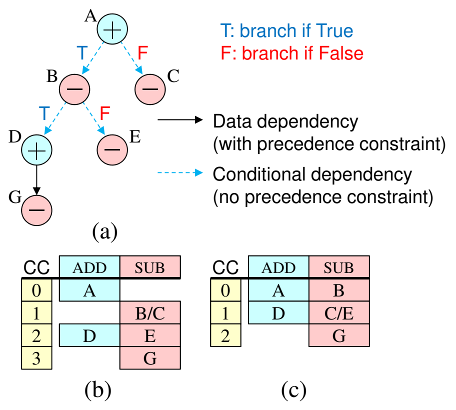
The remainder of the paper is organized as follows. Conditional processing and DMR are briefly introduced in Sect. 2. The proposed DMR conditional processing design method and its ILP formulation is presented in Sect. 3. Experimental results are shown in Sect. 4 and Sect. 5 concludes the work.

## II. Conditional Processing and DMR

Conditional processing and DMR are briefly described.

### A. *Conditional Processing*

In conditional processing, according to the execution of an operation $Z$, some other operations are divided into two groups: those executed when $Z$ is true and those executed when $Z$ is false. Let such an operation $Z$ be called a *condition judgment operation.* Examples of condition judgment operation are the comparison result of two values, the sign (positive or negative) of the result of addition or subtraction, the existence of overflow after addition. Let an operation which is executed only when some condition judgment operation results in true (or false) be called a *condition-dependent operation.* Conditional dependencies and data dependencies between operations are described by a control data flow graph (CDFG). In the conditional processing shown in Fig. 1(a), condition-dependent operations {B, D, E, G} are performed when condition judgment operation A is true, and condition-dependent operation C is performed when A is false. The operations A and D are additions, and B, C, E and G are subtractions. Assuming the execution time of each operation is one clock cycle (CC), the


A
T
F
T: branch if True
F: branch if False
B
C
T
F
D
E
G
Data dependency (with precedence constraint)
Conditional dependency (no precedence constraint)
(a)
(b)
(c)

| CC | ADD | SUB |
|---|---|---|
| 0 | A | |
| 1 | | B/C |
| 2 | D | E |
| 3 | | G |

| CC | ADD | SUB |
|---|---|---|
| 0 | A | B |
| 1 | D | C/E |
| 2 | | G |

Fig. 1. An example conditional operation. (a) CDFG. (b) conditional FU sharing. (c) speculative execution.

execution schedule for one adder and one subtractor is shown in Fig. 1(b). Condition judgment operation A is executed at CC 0. Condition-dependent operation B is executed when A is proven to be true, and condition-dependent operation C is executed exclusively when A is false, thus B and C can share a subtractor and are scheduled at identical CC 1. This situation is said the same FU is *conditionally shared* [8]. B is now a condition judgement operation for {D, E, G} and D and G are executed if B is true or E is executed if B is false. The total processing execution time is 4 CCs.

The execution of condition-dependent operation needs not wait for the completion of the corresponding condition judgment operation. Executing condition-dependent operation before or at the same time as the corresponding condition judgment operation is called *speculative operation* [8]. For example, as shown in Fig. 1(c), the condition judgment operation A is executed at CC 0 and its condition-dependent operation B is speculatively executed at the same CC 0. At CC 1, C and E, which are conditionally dependent on A, conditionally share a subtractor. The use of speculative execution reduces the total execution time to 3 CCs.

While the speculative execution increases the degree of freedom in scheduling the execution time of condition-dependent operations, there is a demerit that more energy is consumed by the speculatively executed operations which are originally unnecessary.

## B. *Double Modular Redundancy*

The soft error in LSIs are modeled as follows. A soft error is assumed to occur only in one of the data stored in a register or the execution of the operation in an FU in the same CC, and there is a sufficiently long time between successive errors. In general, an error in a combinational circuit including an FU continues for more than one CCs. The error duration is set to 1 CC for the sake of simplicity in this work.

DMR duplicates the operation execution and data storage, and detects errors by comparison. Figure 2(b) shows an example in which the operation execution and data retention of the processing shown in Fig. 2(a) are duplicated. Duplicated operation executions are respectively called *primary execution* and *secondary execution*. The primary execution of operation A is denoted as $A_P$, and the secondary execution as $A_S$. In Fig. 2(b), $A_P$ is executed at CC 0 and $A_S$ is executed at CC 1, and the execution results are stored in registers. At CC 2, the data are read from the registers and the values are compared by the comparison operation $A_E$.

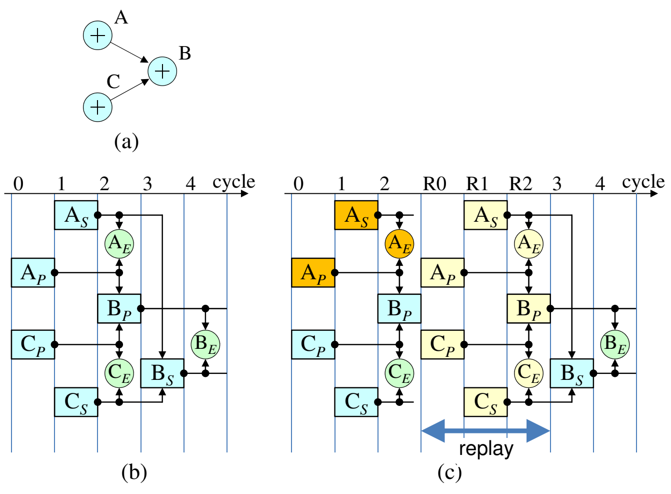


Fig. 2. DMR design. (a) DFG. (b) DMR scheduling of operations. (c) re-execution to correct an error (replay).

If the $A_E$ detects a value mismatch, it means that there is an error in either $A_P$, $A_S$, or the resultant data of these executions stored in registers. The error is corrected by executing all the operations suspected to be an error and storing the results again. This is called the *replay*. Figure 2(c) shows the replay when $A_E$ detected an error. The re-execution of operations is performed from CC R0 to R2, and the execution of the processing is delayed by this time duration. This is called a *delay penalty* associated with re-execution, and in the example of Fig. 2(c), the delay penalty is 3 CCs. When the error detection comparison is performed immediately after the secondary execution, the difference in execution time between the primary and the secondary executions causes the delay penalty. If a large delay penalty is allowed, it is possible to execute the primary and the secondary of the operation at different CCs, thereby increasing the degree of freedom of the scheduling and a good operation schedule may be obtained. On the other hand, it would be necessary to set an appropriate upper limit of the delay penalty, for example in real time processing.

## III. Considerations in DMR design of Conditional Processing

### A. *DMR for Conditional Processing*

In DMR, data dependency is closed in each of the primary executions and the secondary executions of the operations. That is, when operation B uses the operation result of operation A as input data, $B_P$ depends on $A_P$ and $B_S$ depends on $A_S$. This is because, when $B_P$ and $B_S$ are executed using the same input data, $B_E$ reports the coincidence even if there is an error in the data, thereby overlooking the error. On the other

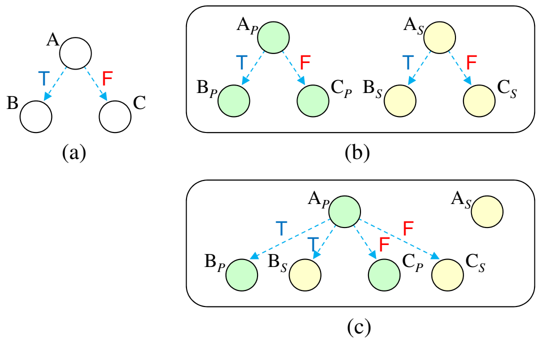
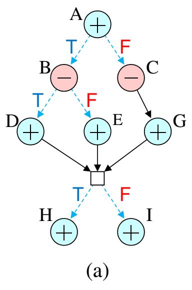


Fig. 3. DMR for conditional operation. (a) CDFG. (b) simple double execution. (c) operations depend on the primary execution of condition.



| CC | ADD0 | ADD1 | ADD2 | SUB0 | SUB1 |
|---|---|---|---|---|---|
| 0 | $A_P$ | $A_S$ | $I_P$ | | |
| 1 | $H_P$ | $H_S$ | $I_S$ | $B_P/C_P$ | $B_S/C_S$ |
| 2 | $D_P/E_P$ | $G_P$ | $D_S/E_S/G_S$ | | |

(b)

| CC | ADD0 | ADD1 | ADD2 | SUB0 | SUB1 |
|---|---|---|---|---|---|
| 0 | $A_P$ | | | $C_P$ | |
| 1 | $A_S$ | $D_P/G_P$ | $E_P$ | $B_P$ | $C_S$ |
| 2 | $H_P/I_P$ | $H_S/I_S$ | $D_S/E_S/G_S$ | $B_S$ | |

(c)

Fig. 4. DMR schedule of conditional processing. (a) CDFG. (b) conditional dependency in the primary and the secondary separately. (c) operations depend on the primary execution of condition.

hand, for condition dependency, the secondary execution of condition-dependent operations may conditionally depend on the primary execution of condition judgment operations. This is the proposal of this work.

For example, for the process shown in Fig. 3(a), the result of simple DMR design is shown in Fig. 3(b), where the primary executions $B_P$ and $C_P$ are conditionally dependent on $A_P$, and the secondary executions $B_S$ and $C_S$ are conditionally dependent on $A_S$. However, as shown in Fig. 3(c), $B_P$, $C_P$, $B_S$ and $C_S$ can be conditionally dependent on the primary execution $A_P$. To detect an error in $A_P$, the secondary $A_S$ is executed and the results are compared. If an error is detected, $A_P$ and operations conditionally dependent on A are re-executed.

The effectiveness of the proposed method is presented using a simple example. When the conditional processing shown in Fig. 4(a) is duplicated and scheduled in 3 CCs with 3 adders and 2 subtractors, if the condition dependencies are separated to the primary and the secondary, the resultant schedule to achieve the minimum energy consumption is as shown in Fig. 4(b). Here it is assumed that true and false branching probabilities of the conditions are 0.5 for true and 0.5 for false in all condition judgments, and that the energy consumption of an operation execution is 1 unit of energy (u.e.) for addition and subtraction. $B_P$ and $C_P$, and $B_S$ and $C_S$ respectively share subtractors conditionally, and the execution probabilities (EPs) of these operations are 0.5. To execute all the operations within 3 CCs, $H_P$, $I_P$, $H_S$, and $I_S$ must be executed speculatively. Therefore the EPs of $A_P$, $A_S$, $H_P$, $I_P$, and so on are 1. The total probabilitistic energy consumption of the schedule is 10 u.e.

Figure 4(c) shows the schedule with the minimized energy consumption by the proposed method. Since all the condition judgement operations $D_P$, $E_P$, and $G_P$ have been executed before CC 2, the condition-dependent operations $H_P$ and $I_P$, and $H_S$ and $I_S$ respectively can share adders conditionally at CC 2, and the EPs of $H_P$, $I_P$, $H_S$, and $I_S$ are 0.5. The total probabilistic energy consumption of the schedule is only 9 u.e.

The example shows the fact that the proposed method increases the degree of freedom in scheduling conditional processing. The proposed method is utilized for minimizing the probabilitistic energy consumption of DMR conditional processing.

## B. *ILP Formulation for Energy Minimization*

The operation scheduling problem is NP-hard, and no heuristic method for finding the optimum solution has been obtained. Optimum operation scheduling problem of DMR conditional processing becomes an advanced combination optimization problem. In order to know the optimum solution, this combined optimization problem is formulated as an integer linear programming problem (ILP), and the optimum solution is obtained by an ILP solution tool (an ILP solver). Since the solution time of the ILP solver generally depends exponentially on the number of variables in the ILP model, it is important to reduce the number of variables required in the ILP formulation.

The following ILP variables and parameters are used to describe the ILP model. With the constraints in Eqs. (1) – (16), the total operation execution energy consumption shown in Eq. (17) is minimized.

Note that an S range of an operation is the set of CCs at which the operation can be executed. It is obtained as the interval between the earliest and the latest possible execution CCs of each operation execution determined by considering the precedence constraint between operations derived from data dependency.

- DFG $(N, E, B)$: a given processing algorithm is denoted as a data-flow graph (DFG) where $N$ is the set of nodes representing operations, $E$ the set of edges representing data dependencies among operations, and $B$ the set of edges representing conditional dependencies among operations.
- $F$: the set of operation types.
- $p_D$: a constant. The upper bound of the delay penalty.
- $R_{i,m}$: S range of the execution of operation $i_m$. $(m \in \{P, S\})$
- $x^t_{i,m}$: a binary variable and becomes 1 when $i_m$ starts at time $t$. $(m \in \{P, S\})$
- $u^t_{i,m}$: a binary variable and becomes 1 when the execution of $i_m$ requires an FU at time $t$. $(m \in \{P, S\})$
- $s_{i,m1,j,m2}$: a binary variable and becomes 1 when the executions of $i_{m1}$ and $j_{m2}$ conditionally share an FU.
- $b_{i,m,c,mc}$: a binary variable and becomes 1 when $i_m$ is executed speculatively with respect to the condition $c_{mc}$.

- $y^p_{i,m}$: a binary variable and becomes 1 when $i_m$ is executed in the speculative execution pattern $p$.
- $NC$: the set of condition judgment operation nodes. $(NC \subset N)$
- $NT_c$ ($NF_c$): a set of operations which are executed if the condition $c$ is True (False).
- $SE_i$: a set of speculative execution patters of operations $i$.
- $SE^c_i$: a subset of $SE_i$ when $i$ is speculatively executed with respect to condition $c$.
- $Q_i$: a constant. The operation execution duration of operations $i_P$ and $i_S$.
- $K_f$: a constant. The maximum number of FUs of the operation type $f$.
- $L_f$: a constant. The duration for which an execution of an operation occupies an FU of the operation type $f$.
- $E^p_{i,m}$: a constant. The energy consumed when $i_m$ ($m \in \{P,S\}$) is executed in the speculative execution pattern $p$.

**Operation execution:**

Every operation $i_m$ is executed exactly once at some time $t$ within its S range.

$$\sum_{t \in R_{i,m}} x^t_{i,m} = 1 \qquad \forall i \in N, m \in \{P,S\} \tag{1}$$

$T_{i,m}$, the execution start time of operation $i_m$, is given by the following equation.

$$T_{i,m} = \sum_{t \in R_{i,m}} t x^t_{i,m}$$

**Precedence constraint:**

If there exists a data dependency from an operation $i$ to an operation $j$, the execution of $i$ must be completed before $j$ starts. This precedence is constrained for every edge $(i,j) \in E$ and for primary operations and secondary operations, respectively.

$$T_{j,m} \geq T_{i,m} + Q_i \qquad \forall (i,j) \in E, m \in \{P,S\} \tag{2}$$

**Constraint on the primary and secondary executions:**

Eq. (3) constrains that $i_S$ starts no earlier than $i_P$.

$$T_{i,S} \geq T_{i,P} \qquad \forall i \in N \tag{3}$$

The duration from the start of the primary $i_P$ $(T_{i,P})$ to the end of the secondary $i_S$ $(T_{i,S} + Q_i)$ is constrained to be within $p_D - 1$.

$$T_{i,S} + Q_i \leq T_{i,P} + p_D - 1 \qquad \forall i \in N \tag{4}$$

**Conditional FU sharing:**

**(a)** If an operation $i$ exists on the True side and an operation $j$ exists on the False side of a common condition operation $c$, $i_{m1}$ and $j_{m2}$ can share an FU if these are executed at the same time.

$$s_{i,m1,j,m2} \leq 1 + x^t_{i,m1} - x^t_{j,m2} \tag{5}$$
$$\forall c \in NC, i \in NT_c, j \in NF_c, m1, m2 \in \{P,S\}$$

When $j_{m2}$ is executed at $t$ ($x^t_{j,m2} = 1$), the right hand side of Eq. (5) becomes 0 if $i_{m1}$ is not executed at the same time $t$ ($x^t_{i,m1} = 0$), and $s_{i,m1,j,m2} = 0$ is constrained.

**(b)** An FU is not shared by operations $i$ and $j$ if $j$ is speculatively executed with respect to the condition operation $c$.

$$s_{i,m1,j,m2} + x^t_{j,m2} + \sum_{\tau \geq t} x^\tau_{c,mc} \leq 2 \tag{6}$$
$$\forall c \in NC, i \in NT_c, j \in NF_c, m1, m2 \in \{P,S\}$$

If $j_{m2}$ is executed before the completion of the condition operation $c_{mc}$, the sum of the 2nd and 3rd terms of the left hand side of Eq. (6) becomes 2 and $s_{i,m1,j,m2} = 0$ is constrained.

**(c)** More than one operations on the False side cannot share an FU with an identical execution $i_{m1}$ on the True side.

$$\sum_{j \in NF_c} \sum_{m2 = \{P,S\}} s_{i,m1,j,m2} \leq 1 \tag{7}$$
$$\forall c \in NC, i \in NT_c, m1 \in \{P,S\}$$

**(d)** If $j_{m3}$ is speculatively executed with respect to a condition operation $c$, then for any operation $p_{m1}$, it is not allowed to share an FU by $p_{m1}$, $j_{m3}$, and the operation $i_{m2}$ which depends on the same condition $c$.

$$s_{p,mp,i,m1} + s_{p,mp,i,m2} + \sum_{\tau < t} x^\tau_{j,m3} + \sum_{\tau \geq t} x^\tau_{c,mc} \leq 3 \tag{8}$$
$$\forall p, i, j, c \in N, n \in NF_j, m1, m2, m3, mc \in \{P,S\}$$

By assuming $p_{m1}$ and $i_{m_2}$ share an FU, and $p_{m1}$ and $j_{m_3}$ share an FU, and $j_{m3}$ is speculatively executed with respect to a condition operation $c$, the left hand side of Eq. (8) becomes 4, but in that case, $i_{m2}$ and $j_{m3}$ cannot share an FU.

**(e)** If an operation $j$ is conditionally dependent on an operation $c$, they cannot share an FU.

$$s_{p,mp,j,m1} + s_{p,mp,c,mc} \leq 1 \tag{9}$$
$$\forall p, j, c \in N, mp, m1, mc \in \{P,S\}$$

**FU requirement:**

If an operation does not share an FU with other operations, a dedicated FU is required to execute the operation.

$$u^t_{j,m2} + \sum_{i,m1} s_{i,m1,j,m2} \geq x^t_{j,m2} \tag{10}$$
$$\forall c \in NC, j \in NF_c, m2 \in \{P,S\}$$

When $j_{m2}$ does not share an FU with other operation executions, the 2nd term of the left hand side becomes 0 and it constrains $u^t_{j,m2} = 1$ at the execution time $t$ of $j_{m2}$.

$$s_{i,m1,j,m2} \leq \sum_t u^t_{i,m1} \tag{11}$$
$$\forall i \in N, c \in NC, j \in NF_c, m1, m2 \in \{P,S\}$$

When $j_{m2}$ shares an FU with $i_{m1}$, an FU must be allocated to $i_{m1}$, and it constrains $u^t_{i,m1} = 1$ at some execution time $t$.

**Judgement for speculative execution:**

Judge whether $i_{m1}$ is speculatively executed with respect to the condition operation $c_{mc}$.

$$x^t_{i,m1} + \sum_{\tau \geq t} x^\tau_{c,mc} \leq 1 + b_{i,m1,c,mc} \tag{12}$$
$$\forall c \in NC, i \in NT_c \cup NF_c, m1 \in \{P,S\}$$

Let $mc = m1$ in the case of the conventional DMR, and $mc = P$ in the case of the proposed primary condition dependence.

**Identifying speculative execution pattern:**

Identify the combination pattern of whether $i_{m1}$ is speculatively executed with respect to the condition operations.

$$\sum_{p \in SE^c_i} y^p_{i,m} \geq b_{i,m,c,mc} \tag{13}$$
$$\forall c \in NC, i \in N, m \in \{P,S\}$$

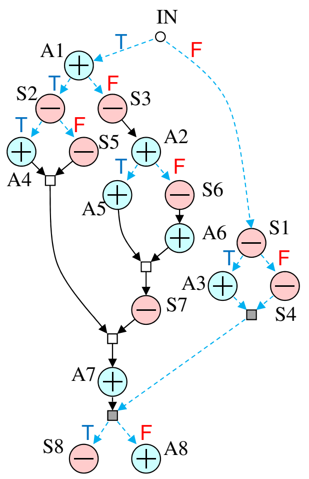


Fig. 5. Conditional operation MAHA

$$\sum_{p \in SE_i} y^p_{i,m} = 1 \qquad \forall i \in N, m \in \{P,S\} \qquad (14)$$

For example, if operation $i$ conditionally depends on three conditions, there exist 8 patterns of the combination of whether $i$ is speculatively executed or not with respect to each conditions. Eq. (14) constrains that only one of 8 patterns is identified.

**FU constraint:**

For each time $t$ and operation type $f$, the number of operations executed simultaneously at $t$ should not exceed the specified constraint $K_f$.

$$\sum_{i \in N_f} \sum_{m=\{P,S\}} \sum_{t'=0}^{L_f-1} \sum_{t-t' \in R_{i,m}} u^{t-t'}_{i,m} \leq K_f \qquad (15)$$
$$t = 0, 1, \ldots, Tr-1, \forall f \in F$$

**Execution time constraint:**

Every operation execution must be done by the specified time $TE$.

$$T_{i,m} + Q_i \leq TE \qquad \forall i \in N, m \in \{P,S\} \qquad (16)$$

**Objective:**

The objective is to minimize the total probabilistic energy consumption $C$ given as follows.

$$C = \sum_{i \in N} \sum_{m=\{P,S\}} \sum_{p \in SE_i} E^p_{i,m} y^p_{i,m} \qquad (17)$$

## IV. Experimental Results

An ILP solver IBM ILOG CPLEX 12.6.0.0 [10] was used to solve ILP models. ILP 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 3.4 GHz microprocessor running 8 threads on 4 physical cores and 16 GB of main memory. The conditional processing algorithms used were MAHA (8 additions, 8 subtractions) [12] shown in Fig. 5, and FIG17P (15 additions, 12 subtractions (including 4 comparisons), and 14 multiplications), which is a part of FIG17 [11], shown in Fig. 6.

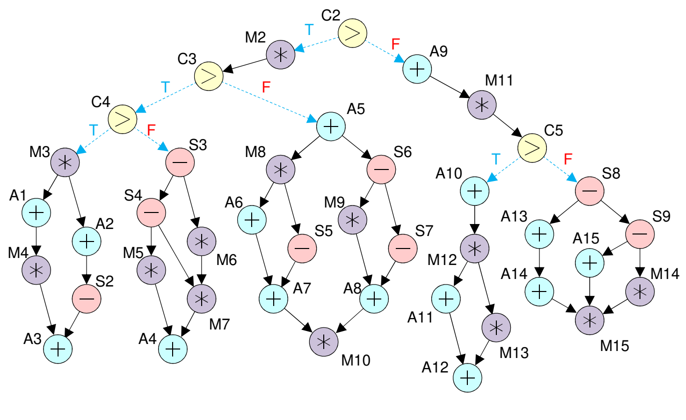


Fig. 6. Conditional operation FIG17P

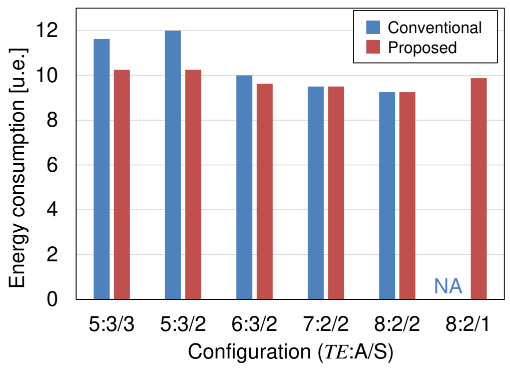


Fig. 7. Minimized total energy consumption of MAHA ($p_D = 3$).

The operation execution time is 1 CC for addition and subtraction and 2 CCs for multiplication. The energy consumption of the operation execution is 1 unit of energy (u.e.) for addition and subtraction and 10 u.e. for multiplication. The true and false branching probabilities of the conditions were 0.5 for true and 0.5 for false in all condition judgments.

All the ILP solver runs successfully completed with the optimum solutions or no results when the ILP models were infeasible because of the constraints are too tight to be solved.

Figure 7 shows the results for MAHA for $p_D = 3$. The configuration $TE$:A/S indicates the given constraints of the total execution CCs $TE$ and the numbers of adders and subtractors. For example, '5:3/2' indicates that every operation execution completes by CC 4($= 5-1$), and 3 adders and 2 subtractors are used. While smaller energy consumption is obtained by the proposed method in many configurations, there is no difference between the result of the conventional and the proposed methods in the cases of 7:2/2 and 8:2/2. It is important to note that the proposed method can derive a feasible DMR design for 8:2/1, with that configuration the conventional method cannot derive a solution.

Some of the obtained operation schedules are shown in Figs. 8 and 9. From the results obtained by the proposed method, it can be seen that the primary execution of condition

TABLE I
ENERGY CONSUMPTION OF FIG17P

| | | $E$ ($p_D=3$) | | $E$ ($p_D=5$) | |
|---|---|---|---|---|---|
| $TE$ | A/S/M | conv. | proposed | conv. | proposed |
| 7 | 4/4/4 | 124.0 | 124.0 (−0.0%) | 124.0 | 118.5 (−4.4%) |
| 7 | 4/3/4 | 152.5 | 134.25(−12.0%) | 152.5 | 118.75(−22.1%) |
| 8 | 3/3/3 | — | **115.25** | 108.75 | 104.0 (−4.4%) |
| 8 | 2/3/3 | — | — | — | **110.25** |
| 8 | 4/3/2 | 117.125 | 109.5 (−6.5%) | 114.0 | 108.75 (−4.6%) |
| 8 | 2/4/2 | — | — | — | **117.25** |
| 9 | 2/2/4 | 101.75 | 101.5 (−0.2%) | 101.75 | 93.0 (−8.6%) |
| 9 | 2/2/2 | 106.25 | 103.25 (−2.8%) | 106.25 | 94.125(−11.4%) |
| 10 | 3/2/2 | 89.5 | 89.5 (−0.0%) | 87.125 | 84.25 (−3.3%) |
| 10 | 2/2/2 | 91.0 | 90.0 (−0.6%) | 88.875 | 87.0 (−2.1%) |
| 11 | 2/2/2 | 83.75 | 83.25 (−0.6%) | 83.5 | 83.0 (−0.6%) |
| 11 | 2/2/1 | — | — | — | **93.0** |
| 12 | 2/2/1 | — | — | 91.0 | 85.0 (−6.6%) |

| CC | ADD0 | | ADD1 | | ADD2 | SUB0 | | SUB1 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | A1p | | A1s | | | S1p | S3p | S1s | S3s |
| 1 | A2p | | A2s | | A3p | S2p | S6p | S2s | S6s |
| 2 | A5p | A6p | A5s | A6s | A3s | S5p | | S5s | |
| 3 | A4p | | A4s | | A8p | S7p | S4p | S7s | S4s |
| 4 | A7p | | A7s | | A8s | S8p | | S8s | |

(a)

| CC | ADD0 | | ADD1 | | ADD2 | SUB0 | | SUB1 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | A1p | | A1s | | | S6p | S1p | S3p | S1s |
| 1 | A2p | A3p | A6p | A3s | A5p | S2p | S3s | S2s | S6s |
| 2 | A4p | A2s | A5s | A6s | | S5p | S7p | S5s | |
| 3 | A4s | | A7p | | | S7s | S4p | S4s | |
| 4 | A8p | | A8s | | A7s | S8p | | S8s | |

(b)

Fig. 8. Obtained DMR schedules for MAHA with $TE = 5$, $p_D = 3$, 3 adders and 2 subtractors. (a) conventional. (b) proposed.

| CC | ADD0 | | | ADD1 | | SUB0 | | |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | S3p | | |
| 1 | A1p | | | | | S3s | S1p | |
| 2 | A1s | | | A2p | | S2p | S6p | S1s |
| 3 | A2s | | | A5p | A6p | S2s | S6s | S4p |
| 4 | A4p | A5s | A3p | A6s | A3s | S5p | S7p | S4s |
| 5 | A4s | | | A7p | | S5s | S7s | |
| 6 | A7s | | | | | S8p | | |
| 7 | A8p | | | A8s | | S8s | | |

Fig. 9. The DMR schedule for MAHA obtained by the proposed method with $TE = 8$, $p_D = 3$, 2 adders and 1 subtractor.

judgment operations are scheduled at early CCs to eliminate speculative execution to reduce unnecessary energy consumption while the secondary execution of condition judgement operations are scheduled at later CCs.

Table I shows the results for FIG17p. ‘A/S/M’ indicates the constraints on the number of adders, the number of subtractors, and the number of multipliers. Up to 22% energy reduction is achieved by the proposed method. Again, the proposed method can derive a feasible DMR design for configurations with which the conventional method cannot derive a solution. These are shown in bold in the table.

## V. CONCLUSIONS

To the minimization of energy consumption for DMR design of conditional processing, we proposed a method to make the secondary execution of condition-dependent operation also dependent on the primary execution of condition judgment operation. The ILP formulation of the minimization problem was presented and the optimal solutions were obtained by solving the ILP models.

Consideration of the technique to further utilize the nature of DMR conditional processing to obtain better design, development of a heuristic algorithm for the solution remain as future works.

## REFERENCES

[1] R. Baumann, “Soft errors in advanced computer systems,” IEEE Design & Test of Computers, vol.22, no.3, pp.258–266, 2005.

[2] F. Wang and V.D. Agrawal, “Single event upset: An embedded tutorial,” Proc. Int. Conf. VLSI Design, pp.429–434, 2008.

[3] F.L. Kastensmidt, L. Sterpone, L. Carro, and M.S. Reorda, “On the optimal design of triple modular redundancy logic for SRAM-based FPGAs,” Proc. DATE 2005, pp.1290–1995, 2005.

[4] S. Golshan and E. Bozorgzadeh, “SEU-aware resource binding for modular redundancy based designs on FPGAs,” Proc. DATE 2009, pp.1124–1129, 2009.

[5] T. Imagawa, H. Tsutsui, H. Ochi, and T. Sato, “A cost-effective selective tmr for coarse-grained reconfigurable architectures based on dfg-level vulnerability analysis,” IEICE Trans. Electron., vol.E96-C, no.4, pp.454–462, 2013.

[6] J. Oh and M. Kaneko, “Area-efficient soft-error tolerant datapath synthesis based on speculative resource sharing,” IEICE Trans. Fund., vol.E99-A, no.7, pp.1311–1322, 2016.

[7] K. Ito, Y. Ishihara, and S. Nishizawa, “Minimization of vote operations for soft error detection in dmr design with error correction by operation re-execution,” IEICE Trans. Fund., vol.E101-A, no.12, pp.2271–2279, 2018.

[8] K. Wakabayashi, “Unified representation for speculative scheduling: Generalized condition vector,” IEICE Trans. Fundamentals, vol.E89-A, no.12, pp.3408–3415, 2006.

[9] K. Ito and K. Kameda, “A method to reduce energy consumption of conditional operations with execution probabilities,” IPSJ Trans. on System LSI Design Methodology, vol.6, pp.60–70, 2013.

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

[11] T. Kim, N. Yonezawa, J.W.S. Liu, and C.L. Liu, “A scheduling algorithm for conditional resource sharing — a hierarchical reduction approach,” IEEE Trans. Computer Aided Design, Int. Circuit. Syst., vol.13, no.4, pp.425–437, 1994.

[12] A.C. Parker, J.T. Pizarro, and M. Mlinar, “MAHA: a program for datapath synthesis,” Proc. Design Auto. Conf., pp.461–466, 1986.