Vol.2011-SLDM-149 No.19
Vol.2011-EMB-20 No.19
2011/3/18

Soft Error-Aware Scheduling in High-Level Synthesis

Yuko Hara-Azumi,[†1,†2,†3] Hiroyuki Tomiyama,[†3]
Takuya Azumi,[†3] Shigeru Yamashita,[†3]
Nikil D. Dutt[†4] and Hiroaki Takada [†2]

Due to the continuous reduction in chip feature size and supply voltage, soft errors are becoming a serious problem in the today's LSI design. This paper proposes a soft error-aware scheduling method in high-level synthesis. The reliability of the datapath circuit is determined not only by those of its computations, which depend on their assigned hardware resources, but also by those of its values, which are affected by their lifetime length. By considering both influences, our proposed method schedules operations for maximizing the reliability of the datapath circuit.

1. Introduction

The vulnerability of circuits against soft errors induced mainly by cosmic rays is becoming a serious problem not only for safety critical applications but also for commercial consumer ones[1)]. So far, various reliability-aware LSI design techniques have been studied for mitigating the potential consequences of soft errors. Most existing works[2),3)] focus on memory elements since the soft error susceptibility of memory elements has been considered more serious than those of combinational circuits (i.e., datapath circuits). However, 4) warned that datapath circuits will become more susceptible due to the continuous reduction in geometries, higher density, and lower supply voltage of the circuits.

Incorporating reliability concerns into the higher level of the LSI design can more effectively design fault-tolerant circuits[5)]. Several works recently have studied High-Level Synthesis (HLS) techniques of fault-tolerant systems against soft errors. Most of the works adopt N Modular Redundancy (NMR), which makes N duplications of hardware resources and takes a vote of the results in order to filter out the propagation of faults to outputs of the circuit. Triple Modular Redundancy (TMR) is the most commonly-used amongst such techniques. Various works such as 6)–8) spacially and/or temporarily employ TMR. The drawback of NMR is that it requires large area or performance overhead, e.g., spacial NMR of a hardware resource with area A requires another $(N-1) \times A$ in area. Since embedded systems generally have strict area and performance constraints, it is difficult to apply NMR techniques to such systems.

Instead of adopting NMR techniques, 5) tries to improve the reliability of the datapath circuit by exploiting various versions of Functional Units (FUs) which have different area, performance (latency), and reliability, under area and latency constraints. However, the work focuses on only the reliabilities of operations and does not consider those of values (i.e., operational results), which also contribute the reliability of the datapath circuit. The longer the lifetime of the value is, the more the value has a chance of being affected by soft errors, that is, the lower reliability the value has[9),10)]. Ignoring such influence may disable the work to generate in practice fault-tolerant circuits.

This paper studies a scheduling technique considering the reliabilities of both operations and values. Utilizing various versions of FUs, this technique simultaneously assigns operations to FUs with the highest reliability and minimizes the lifetime of values so that the reliability of the datapath circuit is maximized under the given constraints on the area and latency. Experiments demonstrate that our work is more effective in the area-performance-reliability trade-off exploration than a traditional TMR technique and the existing work 5).

The rest of this paper is organized as follows. Section 2 gives a target error model and an evaluation metric of the reliability of the datapath circuit. Section 3 presents our soft error-aware scheduling method and formulates it as an ILP problem. Section 4 demonstrates the effectiveness of our work over previous works through experiments. Finally, Section 5 concludes this paper.

2. Target Error Model and System Reliability

A soft error, which is mainly induced by cosmic rays such as alpha particles and neutrons, triggers an incorrect result but no hard/permanent damage to the

†1 Research Fellow of the Japan Society for the Promotion of Science
†2 Nagoya University
†3 Ritsumeikan University
†4 University of California, Irvine

circuits[1]. Faults on the circuits caused by soft errors can be classified into two groups by how they affect the circuits: configurational errors and computational errors. The former affects the configuration of the circuits. The fault remains until the circuits are reconfigured and the faults are corrected. This is more likely to occur on FPGA-based designs since FPGAs are composed of SRAMs. The latter affects only computational results of the combinational circuits, e.g., a bit flip on a data being handled in the circuits. A result may be correct even if its precedent result obtained throuh the same part of the circuit is incorrect. This may occur both on FPGAs and ASICs. In this paper, we target at the latter.

The reliability of the datapath circuit depends on those of its operations and values (i.e., operational results), which are determined by their assigned hardware resources because soft error susceptibilities of resources differ even amongst different implementations for the same type of resources, e.g., ripple-carry adder, Brent-Kung adder, and Kogge-Stone adder[5]. In addition, the reliabilities of values are also determined by their lifetime length because the longer the lifetime of the value is, the more the value has a chance of being affected by soft errors[9),10]. Thus, when operation/value i is implemented by FU/register j, the reliability of operation/value i, $TaskReliability_i$ is expressed as follows:

$$TaskReliability_i = \begin{cases} AgentReliability_j & \text{for operation } i \\ (AgentReliability_j)^{ActiveTime_i} & \text{for value } i \end{cases}$$

where $AgentReliability_j$ and $ActiveTime_i$ are the reliability of FU/register j and the lifetime length of value i, respectively. Then, the reliability of the datapath circuit can be obtained by the total products of $TaskReliability_i$ (i.e., $\prod_{i \in Tasks} TaskReliability_i$), which is used as an evaluation metric in this paper. Their more detailed discussion will be given in the next section.

3. Soft Error-Aware Scheduling for Reliability Maximization

This section presents our proposed scheduling method for maximizing the reliability of the datapath circuit.

3.1 Problem Definition

We propose a scheduling method which maximizes the reliability of the datapath circuit by exploiting various versions of FUs. As mentioned in the previous section, our method considers not only the reliabilities of operations but also those of values which depend on their lifetime length. This method is formulated as an ILP problem.

Inputs to our method are a data flow graph (DFG), various versions of FUs which have different area, performance (latency), and reliability, and constraints on the datapath circuit area and latency given by a designer. A DFG is an acyclic directed graph, where nodes and edges represent operations and data dependencies, respectively. Data which are generated in a clock cycle and used in another clock cycle are called *values* and need to be stored in registers. In the remainder of this paper, operations and values are collectively called *tasks*, and FUs and registers are collectively called *agents*. For simplicity, operation chaining is not considered in this paper, i.e., all data dependencies are handled as values according to the above definition.

The goal of our method is to simultaneously determine the operational scheduling (i.e., when and which version of FUs each operation is assigned to) and allocations (i.e., the number of instances for each FU) so that the reliability of the datapath circuit (i.e., the total reliabilities of operations and values) is maximized while meeting the area and latency constraint.

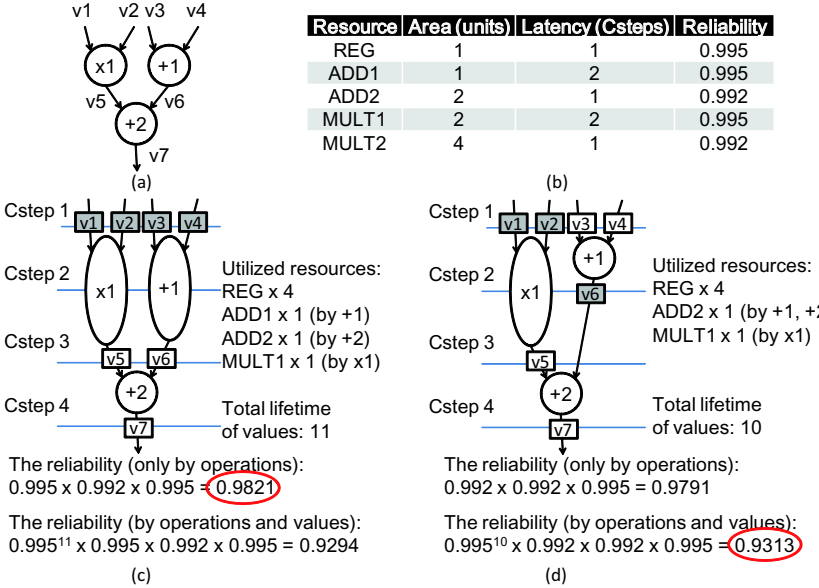
Let us consider an example depicted in **Fig. 1**. Given a DFG in Fig. 1 (a) and a library in Fig. 1 (b), the schedules shown in Fig. 1 (c) and Fig. 1 (d) can be obtained under the constraints of nine area units and four control steps. The lifetime of values in white and gray boxes is one and two control steps, respectively. A previous work 5) which focuses only on the reliabilities of operations regards the schedule in Fig. 1 (c) as the better result. However, by considering the reliabilities of both operations and values, it is obvious that the schedule in Fig. 1 (d) achieves the higher reliability. Therefore, in this example, our work obtains the schedule in Fig. 1 (d) as the better design and overcomes the work 5).

3.2 ILP Formulation

We formulated our operational scheduling technique for maximizing the total reliabilities of operations and values as an ILP problem below. Notations in the following formulas are defined in **Table 1**. The variables are dependent on $x_{i,j}$.

Compatibility between tasks and agents: A task can be executed by one and only one compatible agent.

Vol.2011-SLDM-149 No.19
Vol.2011-EMB-20 No.19
2011/3/18



Resource	Area (units)	Latency (Csteps)	Reliability
REG	1	1	0.995
ADD1	1	2	0.995
ADD2	2	1	0.992
MULT1	2	2	0.995
MULT2	4	1	0.992

Fig. 1 An example: (a)A given DFG, (b)A library, (c)A possible schedule using one `ADD1`, one `ADD2`, and one `MULT1`, and (d)Another possible schedule using one `ADD2` and one `MULT1`.

$$x_{i,j} \leq Compatible_{i,j}, \forall i \in Tasks, \forall j \in Agents \quad (1)$$

$$\sum_{j \in Agents} x_{i,j} = 1, \forall i \in Tasks \quad (2)$$

Active time: The active time of operation i (i.e., the execution time) is the same as that of agent j which executes operation i.

$$ActiveTime_i = \sum_{j \in Agents} (ExecTime_j \times x_{i,j}), \forall i \in Tasks, TaskType_i \neq value \quad (3)$$

For example, $ExecTime_j = 1$ for agent j which completes within a clock cycle.

The active time of value i (i.e., the lifetime) is from when the value is generated to when the value is used for the last time.

$$ActiveTime_i = EndTime_i - InitTime_i + 1, \forall i \in Task, TaskType_i = value \quad (4)$$

For both operations and values, $Active_{i,t}$ is defined as follow:

$$Active_{i,t} = \begin{cases} 1 & \text{if } InitTime_i \leq t \leq EndTime_i \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Table 1 Definition of notations.

$Tasks$	A set of types of tasks (e.g., addition, multiplication, and value)
$TaskType_i$	Type of task i
$Agents$	A set of agents (e.g., ripple-carry adder, Brent-Kung adder, and Kogge-Stone adder)
$AgentType_j$	Type of agent j
$Dataflow_{i1,i2}$	$Dataflow_{i1,i2} = 1$ if there is dataflow from tasks $i1$ to $i2$, otherwise 0
$Compatible_{i,j}$	$Compatible_{i,j} = 1$ if task i can be executed by agent j, otherwise 0
$x_{i,j}$	$x_{i,j} = 1$ if task i is executed by agent j, otherwise 0
$PrimaryIn_i$	1 if task i is a primary input value, otherwise 0
$PrimaryOut_i$	1 if task i is a primary output value, otherwise 0
$Csteps$	A set of control steps
$ExecTime_j$	Execution time of agent j
$InitTime_i$	Initiation time of task i
$EndTime_i$	Termination time of task i
$ProgEndTime$	The time when the input program completes
$ActiveTime_i$	Active time of task i
$Active_{i,t}$	1 if task i is active at control step t
$InstanceNum_j$	The number of instances of agent j
$TaskReliability_i$	Reliability of task i
$AgentReliability_j$	Reliability of agent j
$Area_j$	Area of agent j
$Area\_const$	Constraint of area
$Latency\_const$	Constraint of latency

$InitTime_i$ and $EndTime_i$ are explained below.

Initiation/Termination time: If $Dataflow_{i2,i1} = 1$, value $i1$ is initiated when operation $i2$ generates value $i1$, that is, when the execution of operation $i2$ completes. Note that the initiation time for a primary input value $i1$ (i.e., $PrimaryIn_{i1} = 1$) is given. In experiments in Section 4, $InitTime_{i1}$ is set to 1 for all primary input values.

$$EndTime_{i2} = InitTime_{i2} + ActiveTime_{i2} - 1 = InitTime_{i1}, \forall i1, i2 \in Tasks, Dataflow_{i2,i1} = 1, TaskType_{i1} \neq value, TaskType_{i2} = value \quad (6)$$

If $Dataflow_{i1,i3} = 1$, operation $i3$ starts only after the start of value $i1$.

$$InitTime_{i1} < InitTime_{i3}, \ \forall i1, i3 \in Tasks, Dataflow_{i1,i3} = 1, TaskType_{i1} = value, TaskType_{i3} \neq value \quad (7)$$

For value $i1$ which is not a primary output value (i.e., $PrimaryOut_{i1} = 0$), its termination time is when it is read for the last time.

$$EndTime_{i1} = \max_{i2 \in Tasks} (EndTime_{i2} \times Dataflow_{i1,i2}) - 1, \forall i1 \in Tasks, TaskType_{i1} = value, TaskType_{i2} \neq value \quad (8)$$

For value $i1$ which is a primary output value (i.e., $PrimaryOut_{i1} = 1$), its termination time is when the program completes, $ProgEndTime = \max_{i3 \in Tasks} EndTime_{i3}, TaskType_{i3} \neq value$.

$$EndTime_{i1} = ProgEndTime, \forall i1 \in Tasks, PrimaryOut_{i1} = 1 \quad (9)$$

Constraints on area and latency: The number of instances of agent j is its maximum number required at the same control step.

$$InstanceNum_j = \max_{t \in Csteps} (\sum_{i \in Tasks} Active_{i,t} \times x_{i,j}), \forall j \in Agents \quad (10)$$

The total agent area and all the task execution should not be beyond the constraint on area and latency, respectively.

$$\sum_{j \in Agents} InstanceNum_j \times Area_j \leq Area\_const \quad (11)$$

$$ProgEndTime \leq Latency\_const \quad (12)$$

Reliability: The reliability of operation i depends on its assigned agent j.

$$TaskReliability_i = \sum_{j \in Agents} AgentReliability_j \times x_{i,j}, \forall i \in Tasks, TaskType_i \neq value \quad (13)$$

The reliability of value i varies depending on how long value i is active because register j should keep correct value i while it is active.

$$TaskReliability_i = (\sum_{j \in Agents} AgentReliability_j \times x_{i,j})^{ActiveTime_i}, \forall i \in Tasks, TaskType_i = value, AgentType_j = register \quad (14)$$

Objective: The objective is to maximize the reliability of the overall design.

$$Max : \prod_{i \in Tasks} TaskReliability_i \quad (15)$$

We linearized the formula (15) by logarithm and use the following objective:

$$Max : \sum_{i \in Tasks} TaskReliability_i \quad (16)$$

4. Experiments

This section shows the effectiveness of our work through experiments.

4.1 Experimental Setup

In experiments, two designs, autoregressive lattice (AR) filter[11] and 16-point symmetric Finite Impulse Response (FIR) filter[5], were used as benchmarks. We utilized one type of register (`REG`), three types of adders (`ADD1`, `ADD2`, and `ADD3`), and two types of multipliers (`MULT1` and `MULT2`) for scheduling the two designs by our proposed method. These FUs have different area, performance in control steps (latency), and reliability as shown in the second to seventh columns of **Table 2**, where the values are normalized[5]. The ILP problem described in Section 3 was solved by a commercial ILP solver, CPLEX[12], while varying the constraints on area and latency of the circuits.

Table 2 Area, performance, and reliability of hardware resources.

	REG	ADD1	ADD2	ADD3	MULT1	MULT2	ADD2 (TMR)	MULT2 (TMR)
Latency (CSteps)	1	2	1	1	2	1	1	1
Area (Units)	1	1	2	4	2	4	6	12
Reliability	0.999	0.999	0.969	0.987	0.999	0.969	0.997	0.997

To demonstrate the effectiveness of our work, the two benchmarks were also scheduled by the following two existing works:

- **Spacial TMR** utilizes only one version of FUs for each adder and multiplier and adopts spacial redundancy for maximizing the total reliabilities of operations and values. For the adder and multiplier, `ADD2` and `MULT2` were selected, respectively. The area, latency, and reliability of the TMR-employed `ADD2` and `MULT2`$^{\star 1}$ are described in the last two columns of Table 2.
- **Method 5)** utilizes various versions of FUs other than TMR-employed ones for maximizing the total reliabilities of only operations, i.e., its objective is to maximize $\sum_{i \in Tasks} TaskReliability_i, TaskType_i \neq value$. In comparison, we evaluated this work by the total reliabilities of operations and values (i.e., formula (16)) as well as our work and Spacial TMR.

Each of the above two methods was also solved as an ILP problem by CPLEX[12].

4.2 Experimental Results

In the first sets of experiments, our proposed method was performed for the two designs while varying the constraints on area and latency. **Fig. 2** (a) and Fig. 2 (b) show the area-reliability tradeoffs under $Latency\_const = 11$ and the performance-reliability trade-offs under $Area\_const = 39$, respectively, for AR.

⋆1 For simplicity, the area and reliability overhead by inserting voters are assumed to be zero in this paper.

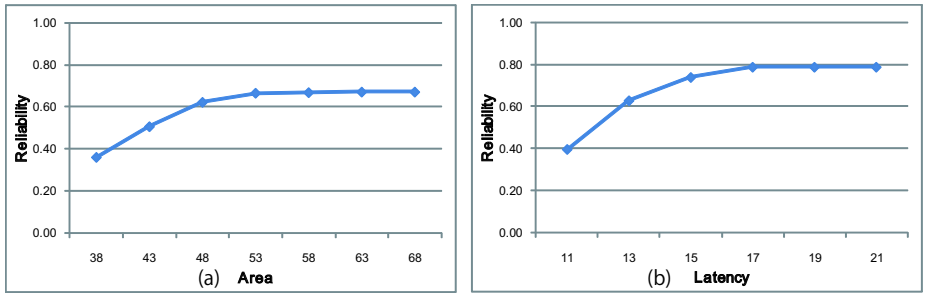


Fig. 2 Trade-offs for *AR*: (a)Area-reliability trade-offs under Latency = 11 and (b)Latency-reliability trade-offs under Area = 39.

Due to the space limitation, the other results are omitted, but the similar results were observed for both *AR* and *FIR*.

Fig. 2 (a) describes that the looser the area constraint, the higher reliability our work achieves even under the same latency constraint. This is because the looser area constraint permits to utilize an FU which has the higher reliability without sacrificing the latency, that is, replacement of `ADD2` with `ADD3`. Similarly, Fig. 2 (b) describes that the looser latency constraint, the more reliable circuits our work explores because FUs with the higher reliability but the longer latency can be utilized. As the latency constraint is relaxed, the improvement rate becomes small. This is because the use of the FUs with the longer latency may lengthen the lifetime of some values, leading to the lower reliabilities of such values. This can be more prominently observed under the tighter area constraint.

In the second sets of experiments, we scheduled the two designs by our work and the two previous works while varying the constraints on the area and latency of the circuits. **Table 3** and **Table 4** summarize the experimental results for *AR* and *FIR*, respectively. The first and second columns of the tables show the constraints on the latency and area, respectively. The third to fifth columns describe the reliability of the circuits obtained by the three methods, and the sixth and seventh columns give the percentage improvements brought by our work over the previous works. The eighth to tenth columns describe the elapsed time of CPLEX for obtaining the results by the three methods. ">1,000" means that it took more than 1,000 seconds to obtain the optimal solution, so for such cases, the shown result is the best solution obtained in 1,000 seconds.

Table 3 Experimental results for *AR*.

Constraints		Reliability			Improvements (%)		Solution time (s)		
Latency	Area	TMR	Method 5)	Ours	TMR	Method 5)	TMR	Method 5)	Ours
9	46	0.37166	0.37274	0.37348	0.49	0.20	0.40	0.50	0.46
9	56	0.52423	0.53383	0.53436	1.93	0.10	0.67	0.67	0.96
9	66	0.61950	0.53758	0.53758	-13.22	0.00	0.79	0.61	0.64
11	38	0.35958	0.35848	0.36136	0.49	0.80	1.12	1.76	2.68
11	48	0.50720	0.61775	0.62272	22.78	0.80	8.14	6.43	19.76
11	58	0.73475	0.65692	0.66885	-8.97	1.82	5.75	1.95	2.14
13	42	0.48699	0.70320	0.71812	47.46	2.12	14.18	46.70	18.40
13	52	0.64604	0.74396	0.75823	17.37	1.92	226.62	4.28	3.14
15	39	0.35958	0.73001	0.73882	105.47	1.21	44.78	878.07	281.89
15	49	0.68735	0.77071	0.77847	13.26	1.01	>1,000	8.48	11.15
15	59	0.78251	0.76995	0.78314	0.08	1.71	63.98	9.17	5.76
17	39	0.35958	0.77019	0.78812	119.17	2.33	265.50	12.78	85.29
17	49	0.70568	0.76787	0.79763	13.03	3.88	>1,000	26.04	11.87
19	36	0.45389	0.74741	0.78496	72.94	5.02	>1,000	114.84	140.34
19	46	0.68525	0.77560	0.79763	16.40	2.84	>1,000	66.00	37.53
21	35	0.32114	0.62875	0.65593	104.25	4.32	>1,000	>1,000	>1,000
21	45	0.70943	0.71809	0.79604	12.21	10.85	>1,000	58.53	152.26

Our work considerably overcomes TMR especially under the tight area constraint since there is no or little space for TMR, leading to the low reliability. When the area constraint is lower than 40 except for when $Latency\_const = 19$ in Table 3, TMR was employed for no operations. On the contrary, TMR achieves the better results in some cases with the loose area and strict latency constraints since our work cannot exploit FUs with the high reliability but the long latency.

Our work overcomes Method 5) because while Method 5) focuses on only operations, our work considers the lifetime of values in addition to the operations. Our work achieves the bigger improvements under the looser latency constraint since the values tend to have long lifetime especially if they are not considered. Studying the results in more detail, we found some cases where our work achieved the better results even though the assignment of operations to FUs is exactly the same between the two works. This is because the operational timing differs, that is, Method 5) scheduled some operations in the later timing, leading to the lower reliabilities of some values by their longer lifetime.

The results demonstrate the effectiveness of our work over the two previous works. Unfortunately, however, the improvements over Method 5) were not as big as expected since the two benchmarks are not large and the influences on the

Table 4 Experimental results for *FIR*.

Constraints		Reliability			Improvements (%)		Solution time (s)		
Latency	Area	TMR	Method 5)	Ours	TMR	Method 5)	TMR	Method 5)	Ours
11	32	0.42769	0.52408	0.52460	22.66	0.10	0.71	2.73	3.78
11	42	0.65745	0.77582	0.77893	18.48	0.40	1.96	1.21	1.68
11	52	0.82688	0.78520	0.78913	-4.57	0.50	1.15	0.93	1.01
11	62	0.84107	0.78520	0.78991	-6.08	0.60	1.51	0.95	1.01
13	32	0.42769	0.67349	0.67686	58.26	0.50	1.68	125.23	149.90
13	42	0.65745	0.78460	0.79805	21.39	1.71	20.06	2.46	6.37
13	52	0.82688	0.79327	0.80769	-2.32	1.82	6.62	2.96	1.85
15	32	0.42769	0.71580	0.71794	67.86	0.30	7.23	300.64	392.46
15	42	0.65745	0.79658	0.81187	23.49	1.92	149.25	6.64	21.84
15	52	0.82688	0.77225	0.82168	-0.63	6.40	12.61	5.50	3.59
17	32	0.42769	0.74473	0.74153	73.38	-0.43	26.70	>1,000	>1,000
17	42	0.78262	0.81444	0.82263	5.11	1.01	822.96	12.23	81.18
17	52	0.82688	0.80876	0.83256	0.69	2.94	37.92	5.67	7.76
19	32	0.42769	0.80969	0.81375	90.27	0.50	22.79	270.84	466.25
19	42	0.78262	0.82441	0.83520	6.72	1.31	>1,000	23.57	49.03
19	52	0.82688	0.82854	0.84023	1.61	1.41	59.37	26.28	8.53
21	31	0.40479	0.78576	0.80888	99.83	2.94	>1,000	212.89	>1,000
21	41	0.65745	0.79684	0.83353	26.78	4.60	335.06	51.04	76.48

reliability of the datapath circuit by those of the values are relatively small. We expect that our work will achieve remarkable effectiveness in practical designs. To evaluate it for larger benchmarks and to develop a heuristic to obtain solutions in practical time are our future work.

5. Conclusions

In this paper, we proposed a soft error-aware scheduling technique in HLS. Our method exploits various versions of FUs with different area, latency, and reliability, and assigns operations to FUs so that the reliability of the datapath circuit is maximized under area and latency constraints. Also, at the same time, our method minimizes the lifetime of values, which also affects the reliability of the circuit. We formulated this method as an ILP problem. Experimental results demonstrated that our work achieves better results than existing works.

We will extend this work so that more general cases such as operation chaining and hierarchical DFGs can be handled. Developing a heuristic to solve the problem in practical time is also the subject of our future work.

Acknowledgements

This work is in part supported by KAKENHI 22700050.

References

1) K.M. Zick and J.P. Hayes, "High-level vulnerability over space and time to insidious soft errors," in *Proc. High Level Design Validation and Test Workshop*, 2008, pp. 161–168.
2) V.Gherman, S.Evain, M.Cartron, N.Seymour, and Y.Bonhomme, "System-level hardware-based protection of memories against soft-errors," in *Proc. Design, Automation and Test in Europe*, 2009, pp. 1222–1225.
3) P.Revirieo, J.A. Maestro, and C.J. Bleakley, "Reliability analysis of memories protected with bics and a per-word parity bit," *ACM Trans. on Design Automation of Electronic Systems*, vol.15, no.2, pp. 18:1–18:15, 2010.
4) P.Sivakumar, M.Kistler, S.W. Keckler, D.C. Burger, and L.Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proc. International Conference on Dependable Systems and Networks*, 2002, pp. 389–398.
5) S.Tosun, N.Mansouri, E.Arvas, M.Kandemir, and Y.Xie, "Reliability-centric high-level synthesis," in *Proc. Design, Automation and Test in Europe*, 2005, pp. 1258–1263.
6) G.Lakshminarayana, A.Raghunathan, and N.K. Jha, "Behavioral synthesis of fault secure controller/datapaths based on aliasing probability analysis," *IEEE Trans. on Behavioral Synthesis of Fault Secure Controller/Datapaths Based on Aliasing Probability Analysis*, vol.49, no.9, pp. 865–885, 2000.
7) W.Kaijie and R.Karri, "Fault secure datapath synthesis using hybrid time and hardware redundancy," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.23, no.10, pp. 1476–1485, 2004.
8) S.Golshan and E.Bozorgzadeh, "SEU-aware resource binding for modular redundancy based designs on FPGAs," in *Proc. Design, Automation and Test in Europe*, 2009, pp. 1124–1129.
9) P.Montesinos, W.Liu, and J.Torrellas, "Using register lifetime predictions to protect register files against soft errors," in *Proc. International Conference on Dependable Systems and Networks*, 2007, pp. 286–296.
10) M.Fazeli, S.N. Ahmadian, and S.G. Miremadi, "A low energy soft error-tolerant register file architecture for embedded processors," in *Proc. International High Assurance Systems Engineering Symposium*, 2008, pp. 109–116.
11) S.Tosun, O.Ozturk, N.Mansouri, E.Arvas, M.Kandemir, Y.Xie, and W.-L. Hung, "An ILP formulation for reliability-oriented high-level synthesis," in *Proc. International Symposium on Quality Electronic Design*, 2005, pp. 364–369.
12) IBM ILOG CPLEX Optimizer. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/