International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 12, Number 2, pages 387–405, July 2022

Evaluation of Directive-based Heterogeneous Redundant Design Approaches for Functional Safety Systems on FPGAs

Taichi Saikai, Kotoko Miyata, Taito Manabe, and Yuichiro Shibata
Graduate School of Engineering, Nagasaki University
1–14 Bunkyo-machi, Nagasaki, 852-8512, Japan
E-mail:
{saikai,kotoko}@pca.cis.nagasaki-u.ac.jp, {tmanabe,yuichiro}@nagasaki-u.ac.jp

Received: February 15, 2022
Revised: May 5, 2022
Accepted: May 26, 2022
Communicated by Ikki Fujiwara

Abstract

Field programmable gate arrays (FPGAs) are now used in a wide range of application fields including aerospace, medical, and industrial infrastructure systems, where not only soft errors but also common cause faults must be treated in systems design. Although the heterogeneous redundant design is preferable in such application fields, it tends to be a large burden on system designers. Even with high-level synthesis (HLS) technologies, which have enabled productive design processes without register transfer level (RTL) descriptions, an efficient design approach for redundant design is not always clear. In this paper, we present and evaluates two heterogeneous redundant circuit design approaches for FPGAs: a resource-level approach and strategy-level approach. The resource-level approach focuses on diversity in FPGA technology mapping. For example, two different implementations for a multiplier, one with look-up tables (LUTs) and the other with digital signal processing (DSP) blocks, can form heterogeneous redundancy. The strategy-level approach makes the use of the optimization options offered by an FPGA design tool, called strategies, as a source of diversity. For example, two different implementations can be derived from the same hardware description with area oriented optimization and performance oriented optimization. With both approaches, heterogeneous implementation variants can be generated from the same hardware description code. For evaluation, we implemented homogeneous and heterogeneous redundant designs for proportional-integral-derivative (PID) control with those approaches and evaluated their error detection capability and reliability with overclock simulation. The resource-level approach showed that heterogeneous redundant designs by the proposed method have a high error detection rate in both RTL and HLS implementations in an application-level circuit. Although the detection rate of the strategy-level approach was not as high as that of the resource-level one, it was shown to have a certain diversification effect.

Keywords: redundant design, diversity, high-level synthesis, compiler options

1 Introduction

Flexibility of field programmable gate arrays (FPGAs) have expanded their application fields to various domains including financial, medical, autonomous vehicle driving, and aerospace systems. In such fields, fault-tolerant design is required since stoppage due to system failures or malfunctions is not acceptable. In order to improve safety of systems, a method of arranging multiple same modules is commonly used, as represented by triple modular redundancy (TMR). Such a method is called homogeneous redundant design. Probabilistic failures, such as device degradation and soft errors caused by environmental radiation, can

be detected by comparing outputs between redundant modules, so that the outage of the entire system is effectively prevented. However, homogeneous redundant design can overlook systematic failures caused by common cause faults such as voltage and temperature abnormalities, since each redundant module may output the same error value.

Heterogeneous redundant design is a method that can improve tolerance for this case. This method introduces diversity into the redundancy by arranging redundant modules with different implementation approaches. The diversity for redundant modules can be made in a various ways, such as the use of different algorithms for the same calculation task. However, this increases the design cost and decreases the productivity. Hence it is desirable to keep the cost as low as possible to mitigate designers' burden. High-level synthesis (HLS), which automatically generates register transfer level (RTL) code from high-level languages, allows designers to easily design hardware without being aware of the complex circuit design issues. However, a concise and efficient design method for heterogeneous redundant designs using HLS is not necessarily obvious.

This paper presents and evaluates two heterogeneous redundant circuit design approaches for FPGAs: a resource-level approach and strategy-level approach. The fundamental difference between these approaches lies in the way in which diversity is introduced. In the resource-level approach, the diversity that exists in technology mapping, which is a process to bind each logic function to FPGA resources, is focused. By inserting directives into RTL or HLS code, heterogeneous redundancy can be introduced with relatively low design costs. For instance, two different implementations of a multiplier, one with digital signal processing (DSP) blocks and the other with look-up tables (LUTs), can be derived from the same code.

On the other hand, in the strategy-level approach, the optimization options offered by an FPGA design tool, called strategies, are focused as a source of diversity. The choice of the strategy can affect various aspects of FPGA design including state machine encoding and the upper limit number of fanouts. Since the strategies are thoughtfully designed by the FPGA tool vendor, quality of each implementation variant produced with different strategies is expected to be high. In this paper, use of strategy-level diversity in two different design phases is discussed: strategy in logic synthesis and strategy in place-and-route. While the resource-level approach has been proposed and discussed in our earlier literature [1, 2, 3], the strategy-level approach is proposal of this paper, inspired by the work in which different compiler optimization levels are used to improve reliability of software [4].

For evaluation, we adopt a proportional-integral-derivative (PID) control module, which is widely used in industrial control systems, as an application example, and implement the module with those redundant design methods. We then evaluate the performance of each design by generating timing errors with overclock simulation, which emulates common cause faults.

The rest of this paper is organized as follows. Section 2 presents the background of this work. In Section 3, we implement modules for PID control, which is the target of redundant design. In Section 4, the resource-level heterogeneous redundant design approach is introduced and evaluated. Comparison between RTL-based and HLS-based designs is also presented. We propose the strategy-based approach utilizing optimization options offered by an FPGA design tool in Section 5. The strategies in logic synthesis and in place-and-route are evaluated and compared. Finally, the paper is concluded in Section 6.

2 Background

2.1 Functional safety

The application domains of FPGAs are increasingly expanding, so that FPGAs are now used in mission-critical systems such as autonomous vehicles. Such systems can be exposed to harsh environments and need to be designed to be robust against any failure. There are two major failures; probabilistic ones (e.g., bit flip caused by environmental radiation, deterioration of devices) and systematic ones (e.g., design mistakes, program bugs). IEC 61508 [5], an international standard for functional safety, says that countermeasures should be taken for both failures.

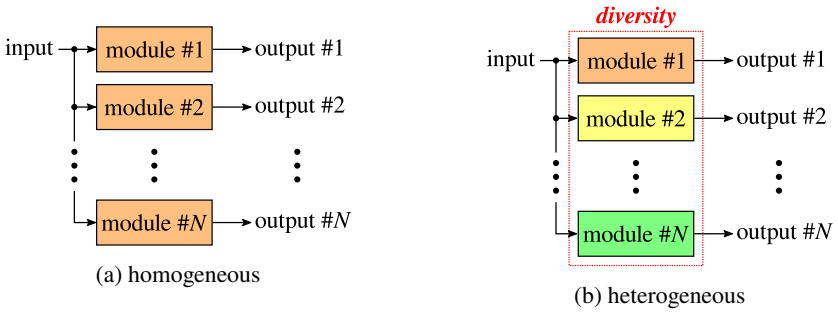


Figure 1: Two types of redundant design

2.2 Redundant design

Redundant design is a typical method to improve the safety of systems. Modules are replicated so that the entire system can continue to function even if one of the modules fails. General redundant design methods are based on multiple identical modules. This kind of methods are called *homogeneous* redundant design. A typical example is triple modular redundancy (TMR), whose structure is shown in Fig. 1a. With such design, even if one of the modules produces an abnormal output value due to a probabilistic failure, the error can be detected by comparing it with normal output values of the others. However, if a failure occurs due to a common factor such as an abnormal voltage drop of the power supply, each module can make the same and wrong output value. This error cannot be detected by comparing the outputs of the redundant modules, making this homogeneous redundancy meaningless.

For system designs that require high reliability, it is necessary to address not only soft errors but also common cause fault. *Heterogeneous* redundant design can be expected to provide fault tolerance against common cause fault. The heterogeneous redundant design introduces diversity into the module redundancy as shown in Fig. 1b. With this design, even if a common cause fault occurs, the diversified modules can output different values so that the system can detect the error. Although circuit topologies, design code, and algorithms can be considered as a source for generating module diversity, this approach increases the design costs. Therefore, for heterogeneous redundant designs, both mitigation of design costs for introducing module diversity and improvement of the tolerance against the common cause fault are required. In terms of design costs, HLS, which automatically generates RTL descriptions from high-level languages such as C, C++, and Java, has made it possible to improve design productivity. However, an efficient design method for heterogeneous redundant design using HLS has not been established yet. Considering that the development of hardware systems using HLS is going to become the mainstream in near future, it is necessary to clarify the heterogeneous redundant design method using HLS as well as RTL.

2.3 Related works

There are a lot of research on high-reliability hardware systems, and also a number of reports on redundant design. Li et al. proposed a CPU-based method for a nuclear plant system [6], and Milluzzi et al. proposed a GPU-based method for image/signal processing in aerospace applications [7]. Redundant design approaches on an FPGA are also actively reported. For example, a design space exploration method for hardware design parameters such as reliability, area, latency, and dynamic power consumption has been proposed [8]. Since introducing redundancy can cause speed performance degradation, a method to mitigate it using partial reconfiguration has also been proposed [9]. In addition, there are research reports on redundant design using HLS from the viewpoint of design productivity [10, 11]. However, the targets of these work are homogeneous redundant designs, and the tolerance for common cause fault is not considered.

Heterogeneous redundant design is an effective method to improve the fault tolerance against common cause fault. Marques et al. proposed a heterogeneous fault-tolerant architecture “Lock-V,” which consists of Arm and RISC-V processors deployed on an FPGA, showing that it has an error correction capability for simulated common cause faults [12]. However, complex circuit configurations, such as multiplexing of different processor architectures, are not very desirable in terms of design cost. On the other hand, for

detecting software faults such as program bugs, Höller et al. proposed a method based on compiler options [4]. They reported that, by combining various optimization levels available in the widely-used C compilers, they were able to detect up to 70% of memory-related software bugs injected into programs. This method is useful for improving reliability while reducing cost, and it is expected that a diversification using FPGA design tool options can be applied to FPGA redundant design.

We have proposed a method to diversify the modules on FPGA circuits by only inserting simple directives into the RTL code [1, 2, 13]. It has been shown that the method can improve reliability against common cause fault while reducing design cost. However, the evaluation targets are simple arithmetic circuits and state machines described in RTL, not including pratical circuits, In the case of redundant design of complex circuits, there is a possibility that the inherent diversity is naturally introduced in the place-and-route process. Therefore, it is necessary to clarify whether the method is effective even for application-level circuits.

For further design cost reduction, HLS-based design is desirable since it provides designers with high level abstraction of hardware. In this paper, we firstly propose a heterogeneous redundant design method using HLS with low design cost. We use a PID control module as an evaluation target. The heterogeneous redundant designs of the target are implemented with the proposed method in RTL and HLS. Then, the implementation results are compared to evaluate the effectiveness of the method. In addition, we evaluate another method based on compiler options of FPGA tools and, together with results of HLS-based approach, aim to establish a directive-based desirable heterogeneous redundant design approach.

In a context of power efficient dynamic voltage scaling (DVS) systems, several circuit mechanisms with redundant structure of flip-flops (FFs) have been proposed. A Razor FF [14] consists a pair of FFs: a main FF and an additional shadow FF. The latter works synchronized with a delayed clock signal and thus has a more relaxed timing constraint than the former has. Therefore, it can be expected that the shadow FF latches the correct value even when the main FF has a timing fault caused by improperly lowered supply voltage, enabling the fault to be detected by comparing the output values of the two FFs. The correct value in the shadow FF can also be used for recovering from the timing fault, allowing the Razor FF to control the supply voltage speculatively based on the observed error rate. In contrast, Canary FF [15] is augmented by the shadow FF which shares the same clock signal with the main FF. On the other hand, the insertion of a delay line into the data input port brings a stricter timing constraint on the shadow FF. This enables to prevent the main FF from having the actual timing fault, by detecting its presage with the shadow FF. These techniques can also be considered as a sort of heterogeneous redundant FF designs, where the diversity is introduced with the delayed clock or delay line. Compared to the modular redundancy approach, their hardware costs are small. On the other hand, when both main and shadow FFs face the timing fault at the same time, they likely to latch the same errant value and fail to detector the error. The quantitative comparison is one of the interesting future work.

2.4 Scope and limit of this work

The main focus of this paper is to evaluate pros and cons of the proposed heterogeneous redundant design approaches. For pros, it is discussed how effectively the proposed approaches can introduce heterogeneity, by evaluating error detection rate with homogeneous redundant designs. For cons, the main concerns is how the heterogeneity lowers the performance of the circuit compared to the homogeneous design. Among various source of common cause fault, we focus on timing errors in this evaluation and perform overclock simulations with random stimulus to mimic the situations such as abnormal distortion in the clock signal or supply voltage. Although this random simulation approach is useful to asses the fundamental characteristics of the heterogeneity, consideration and analysis of the fault model are crucial for realistic systems design [16]. While we partially addressed this issue by using mathematical models in earlier work [1], it is not covered in the rest of this paper. Also, redundant module design covered in this paper is a portion of the design issues of fault tolerance. The topics such as error correction and recovery mechanisms, single points of failure issues, redundancy in other design levels are not covered in this paper and left for the future work.

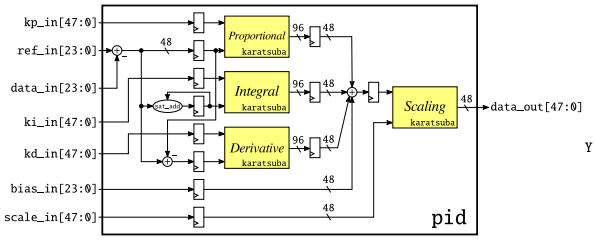


Figure 2: PID controller

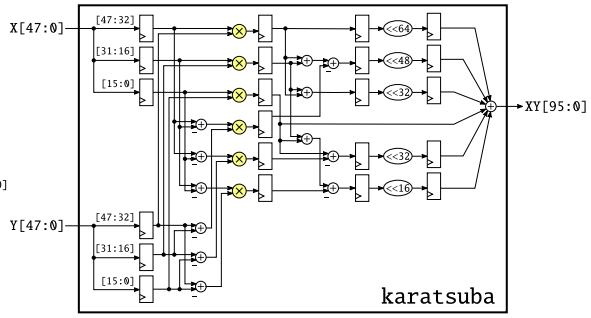


Figure 3: *Karatsuba* multiplier

3 Implementation of PID controller

PID control is a mainstay feedback control method especially in industrial application fields. In PID control, the control value $u(t)$ is found by calculating the proportional, integral, and derivative terms for the error value $e(t)$, which is difference between the observed value and the target value, as shown in the following equation:

$$u(t) = K_p \times e(t) + K_i \times \int_0^t e(\tau)\,\mathrm{d}\tau + K_d \times \frac{\mathrm{d}}{\mathrm{d}t}\,e(t), \tag{1}$$

where K_p, K_i, and K_d are the gains for each control term. Table 1 shows the list of inputs and outputs of the PID control module implemented by RTL (Verilog HDL) and HLS (C++) in this work.

Fig. 2 shows RTL structure of the PID controller module. This module includes saturation addition for $\int_0^t e(\tau)\,\mathrm{d}\tau$ in (1). As shown in Table 1, the PID control module used in this work contains 48-bit signed fixed-point numbers, so the multiplication of 48-bit X and Y is needed. To prevent the degradation of the maximum clock frequency, we divided 48-bit X and Y into three 16-bit segments as $X = x_2b^2 + x_1b + x_0$, $Y = y_2b^2 + y_1b + y_0$, $b = 2^{16}$, and then applied *Karatsuba*-algorithm.

The 9 multiplication operations of ordinary multiplication shown in (2) can be reduced to 6 multiplication operations in the *Karatsuba* multiplication of (3). Note that the multiplication by the power of b can be realized with a shift operation and does not require multipliers. Fig. 3 shows the structure of *Karatsuba* multiplier module in RTL.

$$X \times Y = x_2y_2\,b^4 + (x_2y_1 + x_1y_2)\,b^3 + (x_2y_0 + x_1y_1 + x_0y_2)\,b^2 + (x_1y_0 + x_0y_1)\,b + x_0y_0 \tag{2}$$

$$\begin{aligned} X \times Y = {} & x_2y_2\,b^4 + \left\{x_2y_2 + x_1y_1 - (x_2 - x_1)(y_2 - y_1)\right\} b^3 \\ & + \left\{x_2y_2 + x_1y_1 + x_0y_0 - (x_2 - x_0)(y_2 - y_0)\right\} b^2 \\ & + \left\{x_1y_1 + x_0y_0 - (x_1 - x_0)(y_1 - y_0)\right\} b + x_0y_0 \end{aligned} \tag{3}$$

Table 1: Inputs and output for PID controller

Port Name	Bit Width		Description
	Integer	Decimal	
data_in	24	0	Current observed value
ref_in	24	0	Target value
kp_in	24	24	Proportional gain
ki_in	24	24	Integral gain
kd_in	24	24	Derivative gain
bias_in	24	0	Bias value to remove offset
scale_in	24	24	Multiplied value with PID control result
data_out	24	24	Control value of PID

4 Resource-level approach using RTL/HLS

4.1 Methodology

A design flow of FPGA circuits consists of logic synthesis of design description, technology mapping, place and route, and generation of configuration data. The authors of [1, 2, 13] have proposed a heterogeneous redundant design method focusing on the diversity in the technology mapping process. In the technology mapping process, in which FPGA resources are assigned to the netlist generated by logic synthesis, the diversity can be introduced depending on a design strategy. For example, there are two types of multiplier on FPGAs: one is realized using look-up tables (LUTs), and the other is realized using hardwired digital signal processing (DSP) blocks. Normally, it is left up to the logic synthesis tool to decide which type of multipliers is used. However, by inserting directives into the RTL code, the designer can explicitly allocate resources. In Xilinx Vivado, to implement multiplication by LUTs, add the directive:

```
(* use_dsp = "no" *)
```
(4)

and by DSPs, add the directive:

```
(* use_dsp = "yes" *)
```
(5)

just before the RTL module or wire [17]. Also in HLS design, explicit specification of resources using directives is possible. In Xilinx Vivado HLS, for example, to implement multiplication `c=a*b` by LUTs, insert the pragma:

```
#pragma HLS RESOURCE variable=c core=Mul_LUT
```
(6)

and by DSPs, insert the pragma:

```
#pragma HLS RESOURCE variable=c core=DSP48
```
(7)

into the source code. This achieves the same effect as directives (4) and (5), for the descriptions in a higher abstraction level than RTL. Note that this `RESOURCE` directive is not currently supported for the arbitrary precision fixed-point data type provided by Vivado HLS (i.e., `ap_fixed`). Therefore, it is necessary to implement modules using the C plain old data (C POD) type, to which `RESOURCE` directive can be applied [18].

In addition, FPGA design tools such as Xilinx Vivado optimize the circuit area during logic synthesis and technology mapping. Especially in redundant designs, redundant modules, redundant registers, and redundant wires might be removed from the netlist during the optimization process. To prevent this, we need to add

```
(* dont_touch = "true" *)
```
(8)

to registers that are intentionally redundant, and add

```
(* keep_hierarchy = "yes" *)
```
(9)

to modules that we want to maintain a hierarchical structure. However, all of the above require only simple insertions of directives into RTL code, therefore the cost of implementing redundant designs is still low.

4.2 Evaluation

We evaluate how the proposed method improves the reliability of the target design, focusing mainly on the error detection rate of redundant designs. The evaluation targets are various types of redundant design of PID control modules: duplex/triplex, homogeneous/heterogeneous, and RTL/HLS implementations. Hereinafter, among the homogeneous redundant designs, those using DSPs are referred to as duplex *DSP-DSP* and triplex *DSP-DSP-DSP*, those using LUTs are similarly referred to as *LUT-LUT* and *LUT-LUT-LUT*. In the same way, the heterogeneous redundant designs based on the proposed method referred to as *DSP-LUT*, *DSP-DSP-LUT*, and *DSP-LUT-LUT*. These redundant design modules were synthesized, placed and routed on a Xilinx Spartan-7 FPGA evaluation board (xc7s50csga324-1) using Xilinx Vivado 2019.1. Table 2 shows the required amount of resources and $F_{\max}$ for each design.

Table 2: Required resources and $F_{\max}$ for RTL/HLS-based redundant designs

Design	Type	#LUTs	#FFs	#DSPs	$F_{\max}$ [MHz]
RTL	*DSP-DSP*	2,638	2,644	48	123.9
	DSP-LUT	10,436	4,676	24	100.3
	LUT-LUT	18,152	6,708	0	99.2
	DSP-DSP-DSP	4,108	4,686	72	116.6
	DSP-DSP-LUT	11,994	6,801	48	95.7
	DSP-LUT-LUT	19,501	8,601	24	94.6
	LUT-LUT-LUT	26,510	10,062	0	94.7
HLS	*DSP-DSP*	2,675	2,156	36	112.9
	DSP-LUT	8,041	3,143	18	99.7
	LUT-LUT	12,399	4,070	0	99.1
	DSP-DSP-DSP	3,714	3,234	54	107.2
	DSP-DSP-LUT	8,227	4,148	36	98.3
	DSP-LUT-LUT	12,739	5,062	18	93.1
	LUT-LUT-LUT	19,330	6,105	0	94.5

4.2.1 Evaluation method

As mentioned in Sections 1 and 2, a highly reliable design must handle not only soft errors but also common cause fault. In order to verify the effectiveness of the heterogeneous redundant design by the proposed method, simultaneous failures due to common cause fault should be emulated rather than single failure emulation with bit flip injection. Therefore, we performed gate-level timing simulation with post-mapped netlists with routing delay information, and emulated the common cause fault by setting the clock frequency to above the $F_{\max}$ of the circuit. This overclock simulation emulates timing errors due to delay variations by common cause fault such as abnormal voltage drop and abnormal temperature rise.

The inputs to the redundant module in the simulation were random numbers generated by the SystemVerilog `$random` system task with the same seed value. In addition, since the PID control module includes a register that inherits the previously computed value, once an error occurs in the register, it is considered to remain thereafter. Therefore, for each simulation trial, we directly set random values to registers as well as the input ports shown in Table 1. Considering the $F_{\max}$ values shown in Table 2, we performed simulations by changing the clock frequency from 100 MHz to 200 MHz by 1 MHz, and 100,000 input trials were evaluated for each frequency.

By comparing the outputs in gate-level simulation with those in logic simulation, the number of errors occurred, errors detected, and errors overlooked were counted. From the viewpoint of evaluating the error detectability, the average Hamming distance between the redundant outputs when errors occured was calculated for duplex redundant designs. Let $\mathbf{x}^{(k)} = [x_0^{(k)}, x_1^{(k)}, \ldots, x_{B-1}^{(k)}]$, and $\mathbf{y}^{(k)} = [y_0^{(k)}, y_1^{(k)}, \ldots, y_{B-1}^{(k)}]$ be the results of the k-th B-bit output of the two modules, respectively. The Hamming distance $d_H\left(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\right)$ is calculated as follows:

$$d_H\left(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\right) = \mathrm{popcnt}\left(\mathbf{x}^{(k)} \oplus \mathbf{y}^{(k)}\right), \tag{10}$$

where '$\oplus$' is XOR and $\mathrm{popcnt}(\mathbf{x}) = \sum_{i=0}^{B-1} x_i$ ($x_i \in \{0,1\}$). Then, the average Hamming distance $\bar{H}$ when errors occur n_e times out of n trials is calculated as:

$$\bar{H} = \frac{\sum_{k=1}^{n} d_H\left(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\right)}{n_e}, \tag{11}$$

where $B = 48$ and $n = 100{,}000$ in this evaluation. The larger $\bar{H}$ is, the more easily the errors are detected.

4.2.2 Results and discussion

First, the error detection rate for each redundant design is shown in Table 3. For the duplex redundant design, a large number of errors were overlooked in the homogeneous redundant designs *DSP-DSP* and *LUT-LUT*. This

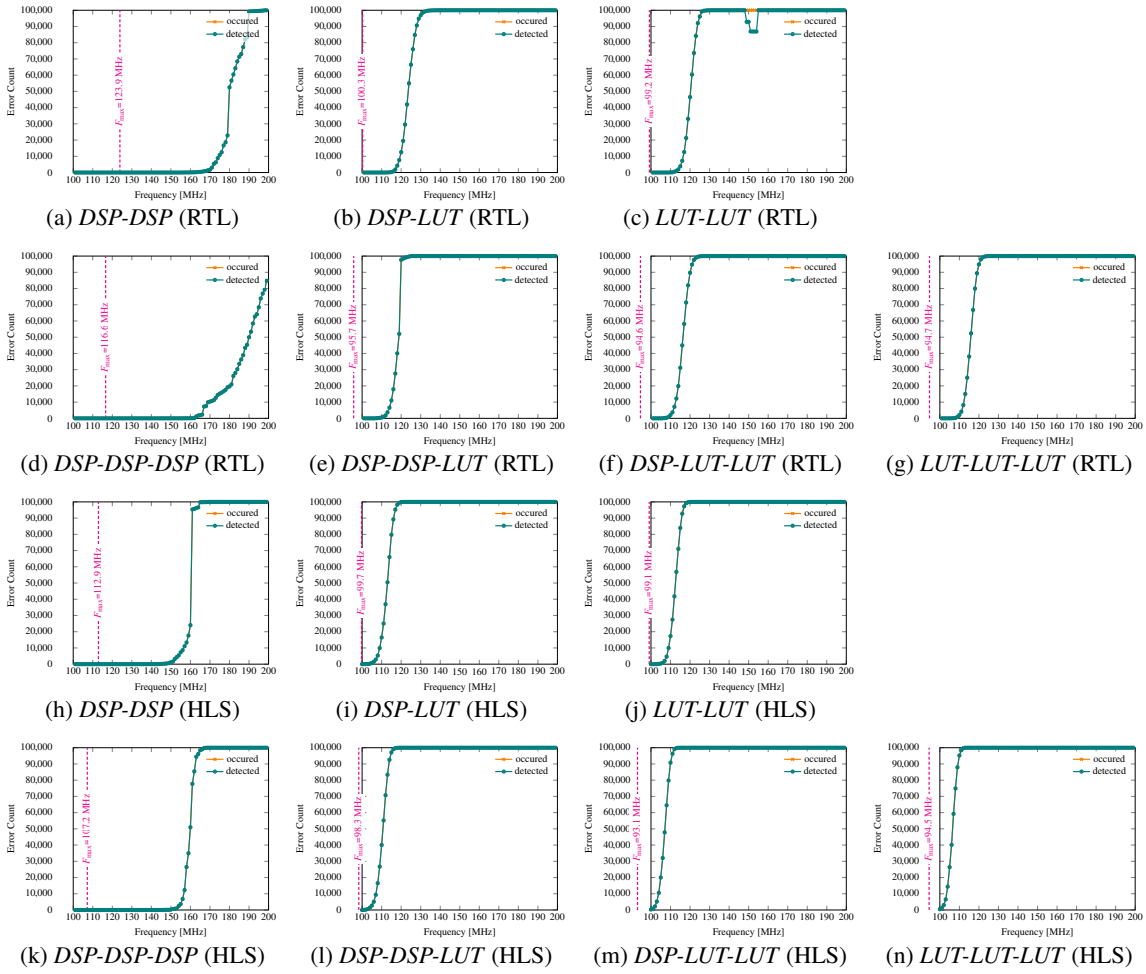
Table 3: Number of errors, detections, overlooks, and detection rate in each RTL/HLS-based redundant design

Design	Type	#Errors	#Detections	#Overlooks	Detection Rate [%]
RTL	*DSP-DSP*	1,797,447	1,796,884	563	99.969
	DSP-LUT	7,579,839	7,579,839	0	**100.000**
	LUT-LUT	7,935,786	7,866,206	69,580*	99.123
	DSP-DSP-DSP	1,170,700	1,170,668	32	99.997
	DSP-DSP-LUT	8,156,813	8,156,813	0	**100.000**
	DSP-LUT-LUT	8,314,671	8,314,671	0	**100.000**
	LUT-LUT-LUT	8,374,202	8,374,096	106	99.998
HLS	*DSP-DSP*	3,983,008	3,982,694	314	99.992
	DSP-LUT	8,677,491	8,677,491	0	**100.000**
	LUT-LUT	8,707,679	8,705,210	2,469	99.972
	DSP-DSP-DSP	3,991,322	3,991,322	0	100.000
	DSP-DSP-LUT	8,900,638	8,900,638	0	**100.000**
	DSP-LUT-LUT	9,249,176	9,249,176	0	**100.000**
	LUT-LUT-LUT	9,307,721	9,307,646	75	99.999

* Breakdown: 2,350 overlooks ∼120 MHz and 67,230 overlooks ∼150 MHz (cf. Fig. 5a)

is almost consistent with the evaluation results for simple circuits described in RTL [2, 13]. We also evaluated homogeneous TMR, which is commonly used for today's practical systems. Although their error detection rate was improved compared to the duplex designs, a certain number of error overlooks were observed in some triplex homogeneous redundant designs. On the other hand, all heterogeneous redundant designs achieved an error detection rate of 100% for both duplex and triplex designs, in RTL and HLS implementations. Next, the distribution of the number of errors and the number of overlooks per frequency for each design is shown in Fig. 4 and Fig. 5, respectively. As can be seen from these two figures, some designs tend to overlook errors in the frequency band "error jump" from the frequency where errors start to occur to the frequency where all outputs become errors. The *LUT-LUT* of RTL (Fig. 4c) overlooked many errors around 150 MHz other than "error jump," but the *DSP-LUT*, where one of LUTs is replaced with a DSP block, did not show this tendency, so the effect of diversifying is significant. These results indicate that heterogeneous redundant design by the proposed method has high fault tolerance against common cause fault even in an application-level circuit.

Fig. 6 shows comparison results of the average Hamming distances when errors occur calculated by (11) for each design. The average Hamming distance converged to around 24 bits in all the designs as the frequency was increased. This is half of the bit width of the output port (48 bits), which means that the flip occurred with a probability of 50% for each bit. The average Hamming distance for the HLS designs is larger than that of the RTL designs, showing that the HLS designs have a better ability to detect errors. One of the reasons for this difference would be the state machine. In HLS, a state machine is generated in the scheduling process to synthesize hardware from procedural descriptions in a programming language, and this process is performed even for arithmetic circuits such as PID controllers. On the other hand, only data paths are synthesized for this kind of simple pipelined arithmetic circuits in RTL designs. The difference is presumed to be caused by the common cause fault on this state machine in an overclocking environment. For the state machines, it has been suggested that diversification by encoding can increase the effectiveness of heterogeneous redundancy [13]. The default state encoding by Vivado HLS is 'one-hot,' and it does not have diversity with respect to the encoding of the redundant designs implemented in this work. However, the Vivado HLS also allows explicit specification of the encoding [18], and it is expected that more reliable heterogeneous redundant designs can be realized by adding diversity not only in resources but also in state encoding.



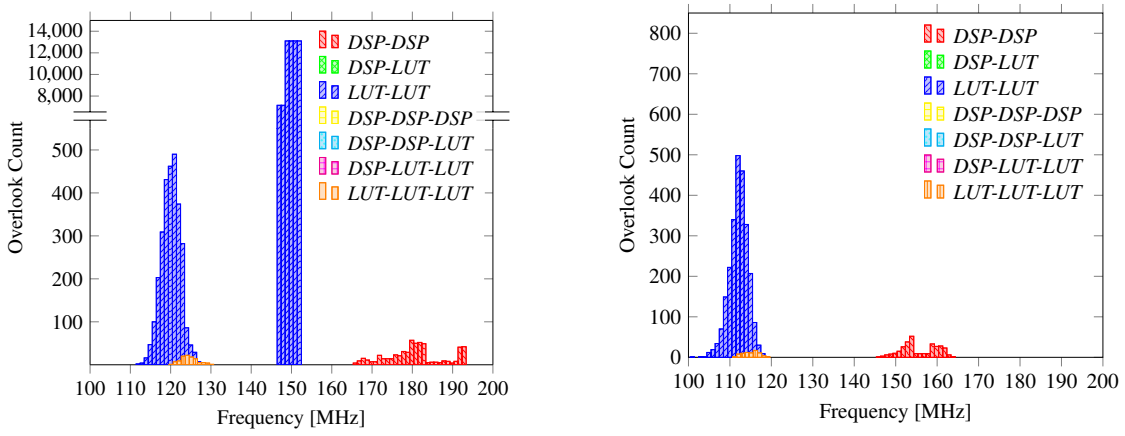
(a) *DSP-DSP* (RTL) (b) *DSP-LUT* (RTL) (c) *LUT-LUT* (RTL)

(d) *DSP-DSP-DSP* (RTL) (e) *DSP-DSP-LUT* (RTL) (f) *DSP-LUT-LUT* (RTL) (g) *LUT-LUT-LUT* (RTL)

(h) *DSP-DSP* (HLS) (i) *DSP-LUT* (HLS) (j) *LUT-LUT* (HLS)

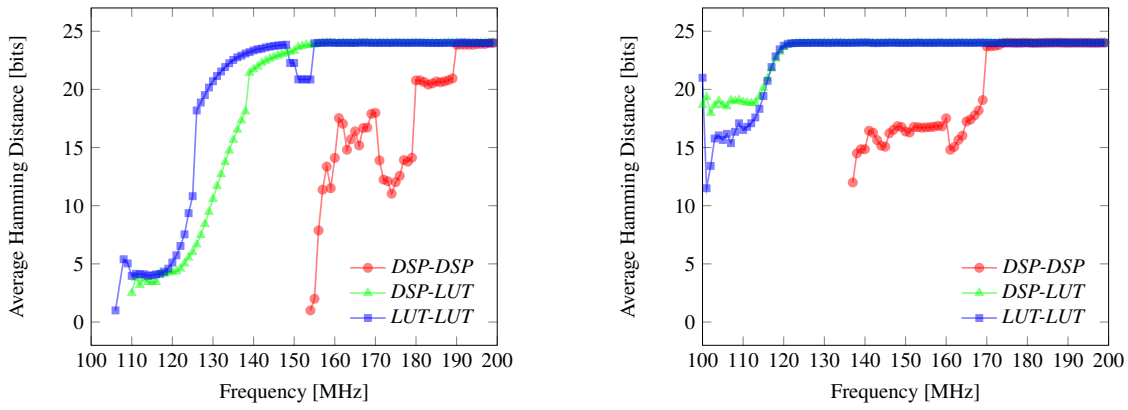
(k) *DSP-DSP-DSP* (HLS) (l) *DSP-DSP-LUT* (HLS) (m) *DSP-LUT-LUT* (HLS) (n) *LUT-LUT-LUT* (HLS)

Figure 4: Error occurences and detection count for RTL/HLS-based redundant designs



(a) RTL-based (b) HLS-based

Figure 5: Overlook count for RTL/HLS-based redundant designs



(a) RTL-based

(b) HLS-based

Figure 6: Average Hamming distance $\bar{H}$ for duplex RTL/HLS-based redundant designs

5 Strategy-level approach

5.1 Methodology

A strategy is a set of circuit optimization options provided by an FPGA design tool. There are several types of strategies, such as increasing operating frequency, reducing circuit area, and reducing power consumption. By setting this "strategy" according to the design objective, it is possible to comprehensively vary FPGA resources, state machine coding, and the limited number of fanouts. In this section, we also propose a *strategy*-based approach that is not specific to resources on FPGAs, and evaluate heterogeneous redundant designs consisting of modules optimized for different criteria such as performance and area.

5.1.1 Diversify logic synthesis strategy

All modules usually share the same strategy. In our method, however, this behavior must be suppressed so that the strategy can be given individually to each module. One way to give different strategies to multiple instances of the same module is to use out-of-context (OOC) design flow in the Vivado environment. In particular, logic synthesis process with the OOC design flow is called out-of-context synthesis, and be referred to as OOC-Synth. OOC-Synth runs independently of top-level logic synthesis for rarely-changing libraries such as Intellectual Property (IP) cores to mainly reduce design time. What is important here is that the logic synthesis strategy can be set separately for each OOC module. OOC-Synth redundant design method focuses on this IP flow and introduces diversity by giving different synthesis strategies to redundant modules. Fig. 7a shows an overview of OOC-Synth redundant design flow with IP cores.

5.1.2 Diversify place-and-route strategy

The OOC-Synth flow utilizing IP is performed independently, but the place-and-route process is performed together on the top-level module. Hence, we also propose an out-of-context implementation (OOC-Impl) redundant method. It focuses on the top-down hierarchical design flow with floorplanning in order to synthesize and place-and-route redundant modules independently of the top-level module and reflect the results in the top-level module.

This OOC-Impl flow includes 5 steps:

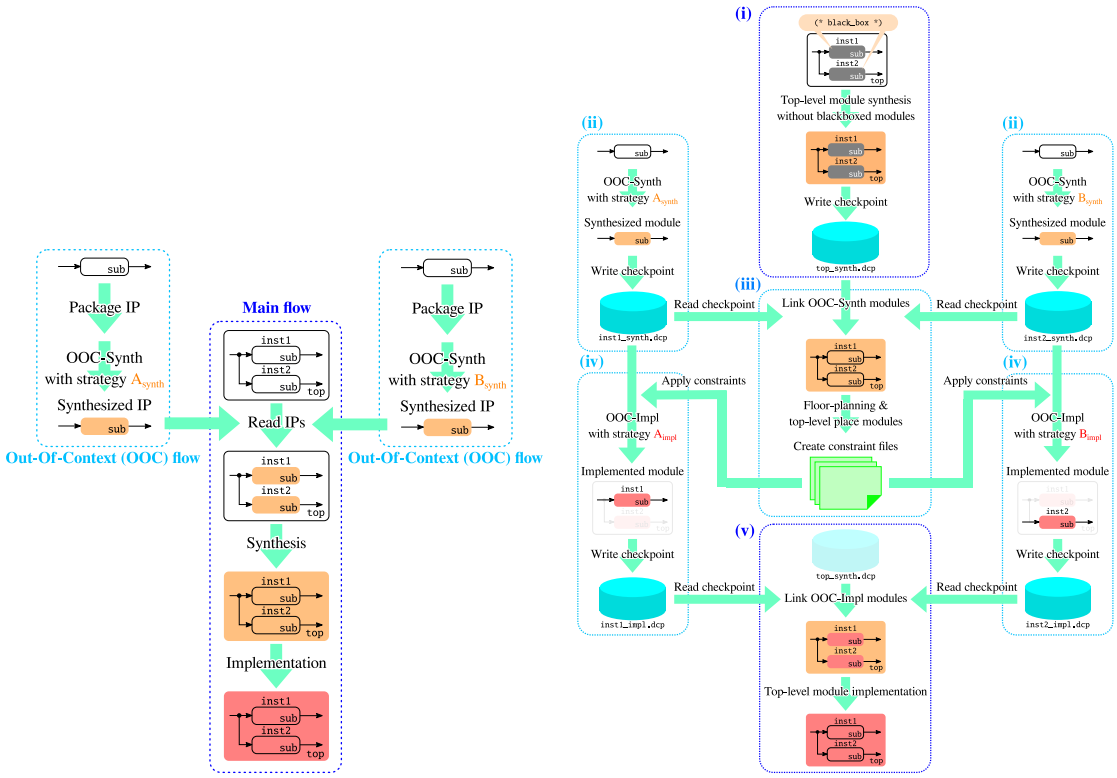
(i) Top-level synthesis

(ii) OOC-Synth of redundant modules

(iii) Top-level placement and generate constraints

(iv) OOC-Impl of redundant modules

(v) Top-level place-and-route



(a) OOC-Synth redundant design (b) OOC-Impl redundant design

Figure 7: Overviews of redundant design flow using out-of-context run

and its overview is shown in Fig. 7b. In the OOC flow, the information of the top-level modules is necessary in order to independently run logic synthesis and place-and-route of redundant modules and marge their results into the top-level module. In step (i), it is necessary to suppress the logic synthesis of redundant modules by inserting the directive:

```
(* black_box *)
```
(12)

just before the instantiation description of redundant modules. In parallel with step (i), step (ii) runs OOC-Synth of redundant modules with appropriate strategy settings. In step (iii), floorplanning is performed so that the place-and-route of each redundant module does not conflict to realize OOC-Impl, and timing-driven constraints used in step (iv) are generated [19]. Then, step (v) applies these constraints and implements each redundant module with its own strategy, and the top-level module is finally configured. These steps include read and write of the design archive files (DCP file in Xilinx Vivado environment).

5.2 Evaluation

The area- and performance-optimized modules are implemented and referred to as *area* and *perf*, respectively. For OOC-Impl design flow, we also prepared a module *power* optimized for dynamic power consumption. Table 4 shows what strategies were selected for redundant modules in OOC-Synth and OOC-Impl flows. These strategies were the optimal pairs for each criteria after running all the combinations. Combining these optimized modules, homogeneous *default-default*, *area-area*, *perf-perf*, *area-area-area*, *perf-perf-perf*, and heterogeneous *area-perf*, *area-area-perf*, *area-perf-perf* and *area-perf-power* redundant designs were configured. For the OOC-Impl redundant design, we floorplanned on the xc7s50csga324-1 FPGA as shown in Fig. 8. The required amount of resources and $F_{\max}$ for these designs are shown in Table 5.

Table 4: Strategy settings for each redundant module

Flow	Name	Strategy	
		Synthesis	Implementation
OOC-Synth	*default*	Vivado Synthesis Defaults	–
	area	`Flow_AreaOptimized_high`	–
	perf	`Flow_PerfOptimized_high`	–
OOC-Impl	*default*	Vivado Synthesis Defaults	Vivado Implementation Defaults
	area	`Flow_AreaOptimized_medium`	`Area_Explore`
	perf	`Flow_PerfOptimized_high`	`Performance_EarlyBlockPlacement`
	power	Vivado Synthesis Defaults	`Power_DefaultOpt`

Table 5: Required resources and $F_{\max}$ for each strategy-based redundant design

Flow	Type	#LUTs	#FFs	#DSPs	$F_{\max}$ [MHz]
OOC-Synth	*default-default*	2,639	2,908	48	123.9
	area-area	2,579	2,908	48	120.1
	area-perf	2,688	3,068	48	121.7
	perf-perf	2,812	3,228	48	132.1
	area-area-area	3,857	3,966	72	119.5
	area-area-perf	3,966	4,126	72	119.8
	area-perf-perf	4,085	4,286	72	123.5
	perf-perf-perf	4,201	4,446	72	122.1
OOC-Impl	*default-default*	2,637	2,908	48	119.7
	area-area	2,575	2,908	48	125.2
	area-perf	2,681	3,068	48	120.9
	perf-perf	2,798	3,228	48	132.9
	area-area-area	3,842	3,966	72	119.5
	area-area-perf	3,963	4,126	72	119.5
	area-perf-perf	4,080	4,286	72	124.7
	perf-perf-perf	4,194	4,446	72	131.4
	area-perf-power	3,987	4,126	72	120.2

5.2.1 Evaluation method

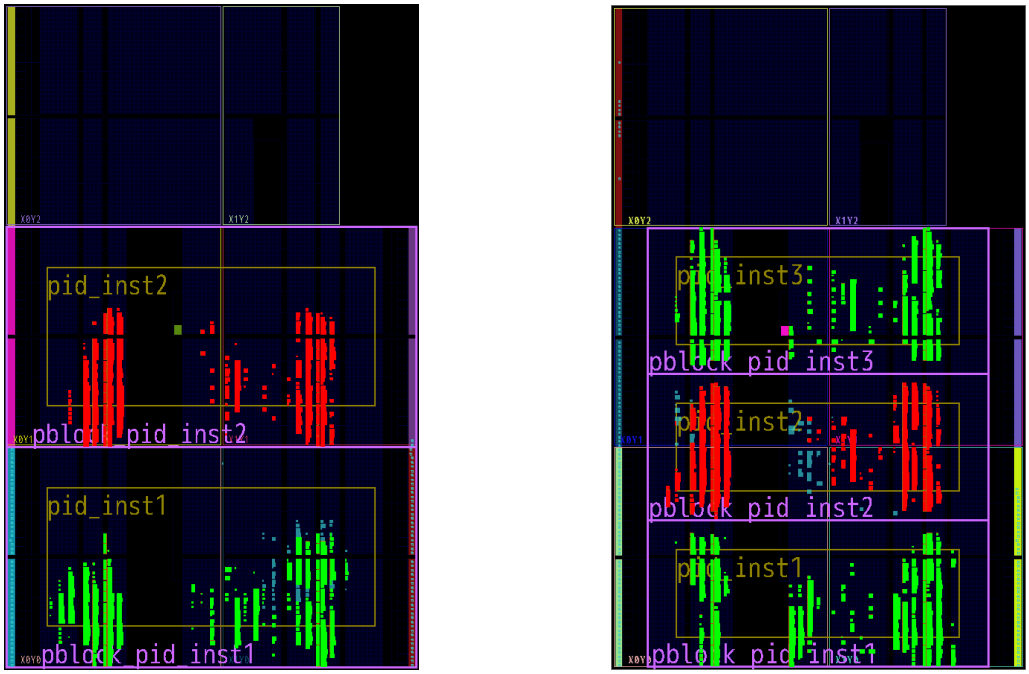
The evaluation range of 100 MHz to 200 MHz in Section 4.2 was not enough to show the difference in detection capability of redundant designs, so we used the range from 100 MHz to 300 MHz in this section. One of the reason for this is that "error jumps" of each design did not end within 200 MHz due to optimizations. In addition to the number of error detections and average Hamming distance, we also evaluated the number of correctable errors for TMR designs in this section. Error correction is typically performed by majority voting of outputs. When 3 redundant module outputs are $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$, respectively, the majority vote output $\mathbf{y}$ is given by the following equation:

$$\mathbf{y} = (\mathbf{y}_1 \,\&\, \mathbf{y}_2) \mid (\mathbf{y}_2 \,\&\, \mathbf{y}_3) \mid (\mathbf{y}_3 \,\&\, \mathbf{y}_1). \tag{13}$$

If an error occurs and the majority output $\mathbf{y}$ matches the true value, then the error is correctable.

5.2.2 Results and discussion

The number of error detections and corrections for each redundant design are shown in Table 6, and their distributions are shown in Fig. 9 and Fig. 10. In addition, Fig. 11 shows the distribution of overlooks, and the number of overlooked errors in the *homogeneous* redundant design was higher in OOC-Impl designs than



(a) duplex redundant design (*area-perf*)

(b) triplex redundant design (*area-area-perf*)

Figure 8: Floorplan of OOC-Impl redundant designs (green: area-optimized, red: performance-optimized)

OOC-Synth ones. One possible reason for this is the fact that the redundant modules were implemented independently without interaction, resulting in little diversity in place-and-route process, leading to more overlooked errors. However, thanks to the diversification of both logic synthesis and place-and-route strategies, the number of overlooked errors in *heterogeneous* redundant designs was limited to a few: *area-perf-perf* TMR design achieved a detection rate of 100%, and *area-perf-power* was extremely close to 100%.

TMR designs can correct errors, and Fig. 9e–9h and Fig. 10e–10i show the correctable error distributions. The correctable errors are distributed over a slightly wider frequency range than the overlooked errors, and the number of correctable errors decreases as the frequency increases. Since most of the errors are uncorrectable in an excessively overclocked environment, it can be inferred that the majority vote in (13) was not determined. For error correction rate of TMR designs shown in Table 6, the average of the *homogeneous* redundant designs is 10.91% while that of *heterogeneous* ones is 17.96%, and the error correction capability of OOC-Impl *heterogeneous* redundant design tends to be higher.

However, in critical situations where common cause faults actually occur, the error detection should be considered more important than the error correction because once an error begins to appear, it must be stopped even if it can be corrected. From the results of the average Hamming distance shown in Fig. 12, the strategy-based heterogeneous redundancy design appears to have a slight advantage over the homogeneous redundancy designs in terms of error detection easiness when errors occurring, but the error detection capability is inferior to the multiplier resource diversification method. Although there are still many combinations of strategies to be applied to redundant modules, the introduction of strategy diversification alone was not sufficiently effective to improve the fault tolerance against common causes in this evaluation. Hence, this approach can be combined with other approach as needed to complement the system reliability. It should be emphasized that we can easily automate the OOC flows and we can apply this approach to circuits other than DSP-friendly applications, such as PID controller.

The results so far can support the conjecture that diversity of multiplier resources was more effective and the greater the difference in the place-and-route between redundant modules, the better the detection capability. If we consider the netlists after place-and-route process as a graph, for example, and define the "similarity" between redundant modules, we expect the number of overlooked errors to increase in proportion to this similarity. It is important to establish more desirable design approaches for directive-based diversification by

Table 6: Number of errors, detections, correctables, and these rates in each strategy-based redundant designs

Flow	Type	#Errors	#Detections	#Overlooks	#Correctables	#Uncorrectales	Detection Rate [%]	Correction Rate [%]
OOC-Synth	*default-default*	11,697,443	11,696,826	617	–	–	99.9948	–
	area-area	10,283,747	10,282,365	1,382	–	–	99.9866	–
	area-perf	11,441,703	11,439,822	1,881	–	–	99.9836	–
	perf-perf	9,292,148	9,287,108	5,040	–	–	99.9458	–
	area-area-area	10,177,715	10,177,047	668	831,971	9,345,744	99.9934	8.1744
	area-area-perf	11,302,295	11,302,279	16	1,607,916	9,694,379	99.9999	**14.2265**
	area-perf-perf	10,448,374	10,447,915	459	989,657	9,458,717	99.9956	**9.4719**
	perf-perf-perf	11,374,566	11,374,552	14	969,082	10,405,484	99.9999	8.5197
OOC-Impl	*default-default*	12,173,796	12,166,366	7,430	–	–	99.9390	–
	area-area	10,884,226	10,871,363	12,863	–	–	99.8818	–
	area-perf	10,972,983	10,972,259	**724**	–	–	**99.9934**	–
	perf-perf	9,035,186	9,023,609	11,577	–	–	99.8719	–
	area-area-area	13,195,142	13,194,979	163	1,958,094	11,237,048	99.9988	14.8395
	area-area-perf	12,874,577	12,874,577	**0**	2,205,680	10,668,897	**100.0000**	**17.1321**
	area-perf-perf	11,844,096	11,844,069	**27**	3,795,156	8,048,940	**99.9998**	**32.0426**
	perf-perf-perf	9,311,616	9,311,305	311	1,128,074	8,183,542	99.9967	12.1147
	area-perf-power	13,279,890	13,279,889	**1**	2,249,211	11,030,679	~**100.0000**	**16.9370**

conducting more quantitative analysis for heterogeneous redundant design.

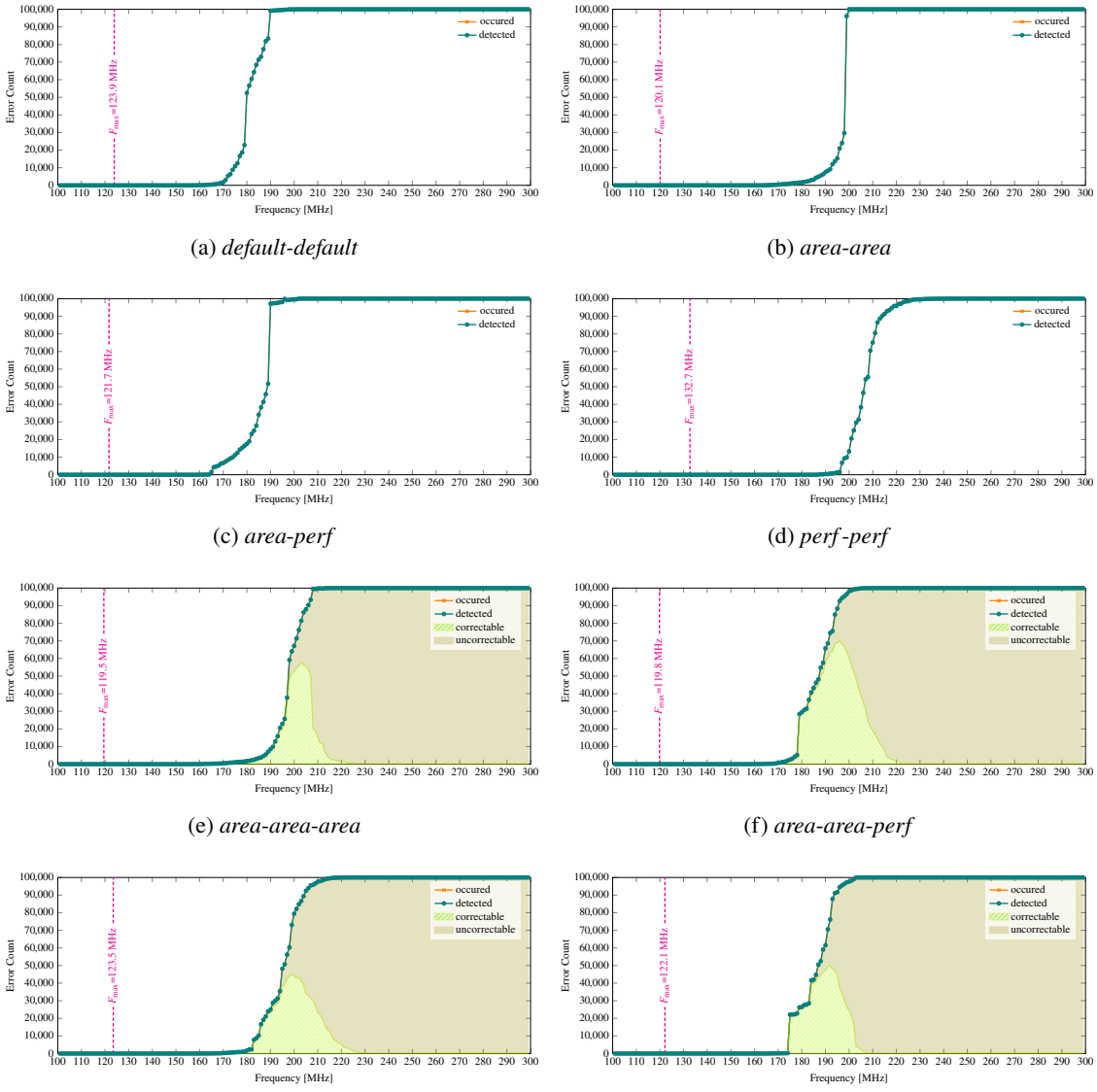
5.2.3 Comparison to the resource-level approach

To make comparative discussions between the resource-level and strategy-level approaches, the results for two representative implementations, the RTL-base resource-level and OOC-Impl strategy-level redundant design, are summarized in Table 7. In terms of the error detection rate, the heterogeneous designs (*DSP-LUT* and *area-perf*) show the best rate in each group. However, while *DSP-LUT* achieves the 100% of detection rate, *area-perf* only shows 99.993% of detection rate, overlooking 724 errors. Although the heterogeneity provided by the strategy-level approach shows an advantage compared to the homogeneous designs, the heterogeneity offered by the resource-level approach is superior to the strategy-level counterpart. This means that the resource-level approach makes more significant difference between the two redundant modules.

The Fmax for *area-perf* (120.9 MHz) is decreased by 8.5% compared to that for the fastest homogeneous counterpart, which is *perf-perf* (132.1 MHz). On the other hand, the Fmax for *DSP-LUT* (100.3 MHz) is degraded by 19% compared to the fastest homogeneous design, which is *DSP-DSP* (123.9 MHz). The increase in the error detection rate that heterogeneous redundant designs achieve is a result of a tradeoff with this performance overhead, and the overhead to be paid is much smaller for the strategy-level approach. This is one of apparent advantages that the strategy-level approach has over the resource-level approach.

Another tradeoff is reflected in resource utilization. For example, heterogeneous *DSP-LUT* requires nearly 4 times LUTs compared to homogeneous *DSP-DSP* as a result of prohibiting the use of half of the DSP blocks. On the other hand, among the implementations in the strategy-approach, differences in resource utilization are not considerable. Even considering out-of-context (OOC) implementation for the strategy-level approach has an area overhead related to fixed floor planning, the circuits generated with the strategy-level approach seem to be more efficient than those of the resource-level approach in terms of the balance between speed and area.

In terms of ease of design, the resource-level approach might be preferable, since circuit designers need to handle floor planning issues for the out-of-context implementation in the strategy-level approach. However, most steps can be automatically carried out using a script file, which effectively reduces the difference in ease of design between two approaches. As a conclusion, the resource-level and strategy-level approaches offer a tradeoff between reliability and efficiency. A desirable choice may depend on which factor is more important in each application. Theoretically, it is also possible to combine these two approaches. For instance, a heterogeneous redundant design consisting of "LUT" module in the resource-level approach and "*perf*" module in the strategy-level approach can be implemented. However, given that tradeoff between the two approaches, it may become halfway measures.



(a) *default-default*

(b) *area-area*

(c) *area-perf*

(d) *perf-perf*

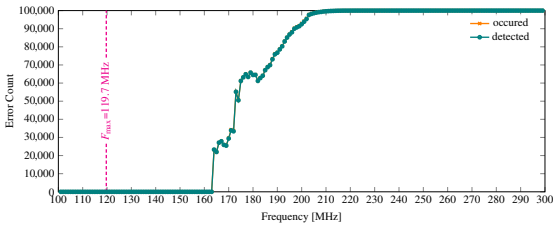
(e) *area-area-area*

(f) *area-area-perf*

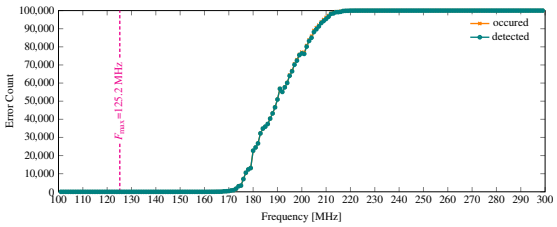
(g) *area-perf-perf*

(h) *perf-perf-perf*

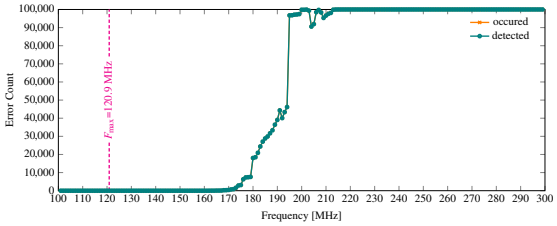
Figure 9: Error occurences and detection/correction count for redundant designs using *OOC-Synth*



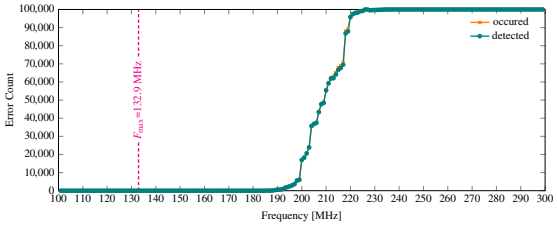
(a) *default-default*



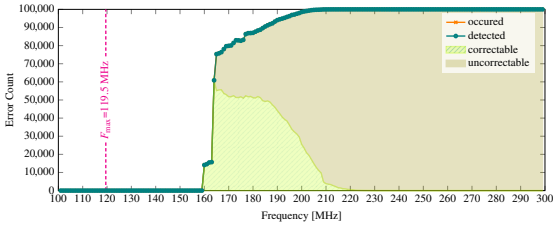
(b) *area-area*



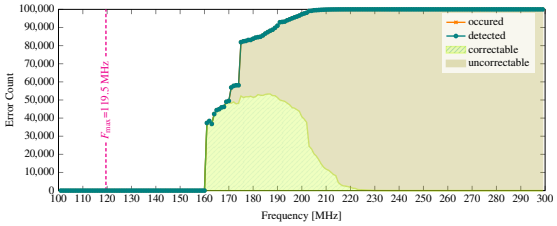
(c) *area-perf*



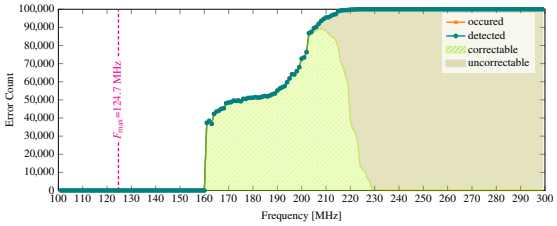
(d) *perf-perf*



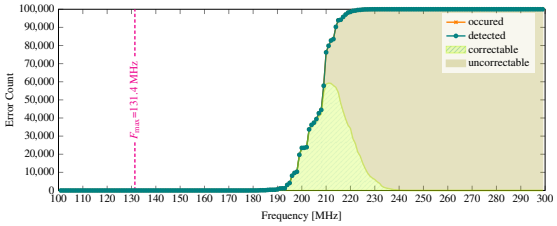
(e) *area-area-area*



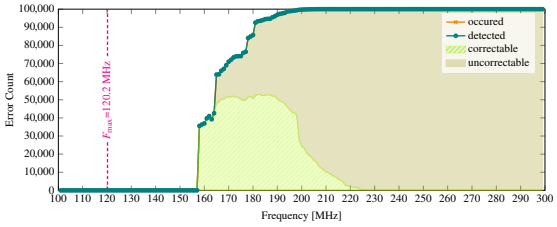
(f) *area-area-perf*



(g) *area-perf-perf*

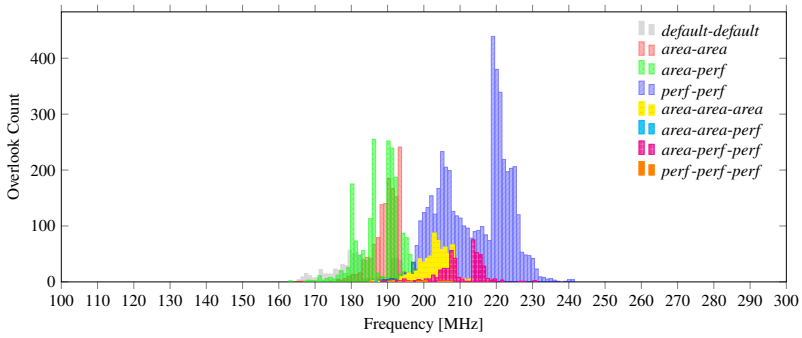


(h) *perf-perf-perf*

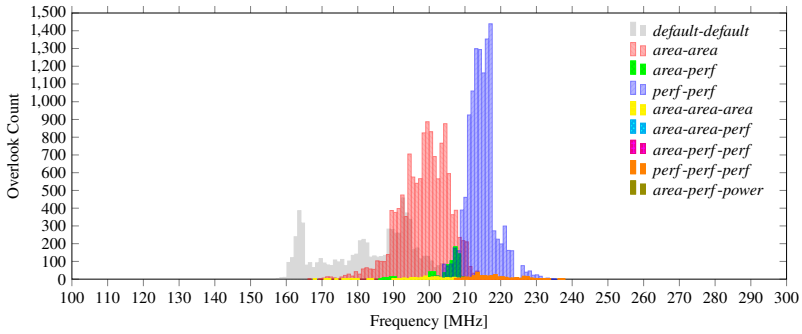


(i) *area-perf-power*

Figure 10: Error occurences and detection/correction count for redundant designs using *OOC-Impl*

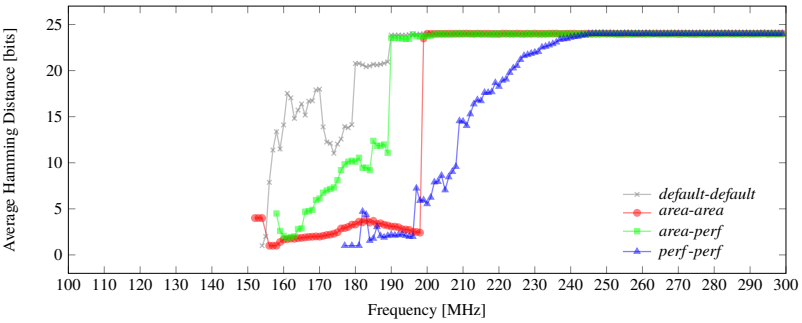


(a) OOC-Synth

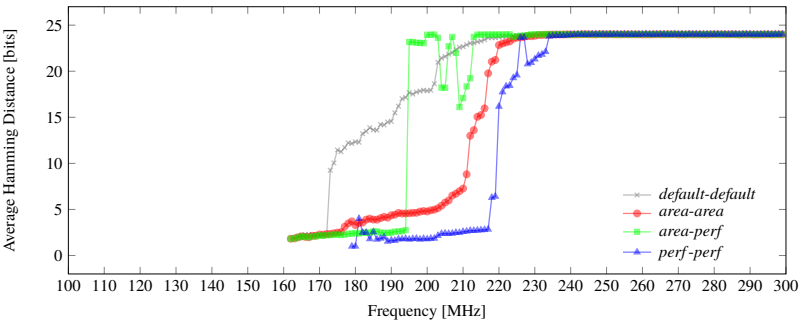


(b) OOC-Impl

Figure 11: Overlook count for strategy-based redundant designs



(a) OOC-Synth



(b) OOC-Impl

Figure 12: Average Hamming distance $\bar{H}$ for strategy-based duplex redundant designs

Table 7: Comparison between resource-level and strategy-level approaches

Approach	Type	Fmax [MHz]	Detection rate [%]	#Overlooks	LUTs	FFs	DSPs
Resource	DSP-DSP	123.9	99.969	563	2,638	2,644	48
	DSP-LUT	100.3	100.000	0	10,436	4,676	24
	LUT-LUT	99.2	99.123	69,580	18,152	6,708	0
Strategy	default-default	119.7	99.939	7,430	2,639	2,908	48
	area-area	125.2	99.882	12,863	2,579	2,908	48
	area-perf	120.9	99.993	724	2,688	3,068	48
	perf-perf	132.1	99.872	11,577	2,812	3,228	48

6 Conclusion

We firstly proposed a heterogeneous redundant design method with lower design cost by HLS. We evaluated the heterogeneous redundant design by utilizing the diversity of logic elements on FPGAs, targeting application-level PID controller rather than simple circuits. Gate-level simulation with delay information was used to emulate the occurrence of common cause fault in redundant modules by running them at frequencies exceeding their $F_{\max}$. The simulation results show that, compared to conventional homogeneous redundant design, the heterogeneous ones by the proposed method have a higher ability to detect errors caused by common fault in both RTL and HLS, even for application-level circuits. Our future work includes the verification of the reliability of heterogeneous redundant design that diversifies the state encoding or that combines encoding and resources using HLS. A strategy-based approach was also evaluated, using the same RTL PID controller. Although the error detection rate of the strategy-based approach was lower than that of the resource level, a certain improvement in error detection/correction capability was evaluated in the heterogeneous redundant design by diversifying both synthesis and place-and-route optimization. It is suggested that reducing the similarity between redundant modules contributes to the reliability of heterogeneous redundant designs, and topological analysis will help to establish a more desirable heterogeneous redundant design approach.

References

[1] Kenichi Morimoto, Yuichiro Shibata, Yudai Shirakura, Hidenori Maruta, Masaharu Tanaka, and Fujio Kurokawa. Diversity diagnostic for new FPGA based controller of renewable energy power plant. *Int'l Journal of Renewable Energy Research (IJRER)*, 7:1403–1412, 2017.

[2] Yudai Shirakura, Taisei Segawa, Yuichiro Shibata, Kenichi Morimoto, Masaharu Tanaka, Masanori Nobe, Hidenori Maruta, and Fujio Kurokawa. A redundant design approach with diversity of FPGA resource mapping. In *Int'l Symposium on Applied Reconfigurable Computing (ARC)*, volume 9625, pages 119–131, 2016.

[3] Taichi Saikai, Kotoko Miyata, Taito Manabe, and Yuichiro Shibata. Evaluation of an HLS-based heterogeneous redundant design approach for functional safety systems on FPGAs. In *Proc. Int'l Symposium on Computing and Networking (CANDAR)*, pages 162–167, 2021.

[4] Andrea Höller, Nermin Kajtazovic, Tobias Rauter, Kay Römer, and Christian Kreiner. Evaluation of diverse compiling for software-fault detection. In *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 531–536, 2015.

[5] Int'l Electronical Commission. Functional Safety. IEC 61508, 2010.

[6] You-ran Li, Wei Sun, Liang Zhou, and Long-quang Zhang. Analysis of CPU redundant configuration for the safety DCS of NPP. In *Proc. International Symposium on Software Reliability, Industrial Safety, Cyber Security and Physical Protection for Nuclear Power Plant (SICPNPP)*, pages 33–40, 2017.

[7] Andrew Milluzzi, Alan George, and Alan George. Exploration of TMR fault masking with persistent threads on Tegra GPU SoCs. In *Proc. IEEE Aerospace Conference*, pages 1–7, 2017.

[8] Jahanzeb Anwer, Marco Platzner, and Sebastian Meisner. FPGA redundancy configurations: An automated design space exploration. In *Proc. IEEE Int'l Parallel Distributed Processing Symposium Workshops*, pages 275–280, 2014.

[9] Atsuhiro Kanamaru, Hiroyuki Kawai, Yoshiki Yamaguchi, and Morisothi Yasunaga. Tile-based fault tolerant approach using partial reconfiguration. In *Proc. Int'l Workshop on Applied Reconfigurable Computing (ARC)*, pages 293–299, 2009.

[10] David Wilson, Aniruddla Shastri, and Greg Stitt. A high-level synsthesis schedulling and binding heuristic for FPGA fault tolerance. *Int'l Journal of Reconfigurable Computing*, 2017:1–17, 2017.

[11] Jakub Lojda, Jakub Podivinsky, Martin Krcma, and Zdenek Kotasek. HLS-based fault tolerance approach for SRAM-based FPGAs. In *Proc. Int'l Conference on Field-Programmable Technology (FPT)*, pages 301–302, 2016.

[12] Ivo Marques, Rodrigues Cristiano, Adriano Tavares, Sandro Pinto, and Tiago Gomes. Lock-V: A heterogeneous fault tolerance architecture based on Arm and RISC-V. *Microelectronics Reliability*, 120:114–120, 2021.

[13] Takanori Itagawa, Ryo Kamasaka, and Yuichiro Shibata. A simple heterogeneous redundant design method for finite state machines on FPGAs. In *Proc. Int'l Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 453–461, 2019.

[14] Dan Ernst, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler, David Blaauw, Todd Austin, Krisztian Flautner, and Trevor Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proc. Annual IEEE/ACM Int'l Symposium on Microarchitecture (MICRO-36)*, pages 7–18, 2003.

[15] Toshinori Sato and Yuji Kunitake. A simple flip-flop circuit for typical-case designs for DFM. In *Proc. International Symposium on Quality Electronic Design (ISQED)*, pages 539–545, 2007.

[16] Josef Börcsök, Sebastian Schaefer, and Evzudin Ugljesa. Estimation and evaluation of common cause failures. In *Proc. Second International Conference on Systems (ICONS)*, pages 41–41, 2007.

[17] Xilinx Inc. *Vivado Design Suite User Guide Synthesis*, UG901 (v2019.1) edition, 2019.

[18] Xilinx Inc. *Vivado Design Suite User Guide High-Level Synthesis*, UG902 (v2019.1) edition, 2019.

[19] Xilinx Inc. *Vivado Design Suite User Guide Hierarchical Design*, UG946 (v2018.3) edition, 2018.