In-Place Decomposition for Robustness in FPGA

Ju-Yueh Lee, Zhe Feng, and Lei He
Electrical Engineering Department
University of California, Los Angeles

ABSTRACT

The programmable logic block (PLB) in a modern FPGA features a built-in carry chain (or adder) and a decomposable LUT, where such an LUT may be decomposed into two or more smaller LUTs. Leveraging decomposable LUTs and underutilized carry chains, we propose to decompose a logic function in a PLB into two subfunctions and to combine the subfunctions via a carry chain to make the circuit more robust against single-event upsets(SEUs). Note that such decomposition can be implemented using the decomposable LUT and carry chain in the original PLB without changing the PLB-level placement and routing. Therefore, it is an in-place decomposition (IPD) with no area and timing overhead at the PLB level and has an ideal design closure between logic and physical syntheses. For 10 largest combinational MCNC benchmark circuits with a conservative 20% utilization rate for carry chain, IPD improves MTTF (mean time to failure) by 1.43 and 2.70 times respectively, for PLBs similar to those in Xilinx Virtex-5 and Altera Stratix-IV.

1. INTRODUCTION

Modern field programmable gate arrays (FPGA) uses ever-advancing fabrication technologies for higher density and reduced power, but at the cost of more vulnerability to single event upsets (SEU) caused by supply voltage fluctuations, electromagnetic coupling and environmental radiation. Because FPGAs apply memory cells (primarily SRAM) to implement logic functions and interconnects, an SEU can have a permanent impact on the logic function and interconnect, which can only be resolved by a reprogramming operation. Therefore, SEUs have a much bigger impact for FPGA than for ASIC, and it significantly reduces MTTF, a system level measurement of reliability. Since an increasing number of FPGA chips are used for deployed systems ranging from internet line cards to enterprise severs [1] rather than prototypes, MTTF is among the most important design objectives for FPGA.

Robustness with respect to SEUs in FPGAs have been extensively studied in the literature [2]. Specific FPGA architectures have been developed such as radiation hardened FPGAs from Xilinx [3] and anti-fuse based FPGAs from Actel [4]. Circuit redundancy such as triple modular redundancy (TMR) [5] and quadruple modular redundancy (QMR) have also been proposed. However, the aforementioned techniques have high overhead on cost, area or power. This makes them too expensive for non-mission critical applications such as communication systems.

Recently, several logic resynthesis algorithms have been proposed to improve robustness for FPGA with minimal area, power, or performance penalties. ROSE [6] iteratively remaps logic blocks using a robust block template with path reconvergence. However, ROSE can change the topology of the LUT-level network, resulting in physical re-synthesis and slowing down design closure. IPR [7] performs logic transformation while preserving the topology of the LUT-level network, and removes the aforementioned design closure problem. In essence, ROSE and IPR use logic masking/redundancy to minimize the impacts of SEUs, and their MTTF improvements could be limited when circuits are heavily optimized for area and therefore have little implicit logic redundancy.

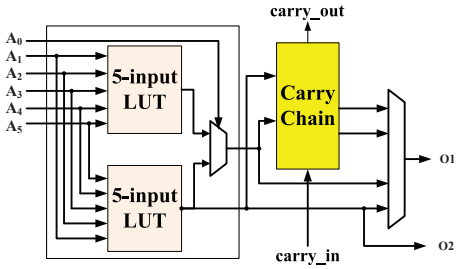
The state of art FPGAs such as Xilinx Vertix-5 and Altera Stratix-IV [8, 9] use decomposable LUTs, where an LUT can be decomposed into two or more smaller LUTs and a second output pin is also provided (see Fig. 1(a) and Fig. 1(b)). Leveraging decomposable LUTs and under-utilization of large-sized LUTs, [10] proposed to duplicate the logic inside a partially used LUT, and increase MTTF without increasing the number of LUTs. However, the encoding to combine the duplicated logic is done in the fanout LUTs. This leads to extra interconnects and potentially heavy area penalty and slows down design closure between logic and physical resyntheses.

In addition to the decomposable LUT, the programmable logic block (PLB) in modern FPGAs also have a dedicated carry chain or adder. While the carry function can be implemented by LUTs,these carry chain circuits are built in as alternative circuits to obtain high speed for applications - such as networking - with extensive carry computation. Although the state of art synthesis tools have been developed to leverage carry chains for arithmetic computation, the chip level carry chain utilization is typically less than 20% and LUT and carry chain inside a PLB are selfomly used simultaneously.

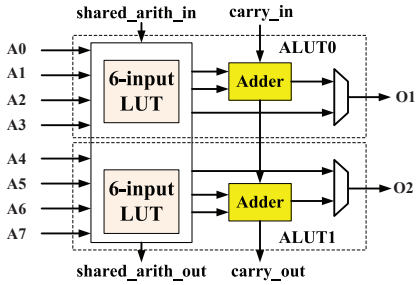
This paper proposes an in-place decomposition (IPD) for robustness in FPGAs. IPD decomposes the logic function originally implemented by one decomposable LUT into two subfunctions to be implemented by smaller LUTs and to combine (also called converge in this paper) the subfunctions by the carry chain. This decomposition introduces re-

This work was partially sponsored by Cisco and JPL.

978-1-4244-8194-1/10/$26.00 ©2010 IEEE



(a) Xilinx Virtex-5 6-input LUT architecture



(b) Altera Stratix-IV ALM architecture

Figure 1: Dual-output LUT FPGA architecture

dundancy as the original logic function is now implemented with extra circuitry (i.e., carry chain) and such redundancy is used explicitly to make the circuit more robust for SEUs.

IPD offers clear advantages over duplication in [10]. Firstly, duplication implies two identical subfunctions. It is a simplified case of decomposition which enables distinguished subfunctions with more potential for improvement. Secondly, IPD performs logic decomposition and converging within the original PLB. It does not change PLB level placement and routing, and therefore there is no PLB level penalty on timing, area and design closure. In contrast, the duplication (namely FMD) algorithm from [10] carries out duplication and encoding (called converging in this paper) in multiple PLBs. This needs extra interconnects between PLBs, resulting in a heavy penalty in area, delay and design closure.

We have developed an ILP (integer linear programming) algorithm to solve IPD optimally inside each PLB and applied ILP iteratively to PLBs at the chip level. For 10 largest combinational MCNC benchmark circuits synthesized by ABC mapper, FMD from [10] improves MTTF by 10% on average, but IPD improves MTTF by 1.43× for PLB similar to that in Xilinx Vertex-5 and by 2.70× for PLB similar to that in Altera Stratix-IV, when the conservative 20% utilization rate of carry chain is assumed.

The rest of this paper is organized as follows. Section 2 presents the preliminaries and problem formulation. Section 3 proposes IPD algorithms and properties. The experimental results are given in Section 4 and the paper is concluded in Section 5. To the best of our knowledge, this paper is the first systematic study on robustness using both the built-in carry chain (or adder) and decomposable LUT features of modern FPGAs.

2. PRELIMINARIES AND PROBLEM FORMULATION

2.1 Fault Modeling

We assume a *stochastic single fault* model for SEU-induced faults on FPGA configuration SRAM bits since it has been shown that simultaneous multiple SRAM flips almost never occur [11]. In other words, we assume that at most one fault occurs on the FPGA configuration bits at a given time.

For a circuit $\mathbb{C}$ with n primary inputs, the sensitivity of an SRAM bit b to an SEU is defined by its criticality c_b:

$$c_b = \frac{1}{2^n}|\{x \mid \mathbb{C}(x) \neq \mathbb{C}_{\bar{b}}(x)\}|, \quad (1)$$

where $x \in \{0,1\}^n$ is the primary input vector and $\mathbb{C}(x)$ is the value of the primary outputs when the input vector x is applied to $\mathbb{C}$. In general, the criticality of an SRAM bit is the probability that an error can be observed at the primary outputs due to a flip in the SRAM bit. Accordingly, the criticality c_L of an LUT L and the full-chip fault rate of a circuit $\mathbb{C}$ are defined as

$$c_L = \frac{1}{2^K}\sum_i c_{L_i} \quad (2)$$

$$fault\_rate(\mathbb{C}) = \frac{\sum_{L \in Luts(\mathbb{C})} c_L}{|Luts(\mathbb{C})|} \cdot P_F, \quad (3)$$

where K is the number of inputs of L, L_i is the ith SRAM bit of LUT L, $Luts(\mathbb{C})$ is the set of LUTs in circuit $\mathbb{C}$, and P_F is the probability that an SEU occurs on an occupied SRAM bit. The full-chip fault rate of a circuit is the percentage of the primary input vectors that cause observable erroneous outputs due to faults under the single fault assumption.

In this paper, we focus on improving the reliability of the LUT configuration SRAM bits. Hence, we consider only SEU on LUT configuration SRAM bits to best show the effectiveness of our algorithm.

2.2 Robust Synthesis for Dual-output LUTs

Modern FPGAs have dual-output features on their PLBs. For example, Xilinx Virtex-5 has a decomposable 6-input LUT architecture with dual-output capability that can be configured as a single 6-input function or two independent functions with a total number of 5 unique inputs (see Fig. 1(a)). Altera Stratix-IV uses an 8-input adaptive logic module(ALM), which contains two adaptive LUTs (ALUT) as shown in Fig. 1(b). ALM has similar features like Virtex-5 6-input LUTs but with different input sharing constraints. An ALM can implement any single 6-input function but only a subset of single 7-input functions. Table 1 summarizes the input sharing constraints when all the input pins are utilized for both architectures. For example, when the second output is used, an ALUT in an ALM can implement up to two 6-input functions with four inputs shared between the two ALUTs.

Virtex-5 6-input LUT	size of dual-function	5,5				
	# of shared inputs	5				
Stratix-IV ALM	size of dual-function	4,4	5,3	5,4	5,5	6,6
	# of shared inputs	0	0	1	2	4

Table 1: Input sharing conditions for Virtex-5 LUT and Stratix-IV ALM

To improve circuit reliability against SEU faults, Fully Masked Duplication (FMD) has been proposed in [10], where duplication of a function is performed by utilizing the unused second output of an LUT. Then, AND or OR encodings for logic masking and therefore robustness enhancement are added at all fanout LUTs of a duplicated LUT when unused input pins are available for the fanout LUTs. A more flexible PMD (partially masked duplication) has also been developed. Note that the encoding of AND or OR operation is called "converging logic" in this paper.

2.3 Formulation of In-place Decomposition

To formulate the in-place decomposition (IPD) method, we first define the decomposition and converging of a function. Given an n-input function F, decomposition transforms F into two subfunctions, $F1$ and $F2$, where the total number of unique inputs is equal to n, and converging logic $\odot$ is applied such that $F = F1 \odot F2$. Fig. 2 is an example of decomposition, where a 5-input function F is transformed into a 4-input subfunction $F1$ and a 3-input subfunction $F2$ with two inputs shared. Then, the outputs of $F1$ and $F2$ are combined by the converging logic.

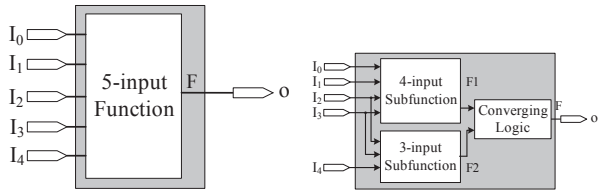
By taking advantage of the dual-output feature of LUTs, the two subfunctions can be implemented in a single dual-output LUT. Decomposition can be divided into two categories, fully-input-shared decomposition where all inputs are the same for the two subfunctions, and partially-input-shared decomposition, which includes the case where there is no common input for the two subfunctions.

Unlike FMD and PMD, which need the spare input pins of LUTs for encoding, IPD utilizes the unused built-in carry chain (or adder) within the same PLB to converge two subfunctions. Consider the boolean operation of a carry chain, where $C_{out} = A \cdot B + B \cdot C_{in} + A \cdot C_{in}$, the carry operation becomes AND operation when $C_{in} = 0$ ($C_{out} = A \cdot B$) and OR operation when $C_{in} = 1$ ($C_{out} = A + B$).

With respect to the above discussion, we formulate the IPD problem as follows.

FORMULATION 1. *Given a circuit C, IPD decomposes the logic function for a PLB into two subfunctions such that the two subfunctions are implemented by the decomposable dual-output LUT in the PLB and are combined (or converged) via the carry chain in the PLB, and the resulting full-chip fault rate is minimized.*

While our formulation and algorithms presented here apply to any converging logic, for simplicity of presentation, we consider only AND and OR converging logic in our discussion and experiment without losing the optimality.



(a) Original 5-input Function (b) Decomposed Function

Figure 2: An Example of Decomposition

3. ALGORITHMS AND PROPERTIES

We first introduce an efficient criticality update method in Section 3.1, and then present an ILP algorithm for IPD in Section 3.2. The reader familiar with criticality can start with Section 3.2 directly. We reveal the condition when in-place duplication is optimal in Section 3.3, and summarize the chip level IPD algorithm combining ILP and in-place duplication in Section 3.4.

3.1 Criticality Update for Decomposition

By the definition of the criticality, we can explain the criticality of an LUT SRAM bit in the following way. The criticality of an LUT SRAM bit c_{L_i} is the probability that the erroneous content of SRAM bit L_i is retrieved, multiplied by the probability that the erroneous output of L is propagated to the primary output. Thus, c_{L_i} of the an LUT before decomposition can be rewritten into the following equation.

$$c_{L_i} = I(L, x_k) \cdot O(L, x_k) \cdot P(L, x_k), \tag{4}$$

where $x_i \in \{x_1, \ldots, x_{2^K}\}$ is the input vector to LUT L which retrieves the content of L_i, and $I(L, x_i)$ is the probability of x_i occurring at L. The probability of passing an fault from LUT L to the primary outputs under LUT input vector x_i can be divided into two parts, the internal observability $O(L, x_i)$ and the external observability $P(L, x_i)$. The internal observability $O(L, x_i)$ is the probability that the fault on L_i can be observed at the PLB output under LUT input vector x_i, and the external observability $P(L, x_i)$ is probability to propagate the fault from the PLB output to the primary outputs, respectively. Note that each x_i is one-to-one mapped to an LUT SRAM bit L_i before decomposition, and the internal observability $O(L, x_i)$ equals to 1 since before decomposition there is no logic masking capability from an LUT to its PLB output. Therefore, c_{L_i} before decomposition is

$$c_{L_i} = I(L, x_k) \cdot P(L, x_k), \tag{5}$$

After decomposition, assuming that the decomposed two subfunctions are implemented by the two sub-LUTs, $L1$ and $L2$, of the decomposable LUT L, an LUT input vector x_i retrieves two LUT SRAM bits, one from $L1$ and the other from $L2$. Moreover, an LUT SRAM bit in $L1$ or $L2$ may have more than one LUT input vector which can retrieve its content. The criticality of an LUT SRAM bit after decomposition becomes the summation of the probabilities of its retrieving input vectors multiplied by its corresponding internal and external observabilities, and the average criticality a decomposed LUT L' is

$$\begin{aligned} c_{L'} &= \tfrac{1}{2^K} \Big(\sum_{i}^{m(L1)} c_{L1_i} + \sum_{j}^{m(L2)} c_{L2_j} \Big) \\ &= \tfrac{1}{2^K} \Big(\sum_{i}^{m(L1)} \sum_{x_k \in \psi(L1_i)} I(L, x_k) \cdot O(L1, x_k) \cdot P(L, x_k) + \\ &\qquad \sum_{j}^{m(L2)} \sum_{x_k \in \psi(L2_j)} I(L, x_k) \cdot O(L2, x_k) \cdot P(L, x_k) \Big), \end{aligned} \tag{6}$$

where $m(L1)$ and $m(L2)$ are the number of SRAM bits of $L1$ and $L2$, and $\psi(L1_i)$ denotes the set of input vectors that can retrieve the content of LUT SRAM bit $L1_i$ ($\psi(L2_j)$ for $L2_j$, respectively). Specifically, since each input vector x_k retrieves SRAM bits from $L1$ and $L2$, each x_k contributes $I(L, x_k) \cdot O(L1, x_k) \cdot P(L, x_k)$ and $I(L, x_k) \cdot O(L2, x_k) \cdot P(L, x_k)$ to the criticality calculation of $c_{L'}$. Therefore, the average criticality of an LUT after decomposition can be rewritten to

$$\begin{aligned} c_{L'} &= \tfrac{1}{2^K} \sum_{i=1}^{2^K} (I(L, x_i) \cdot O(L1, x_i) \cdot P(L, x_i) \\ &\qquad + I(L, x_i) \cdot O(L2, x_i) \cdot P(L, x_i)) \\ &= \tfrac{1}{2^K} \sum_{i=1}^{2^K} I(L, x_i) \cdot P(L, x_i) \cdot (O(L1, x_i) + O(L2, x_i)) \end{aligned} \tag{7}$$

Combining Eq.(5) and Eq.(7), and based on the fact that x_i and L_i are one-to-one mapped before decomposition. we can rewrite (7) to the following equation.

$$c_{L'} = \frac{1}{2^K} \sum_{i=0}^{2^K} c_{L_i} \cdot (O(L1, x_i) + O(L2, x_i)), \tag{8}$$

According to (8), the criticality update after decomposition depends on the internal observabilities $O(L1, x_i)$ and $O(L2, x_i)$. Since converging logic is applied to the outputs of $L1$ and $L2$ after decomposition, internal observabilities could be reduced to 0 under certain LUT input vectors according to the type of the converging logic and the truth tables of $L1$ and $L2$. $O(L1, x_i)$ depends on the output value of $L2$ under input vector x_i, and $O(L2, x_i)$ depends on that of $L1$, respectively. For example, in the case of using AND as the converging logic, $O(L1, x_i)$ can be reduced to 0 if the output of $L2$ is '0' under input vector x_i, and $O(L1, x_i)$ equals to 1 if the output of $L2$ is '1'. Therefore, the observability and the average criticality of LUT can be updated very efficiently after decomposition.

3.2 ILP Algorithm

Based on Eq.(8), we formulate the IPD optimization problem for a given LUT $L \in \mathbb{C}$ to the following Integer Linear Program (ILP) problem:

$$\text{Minimize} \sum_{i=1}^{2^K} c_{L_i} \cdot (O(L1, x_i) + O(L2, x_i)),$$

subject to the following five sets of constraints.

3.2.1 Decomposition selection constraint

$$\sum_{j=1}^{\phi(L)} d_j \leq 1$$

$\phi(L)$ is the set of decomposition templates with different input sharing conditions of a decomposable LUT architecture, such as those in Table 1 for Xilinx and Altera PLBs. $d(L, j) \in \{0, 1\}$ is '1' if LUT L is decomposed with the j_{th} decomposition template, and this constraint guarantees that there is at most one decomposition template is selected and applied.

3.2.2 Boolean matching constraints for the LUT

$$\sum_{i=1}^{2^K} t(d_j, x_i) \geq 2^K \cdot d_j, \quad 1 \leq j \leq \phi(L)$$

$t(d_j, x_i)$ is a binary variable, where $t(d_j, x_i)$ equals to '1' indicates that the Boolean function of L and that of decomposition d_j are equivalent under input vector x_i. This set of constraint guarantees that a decomposition can be selected only if its Boolean function is equivalent to that of L under all LUT input vectors.

3.2.3 Boolean matching constraints for each LUT SRAM bit of the LUT

$$0 \leq L1(d_j, x_i) + L2(d_j, x_i) - 2 \cdot t(d_j, x_i) \leq 1, if L(x_i) = 1$$

$$2 \leq L1(d_j, x_i) + L2(d_j, x_i) + 2 \cdot t(d_j, x_i) \leq 3, if L(x_i) = 0$$

$$1 \leq j \leq \phi(L), \quad 1 \leq i \leq 2^K$$

The third set of constraints perform Boolean matching for decomposition d_j with AND converging logic for each input vector, where $L(x_i)$ is the output value of LUT L under LUT input vector x_i, and $L1(d_j, x_i)$ and $L2(d_j, x_i)$ are the output values of $L1$ and $L2$ with decomposition d_j under input vector x_i. For simplicity, we show only constraints for AND converging logic since constraints for OR converging logic can be easily inferred from the above constraints.

3.2.4 Observability update constraints

$$0 \leq d_j + L2(d_j, x_i) - 2 \cdot P1(d_j, x_i) \leq 1,$$

$$0 \leq -1 \cdot d_j + L2(d_j, x_i) + 2 \cdot B1(d_j, x_i) \leq 1,$$

$$-1 \cdot P1(d_j, x_i) + O(L1, x_i) \geq 0$$

$$B1(d_j, x_i) + O(L1, x_i) \leq 1,$$

$$1 \leq j \leq \phi(L), \quad 1 \leq i \leq 2^K$$

If decomposition d_j with AND converging logic is applied, the above constraints calculate the internal observability for each input vector according to the function of $L1$ and $L2$. $P1(d_j, x_i)$ and $B1(d_j, x_i)$ are binary variables, which represent the propagation and masking of signal from $L1$ ($P2^L_{x_i}$ and $B2^L_{x_i}$ for $L2$, resp). Again, we show only constraints for AND converging logic and constraints for OR converging logic can be generated accordingly.

3.2.5 Default observability constrains

$$O(L1, x_i) + \sum_{j=1}^{\phi(L)} d_j \geq 1, \quad O(L2, x_i) + \sum_{j=1}^{\phi(L)} d_j \geq 1,$$

$$1 \leq j \leq \phi(L), \quad 1 \leq i \leq 2^K$$

The last set of constraints imply that when none of the decompositions is applied, the observabilities are one under any input vector.

3.3 Optimality of in-place duplication

Given an N-input function, multiple potential decomposition solutions exist. For example, a 5-input function can be decomposed into one 3-input subfunction and one 2-input subfunction with no shared inputs. However, the function can also be decomposed into two identical 5-input subfunctions.

To prune the solution space for decomposition, we reveal the following properties.[1]

Property 1: Considering all types of converging logic, the optimal decomposition can be obtained by decomposition with AND or OR converging logic.

Property 2: The optimal fully-input-shared decomposition with AND or OR converging logic can be obtained by decomposing to two identical subfunctions.

For optimal fully-input-shared decomposition, the identical subfunctions are simply duplication of the original function with AND or OR converging logic. This is called in-place duplication or in-place FMD (iFMD) in this paper.

3.4 Overall IPD Algorithm

The detailed IPD algorithm flow is illustrated in Fig. 3. Note that we use AND or OR converging logic just for simplicity, and the IPD algorithm can be applied to any converging logic with slight modification.

Given a circuit $\mathbb{C}$, the IPD algorithm starts with fault simulation to calculate the criticality of each LUT SRAM

[1]The proofs of the properties are provided in our technical report.

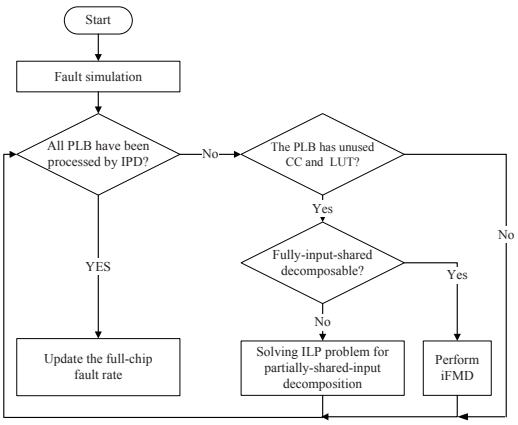


Figure 3: IPD Algorithm Flow

bit. We first check whether any PLB has not been processed and whether it has an un-used carry chain and LUT. For a PLB that has un-used carry chain and LUT, we apply iFMD to the PLB if it is applicable. If iFMD can not be applied, we solve the ILP problem presented in Sec. 3.2 to obtain the optimal partially-shared-input decomposition for the PLB. After all the PLBs are processed, we update the full-chip fault rate of $\mathbb{C}$.

4. EXPERIMENTAL RESULTS

The proposed IPD algorithm is implemented in C++ with the mosek [12] solver on an Ubuntu server with Xeon 2.4GHz CPU and 2Gb memory. The 10 largest combinational circuits in MCNC benchmark set are tested on both Virtex-5 LUT and Stratix-IV ALM architectures. All the benchmarks are first mapped to 6-input LUT logic networks by the Berkeley ABC technology mapper [13]. Then, the LUT merge algorithm in [14] is applied to merge pairs of small functions (<4 inputs) into dual-output LUTs. After that, FMD[10] and our IPD are performed separately for comparison. Assuming single fault and $P_F = 100\%$ in Equation (3), the full-chip fault rate is obtained by Monte Carlo simulation with 5K input vectors for each SRAM bit. For FPGA architectures under study, we apply IPD under different utilization rates of built-in carry chain, e.g., 10% means that randomly, 10% of build-in carry chain inside the used logic blocks are not available to be used by IPD. We compute MTTF as reversely proportional to the chip level fault rate since there is no area change.

4.1 Characteristics of Circuits and Architectures

function size	2	3	4	5	6
distribution	10.14%	17.32%	17.48%	17.72%	37.43%
fault rate contribution	5.39%	6.22%	13.74%	22.05%	48.26%

Table 2: Distribution and fault rate contribution of different sizes of functions in 10 largest MCNC combinational circuits

Table 2 summarizes the distribution of different size of logic functions and their fault rate contribution in the mapped 10 largest MCNC combinational circuits before applying FMD or IPD. It shows that 6-input functions have the greatest impact on the circuits in terms of criticality, due to their large number and the greater amount of SRAM bits they utilize.

For Xilinx Virtex-5 LUT architecture with five shared input pins, in-place duplication can be used for functions with up to 5 inputs without losing optimality. Also note that no decomposition is available for 6-input functions for this architecture. Therefore, IPD is in fact the in-place duplication with AND or OR as the converging logic in this architecture.

Fig. 4 shows the arithmetic mode of Stratix-IV ALM. Each ALUT contains two 4-input LUTs, and three of the inputs are shared among all of the 4-input LUTs, i.e., the two 4-input LUTs in an ALUT have a total of five unique inputs. Therefore, a function with up to five inputs can be decomposed and implemented by one ALUT. For functions that have number of input larger than five, we can perform decomposition by using both ALUTs plus two converging gates implemented by the two adders. Note that an ALM can implement a subset of 7-input functions, which can also be decomposed in the similar way like 6-input functions. However, because ABC mapper is not able to map to a subset of 7-input functions, we are not able to perform our experiments for 7-input functions while our IPD algorithm is capable of dealing with 7-input functions.

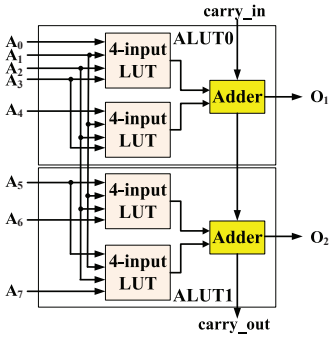


Figure 4: Altera Stratix-IV ALM arithmetic mode

4.2 MTTF Improvement and Discussions

The experimental results for the baseline algorithm, FMD and the proposed IPD are summarized in Table 3. While FMD improves MTTF only by 10% on average, IPD (with a conservative 20% utilization rate for carry chain) improves average MTTF by 1.43× and 2.70× for Virtex-5 LUT and Stratix-IV ALM architectures for 10 largest combinational MCNC benchmark circuits. When all carry chains are available (utilization rate is 0%), IPD improves MTTF by up to 2.43× (see "ex1010") for Virtex-5 LUT architecture, and up to 9.67× (see "apex2") for Stratix-IV ALM architecture.

Because in-place duplication is used exclusively for Virtex-5 LUT architecture as discussed in Section 4.1, the gap between 1.1× and 1.43× is the improvement due to performing logic converging within the same PLB. On the other hand, Stratix-IV ALM architecture utilize both in-place decomposition and in-place duplication, the gap between 1.43× and 2.70× is a good indicator of improvement due to decomposition.

Fig. 5 presents a good indicator of how much IPD could improve the reliability of a system. The (blue) bars represent the absolute difference between criticalities of "on" set and "off" set of circuits. The "on" (resp. "off") set is the SRAM bit set with logic "1" (resp. "0"). The (red) squares

Circuit Characteristics					Fault Rate										Runtime(s)
Circuit	PI#	PO#	Reg#	LUT#	BASE	FMD	IPD								
							Virtex-5 LUT				Stratix-IV ALM				
							0%	10%	20%	30%	0%	10%	20%	30%	
alu4	14	8	-	507	0.34%	0.33%	0.25%	0.26%	0.25%	0.27%	0.09%	0.11%	0.13%	0.17%	1466
apex2	39	3	-	687	0.29%	0.26%	0.18%	0.19%	0.20%	0.21%	0.03%	0.05%	0.07%	0.12%	1137
apex4	9	19	-	594	1.16%	1.10%	0.93%	0.95%	0.97%	0.99%	0.31%	0.41%	0.49%	0.60%	1430
des	256	245	-	556	1.42%	1.41%	1.17%	1.20%	1.21%	1.27%	0.80%	0.85%	0.92%	0.95%	2022
ex1010	10	10	-	668	1.24%	1.05%	0.51%	0.57%	0.65%	0.72%	0.27%	0.37%	0.47%	0.54%	1635
exp5p	8	63	-	384	0.73%	0.62%	0.46%	0.48%	0.51%	0.52%	0.24%	0.30%	0.32%	0.39%	795
misex3	14	14	-	490	0.55%	0.49%	0.32%	0.34%	0.37%	0.38%	0.10%	0.15%	0.16%	0.23%	1235
pdc	16	40	-	1515	0.91%	0.83%	0.52%	0.56%	0.61%	0.63%	0.16%	0.22%	0.31%	0.38%	3429
seq	41	35	-	705	0.63%	0.56%	0.39%	0.42%	0.44%	0.45%	0.11%	0.15%	0.21%	0.28%	1659
spla	16	46	-	1436	1.14%	1.05%	0.70%	0.74%	0.78%	0.82%	0.20%	0.31%	0.40%	0.48%	3270
GeoMean	21	24	-	683	0.75%	0.68%	0.47%	0.50%	0.52%	0.55%	0.17%	0.22%	0.28%	0.35%	1808
Fault Rate Ratio					1.00	0.91	0.63	0.67	0.70	0.73	0.22	0.30	0.37	0.47	
MTTF Ratio					1.00	1.10	1.59	1.50	1.43	1.36	4.51	3.32	2.70	2.12	

Table 3: Summary of experimental results

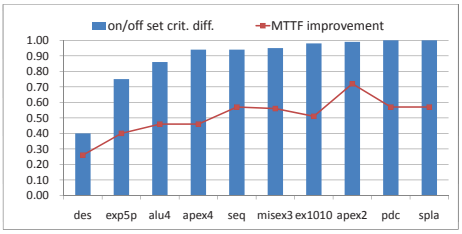


Figure 5: IPD improvement indicator

and the curve represent the MTTF improvement for the corresponding circuits. Both values are normalized for better demonstration. One conclusion that we can draw from the figure is that the more the gap between the criticalities of "on" set and "off" set is, the more IPD increases the MTTF. When the "on" set dominates the criticality of an LUT, faults on the LUT can be masked by OR converging logic and then effectively reduce the criticality. On the other hand, AND converging logic can effectively reduce criticality when "off" set dominates.

4.3 Runtime

Table 3 also presents runtime of our algorithm in seconds. Generally, each PLB takes about 10s to obtain the optimal solution for duplication or decomposition. This can be improved by more efficient algorithms in the future.

5. CONCLUSIONS AND FUTURE WORK

Leveraging decomposable LUTs and built-in carry chain in modern FPGAs, we have developed an in-place decomposition (IPD) to decompose a logic function into two subfunctions and to combine the two subfunctions via carry chain inside a PLB. An ILP algorithm is proposed to solve IPD for each PLB in a circuit. We have also revealed the condition when in-place duplication is optimal and duplication can therefore be used instead of ILP-based IPD for better algorithm efficiency. The primary contribution of this paper is to reveal the potential of IPD for robustness in FPGA. For 10 largest MCNC combinational circuits synthesized by ABC mapper and with a conservative 20% carry chain utilization rate, IPD improves MTTF to 1.43 times for PLBs similar to those in Xilinx Virtex-5 and 2.70 times for PLBs similar to those in Altera Stratix-IV.

The current algorithms and implementation can be applied to sequential circuits as well by treating register outputs and inputs as primary inputs and outputs. When register outputs and inputs are treated as primary inputs and outputs, we observed simular but slightly less improvement on MTTF of sequential circuits than that of combinational circuits. However, this may lead to a over-constrained solution and reduced MTTF improvement. Our future work will develop a more accurate fault rate analysis and a more efficient algorithm for sequential cirtuis.

In our experiments, we assumed that selected carry chain circuits have been used for other purposes in a random fashion. Industrial benchmarks will be used in the future to take into account the real carry chain utilization. In addition, new problem formulations and algorithms (similar to those in [10]) could be developed to implement converging logic in different PLBs when intra-PLB carry chain has been occupied, with consideration of trade-off between area, timing and robustness. Finally, the interaction between IPD and existing ROSE and IPR [6, 7] will be studied for robustness optimization.

6. REFERENCES

[1] Shubu.Mukherjee, *Architecture Design for Soft Errors*. Morgan-Kaufman, 2008.

[2] A. Djupdal and P. C. Haddow, "Yield enhancing defect tolerance techniques for FPGAs," in *MAPLD International Conference*, 2006.

[3] "QPRO XQR4000XL Radiation Hardened FPGAs Datasheet," in *http://www.xilinx.com*.

[4] "Radiation Hardened FPGAs Datasheet," in *http://www.actel.com*.

[5] R. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM Journal of Research and Developmen*, 1962.

[6] Y. Hu, Z. Feng, R. Majumdar, and L. He, "Robust FPGA resynthesis based on fault tolerant boolean matching," in *ICCAD*, 2008.

[7] Z. Feng, Y. Hu, R. Majumdar, and L. He, "IPR: In-place reconfiguration for FPGA fault tolerance," in *ICCAD*, 2009.

[8] A. Cosoroaba and F. Rivoallon, "Achieving higher system performance with the virtex-5 family of FPGAs," in *http://www.xilinx.com*.

[9] "Altera stratix IV features," in *http://www.altera.com*.

[10] J.-Y. R. Lee, Y. Hu, R. Majumdar, L. He, and M. Li, "Fault-tolerant resynthesis with dual-output LUTs," in *ASP-DAC*, 2010.

[11] K. Chapman and L. Jones, "SEU Strategies for Virtex-5 Devices," 2009.

[12] "mosek," in *http://www.mosek.com*.

[13] "ABC: A system for sequential synthesis and verification."

[14] T. Ahmed, P.D.Kundarewich, J.H.Anderson, B.L.Taylor, and R.Aggarwal, "Architecture-specific packing for virtex-5 FPGAs," in *FPGA*, 2007.