Fault-Tolerant Resynthesis for Dual-Output LUTs

Ju-Yueh Lee[1], Yu Hu[1], Rupak Majumdar[2], Lei He[1] and Minming Li[3]
1. Electrical Engineering Department, University of California, Los Angeles
2. Computer Science Department, University of California, Los Angeles
3. Computer Science Department, City University of Hong Kong

***Abstract*— We present a fault-tolerant post-mapping resynthesis for FPGA-based designs that exploits the dual-output feature of modern FPGA architectures to improve reliability of a mapped circuit against faults. Emerging FPGA architectures, such as 6-LUTs in Xilinx Virtex-5 and 8-input ALMs in Altera Stratix-III, have a secondary LUT output that allows access to non-occupied SRAM bits. We show that this architectural feature can be used to build redundancy for fault masking with limited area, power, and performance overhead. Our algorithm improves reliability of a mapping by performing two basic operations: *duplication* (in which free configuration bits are used to duplicate a logic function whose value is obtained on the secondary output) and *encoding* (in which two copies of the same logic function are ANDed or ORed together in the fanout of the duplicated logic). The problem of fault tolerant post-mapping resynthesis is then formulated as the optimal duplication and encoding scheme that ensures minimal circuit fault rate w.r.t. a stochastic single fault model. We show an ILP formulation of this problem and an efficient algorithm based on generalized network flow. On MCNC benchmarks, experimental results show that for combinational circuits the proposed approach improves mean time-to-failure by 25% with 4% area overhead, and the proposed approach with explicit area redundancy improves MTTF by 113% with 36% area overhead, compared to the baseline mapping by ABC. To the best of our knowledge, this represents the first systematic study that exploits dual-output LUT architectures for FPGA fault tolerance.**

I. Introduction

Faults are becoming increasingly pronounced in emerging applications and technologies, such as permanent faults arising from circuit processing at nanometer scales or transient soft errors arising from high-energy particle hits. This has spurred a lot of research to improve fault tolerance without incurring substantial area, power, or performance penalties [1].

In this paper, we explore a new opportunity for achieving fault tolerance w.r.t. a stochastic fault model by exploiting architectural features of emerging FPGA architectures. For example, in Xilinx's Virtex-5 FPGA [9], the 6-input LUT has two usable output pins (see Figure 1). It can implement two independent LUTs if the total number of unique pins in each does not exceed five. Similarly, in Altera's Startix II FPGA [3], an ALM can implement two independent LUTs if the total number of input pins does not exceed 8 and constraints on input sharing and LUT sizes are met. One use of dual-output LUTs is to map circuits that contain many small logic functions, for example, with 2 or 3 inputs, to produce higher logic density and reduced circuit power [11]. Unfortunately, it is not easy to pack logic functions with 4 or 5 inputs to the same dual-output LUT. In fact, empirical observation on a wide collection of benchmark sets shows that fully-occupied LUTs under the Virtex-5 architecture are less than 50%, even with a sophisticated "dual-output-aware" technology mapping and a LUT merging procedures [12].

The low logic utilization rate in practice motivates us to utilize non-occupied SRAM bits of dual-output LUTs for fault masking. Specifically, redundant logics can be implemented in non-occupied SRAM bits of these LUTs and can be accessed via the secondary LUT outputs. Logic duplication can effectively mask $0 \rightarrow 1$ (resp. $1 \rightarrow 0$) single event upsets (SEUs) by ANDing (resp. ORing) the original and the copy of a duplicated LUT. The additional AND and OR logic can be encoded into LUTs at the next logic stage. We consider two versions of our algorithm. In the *fully masked* version (FMD) we assume the duplicated LUT is encoded in the same way (AND or OR) in all fanouts. In the *partially masked* version (PMD) a duplicated LUT may be encoded differently in different fanouts.

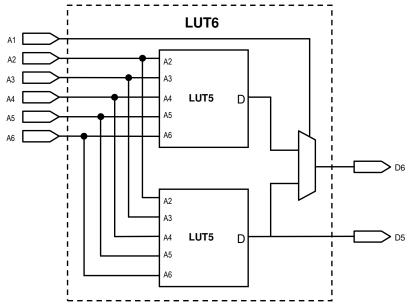


Fig. 1. Xilinx Virtex-5 dual-output LUT

We make the following contributions in this paper. Assuming a stochastic single fault model, we formulate the fault masking problems for dual-output LUT-based circuits using duplication and (full or partial) encoding as an ILP optimization problem. We show how the structure of the ILP problem allows an efficient generalized network flow-based algorithm. We test our algorithm on MCNC benchmarks under two different CAD flows with and without explicit area overhead, respectively. Experimental results show that for combinational circuits the proposed algorithm improves mean time to failure (MTTF) by 25% with 4% area overhead, and the proposed approach with explicit area overhead improves MTTF by 113% with 36% area overhead, when compared to the technology mapping obtained by [12].

In contrast to related fault-tolerance techniques proposed for FPGAs such as triple-modular redundancy (TMR) [2], our approach is lightweight and incurs minor area and performance overhead. In contrast to chip-wise synthesis [4], [5], our generic solution has much lower test complexity. In contrast to a recent stochastic synthesis algorithm (ROSE) [8], our algorithm is much faster on wide input LUTs (by exploiting architectural features). Our use of architectural features is similar to manufacturing fault masking by built-in redundancy, however, our approach has a more global view.

The rest of this paper is organized as follows. Section II presents the problem formulation. Section III proposes our primary algorithm for dual-output LUT-based fault masking, and it is generalized in Section IV. The experimental results are given in Section V and the paper is concluded by Section VI. To the best of our knowledge, this paper is the first systematic study on the stochastic fault tolerance using the dual-output architectural feature of modern FPGAs.

II. Problem Formulation

In this paper, we assume a *stochastic single fault* model for faults, i.e., there is at most one fault occurring at a time for the entire circuit with the identical fault rate for each SRAM bit. In addition, we only consider faults in LUT configuration bits. (We will show that some of faults in the interconnects can also be handled by the presented approach, though they are not considered explicitly in our optimization algorithms.) For the sake of simplicity, we only consider combinational circuits. We assume the circuit has n primary inputs and r primary outputs and that it is mapped to LUTs. Sequential circuits are first converted to combinational ones by adding register outputs/inputs to primary inputs/outputs. The maximal number of inputs allowed for a LUT in the architecture is denoted by K. For a LUT L, we write L_i for the ith configuration bit of L. For a circuit C and a primary input vector $x \in \{0,1\}^n$, we write $C(x)$ to denote the value of the primary outputs when the input vector x is applied to C. For a circuit C and a LUT L in the circuit C, we write $C(\bar{L}_i)$ for the circuit which is identical to C except that the ith configuration bit of L is flipped. We write $Luts(C)$ for the set of LUTs in circuit C. To quantitatively measure the sensitivity of a LUT with respect to the SEU, we define the criticality of a LUT as follows.

Definition 1: The *criticality* c_{L_i} of a LUT configuration bit L_i is the percentage of the primary input vectors which cause observable erroneous circuit outputs due to the flip of L_i, i.e.,

$$c_{L_i} = \frac{1}{2^n}|\{x \mid C(x) \neq C(\bar{L}_i)(x)\}|$$

The criticality c_L of a LUT L is the average criticality of its configuration bits L_i:

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

Definition 2: The *full-chip fault rate* of a circuit is the percentage of the primary input vectors which cause observable erroneous outputs due to faults. Based on the single fault assumption, the full-chip fault rate of a circuit C is calculated by the average criticality of all LUTs in C and it is defined by the following equation.

$$\text{fault\_rate}(C) = \Big(\frac{1}{|Luts(C)|}\sum_{L\in Luts(C)} c_L\Big)\cdot P_F, \qquad (1)$$

where P_F is the probability of a single fault occurs.

The criticality of each LUT in a mapped circuit can be obtained by random simulation or other more efficient approaches, such as the analytical models based on signal masking probability [13] or the hardware-based emulator. The computation of criticality needs only to be performed once. During the course of the optimization, criticality values can be updated efficiently as will be described later in the paper.

In our optimization, we shall use two atomic operations: *duplication* and *encoding*. Duplication computes the same function twice in a LUT, such that the two outputs produce two independently copied versions of the function. Encoding takes the two outputs (representing two independently computed versions of the same logic function) from a dual-output LUT and computes their AND or OR in the fanout nodes. One of the two different encoding schemes, i.e., AND-encoding and OR-encoding, is applied to the fanout LUTs of a duplicated LUT based on the logic masking effectiveness. As shown in Figure 2, any 0→1 fault occurs in LUT A can be masked. Note that any 0→1 fault occurs in intermediate wire z_{org} or z_{cpy} can also be masked as a by-product of the optimization, although only faults in LUT configuration bits are explicitly considered in our optimization.

Given a LUT L with duplication, if the same encoding (AND or OR) is applied to encode all its fanout LUTs, LUT L is called *fully masked*. Otherwise, LUT L is called *partially masked*. In this paper, the duplication scheme where all duplicated LUTs are fully masked is

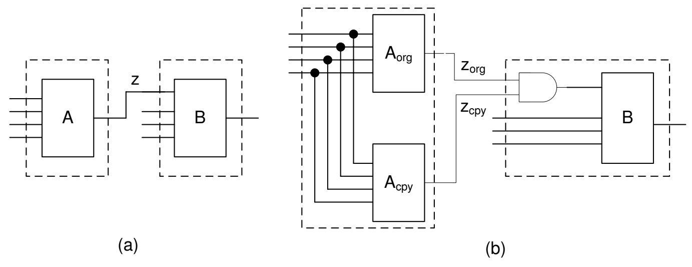


Fig. 2. LUT duplication (LUT A) and AND-encoding of its fanout (LUT B)

called *fully masking-based duplication* (FMD), otherwise it is called *partial masking-based duplication* (PMD). FMD is a special case of PMD and therefore FMD can be solved more efficiently but will result in lower quality for fault tolerance, as will be shown later. In general, the duplication-based fault tolerance problem can be formulated as follows.

Formulation 1: Given a circuit, perform duplication and encoding on LUTs (if possible), such that the full-chip fault rate (defined in (1)) is minimized.

III. FMD Algorithm

After the FMD of a LUT, the following lemmas show that criticalities of this LUT and its fanout LUTs can be updated efficiently. In addition, the update of the criticality of one duplicated LUT is independent of that of the other duplicated LUT. Therefore the FMD of the full-chip can be performed optimally via a mathematical formulation.

Lemma 1: The criticality of LUT L after a duplication operation is $2\cdot(\sum_{L_i\in \text{OnSet}} c_{L_i})/2^K$ (resp. $2\cdot(\sum_{L_i\in \text{OffSet}} c_{L_i})/2^K$) if all fanouts of LUT L are AND-encoded (resp. OR-encoded) assuming the single fault model.

Proof: Suppose LUT L is AND (resp. OR)-encoded, all $0\longrightarrow 1$ (resp. $1\longrightarrow 0$) flip due to SEU can be masked by its fanout AND (resp. OR) gate, and therefore the criticality of $L_i \in$ OffSet (resp. OnSet) is $c_{L_i} = 0$, where $L_i \in$ OffSet (resp. OnSet) means $L_i = 0$ (resp. $L_i = 1$). ■

Lemma 2: The encoding operation preserves the criticality of LUT L assuming the single fault model.

Proof: Suppose the Boolean function implemented by LUT L is $f(i_1,\cdots,i_m)$, where m is the number of occupied inputs of LUT L before encoding, and the criticality of configuration bit L_i is c_{L_i}. Without loss of the generality, the Boolean function after encoding can be expressed $f(i_i,\cdots,i_m \odot i_{m+1})$, where i_{m+1} is a newly occupied input pin due to the addition of $\odot$ logic (AND or OR). The criticality of the encoded LUT, L^e, is

$$\frac{1}{2^K}\cdot\Big(\sum_{L_i^e\in\{i_m\neq i_{m+1}\}} c_{L_i^e} + \sum_{L_i^e\in\{i_m\equiv i_{m+1}\}} c_{L_i^e}\Big),$$

where $\{i_m \neq i_{m+1}\}$ (resp. $\{i_m \equiv i_{m+1}\}$) is the set of min-terms where LUT pins i_m and i_{m+1} have the different (resp. same) logic values. Assuming the single fault, any fault occurs in LUT L indicates that no faults occur in its fanin LUTs and therefore we always have $i_m \equiv i_{m+1}$. As a result, we have $c_{L_i} = 0$ for all $L_i \in \{i_m \neq i_{m+1}\}$, and therefore the criticality of the encoded LUT L is

$$\frac{1}{2^K}\cdot\sum_{L_i^e\in\{i_m\equiv i_{m+1}\}} c_{L_i^e} = \frac{1}{2^K}\cdot\sum_{\forall L_i} c_{L_i} = c_L,$$

and this completes the proof. ■

Based on Lemma 1 and Lemma 2 we have the following theorem to show that the criticality can be updated efficiently.

Theorem 1: The criticality update can be done in constant time when duplicating one LUT with full masking.

A LUT cannot be duplicated more than once since it can only use up to two outputs. On the other hand, a LUT can be encoded more than once. In addition, duplication and encoding can be performed simultaneously on a LUT. The possibility of duplication and encoding is constrained by the number of spare (non-occupied) input pins of a LUT. Particularly, we have the following lemma to quantitatively express such constraints.

Lemma 3: If a K-LUT has p occupied input pins and $p < K$, the total number of atomic operations (duplication and encoding) that can be applied to this LUT cannot exceed $(K - p)$, and the duplication can only be done at most once.

Proof: We show that each atomic operation occupies one spare input pin. It is clear that each encoding operation occupies exactly one original spare input pin. A duplication operation effectively occupies one input pin, e.g., A_1 must be set as constant 0 in a Xilinx Virtex-5 LUT shown in Figure 1. ■

For each LUT $L \in Luts(C)$, we introduce a 0-1 variable d_L where $d_L = 1$ if LUT L is duplicated, and $d_L = 0$ otherwise. Based on these variables and Lemma 3, FMD problem can be formulated as the following ILP problem:

$$\begin{array}{ll} \text{Maximize} & \sum_{L \in Luts(C)} w_L \cdot d_L \\ \text{Subject to} & d_L + \sum_{f \in \text{fanin}(L)} d_f \le S_L, \forall L \in Luts(C), \\ & d_L \in \{0, 1\}, \end{array} \qquad (2)$$

where S_L is the number of spare input pins of LUT L, and $w_L = |\sum_{L_i \in \text{OnSet}} c_{L_i} - \sum_{L_i \in \text{OffSet}} c_{L_i}|/2^K$ is the reduction of criticality after the duplication of LUT L, which can be obtained based on Lemma 1. Formulation (2) guarantees that the encoding scheme which leads to the maximal criticality reduction between AND-encoding and OR-encoding is selected.

A. Analysis of Complexity

Theorem 2: The decision version of problem (2) is NP-complete.

Proof: We prove the NP-hardness of the Problem (2) by a reduction from 3-SAT. Given a 3-SAT instance with variables $x_1, \cdots, x_n$ and clauses $C_1, \cdots, C_m$, we construct our Problem (2) as follows. We construct DAG $G = (V, E)$ where $V = \{S, T\} \cup \{x_1, \cdots, x_n, \overline{x_1}, \cdots, \overline{x_n}\} \cup \{C_1, \cdots, C_m\}$. For the edge set E, we have $(S, x_i) \in E, (T, x_i) \in E, (\overline{x_i}, x_i) \in E$ for $1 \le i \le n$ and $(x_i, C_i) \in E$ if C_j contains literal $\overline{x_i}$ ($(\overline{x_i}, C_i) \in E$ if C_j contains literal x_i). The weight for each node is defined as follows. $w_S = w_T = (m+1)(2n+2)$, $w_{x_i} = w_{\overline{x_i}} = m+1$ and $w_{C_j} = 1$. We finally set $K = 6$. Then the constraints in the ILP can be expressed as follows.

$$d_T + d_S + d_{x_i} + d_{\overline{x_i}} \le 3, 1 \le i \le n$$
$$d_{\overline{z_1}} + d_{\overline{z_2}} + d_{\overline{z_3}} + d_{C_j} \le 3, 1 \le j \le m,\ z_1, z_2, z_3 \text{ are literals in } C_j.$$

Based on the weight selection, we can see that the optimal solution of the constructed Problem (2) will duplicate both S and T; furthermore, it will duplicate exactly one out of each pair x_i and $\bar{x}_i$; finally it will try to duplicate as many C_j's as possible. By the second set of ILP constraints, A C_j can be duplicated if and only if one of its literals is duplicated. Hence, the given 3-SAT instance is satisfiable if and only if the constructed Problem (2) can achieve objective value $2(m + 1)(2n + 2) + (m + 1)n + m$, and a literal is set true in the truth assignment if and only if its corresponding LUT is duplicated. ■

Although Theorem 2 proves that Problem (2) is NP-Hard under arbitrary weights w_i, the complexity of FMD remains unknown as it is not clear if there exists a configuration which results in a weight assignments for w_i used in the proof.

Problem (2) exhibits a nice structure of the min-cost discrete generalized network flow (GNF) as shown in Figure 4(a). Although the worst case complexity of a discrete GNF problem remains NP-Hard [15], there exist efficient combinatorial algorithms to solve it in polynomial behavior [16].

B. Example

Let us illustrate the proposed FMD algorithm using the circuit shown in Figure 3 assuming $K = 4$. The FMD problem can be transformed to a min-cost discrete GNF shown in Figure 4(a) as follows. Each level-1 node d_L in the flow network represents the duplicability of LUT L, and each level-2 node S_L expresses the spare pin number of LUT L. Note that only d_A, d_B, d_D are shown in level-1 because LUT C and LUT E are in the last logic level, and therefore cannot be duplicated. The tuple in each arch denotes (capacity, cost). The capacity of a (S_L, t)-arch is S_L, which serves as the constraints in the above ILP formulation. The weight of a (s, d_L)-arch is $-w_L$ and the supply flow in source node s is ∞. The value γ associated with each node is the gain factor, which means that flow f becomes $\gamma \cdot f$ when it exits from this node. In FMD the gain factor for node d_L is equal to the number of fanins of LUT L. It is easy to verify that this GNF is equivalent to the integer programming-based Formulation (2). The solution of the FMD problem can be obtained by the flow amount in (s, d_L)-arches after solving this min-cost GNF problem with integer flow values, i.e., discrete GNF. One feasible solution of this GNF problem is shown by the bold arches in Figure 4(a) and the corresponding FMD result is shown in Figure 4(b). For instance, LUT D is encoded due to flow in (d_A, S_D)-arch and it is also duplicated due to flows in (s, d_D)-arch and (d_D, S_D)-arch.

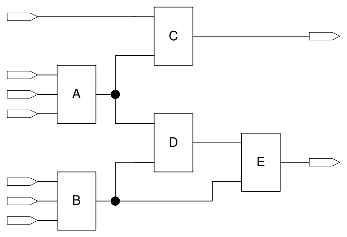


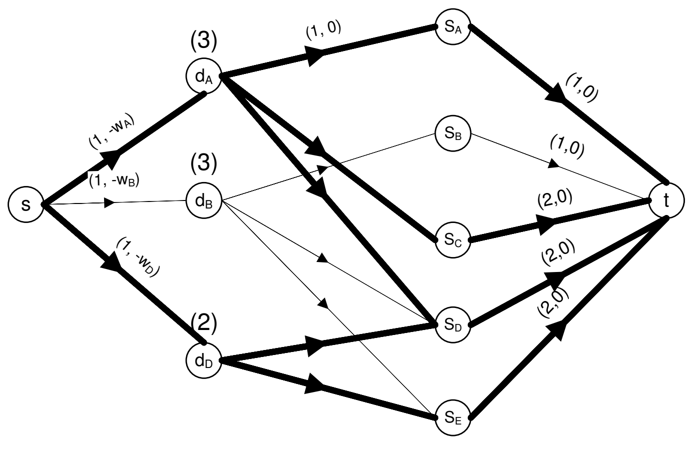
Fig. 3. A sub-circuit example

IV. PMD Algorithm

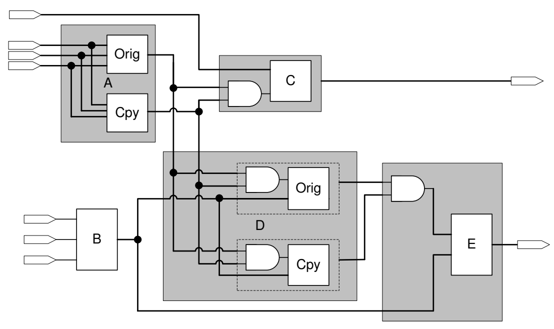
As a generalization of FMD, PMD allows partial encoding of the fanout LUTs of a duplicated LUT, and therefore it is more flexible. It can be shown that Lemma 2 remains true under PMD but Lemma 1 no longer holds under PMD. Essentially the PMD of LUT L does not mask faults observable along paths starting from its non-encoded fanouts, and therefore the update of criticality cannot be simply obtained based on Lemma 1.

Given a LUT L with non-saturated[1] fanouts $f_1, \cdots, f_p$, the duplication of LUT L requires the encoding of at least one of its non-saturated fanouts. In general, each fanout can be AND-encoded, OR-encoded or non-encoded for LUT L, and therefore there are $(3^p - 1)$ fanout encoding options, which are denoted as $\Phi(L) =$

[1]LUT is saturated if there is no spare pins for duplication or encoding. Otherwise, it is non-saturated.



(a) Min-cost generalized flow network for FMD



(b) A FMD solution corresponding to the flow (bold arches) in Figure 4(a) for the circuit in Figure 3

Fig. 4. FMD example

$\{\phi(L,1),\cdots,\phi(L,3^p-1)\}$. For each option $\phi(L,j)$, we can pre-compute the modified criticality of LUT L after applying such an encoding, and denote the reduction of criticality as w_L^j. Since AND(OR)-encoding only changes the criticalities of 0(1) configuration bits of a LUT, we only need to compute the modified criticality of LUT L by applying purely AND(OR)-encoding, and the criticality due to the mixed encoding can be calculated by merging the corresponding pure encodings as shown in Figure 5. As a result, we only need to perform 2^{p+1} computations for the criticality instead of 3^p.

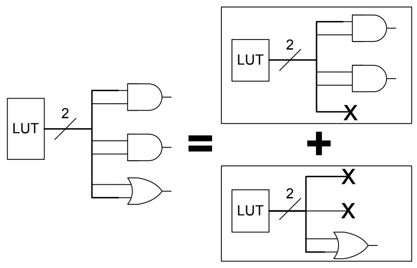


Fig. 5. The modified criticality after two AND-encoding and one OR-encoding can be calculated by merging the modified bit criticalities in OffSet due to two AND-encodings and those in OnSet due to one OR-encoding.

Note that w_L^j can be computed by performing $O(N\cdot\max(|\Phi(L)|))$ iterations of full-chip random simulations, where N is the number of LUTs and $\max(|\Phi(L)|)$ is bounded by the maximal fanout number, which is a small constant in practice. In addition, only those options which has positive w_L^j are kept since our objective is to maximize the criticality reduction. As shown in Figure 6, most LUTs have less than 9 fanouts and a threshold 512 is used to control the size of $\Phi(L)$ taken into account. We use an emulator-based approach to perform logic simulation and therefore can pre-compute w_L^j very efficiently.

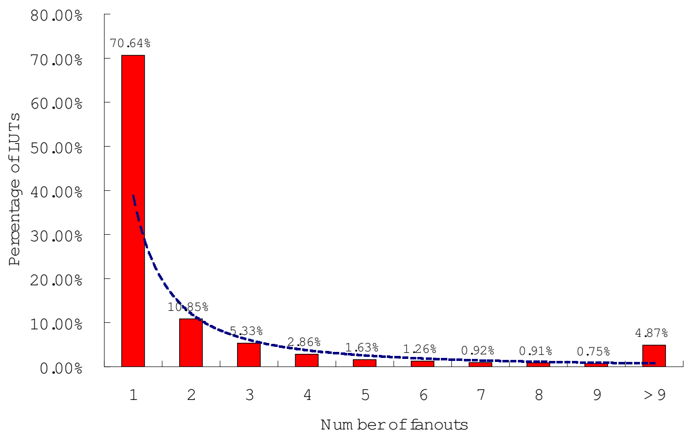


Fig. 6. Fanout distribution of 20 MCNC benchmarks (mapped by 6-LUTs)

Similar to FMD, the update of the criticality for one LUT after PMD is independent to that for another LUT under the single fault assumption, and therefore PMD can be solved optimally by a mathematical formulation. Essentially, PMD selects a subset of LUT to duplicate, and for each of those duplicated LUT encoding schemes for its fanout LUTs are decided. We formulate the PMD problem to an ILP shown in (3) by introducing a 0-1 variable d_L with the same meaning as it is in Formulation (2), and a 0-1 variable e_L^j, where $e_L^j = 1$ if option $\phi(L,j)$ is used for encoding of the fanouts for LUT L.

$$\begin{aligned}
&\text{Maximize} && \sum_{L\in Luts(C)}\sum_{j=1}^{|\Phi(L)|} w_L^j\cdot e_L^j && \\
&\text{Such that} && d_L = \sum_{j=1}^{|\Phi(L)|} e_L^j, && \forall \mathrm{L}\in Luts(C)\\
& && d_L + \sum_{f\in fanin(L)}\sum_{j=1}^{|\Phi(L)|} g_{f,j}^L\cdot e_f^j \le S_L, && \forall \mathrm{L}\in Luts(C)\\
& && d_L\in\{0,1\}\quad e_L^j\in\{0,1\} &&
\end{aligned} \tag{3}$$

where the first set of constraints guarantee that one and only one encoding scheme is used if a LUT L is duplicated, and the second set of constraints guarantee that the required resource (duplication and encoding) for a LUT does not exceed the available amount (similar to the FMD constraints), and constant coefficient $g_{f,j}^i$ is defined as

$$g_{f,j}^L = \begin{cases} 1, & \text{if } L\in\phi(f,j)\\ 0, & \text{otherwise.}\end{cases}$$

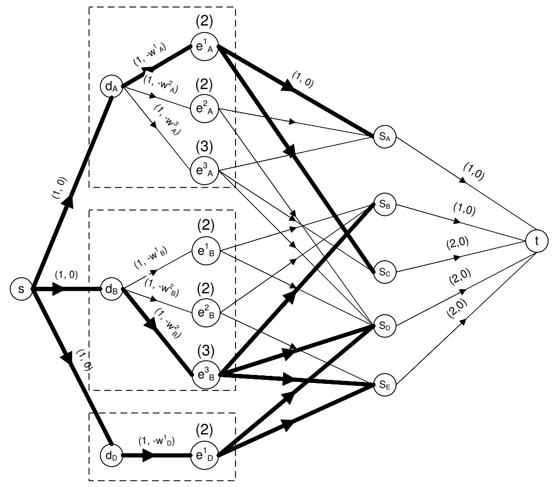
It is easy to show that Formulation (3) is a generalized case of Formulation (2). Particularly, for a circuit with a tree structure, where each LUT has only one fanout, we have $|\Phi(L)|\equiv 1$ and thereby $d_L\equiv e_L$ for all LUT L. In this case, d_L and e_L can be merged as one variable, which follows that Formulation (3) is reduced to Formulation (2). Since we have shown that Formulation (2) is NP-hard in Theorem 2, we have

Theorem 3: The decision version of problem (3) is NP-complete.

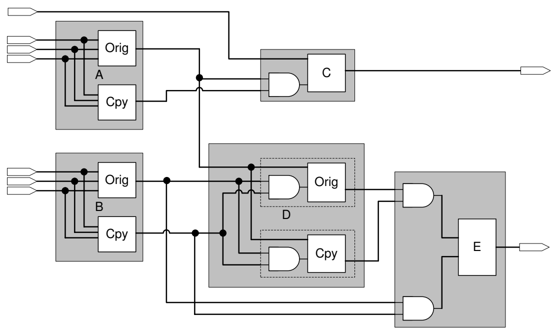
Again we show that Formulation (3) has a nice generalized network flow structure, which means it can be solved efficiently.

A. Example

Again, consider the circuit example shown in Figure 3, and only AND-encoding is assumed for the sake of simplicity. The corresponding GNF for PMD has the similar structure as its counterpart for FMD, except that for each node d_L corresponding to LUT L we introduce a set of new nodes e_L^j to represent encoding options $\phi(L,j)$ for L's fanouts. An arch (e_L^j, s_K) is added if LUT K is one of the fanouts of L and it is encoded in option $\phi(L,j)$. A gain factor is associated with each node e_L^j, and it is equal to the number of encoded fanouts in option $\phi(L,j)$. For instance, there are 2 fanouts of LUT A and therefore $|\Phi(A)| = 3$, i.e., $\phi(A,1) = \{C\}$ (only encoding LUT C), $\phi(A,2) = \{D\}$ (only encoding LUT D) and $\phi(A,3) = \{C,D\}$ (encoding both LUT C and LUT D), if LUT A is duplicated. It is easy to verify that this GNF is equivalent to Formulation (3). A feasible solution of the GNF in Figure 7(a) is shown in bold arches and the corresponding PMD result is shown in Figure 7(b), where three LUTs (A,B,D) are duplicated and three LUTs (C,D,E) are encoded.



(a) Min-cost generalized flow network



(b) A PMD solution corresponding to the flow (bold edges) in Figure 7(a) for the circuit in Figure 3

Fig. 7. PMD example

V. Experimental Results

The proposed FMD and PMD algorithms are implemented in C++ with the mosek [17] solver in a Ubuntu server with Xeon 2.4GHz CPU and 2Gb memory. The 20 biggest MCNC benchmarks are tested. Throughout the experiments, the 6-input 2-output LUT structure in Xilinx Virtex-5 FPGA architecture is assumed. All benchmarks are mapped by the Berkeley ABC technology mapper [18] with the edge

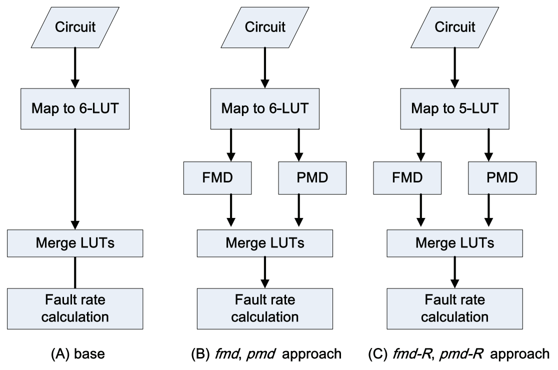


Fig. 8. Experimental flows

flow optimization, which has a special property that the mapped circuits are more suitable to be packed into dual-output LUTs [12]. The full-chip fault rate is obtained by the Monte Carlo simulation assuming the single fault and $P_F = 100\%$ in (1).

Three sets of CAD flows (shown in Figure 8) are compared in our experiments. In the baseline algorithm ("base" in Figure 8 (a)), benchmarks are first mapped to 6-LUTs and then the LUT merge algorithm in [11] is used to merge pairs of small LUTs (<4 inputs) into dual-output LUTs. In the second set of CAD flows ("*fmd*" and "*pmd*" in Figure 8 (b)), the proposed resynthesis (FMD or PMD) is performed on the 6-LUT-mapped circuits, and then the aforementioned LUT merge algorithm is applied on the resynthesized circuit. Note that both *fmd* and *pmd* preserve the logic depths of the baseline flow. To further explore the potential of the proposed algorithms, the third set of CAD flows ("*fmd-R*" and "*pmd-R*" in Figure 8 (c)) adds explicit area redundancy for fault masking by first mapping the benchmarks with 5-LUTs. This reserves at least one spare pin for each LUT to increase the opportunities of duplication by FMD and PMD.

The experimental results for the baseline algorithm and the four proposed resynthesis flows (*fmd*, *pmd*, *fmd-R* and *pmd-R*) are summarized in Table I. It shows that the four proposed resynthesis flows achieve different tradeoff among fault tolerance, area overhead, performance, and runtime. For combinational circuits as shown in the upper part of Table I, *fmd* increases MTTF[2] by 17% without area overhead, and *pmd* increases MTTF by 21% with 4% area overhead, while both preserving the logic depth, compared to the baseline algorithm. However, *fmd* and *pmd* are not effective for sequential circuits due to the low duplication rate (as shown in Figure 9). With explicit area redundancy, *fmd-R* and *pmd-R* become effective for both combinational and sequential circuits. For 19 out of 20 benchmarks[3], *fmd-R* and *pmd-R* results in fault rate reduction. Specifically, *pmd-R* increases MTTF by 113% with 36% area overhead and 15% logic depth increase for combinational circuits, and it increases MTTF by 20% with 14% area overhead and 18% logic depth increase for sequential circuits. Remarkably, *pmd-R* reduces fault rate by about 2× with only 24% area overhead and 14% increase for "apex4". For runtime, *pmd-R* is up to 195× longer than *fmd*, while most cases can be solved very efficiently (within one minute) because of the aforementioned GNF structure. A comparison of the duplication rate is shown in Figure 9, which explains the effectiveness of different algorithms for fault tolerance. It is interesting that although *fmd-*

[2]MTTF ratio is calculated based on the reciprocal of the fault rate for each benchmark circuit [19].

[3]The only exception is "elliptic", for which the 5-LUT mapping results in much more small LUTs than the 6-LUT mapping.

circuit characteristics				# of LUTs					fault rate					runtime(seconds)				logic depth	
circuit	PI#	PO#	reg#	base	*fmd*	*pmd*	*fmd-R*	*pmd-R*	base	*fmd*	*pmd*	*fmd-R*	*pmd-R*	*fmd*	*pmd*	*fmd-R*	*pmd-R*	base/*fmd*/*pmd*	*fmd-R*/*pmd-R*
combinational circuits																			
alu4	14	8		442	442	471	537	594	0.39%	0.38%	0.32%	0.25%	0.19%	0.06	0.06	0.12	47.24	5	6
apex2	39	3		600	602	634	741	779	0.33%	0.29%	0.27%	0.18%	0.16%	0.05	0.06	0.17	20.23	6	7
apex4	9	19		533	537	550	651	664	1.30%	1.11%	0.95%	0.84%	0.55%	0.04	0.4	0.15	15.16	5	6
des	256	245		531	531	531	949	950	1.49%	1.49%	1.49%	0.99%	0.82%	0.01	0.01	3.00	6.12	4	4
ex1010	10	10		642	647	652	735	781	1.29%	1.02%	0.98%	0.89%	0.74%	0.04	0.12	0.11	1.37	5	6
exp5p	8	63		337	340	348	436	437	0.83%	0.78%	0.59%	0.62%	0.38%	0.04	1.46	0.18	0.56	4	5
misex3	14	14		425	428	463	506	553	0.64%	0.56%	0.46%	0.39%	0.26%	0.04	0.18	0.11	10.76	5	6
pdc	16	40		1377	1385	1397	1910	1911	1.00%	0.90%	0.81%	0.60%	0.46%	0.04	1.67	1.06	0.5	7	8
seq	41	35		629	631	661	804	850	0.70%	0.62%	0.55%	0.41%	0.32%	0.04	1.5	0.14	90.47	5	5
spla	16	46		1296	1300	1312	1836	1842	1.27%	1.14%	1.08%	0.77%	0.66%	0.04	0.28	0.74	49.34	7	8
GeoMean	21	24		615	618	638	804	835	0.83%	0.74%	0.66%	0.52%	0.40%	0.04	0.32	0.27	8.34	5.21	5.98
Ratio				1	1	1.04	1.31	1.36	1	0.89	0.79	0.63	0.48	1	7×	6×	195×	1	1.15
MTTF Ratio									1	1.17	1.25	1.63	2.13						
sequential circuits																			
bigkey	263	197	224	518	518	518	798	742	1.36%	1.36%	1.36%	1.18%	1.06%	0.01	0.01	0.05	0.05	3	3
clma	383	82	33	2868	2868	2869	3242	3234	0.08%	0.08%	0.08%	0.07%	0.07%	0.04	0.04	0.16	0.51	9	11
diffeq	28	3	305	534	536	537	573	566	1.05%	1.03%	1.02%	0.98%	0.90%	0.05	0.14	0.39	40.67	8	10
dsip	228	197	224	665	665	665	782	782	1.73%	1.73%	1.73%	1.46%	1.46%	0.01	0.01	0.07	0.08	3	3
elliptic	19	2	194	322	323	323	271	270	1.03%	1.03%	1.02%	1.22%	1.20%	0.04	0.05	0.05	0.32	6	8
frisc	20	116	886	1868	1868	1872	2233	2228	1.20%	1.19%	1.19%	0.89%	0.86%	0.04	0.05	0.23	8.99	14	17
s298	3	6	14	20	20	20	24	23	1.76%	1.72%	1.72%	1.47%	1.44%	0.04	0.04	0.04	0.06	2	2
s38417	29	106	1462	1991	1993	2005	2338	2314	1.48%	1.47%	1.43%	1.24%	1.11%	0.04	2.8	2.05	15.39	6	8
s38584.1	39	304	1260	1972	1976	1985	2408	2378	1.32%	1.30%	1.27%	1.07%	1.00%	0.04	0.08	0.72	176.71	7	8
tseng	52	122	385	640	664	667	743	741	1.37%	1.30%	1.28%	1.16%	1.11%	0.04	0.19	0.73	2.36	7	10
GeoMean	46	47	247	661	664	665	767	755	1.02%	1.01%	1.00%	0.88%	0.84%	0.04	0.11	0.20	1.57	5.63	6.66
Ratio				1	1	1.01	1.16	1.14	1	0.99	0.98	0.87	0.83	1	3×	5×	38×	1	1.18
MTTF Ratio									1	1.01	1.02	1.17	1.20						

TABLE I

SUMMARY OF EXPERIMENTAL RESULTS

R duplicates fewer LUTs than *pmd-R*, it results in larger area for sequential benchmarks (767 vs. 755 on average). This is because more small LUTs (<4 inputs) are used for duplication and encoding in *fmd-R* due to the fully-masking constraint, and as a result the LUT merge algorithm becomes less effective for *fmd-R*.

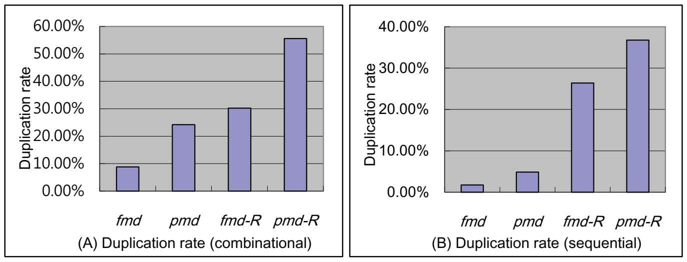


Fig. 9. Duplication rate comparison

VI. Conclusions and Future Work

We have presented a fault-tolerant post-mapping resynthesis which explicitly exploits the dual-output LUT architecture. Our experimental results are encouraging and show up to 2× MTTF increase with 24% area overhead, compared to ABC technology mapping. In the future, the following research directions can be explored. As our experiments present the area-robustness-performance tradeoff of the two extreme cases (mapping with 6-LUTs and 5-LUTs), simultaneous technology mapping and resynthesis should be studied in order to traverse the Pareto points in the solution space and to maximize the robustness under area/performance constraints. In addition, besides `AND`-Encoding and `OR`-Encoding, other logics (e.g., `NAND` and `NOR`, or other more complicated logics) can also be used to mask faults in their upstream circuit. Furthermore, while the proposed technique places masking logic immediately after a duplicated LUT, the area overhead can be improved by placing masking logics at the end of a series of duplications. Finally, we will study the dual-output-aware physical synthesis, including packing, placement and routing, to mitigate the impact of the proposed approach on the performance.

References

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

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

[3] "Altera stratix II device handbook," in *http://www.altera.com*, 2007.

[4] H. Naeimi, "A greedy algorithm for tolerating defective crosspoints in NanoPLA design," in *Master Thesis, California Institute of Technology*, 2005.

[5] M. Joshi and W. Al-Assadi, *Development and analysis of defect tolerant bipartite mapping techniques for programmable cross-points in nanofabric architecture*, Springer Netherlands, 2007.

[6] R. Bonam, Y.-B. Kim, and M. Choi, "Defect-tolerant gate macro mapping and placement in clock-free nanowire crossbar architecture," in *DFT*, 2007.

[7] "Xilinx. easypath FPGAs," in *http://www.xilinx.com*.

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

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

[10] "Altera stratix III features," in *http://www.altera.com*, 2007.

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

[12] S. Jang, B. Chan, K. Chung, and A. Mishchenko, "Wiremap: FPGA technology mapping for improved routability," in *FPGA*, 2008.

[13] S. Krishnaswamy, S. Plaza, I. L. Markov, and J. P. Hayes, "Signature-based SER analysis and design of logic circuits," in *TCAD*, 2009.

[14] D. Sinha and S. Abbaspour, "Constrained aggressor set selection for maximum coupling noise," in *DAC*, 2008.

[15] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[16] T.-Y. Hsu and T.-C. Wang, "A generalized network flow based algorithm for power-aware FPGA memory mapping," in *DAC*, 2008.

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

[18] "ABC: A system for sequential synthesis and verification," in *http://www.eecs.berkeley.edu/ alanmi/abc/*.

[19] S. Mukherjee, J. Emer, and S. K. Reinhardt, "Radiation-induced soft errors: An architectural perspective," in *HPCA*, 2005.