Cross-layer Optimized Placement and Routing for FPGA Soft Error Mitigation

Keheng Huang†‡ Yu Hu† Xiaowei Li†
†Key Laboratory of Computer System and Architecture,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, P.R. China, 100190
‡Graduate University of Chinese Academy of Sciences, Beijing, P.R. China, 100049
{huangkeheng, huyu, lxw}@ict.ac.cn

***Abstract*—As the feature size of FPGA shrinks to nanometers, soft errors increasingly become an important concern for SRAM-based FPGAs. Without consideration of the application level impact, existing reliability-oriented placement and routing approaches analyze soft error rate (SER) only at the physical level, consequently completing the design with suboptimal soft error mitigation. Our analysis shows that the statistical variation of the application level factor is significant. Hence in this work, we first propose a cube-based analysis to efficiently and accurately evaluate the application level factor. And then we propose a cross-layer optimized placement and routing algorithm to reduce the SER by incorporating the application level and the physical level factor together. Experimental results show that, the average difference of the application level factor between our cube-based method and Monte Carlo golden simulation is less than 0.01. Moreover, compared with the baseline VPR placement and routing technique, the cross-layer optimized placement and routing algorithm can reduce the SER by 14% with no area and performance overhead.**

Keywords- cross-layer optimization, cube-based analysis, FPGA, placement and routing, soft error rate.

I. INTRODUCTION

With exponential growth in performance and capacity, SRAM-based field programmable gate arrays (FPGAs) are widely used in many application domains such as telecommunication, industrial control, and embedded applications for their short design cycle and low development cost. Although SRAM-based FPGAs provide all these advantages, due to their storage structures, they are more vulnerable to single event upsets (SEUs) induced by high-energy particles than application-specific integrated circuits (ASICs) [1, 2]. Moreover, an SEU in the configuration bits of FPGA can generate permanent fault effect until re-writing the affected bit. If the SEU affects the predefined function of the design, then it provokes a soft error. With the continuously decreasing of CMOS feature size and threshold voltage, soft errors increasingly become an important concern in SRAM-based FPGAs [3].

Analysis shows that, in SRAM-based FPGAs, more than 98% of all SRAM bits are configuration bits (the rest are user bits, e.g., flip-flops) [4], among which faults in the routing resources (i.e. interconnect) take up about 80% of all soft errors [5]. Several researches have addressed the reliability issue of routing resources. L. Sterpone et al. [6] first observe that an SEU may induce multiple fault effects in routing resources. In the case of TMR designs, one SEU may simultaneously affect two or more modules when the configuration memory bit controls two or more routing segments. In the extreme case, the fault may escape the majority voting. They abstract the circuit into a routing graph, and use different colors in the graph to mark edges and vertices of replica. By checking the number of colors for the SEU propagation tree, the effects of SEU on TMR designs can be accurately predicted. After then, the same authors propose a reliability-oriented placement and routing algorithm namely RoRA [7]. By imposing routing restrictions to reduce the occurrence of SEU induced color collision among modules, the RoRA algorithm can minimize the effect of SEU induced soft errors. However, RoRA is targeted for TMR-based designs only and is not suitable for non-TMR designs.

The algorithms proposed in [8, 9] add the reliability factor into the cost function of the simulated annealing-based placer and the maze-based router. Besides timing and congestion, soft error mitigation is also an objective of placement and routing. In [8], the SEU cost is characterized by the dimensions of bounding boxes and the common dimensions of the intersection area of the bounding boxes for all nets during placement. And in [9], the SEU cost is characterized by the number of SEU-aware configuration bits during routing. Referring to [10], the evaluation criterion of a design against soft errors is soft error rate (SER), which describes the probability that a transient fault occurs and then affects the predefined function of the design. SER is composed of two factors including the application level factor *error propagation probability* (EPP) which describes the influence of a fault on the gate-level netlist of the mapped design, and the physical level factor *node error rate* which describes the fault occurrence probability based on the detailed placement and routing information. As we will show in Section 2, the application level factor varies significantly across different designs. Therefore, prior reliability-oriented placement and routing algorithms considering the physical level factor only cannot accurately characterize the SER of interconnecting wires, and the effectiveness of mitigating soft errors by these algorithms is limited.

To characterize the EPPs at the application level, prior works mainly are Monte Carlo simulation [11, 12] and static analysis [10]. For Monte Carlo simulation, a single fault is

This work was supported in part by National Natural Science Foundation of China (NSFC) under grant No. (61076018, 60803031, 60633060, 60921002, 60831160526), and in part by High-Tech Research and Development Program of China under grant No. (2007AA01Z107, 2007AA01Z113).

978-3-9810801-7-9/DATE11/©2011 EDAA

injected per iteration for a set of input vectors. The simulation attempts to evaluate the percentage of vectors that do not produce the desired output. Monte Carlo simulation usually can achieve a high accuracy but at the cost of the long run time due to feed the whole input set for each fault. Existing works in this category mainly trade the accuracy for the computational complexity, but the acceptable margin is hard to determine. In contrast, static analysis uses probability theory to compute the EPP of each fault [10]. Assuming the signal probability of each input wire is equivalent, the static analysis method proposed in [10] estimates the signal probability of internal wires between LUTs. Afterwards, the EPP of a fault is computed by the signal probability for off-path wires and the error propagation rules for on-path wires. It traverses the entire circuit only twice, so the computational complexity is low. However, based on the signal probability of each wire, this technique does not take the correlation between the signal value and the test vectors into consideration. Furthermore, the roughly estimated signal probability limits the accuracy of EPPs.

In this paper, by incorporating the application level factor into the physical design, we propose a cross-layer optimized placement and routing algorithm for soft error mitigation (COPAR), successfully handling the gap between the SER evaluation criterion and the placement and routing guidance criterion. In order to get the application level factor accurately and efficiently, the cube-based EPP analysis are proposed. Experimental results show that, the difference of the EPPs between our cube-based method and Monte Carlo simulation is less than 0.01 on average. Moreover, compared with the baseline VPR placement and routing technique, our COPAR technique can reduce the SER by 14.39%, without any area or performance overhead.

The rest of this paper is organized as follows. Section 2 describes the background and motivation. Section 3 introduces the procedure of COPAR. Section 4 shows the experimental results, and the paper is concluded in Section 5.

II. BACKGROUND AND MOTIVATION

A. Architecture of SRAM-based FPGAs

SRAM-based FPGAs have fixed number of segments, switching blocks and LUTs. A k-input LUT has 2^k LUT entries and can realize any k-input logic functions. It is configured by the internal 2^k configuration bits (CBs). Alternatively, the interconnection of wires is realized by the configuration bits which control the switches to connect segments into wires. The architecture of SRAM-based FPGAs is shown in Fig. 1. The great flexibility of routing structures gives us the possibility to concurrently consider timing, congestion, and reliability.

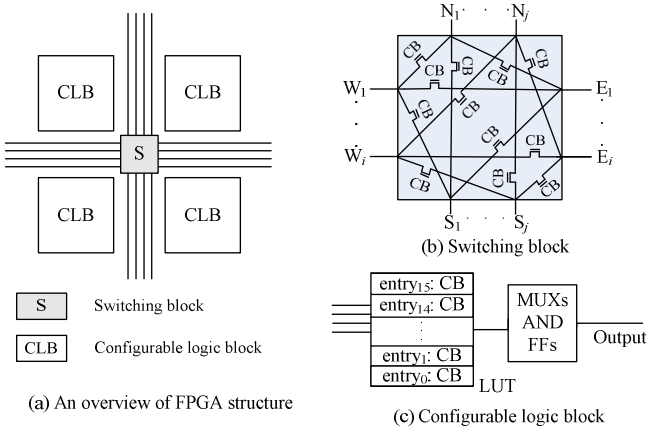

Figure 1. Architecture of SRAM-based FPGAs

TABLE I. INEQUALITY DEGREE ACCORDING TO GINI COEFFICIENT

Gini coefficient	Inequality degree
<0.2	Absolutely equal
0.2-0.3	Relatively equal
0.3-0.4	Moderately unequal
0.4-0.5	The gap is relatively large
>0.6	Quite unequal

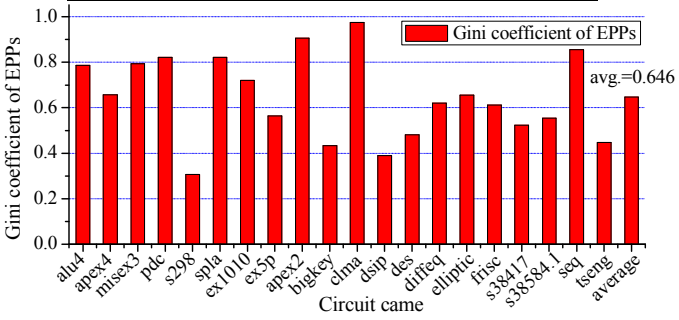

Figure 2. Gini coefficient of EPPs

B. Motivation

SER consists of two factors, the application level factor and the physical level factor. At the application level, an SEU may not have adverse impact on the normal function of the design because of the logic masking effect. So the impact degree of different faults on one application is different, which can be quantified by the application level factor EPP. EPP is device independent and can be calculated based on the netlist information after synthesis. In contrast, based on the layout information given by the detailed placement and routing, the physical level factor *node error rate* defined in [10] describes the fault occurrence probability. For example, if a wire is configured by more CBs, there will be more chance of a fault occurs at it. Similar to [10], [8] adopts the dimensions of bounding boxes and [9] adopts the number of SEU-aware CBs to calculate the *node error rate*, respectively.

As mentioned above, the SER evaluation criterion contains both the physical level factor and the application level factor. However, prior works [8, 9] only use the physical level factor to guide the placement and routing for mitigating soft errors, which actually treat the application level factor of interconnecting wires, in other words, the EPPs, absolutely equal. After conducting full space simulation on the MCNC benchmark circuits with less than 25 combination logic inputs (alu4, apex4, misex3, pdc, s298, spla, ex1010, and ex5p) and conducting 100K input vector simulation on other MCNC benchmark circuits, we observe the EPPs vary significantly across the circuits, which implies a big gap between the SER evaluation criterion and the prior placement and routing guidance criterion. We use the Gini coefficient [13], a measure of static variation, to evaluate the variation of EPPs of interconnecting wires for the MCNC benchmark circuits. Table I shows the inequality degree according to the Gini coefficient, and Fig. 2 shows the Gini coefficient of EPPs for the MCNC benchmark circuits. As we can see, the average Gini coefficient

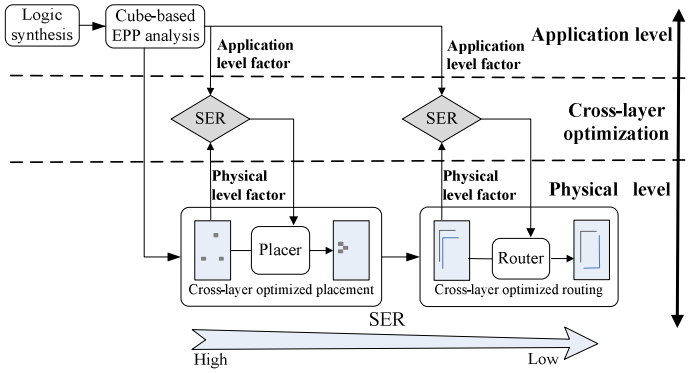

Figure 3. The flowchart of COPAR

of EPPs is 0.646, which indicates a great inequality of EPPs for each application. So unaware of the application level impact, existing reliability-oriented placement and routing approaches that analyze SER only at the physical level, can only obtain suboptimal soft error mitigation.

III. PROPOSED COPAR METHOD

In this work, we propose a cross-layer optimized placement and routing method for soft error mitigation, namely COPAR. By incorporating the application level factor and the physical level factor together, we can handle the gap between the SER evaluation criterion and the placement and routing guidance criterion. As shown in Fig. 3, after logic synthesis, the application level factor computed by cube-based EPP analysis and the physical level factor computed during each stage of placement and routing, jointly guide the placer and router to reduce the SER of the design.

A. Cube-based EPP Analysis

Traditional Monte Carlo simulation [11, 12] has high accuracy but long run time due to feed the whole input set for each fault, while static analysis [10] is time-efficient but its accuracy is poor.

In this work, we leverage the concept of *cube* and *cover* in logic synthesis [14, 15] to perform cube-based EPP analysis in the granularity of input vectors, so the accuracy is high. The same as static analysis method, the cube-based EPP analysis only traverses the entire design two times, so the computational complexity is low. For each possible fault, we first obtain the input vectors that can sensitize the fault, and then keep the input vectors that can propagate the fault effect to the outputs. Finally, the EPP of the fault is the ratio of the kept vectors to the total input vectors.

In the context of combinational logic circuits, a *cube* defines a relationship between the primary inputs and the primary outputs [14]. It is represented as $a_1a_2\ldots a_n|b_1b_2\ldots b_m$, where $a_1,a_2,\ldots,a_n$ are the values of the primary inputs and $b_1,b_2\ldots,b_m$ are the values of the corresponding primary outputs. In the case of single output, the output term can be omitted. A *cover* for the function realized by a circuit is a set of *cubes* that unambiguously defines that function. For example, a 2-input LUT to realize the AND logic is represented as Fig. 4 shows. Each "don't care" symbol "X" in the *cube* can be replaced by a "1" or a "0". Thus a *cube* containing m "X" bits actually equals to 2^m *cubes*. There are two main cubical operations needed in our work, *interface* (**I**) and *adjoin* (**V**) [15], which are analogues to the set-theoretic operations intersection ($\cap$) and union ($\cup$) respectively. The detailed description of cubical operations is in [14]. Note that, the operational result of cubical operations is optimized to have a minimum number of *cubes* each containing a maximum number of free variables.

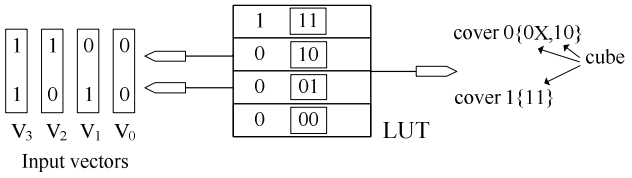

Figure 4. Example of *cube* and *cover*

Our choice of using *cube* and *cover* is motivated by two reasons. First, in addition to the conventional binary values 0 and 1, the "don't care" symbol "X", allows for very compact and economical Boolean representations [14]. Second, by using the two cubical operations, *interface* and *adjoin*, we can compute the EPP of a fault in the granularity of specific input vectors, which is different from the static analysis technique [10] that computes the EPP under the signal probability. Therefore, we can achieve more accurate EPP results.

To perform cube-based EPP analysis, we define four types of *covers* for each wire. The two control-covers namely control-cover 0 and control-cover 1, store the input vectors that control the logic value of a targeted wire to 0 and 1 during logic simulation, respectively. So the control-covers imply the vectors that can sensitize the targeted fault. The two care-covers namely care-cover 0 and care-cover 1, store the vectors in control-covers that can propagate the fault effect at the targeted wire to affect the predefined function. Note that, since the vectors which can propagate the fault effect must sensitize the fault first, the care-cover 1 and the care-cover 0 is the subset of the corresponding control-cover 0 and the control-cover 1, respectively. Then, the EPP of a fault is the ratio of the vectors stored in the care-cover to the total input vectors.

There are only two passes to compute EPPs with our cube-based analysis. The first pass is forward traverse, which performs logic simulation to compute the control-covers of each wire by cubical operations. As Fig. 5 shows, for each input wire, we use cubical operations to pack all input vectors to the two control-covers of the wire. Then, based on the logic value stored in the LUT, the control-cover 1 of the output is the *interface* of the two control-cover 1s for the two inputs. Note that, in the form of control-covers, input vectors are fed into the design all at once, and then are all simulated in one pass through cubical operations, not one vector per time. The second pass, backward traverse, computes the care-covers of each wire from the vectors stored in the previously obtained control-covers. Multiple fanouts and logic masking are considered during back tracing. Take Fig. 6 as an example, the care-cover 0 of the output is the *adjoin* of the care-cover 0 of Fanout0 and the care-cover 0 of Fanout1, which equals to {0X, 10}.

Since computational complexity is an important concern for an algorithm, we analyze the computational complexity of existing EPP analysis algorithms. For a design with V LUTs and E interconnecting wires, the total number of faults is

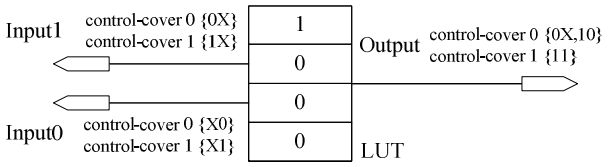

Figure 5. Example of forward traverse

g*(V+E), where g is the number of configuration bits for an LUT or a wire. For Monte Carlo simulation, supposing the number of input vectors is N, we must traverse the whole circuit N times for each fault [16]. For static analysis, the complexity of estimating signal probability is $O(V^2)$ [10]. For cube-based analysis, due to the introducing of "X" bit, the average number of *cubes* in each *cover* is N/C_{avg}, where C_{avg} is the average compression ratio of all *covers*. Then, since the result of cubical operations for two *covers*, such as *covers adjoin* and *covers interface*, is the Cartesian product of their *cubes* respectively, the complexity of a cubical operation is $(N/C_{avg})^2$. The overall complexity of Monte Carlo simulation, static analysis, and cube-based EPP analysis are shown in Table II. In case of a huge N, to minimize the computational complexity of cube-based analysis, the compression ratio C_{avg} must be as large as possible. Note that, the simulated vectors will affect the upper bound of the compression ratio. In the extreme case, take the first input wire as an example, all input vectors that set the wire to 0 can be packed to the control-cover 0, that is {XX..XX0}. Therefore, the upper bound of the compression ratio for a *cover* is N/2. Referring to [11], for smaller designs (<25 inputs), we perform full space simulation. Half of the input vectors will set an input wire as logic 0 and 1 respectively, which satisfies the upper bound condition of compression ratio at all input wires. For larger designs, we impose a constraint to the random selection of input vectors as follows to balance the accuracy and complexity.

The random selection scheme:
For a design with more than 25 inputs
for (int *iter* = 0; iter < *r*; *iter*++)
{
  Randomly set *p* inputs as "X"
  Randomly set the rest inputs as 0 or 1
  Conduct cube-based EPP analysis with these 2^p input vectors
}
Compute the EPP of each fault by averaging the EPPs obtained after iterations.

In the pseudo code of the random selection scheme above, r and p are user-defined parameters. The number of input vectors simulated per iteration is 2^p. The larger the r and p are, the more accurate EPPs can be obtained. By using the selection scheme described above, the upper bound of the compression ratio can also be achieved for the control-covers of all primary input wires. And because the result of cubical operations is optimized to have a minimum number of *cubes* that each contains a maximum number of "X"s, the size of *cover i*, (N/C_i), will not exponentially increase along the EPP analysis. So the complexity of cube-based EPP analysis is acceptable.

TABLE II. COMPARISON OF COMPUTATIONAL COMPLEXITY

Algorithm	Computational complexity
Monte Carlo Simulation [16]	$O(N*V*g*(V+E))$
Static Analysis	$O(V^2*(V+E))$
Cube-based EPP Analysis	$O((N/C_{avg})^2*(V+E))$

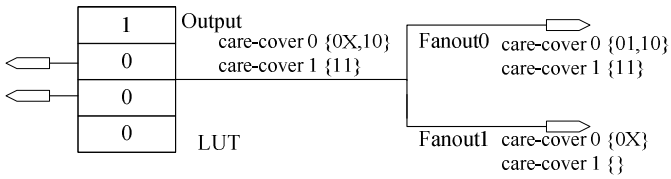

Figure 6. Example of backward traverse

B. Cross-layer Optimized Placement

After cube-based EPP analysis, LUTs are packed into configurable logic blocks (CLBs), while wires are merged into nets. For the cross-layer optimized placement, the major challenge is the integration of the application level factor and the physical level factor into a timing-driven placement algorithm. Traditional placement tools only consider the cost of timing and congestion. To mitigate the soft error of the design, SER cost is taken into account, as shown in Equation (1).

$$Total\ Cost=a*timing\ cost+b*congestion\ cost+c*SER\ cost \quad (1)$$

where a, b, and c are user-defined constants. The SER cost is composed of costs at the application level and the physical level, as shown in Equation (2).

$$SER\ cost = APP\ cost * Phy\ cost \quad (2)$$

where *APP cost* represents the application level cost, while *Phy cost* represents the physical level cost. By incorporating costs at two levels together, we perform cross-layer optimized placement, making the SER evaluation criterion and the placement guidance criterion consistent. Therefore, the objective of the cross-layer optimized placer is to minimize the SER, not the physical level factor only.

In the stage of cube-based EPP analysis, only the stuck-at faults are considered. In the placement stage, we additionally consider wire-open and wire-short faults. Wire-open fault describes a net is broken off due to the SEU impact. Wire-short fault describes an SEU occurs in a configuration bit, and then causes two should-be disconnected nets to be bridged.

For a wire-open fault, the application level factor can be described as the EPP of the net. And the physical level factor can be characterized by the number of configuration bits, which can be simplified as the dimensions of bounding boxes in placement, as shown in Fig. 7. The cost function of wire-open fault is shown in Equation (3).

$$SER\ cost_{\text{open}} = \sum_{i=0}^{i<N_{nets}} [bb_x(i) + bb_y(i)] * EPP_i \quad (3)$$

For a wire-short fault, the application level factor is the EPPs of the two bridged nets, while the physical level factor is measured by the number of configuration bits that can bridge these two nets, which is simply measured by the common dimensions of the intersection area of the two bounding boxes for the two bridged nets, also as Fig.7 shows. The cost function of wire-short fault is given by Equation (4).

$$SER\ cost_{\text{short}} = \sum_{i=0}^{i<N_{nets}} \sum_{j=0, j\neq i}^{j<N_{nets}} \{[cbb_x(i) + cbb_y(i)] * (EPP_i + EPP_j)\} \quad (4)$$

C. Cross-layer Optimized Routing

Comparing with the coarse estimation of cost factors during placement, the estimation of cost factors in routing is performed in a finer granularity way. For a net consisting of many segments, the router traverses all possible

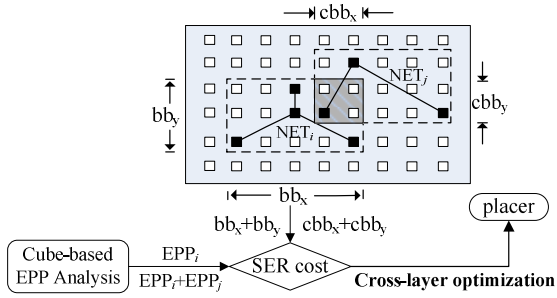

Figure 7. Example of SER cost in placement

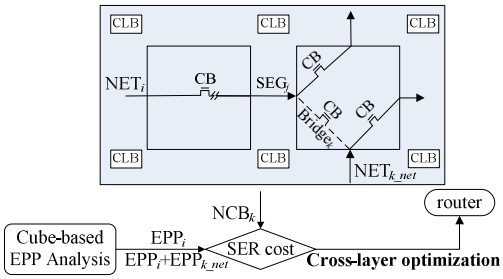

Figure 8. Example of SER cost in routing

implementations and finds the one that gives the lowest cost. Taking reliability into consideration, SER cost is added into the cost function of the router as follows:

$$Total\ Cost = a*timing\ cost + b*congestion\ cost*SER\ cost \quad (5)$$

where a and b are user-defined constants, and SER cost is calculated with Equation (2).

For the wire-open fault of a candidate segment, the application level factor is characterized by the EPP of the segment, while the physical level factor is evaluated by the number of configuration bits (NCB), as Fig. 8 shows. The cost function of wire-open fault is given by

$$SER\ cost_{\text{open}} = \sum_{i=0}^{i<N_{nets}} \sum_{j=0}^{j<N_{segs}} (NCB_j * EPP_j) \quad (6)$$

For the wire-short fault of a candidate segment, we use the wired-AND/wired-OR bridging fault model [17]. All possible fanout branches of the segment are traversed to see if there is a possible bridging fault between two used nets. If so, the application level factor is characterized by the EPPs of the two bridged nets, while the physical level factor is evaluated by the number of configuration bits that can bridge these two nets, as Fig. 8 shows. The cost function of wire-short fault is:

$$SER\ cost_{\text{short}} = \sum_{i=0}^{i<N_{nets}} \sum_{j=0}^{j<N_{segs}} [\sum_{k=0}^{k<N_{bridge}} NCB_k * (EPP_i + EPP_{k\_net})] \quad (7)$$

Consequently, by modifying the cost function of placer and router, the application level factor and physical level factor are concurrently considered. For the cross-layer optimized placement and routing algorithm, the SER evaluation criterion is consistent with the placement and routing guidance criterion. Therefore, the effectiveness of soft error mitigation in placement and routing is maximized.

IV. EXPERIMENTAL RESULTS

In order to validate the proposed COPAR algorithm, we use the combinational part of the 20 largest MCNC benchmark set, and map them into 6-input LUTs by Berkeley ABC mapper [18]. All the algorithms described above are implemented in Java and run on a Xeon Linux workstation with 6GB memory.

As performed in [11], we use full space vectors for the circuits having less than 25 inputs (alu4, apex4, misex3, pdc, s298, spla, ex1010, and ex5p). For larger circuits, to obtain highly accurate EPP results, the algorithm proposed in [11] uses 100M vectors for each fault. In this work, we set r as 100 and p as 25 for the random selection scheme described in Section 3, so the number of generated input vectors is ($100*2^{25}$), which satisfies the number of input vectors to achieve highly accurate EPPs. The quality of cube-based EPP analysis is determined by how accurately the vectors in care-covers are actually the test vectors for the corresponding fault. The difference (scaled to the total number of input vectors) can be used to evaluate the accuracy of cube-based EPP analysis.

$$Gap = \sum_{i=0}^{N_{\text{faults}}} (\frac{N_{\text{cube}} - N_{\text{sim}}}{N_{\text{inputs}}}) \quad (8)$$

where N_{cube} is the number of vectors in care-covers, while N_{sim} is the number of test vectors for the fault in simulation. N_{inputs} is the total number of input vectors. As shown in Table III, the deviation of cube-based EPP analysis (G_{cube}) is no more than $1.0*10^{-4}$ on average, which is better than the windowing based DCOW (G_{DCOW}) [11] and static analysis method in [10] (5% on average). Note that, the accuracy of DCOW is measured by the difference (scaled to the total number of configuration bits) of the number of simulated critical bits between DCOW and Monte Carlo simulation, which is a rough estimation of SER. Although the deviation of the cube-based EPP analysis scheme can be neglected to a certain extent, we also give a brief analysis. The reason why the cube-based EPP analysis will lose accuracy in some cases is the fanout reconvergency [19]. Both the self-masking and multiple-path sensitization will lead to erroneous judgment for test vectors in cube-based analysis. We also compare the run time between Monte Carlo simulation and

TABLE III. COMPARISON OF EPP ACCURACY AND RUN TIME

Circuit	Monte Carlo simulation		Cube-based analysis		Gap(10^{-4})†	
	$N_{\text{sim}}(10^5)$	*Time*(s)	$N_{\text{cube}}(10^5)$	*Time*(s)	G_{cube}	G_{DCOW}
alu4	20.46	3314	20.45	186	0.212	0
apex2	122.85	N/A	122.69	42/itr	0.257	1052
apex4	2.89	461	2.89	54	0	0
clma	126.79	N/A	126.89	35/itr	0.0529	1336
misex3	31.27	98528	31.23	78	0.604	0
pdc	533.11	647142	533.27	4230	0.195	0
s298	36.70	1055	36.70	1	0	0
s38417	2724.2	N/A	2725.37	395/itr	0.533	344
s38584.1	2385.3	N/A	2385.50	252/itr	0.095	122
seq	689.72	N/A	689.41	13/itr	0.236	516
spla	698.48	569302	698.48	1440	0.00013	0
bigkey	876.46	N/A	876.46	14/itr	0	14
des	868.03	N/A	868.03	12/itr	0	0
diffeq	432.85	N/A	432.85	9/itr	0	379
dsip	1158.6	N/A	1158.63	24/itr	0	0
elliptic	240.35	N/A	240.35	3/itr	0	192
ex1010	2.50	109325	2.50	69	0	0
ex5p	0.47	305	0.47	16	0.024	0
frisc	1706.4	N/A	1706.44	107/itr	0.00585	472
tseng	1436.1	N/A	1436.24	13/itr	0.0786	247
Geomean	704.68	**178679**	704.74	**759**	0.115	234

†Full space vectors for smaller circuits (<25 inputs), and 100K vectors for the others.

TABLE IV. COMPARISON OF SER MITIGATION

Circuit	VPR(baseline)	PPL		COPAR	
	SER(FIT†)	*SER*(FIT)	*Ratio*	*SER*(FIT)	*Ratio*
alu4	17.05	17.08	100.17%	14.25	83.58%
apex2	18.40	18.07	98.21%	14.04	76.30%
apex4	118.40	105.98	89.52%	98.08	82.84%
clma	23.31	20.12	86.33%	19.78	84.85%
misex3	24.76	30.05	121.33%	21.56	87.05%
pdc	175.04	212.12	121.18%	141.09	80.60%
s298	2.68	1.92	71.52%	1.69	63.11%
s38417	547.02	503.30	92.01%	524.95	95.96%
s38584.1	620.70	586.47	94.49%	567.22	91.38%
seq	49.23	46.07	93.56%	38.64	78.48%
spla	269.48	247.18	91.72%	200.03	74.23%
bigkey	159.87	151.56	94.80%	130.09	81.38%
des	130.70	118.60	90.75%	123.79	94.72%
diffeq	91.08	82.81	90.92%	81.81	89.83%
dsip	208.98	168.53	80.65%	207.93	99.50%
elliptic	42.66	39.51	92.63%	39.90	93.55%
ex1010	122.41	117.97	96.37%	96.01	78.44%
ex5p	32.02	29.23	91.29%	28.76	89.84%
frisc	488.89	431.86	88.33%	455.16	93.10%
tseng	125.74	106.53	84.72%	117.58	93.51%
Geomean	**163.42**	**151.75**	**93.53%**	**146.12**	**85.61%**

†The raw error rate of an SRAM bit is assumed to be 0.01 (FIT/bit)

cube-based EPP analysis for smaller circuits (<25 inputs). As Table III shows, the speedup ratio is significant.

To verify the cross-layer optimized placement and routing for soft error mitigation, VPR (Versatile Place and Route) [20], the state-of-the-art placement and routing tool for FPGAs, is used. The wire segments are all unidirectional single length, and the routing channel width is set to be 30% larger than the minimum channel width required. Each CLB consists of four 6-input LUTs. After packing the netlist from LUTs into CLBs by T-VPACK [20], we applied three different algorithms: original timing-driven placement and routing algorithm by VPR (baseline), placement and routing algorithm guided by the physical level factor only (PPL) [8, 9], and our COPAR.

As shown in Table IV, compared with the results of VPR, our COPAR algorithm can decrease the SER by 14.39% on average. In contrast, although PPL algorithm reduces the number of care bits by 11.11% [9], the actual SER reduction is only 6.47% on average because of the inconsistence between the SER evaluation criterion and the placement and routing guidance criterion. We also validate the area and delay of the COPAR with VPR, no extra area cost and no extra performance cost are reported.

V. CONCLUSION

In this paper, we present a cross-layer optimized placement and routing algorithm for FPGA soft error mitigation, namely COPAR, successfully handling the gap between the SER evaluation criterion and the placement and routing guidance criterion. To obtain the application level factor accurately and efficiently, the cube-based EPP analysis algorithm is proposed. Experimental results show that, the accuracy of cube-based EPP analysis algorithm is more than 99%, while the computational complexity is low. And compared with the baseline VPR placement and routing technique, the COPAR can reduce the SER by more than 14% on average, without any area or performance overhead. In our work, due to the time limitation, 100K vectors are used in Monte Carlo simulation for circuits with more than 25 inputs. To simulate more test vectors is our future work.

VI. REFERENCES

[1] C. Carmichael, E.Fuller and et al, "Proton Testing of SEU Mitigation Methods for the Virtex FPGA," in Proc. of the Military and Aerospace Applications of Prgrammable Lgic Devices (MAPLD), 1999. pp. 23-25.

[2] E. Normand, "Single Event Upset at Ground Level," IEEE Transactions on Nuclear Science, Vol. 43, No.6, December 1996, pp. 2742-2750.

[3] M. Bellato, P. Bernardi and et al, "Evaluating the effects of SEU's affecting the configuration memory of an SRAM-based FPGA," in Proc. of IEEE Design Automation and Test in Europe Conference and Exhibition (DATE), 2004, pp. 188-193.

[4] H. Asadi, M.B. Tahoori and et al, "Soft Error Susceptibility Analysis of SRAM-Based FPGAs in High-Performance Information Systems," IEEE Transactions on Nuclear Science, Vol. 54, No.6, December 2007, pp. 2714-2726.

[5] P. Graham, M. Caffrey and et al, "Consequences and categories of SRAM FPGA configuration SEUs," in Proc. of Military Aerospace Applications of Prgrammable Logic Devices (MAPLD), 2003, pp. C6.1-C6.11.

[6] L. Sterpone, M. Violante, "A New Analytical Approach to Estimate the Effects of SEUs in TMR Architectures Implemented Through SRAM-Based FPGAs," IEEE Transactions on Nuclear Science, Vol. 52, No.6, December 2005, pp. 2217-2223.

[7] L. Sterpone, M. Violante, "A New Reliability-Oriented Place and Route Algorithm for SRAM-Based FPGAs," IEEE Transactions on Computers, Vol. 55, No.6, June 2006, pp. 732-744.

[8] H. R. Zarandi, S. G. Miremadi and et al, "SEU-Mitigation Placement and Routing Algorithms and Their Impact in SRAM-based FPGAs, " in Proc. of IEEE Int. Symp. on Quality Electronic Design (ISQED), 2007, pp. 380-385.

[9] S. Golshan, E. Bozorgzadeh, "Single-Event-Upset (SEU) Awareness in FPGA Routing," in Proc. of IEEE/ACM Design Automatic Conference (DAC), 2007, pp. 330-333.

[10] Hossein Asadi, M. B. Tahoori, "Analytical Techniques for Soft Error Rate Modeling and Mitigation of FPGA-based Designs," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 15, No. 12, December 2007, pp. 1320-1331.

[11] Jason Cong, Kirill Minkovich, "LUT-based FPGA Technology Mapping for Reliability," in Proc. of IEEE/ACM Design Automatic Conference (DAC), 2010, pp. 517-522.

[12] M. Violante, L. Sterpone and et al, "Simulation-Based Analysis of SEU Effects in SRAM-based FPGAs," IEEE Transactions on Nuclear Science, Vol. 51, No. 6, December 2004, pp. 3354-3359.

[13] N. G. Mankiw, "Principle of Economics," South-Western College Pub, 3 Edition, 2003.

[14] J. S. Takhar, D. J. Gilbert, "Test Pattern Generation for Multiple Output Digital Circuit using Cubical Calculus and Boolean Differences," in Proc. of IEEE Midwest Symp. on Circuits and Systrems, Vol. 1, 1997, pp. 409-412.

[15] J. P. Roth, "Computer Logic, Testing and Verification," Computer Science Press, 1980.

[16] Yan Lin, Lei He, "Device and Architecture Concurrent Optimization for FPGA Transient Soft Error Rate," in Proc. of IEEE/ACM Int. Conference on Computer-Aided Design (ICCAD), 2007, pp. 194-198.

[17] L. T, Wang, C. W. Wu, X. Q. Wen, "VLSI Test Principles and Architectures," Morgan Kaufmann Publishers, 2006.

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

[19] M. Abramovici, P. R. Menon, D. T. Miller, "Critical Path Tracing - An Alternative to Fault Simulation, " in Proc. of IEEE/ACM Design Automatic Conference (DAC), 1983, pp. 214-220.

[20] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," in Proc. of IEEE Int. Workshop on Field-Programmable Logic and Applications, 1997, pp. 213-222.