Reliability-Aware Placement and Routing for FPGAs

A Thesis Presented

by

Mohammed A. Abdul-Aziz

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements
for the degree of

Master of Science

in

Electrical and Computer Engineering

in the field of

Computer Engineering

Northeastern University
Boston, Massachusetts

February 2010
Acknowledgements

First and foremost I would like to express sincere gratitude’s to my advisor, Prof. Mahdi Tahoori, for his valuable guidance and invaluable input throughout my work. Without him this work would not have been possible.

I would like to express my thanks and respect to Dr. David Kaeli, for his extraordinary classes, which helped me gaining perspective and interest in various subjects in the field and for serving in my thesis defense committee.

I would also like to thank the EMC Corporation and Kevin Granlund for providing the co-op opportunity, and my fellow graduate student, Syed Shazili who established the grounds for the work there. I would like to thank the Fulbright Scholarship and the IIE for making this opportunity coming true.

I would like to thank Prof. Kaeli and Prof. Mi for serving in my thesis committee.

I would like to thank my fellow Graduate students in the Dependable Nano-Computing Laboratory, for their feedback, and brief time I shared with them, and my friends in the NUCAR for all the help they provided me, and for their friendship.

And last but not least, my wife and daughter, for the patience and the hard times that we went through, and my family.

I would like to thank and appreciation for the many other people, who I can’t list all of their names here, cause the list will go on, for all the help they showed me along the way.

Thanks.
Abstract

Soft errors are intermittent malfunctions of hardware that are not reproducible. They may affect the data integrity and affect the system operation. These errors are growing reliability threat in VLSI system design. A soft error occurring in a memory cell or register is called a Single Event Upset (SEU).

Designs mapped into Field Programmable Gate Arrays (FPGAs) are more vulnerable to soft errors than ASIC implementations, due to the large number of configuration memory used to map the design into the FPGA. An SEU causes a unique failure mode in the mapped design due to the unique architecture of FPGAs.

In this work, we try to mitigate the effects of SEUs in SRAM based FPGAs. An SEU occurring in one of the configuration bits of the FPGA may cause a permanent error in the implemented circuit. Since the majority of FPGA real estate is dedicated for SRAM configuration bit, mitigating soft error in these configuration bits can improve the reliability of the mapped design. The goal of this work is the development of an SEU-aware placement and routing tool that is capable of producing an implementation that is less vulnerable to SEUs. To achieve this, we modified the cost function for the placement and the routing part of the VPR tool to reduce the effects of SEU in the final mapped design. The VPR tool is a general purpose FPGA placement and routing tool that is widely accepted and used in the academic field.

Two classes of errors that can be caused by an SEU in the routing resources were considered: switch open errors and switch short errors. During placement, we applied a cost function to estimate for the sensitivity to reduce the chances of the occurrence of these errors. We optimize for the switch open errors by minimizing the number of switches used for routing a net of the circuit. For reducing switch short errors, we try to minimize the overlapping area between nets. During routing, we carefully assign routing resources for the nets to minimize these errors. By using this approach we were able to reduce the number of total sensitive errors by 58% in average, but that reduction comes at the cost of increased critical path delay by on 54% on average.
Contents

CHAPTER 1 INTRODUCTION.................................................................................................................. 1

CHAPTER 2 FPGA ARCHITECTURE AND SEU EFFECTS................................................................. 5

2.1 SRAM-BASED FPGA ARCHITECTURE ......................................................................................... 5

2.2 SEUs IMPACT ON INTEGRATED CIRCUITS............................................................................... 10

  2.2.1 SEU Definition, History and Mechanism............................................................................... 10

  2.2.2 Radiation Effects in SRAM Memory Cell............................................................................... 12

  2.2.3 SEU Impact on SRAM-Based FPGAs .................................................................................... 12

2.3 SEU MITIGATION IN FPGAS AND PREVIOUS WORK.............................................................. 17

CHAPTER 3 FPGA DESIGN AUTOMATION TOOL.......................................................................... 22

3.1 THE VPR TOOL .......................................................................................................................... 22

3.2 PLACEMENT ................................................................................................................................ 24

3.3 ROUTING .................................................................................................................................... 31

CHAPTER 4 SEU-AWARE PLACEMENT AND ROUTING ................................................................. 36

4.1 ESTIMATION OF SENSITIVE BITS DURING PLACEMENT ..................................................... 36

  4.1.1 Estimating for Open Sensitive Bits......................................................................................... 37

  4.1.2 Estimating for Short Sensitive Bits......................................................................................... 38

4.2 SEU-AWARE ROUTING............................................................................................................... 42

  4.2.1 Open Sensitive Aware Routing.............................................................................................. 43

  4.2.2 Short Sensitive Aware Routing.............................................................................................. 44

4.3 OPTIMIZATION ......................................................................................................................... 45

  4.3.1 Optimizing for the Short Sensitive Placement....................................................................... 47

  4.3.2 Optimizing for the Short Sensitive Routing......................................................................... 53
CHAPTER 5 EXPERIMENTAL RESULTS ................................................................. 56

5.1 OPEN SENSITIVE COMPARISON ................................................................................. 60
  5.1.1 Open Sensitive Comparison Using Timing Driven Router ................................. 60
  5.1.2 Open Sensitive Placement (Open Sensitive Router) Comparison .................... 63

5.2 SHORT SENSITIVE COMPARISON ........................................................................... 68
  5.2.1 Short Sensitive Placement Comparison Using Timing Driven Router ............... 68
  5.2.2 Short Sensitive Estimate 2 Placement Comparison Using Timing Driven Router .. 71
  5.2.3 Short Sensitive Placement (Short Sensitive Aware Router) Comparison .......... 73

5.3 TOTAL SENSITIVE BITS ......................................................................................... 77

CHAPTER 6 CONCLUSION ....................................................................................... 82

REFERENCES .......................................................................................................... 83
List of Figures

FIGURE 1 Generic FPGA Architecture ................................................................. 6
FIGURE 2 Basic Logic Block Element ................................................................. 7
FIGURE 3 2-input LUT implemented in an SRAM-based FPGA .......................... 7
FIGURE 4 (A) CLB pins (B) Routing architecture for Island Style FPGA ........... 8
FIGURE 5 Three types of PIP used in SRAM-based FPGAs ............................... 8
FIGURE 6 (A) Connection Box (B) Switch block implementation with Pass Transistor ........... 9
FIGURE 7 Channel segmentation distribution .................................................... 9
FIGURE 8 SEU occurring in a combinational logic ............................................. 11
FIGURE 9 SEU effect in SRAM Memory cell .................................................... 12
FIGURE 10 Impact of SEU on routing signals ................................................... 17
FIGURE 11 CAD flow ......................................................................................... 23
FIGURE 12 Simulated Annealing Flow Chart .................................................... 25
FIGURE 13 Net representation in VPR, and Bounding Box ............................... 27
FIGURE 14 Pseudo-code to update the bounding box of net 1 incrementally ....... 29
FIGURE 15 Estimation of open sensitive bits ..................................................... 37
FIGURE 16 Estimation of Short Sensitive bits based on Bounding Box ............. 39
FIGURE 17 Calculation of switch matrix usage probability ............................... 40
FIGURE 18 Usage Probabilities distribution (A) net 1 (B) net 2 ......................... 42
FIGURE 19 Two net topologies routed using Pass Transistor routing switches .... 44
FIGURE 20 Short Sensitive Aware Router operation ........................................ 45
FIGURE 21 Structures Used for Optimization .................................................. 48
FIGURE 22 Array of linked list for nets, sorted based on their XMAX .................. 49
FIGURE 23 Determining the overlapping area, net1 XMAX is less than net2 and net3 XMIN .... 49
FIGURE 24 Pseudo-code for the optimization techniques ............................... 50
FIGURE 25 Four dimensional arrays, points to the switch usage probabilities, computation optimized ................................................................. 51

FIGURE 26 Two dimensional arrays, points to the switch usage probabilities, memory optimized ................................................................. 52

FIGURE 27 Example of the routing file output ......................................................................................................................... 53

FIGURE 28 Addressing of blocks and channels for VPR................................................................. 54

FIGURE 29 Three different switch blocks defined by VPR ................................................................. 58

FIGURE 30 Symmetric switch module algorithm .................................................................................. 58

FIGURE 31 Circuit e64-4lut net 59 (I_43_) routed using Open Sensitive Aware Router .............. 67

FIGURE 32 Circuit e64-4lut net 59 (I_43_) routed using Timing Driven Router ......................... 67

FIGURE 33 Circuit e64-4lut net 163 (N_N366) routed using Timing Driven Router ....................... 76

FIGURE 34 Circuit e64-4lut net 163 (N_N366) routed using Short Sensitive Aware Router .......... 77

FIGURE 35 Plotting the data point averages of total sensitive bits (Table 15) vs average of critical path delay (Table 16) for various placement and routing combinations .......... 81
List of Tables

Table 1 Number of Configuration bits versus the number of flip flops ........................................... 14
Table 2 MTTM for routing, logic and user bits .................................................................................. 15
Table 3 Number of sensitive SRAM bits for each part of the FPGA .................................................. 15
Table 4 Comparison of the Open Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path Timing Driven placement and the Bounding Box placement ........................................................................ 61
Table 5 Comparison of the Open Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Net Timing Driven placement and the Bounding Box placement ........................................................................ 62
Table 6 Comparison of the Open Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Bounding box placement routed using the Timing Driven router, and using the Open Sensitive aware Router ........................................................................................................... 64
Table 7 Comparison of the Open Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path Timing Driven placement routed using the Timing Driven router, and the Bounding box placement using the Open Sensitive Aware router ........................................................................................................... 65
Table 8 Comparison of the Open Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Net Timing Driven placement routed using the Timing Driven router, and the Bounding box placement using the Open Sensitive Aware router ........................................................................................................... 66
Table 9 Comparison of the Short Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing for each channel) between the Path timing driven placement and the Short Estimate 1 placement ........................................................................... 70
TABLE 10 Comparison of the Short Sensitive Bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path timing driven placement and the Short Estimate 2 placement .............................................................. 72

TABLE 11 Comparison of the Short Sensitive Bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path timing driven placement (routed using the Timing Driven Router) and the Short Estimate 1 placement ........................................................................................................................................ 74

TABLE 12 Comparison of the Short Sensitive Bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path timing driven placement (routed using the Timing Driven Router) and the Short Estimate 2 placement ........................................................................................................................................ 75

TABLE 13 Total Sensitive bits for Bounding Box, Path Timing Driven, Net Timing driven, Short Sensitive Estimate 1 and Short Sensitive Estimate 2 routed using the Timing Driven router (TimingR) and the Short Sensitive Aware router ................................................................. 79

TABLE 14 Critical Path Timing for Bounding Box, Path Timing Driven, Net Timing driven, Short Sensitive Estimate 1 and Short Sensitive Estimate 2 routed using the Timing Driven router (TimingR) and the Short Sensitive Aware router ................................................................. 80
Chapter 1

Introduction

The continuous increase in transistor density in the semiconductor devices, following the prediction of Moore’s law, and the continuous shrinking of transistor dimensions, voltage scaling and increase in operating frequency came at a cost of increased susceptibility to various source of noise and radiation induced soft errors.

Field Programmable Gate Arrays (FPGAs) are one of the semiconductor devices that benefited from the continuous advances in VLSI technology, allowing them to provide more functionality, higher performance and lower power consumption, making them more desirable to be adopted by system designer over other alternatives, such as Application Specific Integrated Circuits (ASICs). FPGA offers several advantages over ASICs are as follows:

- Lower non-recurring cost: manufacturing ASICs involves making lithography masks which incurs high cost, while FPGAs can be programmed by means of loading the configuration into the FPGA.

- Lower Time to Market (MTTM): FPGAs are ready to be programmed in seconds once the design is complete and implemented using the Computer Aided Design (CAD) tools, while programming ASICs involves making lithography masks, and entering fabrication process to get the design implemented.

- In case of error in the design, ASIC implementation have to be discarded, and new corrected lithography masks have to be made, and fabricate new corrected ASICs, while design mapped into FPGA can be corrected by means of loading the corrected design.

- FPGA offers flexibility for changing requirement, reprogramming in case of project objective changes, more over it offers great value for space mission in
which an FPGA can service multiple objectives, simply by reprogramming the FPGA, the function of the FPGA can be changed.

- High performance approaching that of ASICs.

Beside the advantages of FPGAs over ASICs, the inclusion of specialized hardware block such as hardcores (PowerPC™ core in the Virtex II from Xilinx) [PowerPC, 10], Phased Locked Loop (PLLs) [Virtex5,09] and the ability to implement softcores (such as Nios™II from Altera [StratixIV’09] and Micro-Blaze™ form Xilinx [MicroBlaze, 07]) in the FPGA. All of these features made FPGAs commonly adopted in many application domains such as industrial, spacecraft, automotive, embedded application, storage systems, networking and prototyping. Three main types of programmable FPGAs are commonly in use: SRAM programmable, flash memory programmable and anti-fuse programmable.

Like other semiconductor devices (integrated circuits), FPGAs also suffered from radiation induced soft errors. Radiation induced soft errors causes a number of different failure modes in semiconductor devices. A soft error occurring in an individual memory cell or register is called Single Event Upset (SEU). SEUs are nondestructive events (thus soft) caused by ionizing radiation sources (such as Alpha particles and Neutrons) striking sensitive junction of CMOS devices. An SEU can cause a bit flip (0 to 1 or vise versa) in the value stored in a memory cell.

Technologically, SRAM programmable FPGAs are usually one generation ahead of other types of FPGAs, because SRAM programmable FPGA is built using standard CMOS process manufactures [Wang, LHCElectronics’03]. The configuration memory of SRAM-based FPGA is susceptible to SEUs that might alter the configuration of the mapped design. Anti-fuse programmable FPGAs configuration, unlike that of the SRAM and flash programmable FPGAs, are not affected by radiation induce SEUs, but on the other hand they are one time programmable.

The failure modes due to SEUs in SRAM programmable FPGA are distinct from those in other semiconductor devices (such as ASICs). An SEU occurring in one of the configuration bits can lead to a permanent error in the mapped design, depending on whether the configuration bit being used by the mapped design or not. This distinct
failure mode, and the fact that a soft error may lead to permanent error in the mapped design, limits the adoption of these devices in mission critical applications.

To implement a circuit into an FPGA, it is essential to use a CAD tools to map the design into the FPGA blocks, and set necessary configuration bits to implement the circuit, these CAD tool are traditionally designed to optimize for certain goals such as: routing resources usage, critical path delay, wirelength, area and power. Now given the sensitivity of FPGA mapped design to soft errors, it is necessary to add another goal to optimize for, the reliability of the design to soft errors.

Majority of SRAM programmable FPGA real estate is dedicated to SRAM memory cells to hold the configuration of the mapped circuit (more than 98% of memory resources in an FPGA are configuration bits), and considering the fact that routing configuration bits accounts for 78% or more of the total configuration bit of a mapped design [Graham, MAPLD, 03]. By carefully allocating the routing resource, one can reduce or mitigate the effect of SEUs in the routing resources, and improving on the reliability of the FPGA mapped design. To do that, we need to understand the effect of SEU on the routing resources, model the failure modes caused by the SEU, and consider these failure modes during mapping the design into the FPGA, and carefully allocate FPGA resources to the mapped design.

In this work, we incorporate reliability constraints to account for SEU in a well know FPGA CAD tool, the VPR tool. These constraints capture the possible error due to SEU during mapping the design into the FPGA, and minimize these errors, while optimizing for the other parameters, namely delay, area, and routing resources.

We analyzed the reliability of the placed and routed circuits using the original algorithms implemented in the VPR tool, and we implemented two new cost functions for the placement part. The two functions aims at reducing the chances of occurrence of bridging between nets, one by reducing the overall overlapping are between nets, while the other tries to predict the routing of the nets, and tries to reduce the number of shared routing switches.

We also modified the cost function of one of the routers implemented in the VPR tool (the shortest path first router). We implemented two routers, an open sensitive bit aware router, which accounts for the number of routing switches used to route a net, and
optimize for the number of these switches thus reducing the number of switch open errors. The second router is a short sensitive aware router, which accounts for the switch short try to route the nets using paths that reduce the chances of forming a switch short error between two nets. That can be achieved by using different tracks (of the routing channel) for nets passing through a routing switch.

The rest of the thesis is organized as follows, in Chapter 2 we describe the FPGA architecture that we used throughout the work, we present some background information on SEU effect on semiconductor devices as well, and we describe the FPGA failure mode due to SEU. Chapter 3 will discuss the operation of the VPR tool that we used in our work. Our approach and the estimates we implemented will be described in Chapter 4. Chapter 5 will present the result obtained using our approach, and compare it against the traditional optimization goals. We will show the tradeoffs of our approach in term of critical path timing and routing resources usage (as a representation of area overhead). The conclusion of our work will be presented in Chapter 6.
Chapter 2

FPGA Architecture and SEU Effects

In this chapter we discuss relevant background related to this work, and also review the various soft error mitigation techniques and previous work. The organization is as follows: Section 2.1 discuss the SRAM based FPGA architecture, Section 2.2 discusses the history of radiation induced soft errors, and its impact on the integrated circuits, their effect on memory element, and the different failure mode they cause in FPGAs. In Section 2.3 we will review the previous work related to mitigating the effects of SEU in semiconductor devices and FPGAs.

2.1 SRAM-Based FPGA Architecture

FPGAs are attractive for designer for their ability to implement any circuit simply by programming it appropriately. The quality of the implemented circuit (in terms of performance) depends on: the quality of the CAD tool used for mapping the circuit, the quality of the FPGA architecture, and the electrical design (i.e. transistor-level design of the FPGA) of the FPGA [Betz, Kluwer, 99].

FPGAs consist of a large number of programmable logic blocks (each of which implement a small amount of digital logic), programmable routing (which allows the logic block inputs and output to be connected to form larger circuit), I/O blocks (to provide interface to the outer world), and memory blocks. FPGAs may include specialized hardware circuitry that is often used in digital systems, such as DSP block [Altera, Handbook’09], DLLs to support complex clocking schemes [Virtex5,09], and more recently embedded processor hardcore (e.g. PowerPC™ 405 cores in the Xilinx Virtex-II Pro) [PowerPC, 10].

A generic island style FPGA (Fig.1) is composed of a two-dimensional array of identical Configurable Logic Blocks (CLB), on the perimeter of this array, there are
Input/Output Blocks, these components are connected through a programmable interconnect network which consists of switch matrices and wires. This arrangement of resources is commonly used by FPGAs from Xilinx, and Lucent. Other styles are the row-based used by Actel, and hierarchical FPGAs from Altera. In this work we focus on island style FPGAs.

The CLBs are the main logic resource for implementing sequential as well as combinatorial circuits in the FPGA, these CLB is composed of one or more Basic Logic Block (BLE) (shown in Fig. 2). Each CLB may contain one or more BLE, in which case BLEs will be arranged into a cluster.

![Generic FPGA Architecture](image)
Most current commercial FPGAs use logic blocks based on look-up tables (LUTs shown in Fig. 3). 4 input lookup tables are most commonly used, recently there is a move toward a 6 input lookup tables [Virtex5, 09]. In the island-style FPGA, CLBs are surrounded by routing channels of pre-fabricated wiring segments on all four sides, a CLB input or output (called an input or output pin Fig. 4(a) ) can connect to some or all of the wiring segments in the channel adjacent to it via a connection block (Fig. 4(b) ).
Programmable routing is realized by the use of PIPs. Three implementations Programmable Interconnect Points (PIP) are shown in Fig.5. To describe the routability of FPGAs, two terms are used: Connection block flexibility ($F_c$) which specify the number of wire tracks that which an input or output pin (from now on called only pin) can connects to ($F_c$ for Fig. 6 (a) is 2 or 50%). Switch block flexibility ($F_s$) specifies the number of outgoing tracks that an incoming track can connects at each switch box, located at the intersection of vertical and horizontal channels ($F_s$ for wire shown Fig. 6 (b) is 3). This can be represented by either an absolute number of tracks that can be connected to, or as a fraction of the total number track of the channel.
Wire track length in the FPGA are represented by the number of Logic blocks it spans, Fig. 7 show an example of channel wire segments of different lengths, where 40% of the tracks are of length 1, 40% are length 2 and 20% are of length 4.
Finally we want to make a distinction between two categories of memory elements [Asadi, VTS’05]:
- Configuration bits: are used for specifying the particular circuit mapped into the FPGA.
- User bits: such as flip-flops (FFs) or on-chip memory bits, holds the current state of the circuit.

After loading a design into an FPGA, the contents of configuration bits are supposed to remain unchanged, while the contents of user bits can be changed at any clock cycle.

2.2 SEUs Impact on Integrated Circuits

2.2.1 SEU Definition, History and Mechanism

Since its introduction, semiconductor devices and Integrated Circuits (IC) witnessed continuous advancement in there fabrication process technology, the most appealing for digital design being CMOS process technology. In this technology, the advancement manifested itself by the continuous shrinking of transistor geometry, scaling of operating voltage, and increase in operating frequency, this development was driven by the increasing demand for higher density (functionality), better performance (usually related to frequency), and lower power consumption (portability).

However, with today’s very deep submicron ICs scaling, lower noise margin, and low circuit node capacitance, the effect of various noise sources are becoming of great concern. Also, the effect of radiation (which used to be an issue for space, and aerospace application) at terrestrial levels is becoming the source of increasing number intermittent errors. Soft errors have recently become a huge concern in advanced commercial ICs because they can induce a product failure rate that is higher than all the other reliability mechanisms combined [Baumann, HSES’04].

These errors are called soft because the circuit itself is not permanently damaged due to the radiation, if the error cause by the soft error is captured and corrected, the system will perform correctly. The mechanism of soft error is as follows, when a single
heavy ion strikes a sensitive region in silicon (of a semiconductor device), it loses its energy via the production of free electron-hole pairs, resulting in a dense ionized track in the local region. Some of the generated charge will recombine to form a very short duration current pulse at the transistor that was struck by the particle. This short pulse can cause a bit flip the value stored in a memory cell, or be interpreted as a change of value at the input of a logic gate, causing the output of the gate to change respectively (Fig. 8).

Three mechanisms are responsible for soft errors has been identified: the reaction of high-energy cosmic neutrons with silicon and other device materials, the reaction of low-energy cosmic neutrons with high concentrations of $^{10}$B in the device, and alpha particles emitted from trace radioactive impurities in the device materials.

The frequency of the soft errors in semiconductor device is denoted by Soft Error Rate (SER). The variation of the SER due to technology scaling has been monitored for different systems. The SER for DRAM systems is relatively unchanged by scaling, while the SER for SRAM and peripheral logic systems is increasing rapidly with each new technology node. In fact, soft errors are now the dominant failure mode of all reliability mechanism combined (in qualified products). [Baumann, DTC’05].

![Figure 8 SEU occurring in a combinational logic](image)

**Figure 8 SEU occurring in a combinational logic**
2.2.2 Radiation Effects in SRAM Memory Cell

When a charged particle strikes one of the sensitive nodes of a memory cell, such as the drain in an off state transistor, it generates a transient current pulse that can turn on the gate of the opposite transistor. The effect can produce an inversion in the stored value, in other words, a bit flip in the memory cell. Memory cells have two stable states, one that represent a stored “0” and one that represents a stored “1”. In each state two transistors are turned on, and two are turned off. A bit flip in the memory element occurs when an energetic particle causes the state of the transistor in the circuit to reverse, this will causes an SEU (as shown in Fig. 9). This is one of the major concerns in digital circuits.

![Figure 9 SEU effect in SRAM Memory cell [KASTENSMIDT, 06]](image)

2.2.3 SEU Impact on SRAM-Based FPGAs

The physical design of FPGAs, i.e. the way that FPGA resources are utilized to map a particular design, has a considerable impact on the reliability of that mapped design. For instance, if an FPGA resource fails, due to either a permanent or transient error, depending on whether this resource is used by the mapped design or not, a failure might or might not occur in the system.

Not all FPGA resources failures cause system error in a mapped design. Based on error susceptibility of a mapped design, the FPGA resources can be categorized into sensitive and non-sensitive resources. Non-sensitive bits act as “don’t care” configuration
bits for that particular design. Hence, the sensitivity of particular configuration bit is application-dependent.

The effects of SEUs on digital circuits implemented in an FPGA can be classified into a) transient and b) permanent errors. SEU can cause transient errors in the combinational logic components, which can be propagated and captured in flip-flops. Also, SEUs can directly make transient errors on memory element and change the contents of memory caches, main memories, register file and flip-flop (FFs). These errors are called transient because they may be overwritten or corrected using error-detection-and correction techniques. So, Transient errors impact the user-defined logic and flip-flops of the FPGA.

Moreover, SEUs can make permanent errors on an FPGA if they alter the contents of configuration bits. Note that these errors differ from those errors which damage the device (hard errors of physical defects). In this case, the configuration bit remains erroneous until the new configuration is downloaded into the FPGA. So, these permanent errors are recoverable. In the rest of this work, when we refer to permanent errors, we mean recoverable permanent errors. We already categorized the memory resources in the FPGA into a) user bits and b) configuration bits. An SEU on user bit cause a transient error and an SEU on configuration bits may lead to a permanent error.

An SEU changing a configuration SRAM cell makes a permanent effect until the original configuration bit stream is re-downloaded into the FPGA. This type of error is the major error type in FPGAs because the number of SRAM cells dominates user-defined memory elements. Typically, the number of SRAM configuration cells is more than 99% of all memory elements inside an FPGA. As an example, Table 1 shows the number of configuration bits and the number of flip-flops for some Xilinx Virtex FPGA devices.

Permanent errors are classified into routing errors, LUT bit-flips, and control/clocking bit-flips. Since routing resources consume more than 80% of FPGA real estate, the majority of FPGA vulnerability is due to errors in the routing resources, especially SEUs in the routing configuration bits.
<table>
<thead>
<tr>
<th>Device</th>
<th>No. of FFs (Nff)</th>
<th>No. of Config. Bits (Ncb)</th>
<th>Nff/Ncb</th>
</tr>
</thead>
<tbody>
<tr>
<td>XCV50</td>
<td>1536</td>
<td>559200</td>
<td>0.27%</td>
</tr>
<tr>
<td>XCV100</td>
<td>2400</td>
<td>781216</td>
<td>0.31%</td>
</tr>
<tr>
<td>XCV200</td>
<td>4704</td>
<td>1335840</td>
<td>0.35%</td>
</tr>
<tr>
<td>XCV300</td>
<td>6144</td>
<td>1751808</td>
<td>0.35%</td>
</tr>
<tr>
<td>XCV400</td>
<td>9600</td>
<td>2546048</td>
<td>0.38%</td>
</tr>
<tr>
<td>XCV800</td>
<td>18816</td>
<td>4715616</td>
<td>0.40%</td>
</tr>
<tr>
<td>XCV1000</td>
<td>24576</td>
<td>6127744</td>
<td>0.40%</td>
</tr>
<tr>
<td>XC2V2000</td>
<td>21504</td>
<td>7492000</td>
<td>0.29%</td>
</tr>
<tr>
<td>XC2V4000</td>
<td>46080</td>
<td>15659936</td>
<td>0.29%</td>
</tr>
<tr>
<td>XC2V8000</td>
<td>93184</td>
<td>29063072</td>
<td>0.32%</td>
</tr>
</tbody>
</table>

Table 1: number of configuration bits versus the number of flip flops

In a recent soft error analysis work in FPGA [Asadi, FPGA’05], a new metric called MTTM (mean time to manifest error) was used to quantify the vulnerability of different resources to error. MTTM for a particular resource (e.g. the SRAM cell controlling the state of a PIP or an LUT cell) is defined as the average time (the number of clock cycles) from an error in that resource to a system failure (error appearing at primary outputs) due to that error. Resources with smaller MTTM are more vulnerable to soft error. Table 2 shows the average MTTM for programmable routing resources, LUTs and user flip flops. As can be seen in this table each configuration routing bit (used by the mapped design) is 7X more vulnerable than logic configuration bits and 3X more vulnerable than user flip-flops. Since the number of routing configuration bits is also far more than logic configuration bits and user bits (Table 3), the reliability of a mapped design is greatly dependent on reliability of configuration routing resources. By reducing the total number of vulnerable routing configuration bits used in the physical mapping of a design, its reliability can be considerably improved.

Routing resources can be inter-CLB or intra-CLB. An inter-CLB routing signal connects two or more CLBs. Those that used inside a CLB are called intra-CLB signals. Switch matrices and line segments are used to route inter-CLB while multiplexers and buffers are mostly used for intra-CLB routing. Select-bits of multiplexers comprise more than half of the total susceptible SRAM cells to SEUs as shown in [Graham, MAPLD, 03].
An SEU changing a configuration routing bit causes a switch open, switch short, or bridging error (wired-or, wired-and), as shown in Fig. 10. Buffer errors are buffer on and buffer off errors.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Routing</th>
<th>LUT</th>
<th>User FFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>s298</td>
<td>2.73</td>
<td>20.22</td>
<td>4.95</td>
</tr>
<tr>
<td>s344</td>
<td>2.59</td>
<td>17.48</td>
<td>5.62</td>
</tr>
<tr>
<td>s349</td>
<td>2.91</td>
<td>20.44</td>
<td>5.62</td>
</tr>
<tr>
<td>s382</td>
<td>3.3</td>
<td>22.11</td>
<td>6.98</td>
</tr>
<tr>
<td>s386</td>
<td>3.88</td>
<td>30.61</td>
<td>15.76</td>
</tr>
<tr>
<td>s400</td>
<td>3.13</td>
<td>20.8</td>
<td>7.23</td>
</tr>
<tr>
<td>s444</td>
<td>3.00</td>
<td>21.92</td>
<td>7.34</td>
</tr>
<tr>
<td>s510</td>
<td>4.87</td>
<td>34.77</td>
<td>3.8</td>
</tr>
<tr>
<td>s526</td>
<td>4.03</td>
<td>27.68</td>
<td>8.14</td>
</tr>
<tr>
<td>s641</td>
<td>2.42</td>
<td>16.96</td>
<td>12.1</td>
</tr>
<tr>
<td>s713</td>
<td>2.41</td>
<td>16.82</td>
<td>13.06</td>
</tr>
<tr>
<td>s953</td>
<td>3.23</td>
<td>21.22</td>
<td>24.16</td>
</tr>
<tr>
<td>s1196</td>
<td>5.17</td>
<td>36.59</td>
<td>23.81</td>
</tr>
<tr>
<td>s1238</td>
<td>5.56</td>
<td>41.23</td>
<td>24.06</td>
</tr>
<tr>
<td>s1488</td>
<td>4.47</td>
<td>29.92</td>
<td>3.81</td>
</tr>
<tr>
<td>s1494</td>
<td>4.6</td>
<td>31.39</td>
<td>3.82</td>
</tr>
<tr>
<td>average</td>
<td>3.64</td>
<td>25.63</td>
<td>10.63</td>
</tr>
</tbody>
</table>

Table 2 MTTM for routing, logic and user bits [Asadi, VTS’05]

<table>
<thead>
<tr>
<th>circuit</th>
<th>Routing</th>
<th>LUT</th>
<th>Control / Clocking</th>
<th>Total Conf Bits</th>
<th>No. of FFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>s298</td>
<td>982</td>
<td>410</td>
<td>133</td>
<td>1525</td>
<td>14</td>
</tr>
<tr>
<td>s344</td>
<td>1035</td>
<td>392</td>
<td>168</td>
<td>1595</td>
<td>15</td>
</tr>
<tr>
<td>s349</td>
<td>1450</td>
<td>520</td>
<td>187</td>
<td>2157</td>
<td>15</td>
</tr>
<tr>
<td>s382</td>
<td>1849</td>
<td>712</td>
<td>207</td>
<td>2768</td>
<td>21</td>
</tr>
<tr>
<td>s386</td>
<td>1689</td>
<td>660</td>
<td>160</td>
<td>2509</td>
<td>6</td>
</tr>
<tr>
<td>s400</td>
<td>1949</td>
<td>700</td>
<td>218</td>
<td>2867</td>
<td>21</td>
</tr>
<tr>
<td>s444</td>
<td>1690</td>
<td>692</td>
<td>208</td>
<td>2590</td>
<td>21</td>
</tr>
<tr>
<td>s510</td>
<td>3122</td>
<td>1244</td>
<td>299</td>
<td>4665</td>
<td>6</td>
</tr>
<tr>
<td>s526</td>
<td>2401</td>
<td>856</td>
<td>227</td>
<td>3484</td>
<td>21</td>
</tr>
<tr>
<td>s641</td>
<td>3038</td>
<td>1056</td>
<td>375</td>
<td>4469</td>
<td>19</td>
</tr>
<tr>
<td>s713</td>
<td>2793</td>
<td>988</td>
<td>355</td>
<td>4136</td>
<td>19</td>
</tr>
<tr>
<td>s953</td>
<td>7906</td>
<td>2644</td>
<td>597</td>
<td>11147</td>
<td>29</td>
</tr>
<tr>
<td>s1196</td>
<td>7980</td>
<td>2976</td>
<td>613</td>
<td>11569</td>
<td>18</td>
</tr>
<tr>
<td>s1238</td>
<td>8608</td>
<td>3224</td>
<td>652</td>
<td>12484</td>
<td>18</td>
</tr>
<tr>
<td>s1488</td>
<td>9660</td>
<td>3688</td>
<td>702</td>
<td>14050</td>
<td>6</td>
</tr>
<tr>
<td>s1494</td>
<td>9670</td>
<td>3628</td>
<td>695</td>
<td>13993</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 3 Number of sensitive SRAM bits for each part of the FPGA [Asadi, VTS’05]
Consider a chain of PIPs connecting two nodes of the circuit. An SEU changing a PIP control from 1 to 0 will cause the connection switch open, resulting in an open error in the gate-level netlist (Fig. 10 (a)). However, the situation for 0 to 1 SEUs is different. Not all switch shorts due to a $0 \rightarrow 1$ SEU cause bridging errors in the netlist. Consider the switch matrix shown in Fig. 10(b). Three different nets are shown which use PIPs (W1, N1), (W2, S1), and (E1, W1) between the two switches (originating from a CLB). An SEU ($0 \rightarrow 1$) on the unused PIP (W1, S1) or (N1, S1) causes a bridging error between nets A and C and (W1, E1) causes a bridging error between nets A and B. These PIPs are called sensitive PIPs and the corresponding controlling configuration bits are sensitive bits (We will call this class of errors Short Sensitive errors). However, an SEU on the unused PIP (N2, E2) does not cause any bridging since no two nets are adjacent to this PIP. Hence (N2, E2) is a non-sensitive PIP.

Based on the above classification, the reliability of a mapped design is inversely proportional to the number of sensitive bits. For both transient (SEUs in the sensitive configuration bits) and permanent (open and bridging faults in the wire segments as well as switch stuck-open and stuck-closed) errors, if a sensitive bit or a device controlled by a sensitive bit fails, there will be an observable error in the system. However, if a non-sensitive bit or a device controlled by (connected to) a non-sensitive bit fails; there will be no error in the system. Therefore, in order to improve the reliability of a design mapped into the FPGA it is extremely important to reduce the number of sensitive bits. In the Chapter 4, we will explain how to achieve this during the placement and routing phases of FPGA physical mapping.
2.3 SEU Mitigation in FPGAs and Previous Work

Radiation hardening is one way to mitigate the effects of soft error. It can be achieved by careful transistor resizing for the sensitive nodes. By increasing the size of the transistor, the critical charge need to be deposited to cause a glitch is also increased, reducing the soft error rate. Another method proposed in [Diril, VTS'05], it suggests varying the supply voltage and using threshold voltage modulation together with variable capacitance.
banks. By controlling these parameters, the soft error tolerance of the circuit can be controlled, varying depending on the soft error rates, which is monitored using a simple on chip circuitry. A time redundancy based mitigation scheme was presented in [Nicolaidis, VTS’99]. By adding a delay component to the signals, one can suppress the effect of soft error glitches (this method represent a simple temporal redundancy scheme).

However the failure mode in the SRAM based FPGAs are unique than those occurring at the ASIC, several studies has been conducted to model and understand these failures.

Several techniques were proposed previously to mitigate the effect of SEU errors. These can be divided into reconfiguration based techniques, or redundancy based technique [Sterpone, TOC’06]. Reconfiguration based technique is based on the scrubbing approach, where the configuration is stored in a reliable memory (Flash or EEPROM). By continually rewriting configuration memory (partially if configuration frame is supported, or entirely if not), an SEU occurring at the configuration memory will be restored after rewriting the configuration. Scrubbing rate define how often the configuration is rewritten. Redundancy based techniques exploit additional hardware techniques or additional computation time for detecting the presences of SEUs propagation to the circuit’s output. Fault Masking can be achieved using Triple Module Redundancy (TMR) approach, where three replicas of the same circuit are implemented and a voter circuit compares the outputs of the circuit, and erroneous output will be voted out.

Studies based on heavy ions and neutron irradiation of SRAM based FPGA, the goal is the characterization of soft errors for both static (without toggling the clock and input kept unchanged) and dynamic (with active clock and inputs). Non-intrusive upset detection and partial reconfiguration in combination with TMR was used to mitigate the SEU occurring at the Virtex II FPGA [Carmichael, MAPLD’01] [Yui, REDW’03].

The approach presented in [Asadi, VTS’05] is based on two FPGAs. One is the main SRAM-Based FPGA and the other an auxiliary anti-fuse based FPGA. The main FPGA performs the user function and have a larger logic density. However it is SEU sensitive (due to the fact that it is SRAM-Based). The auxiliary is of lower logic density and serve as cyclical redundancy check (CRC) checker circuit. The main FPGA is
configured in a way that the CRC code is stored by the end of each configuration frame, and the CRC code is also stored in the memory of the auxiliary FPGA. The auxiliary FPGA read a frame of the configuration memory, calculates the CRC code, and checks it with the stored CRC code in the auxiliary memory. In the case of an SEU error found, the configuration memory is restored by error correction obtained from the CRC checker.

Altera Stratix III introduced a background error detection circuitry, since the introduction of the 130-nm process generation [Altera, 07], where they introduced a CRC engine to enable continual verification of the Configuration RAM (CRAM) contents during device operation, the CRC is guaranteed to detect up to a maximum of three bit errors. The CRC is integrated within the FPGA (on-chip), and the circuitry is robust and not susceptible to soft errors. The engine is self-contained and can be enabled using the programming tool.

The Traditional way to achieve radiation hardening in FPGA is through Triple Module Redundancy (TMR) for the implemented circuit, either full TMR or partial TMR, however even with that a bridging error between two redundant copies of the circuit, or in the voter circuit (majority voter), will turn the redundancy overhead unusable, since error in one circuit can be propagated to another circuit, or the majority voter cannot correctly compares between the results.

Characterization and categorization of the Virtex SRAM based FPGA SEU induced failure modes were studied in [Graham, MAPLD, 03]. The study used a PC-based bitstream SEU simulator. The simulator loads the same design into two Virtex FPGAs, one FPGA is loaded with the correct design (implements the golden run), while injecting errors (simulating SUE) into the configuration memory of the second FPGA by means of partial reconfiguration.

Seven failure modes were observed: MUX select, PIP short, PIP open, buffer off, buffer on, LUT value change and control bit change, and these errors can be further be divided into two categories, “routing errors” and “logic errors”. It was observed that most of the design failures (78% to 84.4% of total failures) are due to failures in routing resources, and the remaining (~20%) are due to upsets in control bits and LUT value changes. They also noted that routing multiplexers changes have the most significant
impact on reliability of all failure modes, accounting for 73% of the sensitive programming data.

An SEU-Aware router was introduced in [Golshan, DAC’07]. It was based on the fact that routing configuration bit contribute to around 90% of the total configuration bits, and the majority of the configuration bits in routing architecture is 0-bits. Most of these 0-bits correspond to switches which do not belong to the routes of the nets in the design. Hence reducing 0-bits that affect the functionality of the design implies reducing adjacency between routed net. Switch box configuration bits were considered.

In the work the sensitive bits were put into three categories:

- 1-per-net care bits: any closed switch connecting two routing resources corresponding to the same net is considered as a 1-per-net care bit.
- 0-per-net care bits: any open switch that hurts the functionality of one net in case it is closed is referred to as 0-per-net-care bits. This mainly occurs in switch boxes. Either the both segments are used by the same net (like T junction of a multi-terminal net) or it is between a used segment and unused segment in buffered switches. If a 0-per-net care bit gets closed, it either provides redundant route between two nodes, or it would generate a combinational closed loop.
- Cross-net care bits: Any interference on nets caused by closing an open switch between two different nets is considered as a cross-net care bit. An open switch between two routing resource corresponding to two different nets is considered as a cross-net care bit.

A maze router-based routing algorithm was proposed for the two terminal nets to find the optimum path. The cost function is the sum of the care bits presented above. For routing multi-terminal net, they introduce a maze-based heuristic approach, where they use the existing routing to connect the remaining terminals. By using the existing routing they eliminate the need to update the path cost. Thus using the same components for the cost used for two terminal nets. They suggested that the component of the timing driven router of the VPR tool (Described in Chapter 3) will also optimize for the 1-per-net care bits, since optimizing for timing implies using fewer number of switches for routing (where each switch connection introduces extra capacitance, increasing the net delay).
They suggest the use of Asymmetrical SRAMS (ASRAM-0) which makes the $0 \rightarrow 1$ upsets less probable than $1 \rightarrow 0$ upsets.

In a similar approach by [Zarandi, ISQED’07], they proposed modifying the VPR tool, so that during placement and routing. Decisions are taken with awareness of SEU-mitigation. Moreover, no redundancies during the placement and routing are used but the algorithms are based on the SEU avoidance. The results suggest that placement and routing can decrease the SEU rate of circuit implemented on FPGAs about 18% and 12% respectively. However, it increases critical path delay and power consumptions of the circuit up to 5% and 8% respectively. It was suggested that by using fault-avoidance method, mitigation of SEU effects would decrease up to 22%, and that this method is notable compared to previous TMR and Duplication with Comparison (DWC).

A new reliability-oriented place and route algorithm called RoRA is introduced in [Sterpone, TOC’06]. It is based on using TMR approach for target circuits. Although TMR technique decreases the effects of SEUs significantly, in some types of FPGAs (that uses MUX in their routing resources) one SEU can cause multiple errors. So it is important to place and route different copies of the TMR implementation in a fashion that reduces this effect. They presented an algorithm that is able to place and route the logic functions and the signals of a design in such a way that the number of SEUs affecting the configuration memory and possibly causing FPGA misbehaviors is drastically reduced with respect to common redundancy-based approach such as TMR. They reported that a reduction of about 22% of the circuit speed, while the capability to tolerate SEU effects in the FPGA’s configuration memory increases up to 85% with respect to TMR approach.

In our work we focus on modifying the CAD tool to account for reliability during the placement and routing phases, while trying to optimize for the area and delay as well, looking at each class of errors separately first, then look at the combined effect of the final placed and routed circuit.
Chapter 3

FPGA Design Automation Tool

Mapping a circuit into an FPGA is a complicated task, and it is usually broken into two steps, placement and routing. Traditionally the goal of these CAD tools was the optimization of area, delay and power. Now given the susceptibility of FPGA to permanent errors cause by SEUs in the configuration bits, we have to account for the reliability of the mapped circuit, which we will discuss in Chapter 4.

In this chapter we will describe the FPGA CAD tool that we will modify. It is arranged as follow: in section 3.1 we will describe the operation of a widely accepted FPGA CAD tool, and the details of the algorithm implemented in this tool. In Section 3.2 we will explain the details of the simulated annealing based placer of the VPR tool, and will discuss the annealing schedule implemented. The two routers (The shortest path first and the timing driven router) will be described in Section 3.3.

3.1 The VPR Tool

To achieve our goal, that is to improve the reliability of an FPGA mapped design against SEU, we implement our approach using a well known FPGA CAD tool, the VPR tool. By introducing the reliability constraints as the cost function of simulated annealing-based placement algorithm, and the shortest-path-first router of the tool.

The Versatile Place and Route (VPR) is a CAD tool that is used for the placement and routing of array FPGA, developed at the University of Toronto [Betz, Kluwer, 99] [Betz,FPL’97]. It was originally designed to evaluate the utility of new architectural feature experimentally.

According to the CAD flow used by the tool (Fig. 11) [Betz,Manual’00], the SIS synthesis package is used to perform technology-independent logic optimization of each
circuit. Then each circuit is technology mapped into 4-LUTs and flip flops by FlowMap. The output of FlowMap is a .blif format netlist of LUTs and flip flops. After that the T-VPack program comes into roll, where it packs this netlist of 4-LUTs and FFs into more coarse-grained logic blocks. It outputs a netlist in the .net format that the VPR uses. VPR can then place the circuit and either globally route it, or perform combined global and detailed routing on the placed circuit.

![Figure 11 CAD flow](BetzManual'00)

The output of VPR consists of a file describing the circuit placement, another file describing the circuit’s routing, and various statistics concerning the minimum number of tracks per channel required to successfully route, the total wirelength, etc. In order to find the minimum number of tracks required for successful routing, VPR actually attempts to
route the circuit several times with different numbers of tracks allowed per channel in each attempted routing.

In our work, we tried to keep much of the desired functionality of VPR, such as the automatic schedule for placement and routing, as well as the automatic generation of routing architecture intact, generally because these parameters have been thoroughly studied before hand, and it the changes that we made were mostly compatible with these techniques.

VPR source code is publicly available, and the tool is widely accepted and used in the academic field and research. Plus it is part of the SPEC2000 benchmark (CINT2000). It provides a frame work to apply our approach to place and route the FPGA, in way that improve the reliability of the final implementation of the circuit, That can be adopted to implement wide selection of FPGA architecture. Since we can easily vary the parameters of the architecture (size of the FPGA in terms of CLB, channel width, switches architecture, etc.).

### 3.2 Placement

The VPR tool uses the simulated annealing algorithm for placement [Kirkpatrick, Science’83], The flow chart for the implemented algorithm is presented in Fig. 12

Simulated annealing is widely accepted algorithm for placement, and floorplaning. Other algorithms that are also used for placement are the mincut and analytic based placers. Simulated annealing mimics the annealing process used to gradually cool molten metal to produce high-quality metal objects.

The algorithm starts by a random initial placement. A large number of moves are then made to gradually improve the placement. A logic block is selected at random and a new location for it is also selected at random. The change in cost function that would result from moving the selected logic block to the proposed new location is computed. If the cost would decrease, the move is always accepted and the block is moved. If the cost would increase there is still a chance of acceptance given by $e^{-\Delta C/T}$, where $\Delta C$ is the (positive) change in cost function the move causes, and $T$ is a parameter called temperature that control the likelihood of accepting a move that make the placement
worse. Initially, $T$ is very high so almost all moves are accepted, it is gradually decreased as the placement is refined so that eventually the probability of accepting a move that makes the placement worse is very low. This ability to accept hill-climbing moves that make a placement worse allows simulated annealing to escape local minima in the cost function.

![Simulated Annealing Flow Chart](image)

**Figure 12 Simulated Annealing Flow Chart**

The rate at which the temperature is decreased, the exit criterion for terminating the placement, the number of moves attempted at each temperature (*InnerLoopCriterion*), and the method by which potential moves are generated are defined by the annealing schedule. A good annealing schedule is crucial to obtain good results in a reasonable amount of CPU time.
VPR allows for a user defined schedule, where the user can input the schedule parameters (number of moves at each temperature, initial temperature and the rate at which the temperature decreases, exit temperature, etc.) and assumes default values for the parameters that wasn’t specified by the user.

The other option is to use an automatic schedule built into the VPR tool, where the parameter for the simulating annealing are automatically generating, based on the parameter of the circuit to be placed (number of nets in the circuit, number of blocks, etc.). The placer starts with a random placement then perform a set of random moves on the initial placement, and set the initial temperature \((\text{Initial}T)\) to \(20\sigma\), where \(\sigma\) is the standard deviation of the cost over these moves. New temperature (which is calculated after each move limit) is dynamically calculated and it reduced by a factor \(\alpha\) which varies dynamically based on the number of accepted moves at the old temperature.

The \textit{InnerLoopCriterion} is fairly complex for this schedule, it involves monitoring the fraction of new states generated that have their costs within certain range of the average cost at that temperature, and there area several cases and fall-back cases define. Finally the anneal terminates (\textit{ExitCriterion}), when:

\[
T < \varepsilon \cdot \frac{\text{Cost}}{N_{\text{nets}}}
\]  

\(3-1\)

The \(R_{\text{limit}}\) parameter in Fig. 12 controls how close together blocks must be considered for swapping. Initially \(R_{\text{limit}}\) is fairly large (being the size of the entire chip), and swaps of blocks far apart on a chip are likely for the first part of the anneal. Shrinking gradually during the middle stages of the anneal, and finally being 1 logic block at low temperature. \(R_{\text{limit}}\) is updated according to the following relation:

\[
R_{\text{limit}}^{\text{new}} = R_{\text{limit}}^{\text{old}} \cdot (1 - 0.44 + \alpha)
\]  

\(3-2\)

In the schedule the number of moves attempted in the “inner loop” is:

\[
\text{MovesPerTemperature} = \text{innerNum} \cdot \left(\text{number \_of \_blocks}\right)^{1.33}
\]  

\(3-3\)

Where the default value for \text{innerNum} is 10.
VPR support 4 types of cost functions: Linear Congestion, Nonlinear Congestion, Net Timing Driven and Path Timing Driven. The Linear Congestion cost function considers the Bounding Box of the nets of the circuit (Fig. 13), and it merely considers minimizing the wirelength of the circuit. The cost function for the Linear Congestion placement is:

$$Cost_{linear\_congestion} = \sum_{i=1}^{N_{nets}} q(i) \left[ \frac{bb_x(i)}{C_{av,x}(i)^{\beta}} + \frac{bb_y(i)}{C_{av,y}(i)^{\beta}} \right]$$

(3-4)

Where the summation is over the $N_{nets}$ in the circuit.

**Figure 13 Net representation in VPR, and Bounding Box**

For each net $i$, $bb_x(i)$ and $bb_y(i)$ denote the horizontal and vertical spans of the bounding box, respectively. The $q(i)$ factor compensate for the fact that the bounding box wire length model underestimates the wiring necessary to connect nets with more than three terminals. Its value depends on the number of terminals of net $i$. $q(i)$ is 1 or nets
with 3 of fewer terminals, and slowly increases to 2.79 for nets with 50 terminals. $C_{av,x}(i)$ and $C_{av,y}(i)$ are the average channel capacities (in tracks) in the $x$ and $i$ directions, respectively, over the bounding box of net $i$.

This cost function penalizes placements which require more routing in areas of the FPGA that have narrower channels. The exponent $\beta$, in the cost function allows the relative cost of using narrow and wide channels to be adjusted. When $\beta$ is zero, the linear congestion cost function reverts to the standard bounding box cost function. The larger the value of $\beta$, the more wiring in narrow channels is penalized relative to wiring in wider channels.

$C_{av}$ depends only on the channel capacities, which do not change during placement, and on the maximum and minimum coordinates of a bounding box. Therefore $C_{av,x}$ and $C_{av,y}$ values are pre-computed and stored in a two-dimensional array, indexed by the bounding box minimum and maximum coordinates.

One more feature the VPR tool implements an optimization technique to speed up the process of computing the nonlinear congestion cost. Even with a good annealing schedule, millions of potential block swaps will be evaluated in a typical placement run. The most computationally expensive part of evaluating a swap is computing the change in cost $\Delta C$, the swap would produce.

In any swap of two blocks taking place in the inner loop (during the operation of VPR), the only terms in the summation (3-4) that change are those corresponding to the nets attached to the two swapped blocks. The bounding boxes of all the nets attached to either of these two blocks must be recomputed, and then the summation can be used to determine $\Delta C$. The re-computation of the net bounding boxes is the key step here, unless care is taken, it can dramatically slow the placer.

The straightforward way to re-evaluate a net’s bounding box is to examine the location of each of its terminals. Unfortunately, this is an $O(k)$ operation for a $k$-terminal net. Large circuits typically have many high-fanout nets, a few which have hundreds of terminals. As well, since high-fanout nets have terminals on so many blocks, swapping any two blocks has a high probability of disturbing some high-fanout nets.

An alternative to this brute-force computation was developed (and we adopted an extension of this technique for our cost function), which is called incremental bounding
box evaluation. Where for each net, the coordinates of each of the four sides of the bounding box \((x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}})\), and the number of net terminals that lie on each of these sides \((N_{x_{\text{min}}}, N_{x_{\text{max}}}, N_{y_{\text{min}}}, N_{y_{\text{max}}})\) is stored. Fig. 14 shows an example of this data storage.

Now consider a move from some terminal of this net is moved via a swap from \((x_{\text{old}}, y_{\text{old}})\) to \((x_{\text{new}}, y_{\text{new}})\). Since the extra information is stored, Usually it is easy to determine the new net bounding box by looking only at the terminal which moved, rather than all \(k\) terminals, by checking the number of terminals on each edge. Fig. 14 lists a pseudo-code used to update the \(x_{\text{min}}\) and \(N_{x_{\text{min}}}\) values for a net \(i\), the same code can be used for the other three sides.

```plaintext
if(x_{\text{new}}!=x_{\text{old}}){ /* Terminal has moved horizontally */
    if(x_{\text{new}}<x_{\text{min}}(i)){ /*Terminal moved left past old \(x_{\text{min}}\) edge */
        x_{\text{min}}(i)=x_{\text{new}};
        N_{x_{\text{min}}}(i)=1;
    }
    else if (x_{\text{new}}==x_{\text{min}}(i)){ /*Terminal moved left to lie on the old \(x_{\text{min}}\) edge */
        N_{x_{\text{min}}}(i)+1;
    }
    else if (x_{\text{old}}==x_{\text{min}}(i)){ /* Terminal was on \(x_{\text{min}}\) edge; moved right*/
        if(N_{x_{\text{min}}}>1){ /* Still terminals on \(x_{\text{min}}\) edge? */
            N_{x_{\text{min}}}-1;
        }
        else { /* Terminal moved within \(x_{\text{min}}\) edge */
            BruteForceBoundingBoxRecompute(i);
        }
    }
}
```

**Figure 14** Pseudo-code to update the bounding box of net \(i\) incrementally

There is only one case for which the net bounding box must be recomputed by the brute-force procedure: when the terminal moved is the only net terminal on a side of the bounding box, and it is moved inward, toward the bounding box center. In this case the recomputation is \(O(k)\), while all other cases it is \(O(1)\). The probability of an arbitrary net terminal being on some side of the bounding box, and being the only terminal on that side
of the bounding box is $=1/k$, however, hence the average bounding box recomputation is $O(1 + \frac{1}{k} \cdot k) = O(1)$.

The net timing driven placement implements a timing analysis to compute the delay of all of the paths in a circuit. These delay values are then be used to guide the algorithm so that it can reduce the critical paths.

To maximize the quality and performance of the placed circuit, VPR model the delay of each connection in the circuit as the circuit is placed. In a tile based FPGA, the FPGA structure is homogeneous, i.e. every $x$, $y$ location in the FPGA is constructed from identical tiles. VPR exploit the uniformity architecture by computing the delay of a connection between two blocks as a function only of the distance $(\Delta x, \Delta y)$ between them.

To allow an efficient assessment of the delay between blocks that are $\Delta x$ and $\Delta y$ distance apart. VPR computes a delay lookup matrix indexed by $\Delta x$ and $\Delta y$. To compute a given $(\Delta x, \Delta y)$ entry in the matrix, VPR router is employed to determine the delay between two blocks that are $(\Delta x, \Delta y)$ distance apart. To do this, a source block is placed at a location $(x_{source}, y_{source})$ in the FPGA, and a sink block is placed at $(x_{source} + \Delta x, y_{source} + \Delta y)$. Then the timing-driven router is used to perform a routing between the two blocks, and the delay is recorded in the delay lookup matrix at location $(\Delta x, \Delta y)$. This process is then repeated for every possible $\Delta x$ and $\Delta y$ value in the FPGA. Note that our connection delay estimate does not depend on a nets fanout.

The timing driven placement cost of a connection $(i,j)$ is as follows:

$$\text{Timing\_Cos}\, (i,\,j) = \text{Delay}\, (i,\,j) \cdot \text{Criticality}\,(i,\,j)^{\text{Criticality\_Exponent}} \quad (3-5)$$

The Criticality reflects the criticality of the current connection, depending on amount of delay that can be added to the connection without increasing the critical path delay (more on Criticality in the next section). And the total Timing\_Cost for a circuit is the sum of the Timing\_Cost of all of its connections.

As for the Path Timing Driven placement, it is a combination of the two cost, the bounding box placement cost (wiring cost) and the timing driven placement cost (timing cost) normalized by the following equation:
\[
\Delta C = \lambda \cdot \frac{\Delta \text{Timing Cost}}{\text{Previous Timing Cost}} + (1 - \lambda) \cdot \frac{\Delta \text{Wiring Cost}}{\text{Previous Wiring Cost}}
\]  \hspace{1cm} (3-6)

The variable $\lambda$ is defined by the user, and it is used as a trade-off variable to determine how much weight to give each component. The $\text{Previous Timing Cost}$ and $\text{Previous Wiring Cost}$ are normalization variables that are updated once every temperature.

The effect of these two normalization components is to make the function weight the two components only with the $\lambda$ variable, independent of their actual values. It automatically adjust the weights of the two components so that the algorithm is always allocating $\lambda$ importance to changes in the Timing Cost, and $(1 - \lambda)$ importance to change in the Wiring Cost [Marquardt, FPGA’00].

### 3.3 Routing

Before the placement, the exact location of the blocks is unknown, so no exact information about how the connection between blocks belonging to one net is known, and neither the exact delay or wirelength information are available, all we can do is to our best in estimating these parameters during placement, and optimize for them. After the placement phase is finished, we need a router that is able to optimize for the parameter which we placed the circuit for.

A big part of the router implemented in VPR is dedicated to make the router, graphics routing, delay extractor and various other routines all operate on a direct graph that describes the FPGA. This graph is called routing resource graph. Its representation is very general and can describe a wide variety of FPGA architectures. However, describing a new architecture by creating a routing-resource graph by hand is not feasible- the routing resource graph to describe a typical FPGA containing 8000 4-LUTs and is almost 30 MB in size. One possibility is to design a basic tile (a single logic block and its associated routing) manually, and create a program to automatically replicate and stick together this tile into a routing-resource describing the entire FPGA. Even creating a basic tile manually is too time-consuming; however a typical tile contains several
hundred PIPs and wires, so it can take hours or days to describe even one tile. Furthermore such a hand-crafted tile is designed for one value of routing channel width, W. While it is desirable to experiment with different channel width, or how well the CAD tool route a given architecture, using a hand-crafted for each and every change in the architecture (like increasing the value of W by 1) would require lots of time.

VPR is designed so that it takes a concise architecture definition file, and uses an internal graph generator to create a highly detailed routing resource graph representation which the router and other CAD routines can use. If a new class of FPGAs that does not fit into the implemented format of the architecture file, then, only the routing graph generator has to be modified; the router, graphics, timing analyzer, and statistics routines will all function correctly, so the VPR router is highly modularize, so that changes in one part can be done independently from the other parts.

The VPR router is based on the Pathfinder negotiated congestion algorithm [Betz, Kluwer, 99] [Betz, FPL’97]. Basically, this algorithm initially routes each net by the shortest path it can find, regardless of any overuse of wiring segments or logic block pins that may result. One iteration of the router consists of sequentially ripping-up and re-routing (by the lowest cost path found) every net in the circuit. The cost of using a routing resource is a function of the current overuse of that resource and any overuse that occurred in prior routing iteration By gradually increasing the cost oversubscribed routing resources, the algorithm forces nets with alternative routes to avoid using oversubscribed resource leaving only the net that most needs a given resource behind. The global routing is a routing stage that plans the approximate routing path of each net to reduce the complexity of routing task and guide the detailed router. While the detailed router finalize the exact pin-to-pin connections (details which pin of the CLB get connected to which track, and details the switch box connections, as well as connection box connection configuration). The VPR router is capable of performing global routing only, or a combined global-detailed routing.

There are two routing algorithm used to route nets in the VPR tool, the first one is the breadth first search router (Dijkstra’s shortest path first algorithm, i.e. a maze router [Betz, Kluwer, 99] [Betz, FPL’97]), which focuses more on the routability of circuit
(routing the net with fewer routing tracks). The second router is the timing driven router, which focuses on routing the circuit with minimum critical path delay. The routers

For a k terminal net, the maze router is invoked k-1 times to perform all the required connections. In the first invocation, the maze routing wavefront expands out from the net source until it reaches any one of the k-1 net sinks. The path from source to sink is now the first part of this net’s routing. The maze routing wave front is emptied, and a new wavefront expansion is started from the entire net routing found thus far. After k-1 invocation of the maze router all k terminals of the net will be connected.

To optimize the run time of the algorithm, in the latter invocation of the maze router the partial routing used as the net source will be very large, and it will take long time to expand the maze router wavefront out to the next sink. The optimization technique used here is that, when a sink is reached, it adds all the routing resource segments required to connect the sink and the current partial routing to the wavefront (i.e. the expansion list) with a cost of 0 and it does not empty the maze routing wavefront, it just continue expanding normally expand normally. Since the new path added to the partial routing has a cost of zero, the maze router will expand around it at first. Since this new path is fairly small, it will take relatively little time to add the new wavefront, and the next sink will be reached much more quickly that if the entire wavefront expansion had bee started from scratch.

The cost function used for the breadth first search router is given below, where $Cost(n)$ is the cost of using routing resource $n$ when it is reached by connecting it to routing resource $m$ is:

$$Cost(n) = b(n).h(n).p(n) + BendCost(n,m)$$ $(03-7)$

Where $b(n)$ is the base cost for resource $n$, and it depends on the type of the routing resource $n$ (depends on whether the resource is input pin, output pin, wire track, etc.), it varies from 0 to 1 depending on the type of the routing resource $n$ being considered. $h(n)$ is the historical congestion and it is increased after every routing iteration in which node $n$ is overused and gives the router “congestion memory.” $p(n)$ is the present congestion cost of node $n$, it is 1 if using this node to route the current
connection will not cause any overuse, and increase with the amount of overuse of the node. \( p(n) \) is also a function of the number of routing iterations that have been performed. In early iterations, \( p(n) \) grows slowly with the current overuse of node \( n \), in later iterations, \( p(n) \) goes up very rapidly with overuse of node \( n \).

The \( \text{BendCost}(n,m) \) term is an enhancement to improve the result of the global routing (it will serve our goal to optimize for reliability later on). It penalizes bends when global routing is being performed, since global routes with many bends make it difficult or impossible for a subsequent detailed routing phase to utilize long wire segments. Hence reducing the number of bends in a global tends to lead to detailed routes that are both faster and require fewer tracks. If global routing only is being performed, \( \text{BendCost}(n,m) \) is 1 if making the connection from node \( m \) to node \( n \) implies a bend- i.e. node \( m \) is a horizontal channel segment and node \( n \) is a vertical channel segment or vice versa. Including this \( \text{BendCost} \) term in the total cost of using a node produces routes with very few unnecessary bends and does not significantly increase the global routing track count. If combined global-detailed routing is being performed there is no need to penalize bends, so \( \text{BendCost}(n,m) \) is always zero in this case.

The timing driven router uses the same components of the breadth first search cost function and introduces another factor of the cost function that is Criticality:

\[
\text{Crit}(i, j) = 1 - \frac{\text{slack}(i, j)}{D_{\text{max}}}
\]  

Where \( D_{\text{max}} \) is the delay of the circuit critical path, and \( \text{slack}(i,j) \) is the amount of delay that could be added to this connection before it affected the circuit’s critical path. \( \text{Crit}(i,j) \) is therefore between 0 and 1.

The cost of using a routing resource node, \( n \), as part of connection \((i,j)\) is:

\[
\text{Cost}(n) = \text{Crit}(i, j) \cdot \text{delay}(n) + \left[ 1 - \text{Crit}(i, j) \right] b(n) \cdot h(n) \cdot p(n)
\]
The first term is the delay sensitive term, the criticality of the connection times the intrinsic delay of the node. The second term is the congestion sensitive term. \( b(n) \) is the base cost of node \( n \), and is set equal to \( delay(n) \).
Chapter 4

SEU-Aware Placement and Routing

In our work, we try to minimize the affects of SEU on the programmable routing resources of the FPGA by modifying an existing, well known FPGA CAD tool (the VPR tool), and using a generic FPGA architecture. We don’t introduce any extra hardware, nor consider any hardware modification. Moreover, the proposed approach don’t introduce any redundancy in the hardware resource of the FPGA.

We modify the placement and routing algorithms in the VPR tool [Betz, Kluwer, 99] [Betz,Manual’00] by introducing the reliability constraints as the cost function of simulated annealing-based placement algorithm, and the shortest-path-first router of the tool. We will show that by using these reliability measures, we are able to improve the reliability of mapped design to FPGAs while consider the effect on both the area and delay parameter of the circuit.

4.1 Estimation of Sensitive Bits During Placement

At the placement phase, the exact routing paths of the signals are not available. The goal of the placement is to assign location of each logic block in the design. The routing of the nets will be done after the placement. However, the information regarding the location of each net at the placement phase can be obtained from its bounding box (BB). Therefore, our estimation of routing sensitive bits this phase is based on the tentative bounding box information of each net.
4.1.1 Estimating for Open Sensitive Bits

The number of open sensitive bits for a particular net is proportional to the number of PIPs used in the routing of that net. Therefore, the number of open sensitive bits for a net can be estimated from the number of switch matrices that the net is routed through.

First let’s consider a two-terminal net and its corresponding bounding box shown in Fig. 15. The BB is specified by the coordinates of its two corners, \{(x_1, y_1),(x_2, y_2)\}, in terms of the CLBs. The number of switch matrices used for routing can be estimated by the Manhattan distance, \(|x_1-x_2|+|y_1-y_2|+1\). This estimate is highlighted in the Fig. 15.

Also, the number of switch matrices, as a lower bound, can be estimated as \(\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}\). This is known as the Euclidean distance.

\[
Open\_Estiamte = |x_1 - x_2| + |y_1 - y_2| + 1 \tag{4-1}
\]

For the multi-terminal nets (three terminals or more), we can either use the estimate based on only the bounding box or use more accurate estimate based on Steiner tree. Since estimation of open sensitive bits during the placement phase is very similar to wire length estimation, we can use any of the available techniques for wire length estimation for this purpose.

![Figure 15 Estimation of open sensitive bits](image)
4.1.2 Estimating for Short Sensitive Bits

A short sensitive bit is formed if two nets pass through the same switch matrix and there is an unused PIP between two used switches in those nets. Therefore, if two nets do not share any switch matrix in their routing paths, there are no short sensitive bits associated with them; however, if they share some switch matrix in their paths, there is a chance of the formation of a short sensitive bit.

4.1.2.1 Short Sensitive Estimate1

During the placement phase, the amount of overlap between the bounding boxes of two nets can be used as a metric to estimate the short sensitive bits. Fig. 16 shows the bounding boxes of two nets and the overlap between them. The width and the height of these nets in terms of the number of CLBs are denoted by W1, L1, W2 and L2, respectively. The width and the height of the overlap area are w and l, respectively. We can simply estimate the number of short sensitive bits as the number of CLBs in the overlap area, i.e.:

\[ \text{Estimate}_1 = w \cdot l \]  \hspace{1cm} (4-2)

By reducing the overall overlapping area between nets, we would be able to reduce the number of short sensitive bits, since the chances that a short sensitive bit is reduced (through the reduction of the shared routing switches), the same estimate should work for multi-terminal nets.
4.1.2.2 Short Sensitive Estimate

To more accurately estimate the number of short sensitive bits, we also need to consider the probability of using each switch matrix in the bounding box. In other words, since the approximate route of a net is expressed by its bounding box, there is a possibility of each switch matrix within the bounding box being actually used in their routing. The distribution of the usage probabilities is not uniform around the bounding box. Consider the bounding box in which a net is supposed to be routed from the top-left corner to the bottom-right corner. The top-left switch matrix as well as the bottom-right one will be always used in the routing path with probability of 1.

Here we explain how to calculate the usage probability for each switch matrix within the bounding box of a net. Consider a bounding box, for the sake of simplicity, the top-left corner is \((I, I)\) and the bottom-right corner is \((m,n)\). Also, assume that the terminals of this net are located in the top-left and bottom right corners. In order to calculate the usage probability of a switch matrix at the position \((i,j)\) (Fig. 17), we need to divide the number of paths going from \((I, I)\) to \((m,n)\) that go through \((i,j)\) over the number of all possible paths from \((I, I)\) to \((m,n)\). Note that since shortest path is used in the routing, we just consider the number of shortest paths in the enumeration. In other words, the direction of routing is always from left to right and from top to bottom. If the number of all shortest
paths from $(l,l)$ to $(m,n)$ is denoted by $f(m,n)$, the usage probability of switch matrix $(i,j)$, denoted by $p(i,j)$, can be calculated as:

$$p(i, j) = \frac{f(i, j) \times f(m-i+1, n-j+1)}{f(m, n)}$$  \hspace{1cm} (4-3)

In eq. (4-3), $f(i,j)$ is the number of shortest paths from $(l,l)$ to $(i,j)$ and $f(m-i+1,n-j+1)$ is the number of shortest paths from $(i,j)$ to $(m,n)$, as shown in Fig. 17. $f(m,n)$ can be recursively expressed as:

$$f(m,n) = f(m-1,n) + f(m,n-1)$$
$$f(1,n) = 1$$
$$f(m,1) = 1$$  \hspace{1cm} (4-4)

Using this recursive formula, we can obtain non recursive equation for $f(m,n)$ as:

$$f(m,n) = \binom{m+n-2}{n-1}$$  \hspace{1cm} (4-5)

Based on equations (4-3) and (4-5), we can calculate the usage probability as follows:
\[ p(i, j) = \frac{\binom{i + j - 2}{j - 1} \binom{m + n - (i + j)}{n - j}}{\binom{m + n - 2}{m - 1}} \] (4-6)

Note that for multi-terminal nets, instead of setting the source and destination points as the top-left and bottom-right corners of the bounding box (i.e. \((1,1)\) and \((m,n)\)), the location of terminals of the nets must be set as the source and destination points.

Assume that \(p^{N_1}(i,j)\) is the probability of using switch matrix \((i,j)\) for net \(N_1\). Therefore, for any switch matrix \((i,j)\) in the overlap area, the probability of both nets being routed through that switch matrix is given by \(p^{N_1}(i,j) \cdot p^{N_2}(i, j)\). Hence, a more accurate estimate for the number of short sensitive bits due to two particular nets \(N_1\) and \(N_2\) can be expressed as follows:

\[
\text{Estimate}_2 = \sum_{\forall (i,j) \in \text{overlap area}} p^{N_1}(i, j) \cdot p^{N_2}(i, j)
\] (4-7)

Fig. 18 shows the usage probabilities of the switch matrices in the BB of the nets in the example of Fig. 16.

We can extend this approach to consider multi-terminal nets by either iterating for each source-destination pair for the network, or by multiplying by a factor that represent the number of terminals.
We show the switch usage probabilities of two nets in Fig. 18. Since during placement we are estimating for the switch usages, we assumed that the net terminals are located on the diagonal corners of the bounding box, which is not actually the case.

**4.2 SEU-Aware Routing**

After the placement phase, the locations of the CLBs become known, so the router is ready to connect block belonging to a net together.

Note that the original routing algorithm discussed above (in section 3.1.2 ), optimized for timing (that is critical path timing) so that we can run our net faster, however, that may not achieve our goal in reliability, and may conflict with it, at least it doesn’t exploit the potential of the reduction achieved by the placement.

So we introduce two routing algorithms, one for the open sensitive bit, and the other for the short sensitive bit. The implementation for these algorithms is similar, and fairly simple, as will be discussed below.
4.2.1 Open Sensitive Aware Routing

To optimize the router for the open sensitive bits, reducing the number of switch used for routing can achieve that (as explained in section 3.2.1), the bounding box placement (open sensitive placement) get the blocks ready for routing that can optimize for the open sensitive bits.

Optimizing for timing during routing, have an implicit component of optimizing for the number of switches used in the routing, however, we emphasized the part for the number of routing switches in the cost function for the breadth first search (shortest path or Dijkstra’s algorithm) by changing the bending part of the cost function, and making it account for the number of routing switches:

\[
Cost(n) = b(n)J(n).p(n) + \text{OpenSensitiveBit}(n,m)
\]  \hspace{1cm} (4-8)

Where the OpenSensitiveBit(n,m) is set to 1 when ever routing node n to node m involve a routing switch (a connection from a wire segment from a routing channel to another wire segment of a routing channel). Note that using a route with less delay, may imply using a short path, with fewer switch, but the OpenSensitiveBit(n,m) emphasize on the cost function to achieve that, but that is not necessarily true for the timing driven router, which uses the Elmor Delay model to estimate the delay of a path, for example, refer to Fig. 19.

According to Elmore delay model, the delay of the net on the left is:

\[
T_{\text{Elmore, left}} = T_{\text{obuf}} + 6R_{\text{obuf}}C_{\text{total}} + 8R_{\text{pass}}C_{\text{total}}
\]  \hspace{1cm} (4-9)

While the Elmor delay for the net on the right:

\[
T_{\text{Elmore, right}} = T_{\text{obuf}} + 7R_{\text{obuf}}C_{\text{total}} + 6R_{\text{pass}}C_{\text{total}}
\]  \hspace{1cm} (4-10)

If the resistance of the logic block output buffer \(R_{\text{obuf}}\) is less than twice the resistance of a routing pass transistor (the typical case), the topology on the right has
lower delay to sink 1. If sink 1 is time-critical therefore, the topology on the right is superior even though it uses more wirelength (thus more switches). [Betz, Kluwer, 99]

![Two net topologies routed using pass transistor routing switches.](image)

**Figure 19** Two net topologies routed using pass transistor routing switches. [Betz, Kluwer, 99]

### 4.2.2 Short Sensitive Aware Routing

Optimizing for the short sensitive bits during routing, we introduce a factor in the breadth first search router cost function that account for the short sensitive bits whenever a PIP switch is considered for routing:

\[ Cost(n) = b(n).h(n).p(n) + \text{ShortSensitiveBit}(n,m) \]  

Where \( \text{ShortSensitiveBit}(n,m) \) is the number of short sensitive bits introduced if connecting routing resource \( n \) and \( m \) incurs using a switch box. It checks for any PIP that could form a bridging connection between two nets caused by the connection under consideration.

When a routing track (wire track of a routing channel) is to be routed through a routing switch, the router checks all the possible connection that can be made by that routing track for for a bridging error. In any net (other than the net being routed), if a possible bridging
error was detected by the router, the router will penalize that path by increasing the cost by the number of possible bridging error detected (caused by the use of that routing switch connection or PIP). This results in favoring routing tracks that have fewer possible bridging error. In the example of Fig. 20, the router will choose path 2 instead of path 1 to route net A. Using path 1 will introduce two short sensitive bits to the path, increasing the cost of that path (path 1 grey, path 2 black).

![Figure 20 Short Sensitive Aware Router operation](image)

### 4.3 Optimization

The modification we made on the original implementation of VPR imposed a large penalty on the run time to be done. That is true especially if we are using the automatic schedule where the initial temperature (that affect the run time) equals 20 times the standard deviation of the change in the cost of the initial placement after the initial moves. Since we changed the cost function (for example for the Estimate 1) for the placement to
be the product of the width and length of the overlapping area, the starting cost will be orders of magnitude more than the starting cost of the original cost function, causing the anneal to start at a much higher temperature, resulting in longer time required for the mapping to cool down. Adding to that the added complexity of computing the cost, determining the overlapping area between nets and updating the overlapping coordinates.

The original implementation of VPR stores the coordinates of the bounding box of the nets in an array, so starting from there, we have to extend the, and add the necessary data structure required for our optimization.

To describe the optimization techniques we implemented, we first discuss the complexity of the original algorithm for the Bounding Box cost function (which is essentially the same for the timing driven placement) [Marquardt, FPGA’00]. At each temperature, the inner loop of the placer is executed $O(n^{4/3})$ times (where $O(n^{4/3})$ swaps is performed), $n$ here is the number of block in the FPGA. The Bounding Box update operation is a $O(k_{\text{max}})$ operation in worst case scenario, while the average case is $O(1)$ implementing the incremental bounding box update (discussed in section 0).

The overall time complexity of the algorithm have a worst case of $O((k_{\text{max}}n)^{4/3})$, but on average it is $O(n^{4/3})$, where $n$ is the number of blocks (CLBs) in the FPGA area, and $k_{\text{max}}$ is the maximum number of fanout of the nets.

We would like to mention a small modification that we made on the original implementation was the modification of the $R_{\text{limit}}$, instead of allowing it to decrease as we proceed with the placement, we kept $R_{\text{limit}}$ fixed to it initial value (the size of the entire chip), so we don’t maintain the update schedule of $R_{\text{limit}}$ implemented originally in the VPR tool. The reason for this is the goal of the placement is different for the two cases (the original implementation, and our estimates). If we take for example the Bounding Box placement, updating $R_{\text{limit}}$ works well for the original cost function of the VPR tool. $R_{\text{limit}}$ starts as being the size of the entire chip for the first part of the anneal, shrinking gradually during the middle stages of the anneal, and finally being 1 logic block at low temperatures. This ensures that only blocks that are close together are considered for swapping. These local swaps tend to result in relatively small changes in the placement cost, increasing their probability of acceptance.
For our implementation, we kept $R_{\text{limit}}$ at its initial value, so swaps can interchange the location of blocks $x$ or $y$ units apart in the FPGA area (where $x$ and $y$ are the dimensions of the FPGA array). We found by keeping the $R_{\text{limit}}$ at its initial value, we are able to obtain better results than decreasing the values of $R_{\text{limit}}$ as we proceed with the anneal. The reason for that is the goal of our placement algorithm is different that the original placement algorithm, where in the original implementation, and considers the Bounding Box cost function for example; we are concerned in reducing the bounding box of individual nets. While our placement algorithm, if we consider Short Sensitive Estimate 1 for example, we would like to reduce the overlapping area of a given net relative to the other nets, so swapping two blocks close together might not give the same results as swapping two blocks further away.

We experimented with both approaches (updating $R_{\text{limit}}$, and keeping $R_{\text{limit}}$ at its initial value), and we did find that keeping $R_{\text{limit}}$ at its initial value does results in a final placement with fewer short sensitive bits.

### 4.3.1 Optimizing for the Short Sensitive Placement

The runtime of the simulated annealing placement, implementing our cost function is magnitudes higher than that of the original algorithm, many optimization techniques were required to bring down the time required to finish the placement, however, even though these techniques were implemented, we couldn’t finish the placement for some of the large circuits (those circuits with number of nets larger than 4000 for the case of Short Sensitive Estimate 1 placement, and 2000 for the case of Short Sensitive Estimate 2).

#### 4.3.1.1 Optimizing for Short Sensitive Estimate 1

The optimization for the cost for Short Sensitive Estimate 1 is two folds, First, we divide the computation of the total cost into two part, and we allocate to data structure for that, one of them is to store the sum of the overlapping area for one net with all other nets (that is a one dimensional array indexed by the net number), and the other array is a two dimensional array of size (number of nets X number of nets), and it keep track of the
overlapping area of each net with each other net (both structures are shown in Fig. 21),
this second array is an extension of the implementation of the incremental bounding box
update, which is an optimization for the original bounding box implementation. The goal
of these two arrays, is that instead of computation the whole cost (that is finding the
overlapping area and adding them together) from scratch, we compute the difference of
the overlap area of the previous placement, and the new overlap area for the net affected
by the move of the CLBs that got swapped, so first we add the overlapping area of the
nets affected by the swap together, then finding how the overlapping of these nets got
affected by the move, find the overlapping nets, compute the new overlapping nets,
compares the two values, and then make a decision if the current swap should be kept or
not, if yes, that is it, go back and perform another swap, if no then we have to restore the
previous values, which we keep track of.

The second, is that, to minimize the search for the nets to check if the current net overlap
with or not, is that to sort the nets based on their maximum coordinate (we keep a list of
the nets based on their max x coordinate and the other for the max y coordinate, as shown
in Fig. 22), note that if a net 1 with a max x coordinate of say maxx1, and net 2 with min x
coordinate of \( \text{minx}_2 \), and if that \( \text{minx}_2 > \text{maxx}_1 \) (better illustrated in Fig. 23), net 2 cannot overlap with net 1. Same applies for the y-coordinates. Using this technique, we save on the effort to find out the nets that do overlap, reducing the complexity from \( O(m^2) \) to \( O(m \log m) \), where \( m \) is the number of the nets of the circuit.

![Figure 22 Array of linked list for nets, sorted based on their xmax](image)

![Figure 23 determining the overlapping area, net1 xmax is less than net2 and net3 xmin](image)
Put in a Pseudo-code:

```c
find affected nets()
for each affected net i
{
  if a change in x_max or y_max of the affected net
    update sorted_xmax;
    update sorted_ymax;
  set the overlapping cost for the affected net i to zero;
  set the overlapping cost of net i with each other net to zero;
  for each net j; sorted_xmax[j] >= net[i].xmin , sorted_ymax[j] >= net[i].ymin
    if net i and net j overlap
      overlapping[i][j]=w.l ; /*where w and l are the overlapping
      area width and length*/
      short_sens_bit[i]= short_sens_bit[i]+w.l ;
    }
}
```

Figure 24 Pseudo-code for the optimization techniques

The techniques implemented above resulted in an overall time complexity of $O((k_{max} n m \log m)^{4/3})$ in worst case, and $O((n m \log m)^{4/3})$ on average.

### 4.3.1.2 Optimizing for Short Sensitive Estimate2

The Optimization of the Short Estimate 2 is similar for the one done for the Short Estimate 1, for the part of finding the overlap area, however we still have to compute the switch usage probabilities, if we compute the switch usage probabilities at the time of determining the overlapping area, we will face a large delay penalties (due to the overhead of computing the overlap area).

There are two optimization techniques that we tried, one is computation optimized, and the other is memory optimized, in the end we picked the memory optimized version in the final implementation.
To implement the computation optimized version, we create a four dimensional array of pointer, the array is indexed by the y maximum, y minimum, x maximum, x minimum of the nets (so its dimensions are: ny, ny, nx, nx, where ny and nx are the x and y length and width of the FPGA array area). The four dimensional array elements points to two dimensional arrays. The two dimensional arrays contains the switch usage probability of a net (Fig. 25).

![Figure 25 Four dimensional arrays, points to the switch usage probabilities, computation optimized](image)

We store the switch usage probabilities on an array of the size of the FPGA (number of CLB in each dimension), we compute the usage probability of the switch inside the bounding box of the net, and then allocate the rest of the array to zero, so to compute the cost, all we need to do is to compute the cost across all the FPGA area, wherever there is a zero that means that the two net does not overlap, where the two nets two overlap, the cost will be non zero.

Remember, the cost for Estimate 2 is:

\[
\text{Estimate } 2 = \sum_{\forall (i, j) \text{in } \text{overlap area}} p_{i, j}^{N_1} \cdot p_{i, j}^{N_2}
\]  

(4-12)
The multiplication of the two probabilities will take care of finding the overlapping area.

For the memory optimized version, instead of indexing the first array (pointers of the switch usage probabilities array) by the absolute values of the y minimum, y maximum, x minimum and x maximum, we index the array by the length and width of the bounding box, and we store the usage probabilities of the area bounded by the bounding box of the net (Fig. 26).

Figure 26 Two dimensional arrays, points to the switch usage probabilities, memory optimized

However, there are extra computation involved, regarding the boundaries of the overlapping area, and the boundaries of the loops to compute the cost.

For the first implementation (the computation optimized implementation), the memory required for a small circuit placement (tseng.net) for the manual schedule was 1.5GB, while the memory required for the same circuit using the memory optimized implementation is around 16,500 KB.

For the computation optimized version it took much longer to execute, possibly due to the longer access time to the memory, so for the computation optimized implementation, it was more saving in terms of programming effort, rather the computation itself.
The time complexity for the Short Sensitive Estimate 2 placement is the $O([k_{\text{max}} n m \log m o \ p]^{4/3})$ in worst case, where o and p are the length and width of the overlapping area, where we have a loop over that area to find the product of the switch usage probability.

### 4.3.2 Optimizing for the Short Sensitive Routing

As we discussed back in section 3.3.2, without implementing an optimization, finding the number of short sensitive bit for each wire track of a channel used for routing a net, with the other wires in the surrounding channels that is used by other nets, so to find that, we have to check all the routing done so far for the other nets, which is a time consuming process.

![Net 17 (tin_pready_0_0_)]

**Figure 27 Example of the routing file output**

Fig.27 show an example of the final routing output, the only resources that are susceptible to short errors are CHANX and CHANY.
Fig. 28 shows the addressing of the IO pads, CLBs and routing channel used by VPR, the CHANY routing channel takes the coordinates of the CLB (or IO block) left to it, and the CHANX routing resource take the coordinates of the CLB (or IO block) below it, note that CLBs start at address (1,1), and IO pads starts at address (1,0) and (0,1). As we have stated earlier, each wire of each channel is represented internally as a node in the routing resource graph, the way we can distinguish between the wires being used by a net, is there is a Track field associated with the CHANX and CHANY resource, giving the track number the net is connected to.

Based on these information (we have two different routing resources, and each of them can have the same coordinates), we created two three dimensional array, indexed by the x coordinate, y coordinate, and the wire track number, one array for CHANX routing resource, and the other for CHANY routing resource.

We store the net number (ID) connected to each wire track for each CHANX and CHANY routing resource, that will make it easier, and faster to find out the number of
Short sensitive bit, but checking if any of the wire track in the neighboring channels that with a possibility of forming a short error, is connected or used by another net, if yes the increment the number of short sensitive errors for that routing resource, if not, check the next routing resource.
Chapter 5

Experimental Results

To compare the final sensitive bits (after placement and routing), we wrote a small programs that parse the final routing file, and find out the number of sensitive bits (open sensitive bits, and short sensitive bits). The total sensitive bits will be the sum of the two classes of sensitive bits.

The routing resources used described by the routing file are one of the following:

- SOURCE: the source of a certain output pin class.
- SINK: the sink of a certain input pin class.
- OPIN: output pin.
- IPIN: input pin.
- CHANX: horizontal channel.
- CHANY: Vertical channel.

Each net routing begins with a SOURCE and ends on a SINK. After determining the routing resource type (using the key words listed above) the location of the routing resource is given in a bracket in the for of (x,y) coordinates. Finally the pad number (if the SOURCE, SINK, IPIN or OPIN was on an I/O pad), pin number (if the IPIN or OPIN were on a CLB), class number (if the SOURCE or SINK were on a CLB) or track number (for CHANX or CHANY) is listed – whichever appropriate.

If our architecture allow for two I/O pads in one row or column, then the pad number identify which pad at that location we are attached.

For the CHANX routing resource (the horizontal channel) track 0 is the bottommost track, while in the case of CHANY (vertical channel) track 0 is the left most track. Note that if only global routing was performed the track number for each of the CHANX and CHANY resources listed in the routing will be 0, as global routing does not assign tracks to the various nets.
Please note that there is no description for the programmable interconnect switches (PIP) in the final routing file, and they are represented internally by the VPR router as an edge from one routing resource to another, so for our tool we used our understanding that in the listing of the routing resource a transition from one routing resource to another (CHANX followed by a CHANX, CHANY followed by CHANY, CHANX followed by CHANY, or CHANY followed by CHANX) as a switch PIP, and obviously a CHAN (CHANY or CHANX) to IPIN, or OPIN to CHAN (CHANX or CHANY) will represent a connection box PIP.

For the case of Open sensitive bits, to count for the actual open sensitive bits in the final routing, we parse the routing file, and find out the channel to channel transitions (CHANX to CHANX, CHANY to CHANY, CHANX to CHANY, or CHANY to CHANX), and count them as an open sensitive bit for our final sensitive bit.

For the short sensitive bits, the case is a bit different; we need to check for every channel (CHANX or CHANY) of the surrounding channels, for a possibility of forming a short error with any of the tracks in that channel that is connected to a net other than the net using the current channel track.

VPR supports 3 routing switch box blocks: Subset, Universal and Wilton (Fig. 26), all of which have a connection flexibility of 3, the difference between them is to which outgoing track the incoming track get connected to.

The subset switch box block is the planar or domain-based switch box used in the Xilinx 4000 FPGAs – a wire segment in track 0 can only connect to other wire segments in track 0 and so on. The Universal switch block (also called Symmetric Switch Module) presented in [Chang, DAES’96] and described in Fig. 29 and Fig. 30, while the Wilton [Wilton, PhD’97] switch box is similar to the Universal switch box, except the diagonal connections have been “rotated” by one track.
Algorithm: Symmetric_Switch_Module (W)

Input: W -- size of the switch module.
Output: M(T, S) -- the symmetric switch module of size W;

T: set of terminal;
S: set of switches.

1. T ← ∪_{i∈{L,T,R,B}} T_i;
2. S ← ∅
3. For i ← 1 to W do {
4. S← ∪ {t_i, 3W-i+1}; /* Type-1 connections */
5. S← ∪ {t_{W+i}, 4W-i+1}; /* Type-2 connections */
6. S← ∪ {t_i, 2W-i+1}; /* Type-3 connections */
7. S← ∪ {t_{W+i}, 3W-i+1}; /* Type-4 connections */
8. S← ∪ {t_{2W+i}, 4W-i+1}; /* Type-5 connections */
9. S← ∪ {t_i, 4W-i+1}; /* Type-6 connections*/
10. } /* for */
11. Output M(T, S);

Figure 30 Symmetric Switch Module Algorithm

We based our comparison of actual sensitive bits of the final placement and routing we used the subset switch box, for it is similar to the one used in Xilinx XC4000 Family FPGA’s [Chang,DAES’96], and it is the default switch box type in the included architecture file accompanying the VPR tool. None of the other work discussed in the previous work section mentioned the switch box type used, nor it is the intension of this work, at this point, to discuss the effect of the switch box type.
The type of the routing switches considered for finding the actual short sensitive bits are tri-state buffers, the difference here, if the routing switch used is a pass transistor, then it is a bidirectional connection, meaning, a short sensitive bit from net a to net b, is the same short sensitive bit from net b to net a, while for the tri-state buffer case, which is a one directional connection, a short sensitive bit from net a to b, is different than the short sensitive bit from net b to a, however, we can consider the final sensitive bits as a measure for the reliability of the circuit mapped into an FPGA, and comparing between different implementation of the circuit mapped into the FPGA, if we used the same criteria for comparing, it doesn’t matter, however if the exact number of sensitive bit is what matter for us, and we are using pass transistor for our routing, then we can divide the final short sensitive bits by 2, any other variation on this, that is if both tri-state buffer and pass transistor are used as routing switches (that is there are two type of switch boxes, one uses tri-state buffers, and the other uses pass transistors), then the tool we wrote will need to be changed, and it will be case dependent (depending on the distribution of the tri-state buffers and pass transistors in the FPGA).

To compare the reliability of the final placed and routed circuit, for this work, we will use the actual sensitive bits as a measure for reliability, we will compare the open sensitive bits for the Bounding Box placement (as our optimization for Open sensitive bits) against the Path timing driven and net timing driven placements, all of these placement using the timing driven router, we will compare the difference in open sensitive bits, critical path delay, and number of tracks used to route the placement, Then we will used the same placement (the bounding box placement) and route it using our open sensitive router. And compare the final output based on the previous comparison.

After the comparison of the open sensitive bit, we will move to our short sensitive estimates (Estimate 1 and Estimate2), and compare the final short sensitive bits of the routed circuit (routed using the timing driven router) with that of the path timing driven, one reason is that it has the components of the bounding box placement and the net timing driven placement, it provides a balanced final output of the total wiring length required to route the circuit and the critical path delay. We will compare for the final short sensitive bits, critical path timing delay and the number of tracks (in term of
channel width). We used the number of tracks used for routing as our measure of the area required for the routing resources.

Following that, we will use our Short Sensitive aware router to route the placement of the previous step to find out the improvement on the Short sensitive bits, and the other parameters we are comparing against.

Finally, we will present the same results on the basis of total sensitive bits (open sensitive bit + short sensitive bits), while the comparison of the other parameters (critical path delay and area) has already been establish.

The router we are comparing against is the timing driven router, because it optimizes for the critical path delay and it is the default router in the VPR tool.

We are basing our comparison on the circuits accompanying the VPR tool which are the 20 largest MCNS circuits [Betz, Kluwer, 99]. However we were able run our Short Sensitive Estimate 1 placement on only 16 of these circuits, and 13 of them for the Short Sensitive Estimate 2 placement. The total sensitive bit comparison will be on the 13 circuits we were able to obtain the Short Sensitive Estimate 2 placement.

The reason that we excluded these circuits is the huge time required run the placement using our estimate; the placement didn’t finish within a reasonable amount of time (the run time is a function of the net size).

5.1 Open Sensitive Comparison

5.1.1 Open Sensitive Comparison Using Timing Driven Router

The comparison for the open sensitive is between the bounding box placement and the path timing driven placement, and between the bounding box placement and the net timing driven placement.

In Tables 4, we will compare the Path timing driven placement (Path) against the Bounding Box placement (BB), and we will also compare the Net timing driven placement (Net) against the Bounding Box driven placement (BB) in Table 5.
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Open Sensitive bits</th>
<th>Critical Path Delay</th>
<th>Number of Tracks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Path$^1$</td>
<td>BB$^2$</td>
<td>$\Delta$</td>
</tr>
<tr>
<td>alu4</td>
<td>21593</td>
<td>19414</td>
<td>11.22%</td>
</tr>
<tr>
<td>apex2</td>
<td>32605</td>
<td>28691</td>
<td>13.64%</td>
</tr>
<tr>
<td>apex4</td>
<td>21444</td>
<td>20271</td>
<td>5.79%</td>
</tr>
<tr>
<td>bigkey</td>
<td>18841</td>
<td>16933</td>
<td>11.27%</td>
</tr>
<tr>
<td>des</td>
<td>26244</td>
<td>25917</td>
<td>1.26%</td>
</tr>
<tr>
<td>diffeq</td>
<td>13692</td>
<td>12802</td>
<td>6.95%</td>
</tr>
<tr>
<td>dsip</td>
<td>21026</td>
<td>12788</td>
<td>64.42%</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>2957</td>
<td>2857</td>
<td>3.50%</td>
</tr>
<tr>
<td>elliptic</td>
<td>53380</td>
<td>42881</td>
<td>24.48%</td>
</tr>
<tr>
<td>ex5p</td>
<td>19067</td>
<td>17403</td>
<td>9.56%</td>
</tr>
<tr>
<td>frisc</td>
<td>58328</td>
<td>50022</td>
<td>16.60%</td>
</tr>
<tr>
<td>misex3</td>
<td>22091</td>
<td>19390</td>
<td>13.93%</td>
</tr>
<tr>
<td>s298</td>
<td>23741</td>
<td>20306</td>
<td>16.92%</td>
</tr>
<tr>
<td>seq</td>
<td>26746</td>
<td>25547</td>
<td>4.69%</td>
</tr>
<tr>
<td>spla</td>
<td>63961</td>
<td>61933</td>
<td>3.27%</td>
</tr>
<tr>
<td>tseng</td>
<td>8724</td>
<td>7909</td>
<td>10.30%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>13.61%</td>
</tr>
</tbody>
</table>

Table 4 Comparison of the Open Sensitive bits, Critical Path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path Timing Driven placement and the Bounding Box placement

---

1. Open Sensitive bits for the Path Timing Driven placement routed using the Timing Driven router
2. Open Sensitive bits for the Bounding Box placement routed using the Timing Driven router
3. Critical Path delay for the Path Timing Driven placement routed using the Timing Driven router
4. Critical Path delay for the Bounding Box placement routed using the Timing Driven router
5. Number of routing tracks used to route the Path Timing Driven placement routed using the Timing Driven router
6. Number of routing tracks used to route the Bounding Box placement routed using the Timing Driven router
Table 5 Comparison of the Open Sensitive bits, Critical Path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Net Timing Driven placement and the Bounding Box placement

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Open Sensitive Bits</th>
<th>Critical Path Timing</th>
<th>Number of Tracks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Net(^7) BB Δ T(<em>{\text{Crit Net}})^8 T(</em>{\text{Crit BB}}) Δ Tracks Net(^9) Tracks BB Δ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alu4</td>
<td>19269 19414 -0.75% 1.12E-07 1.22E-07 7.96% 12 10 20.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>apex2</td>
<td>29498 28691 2.81% 1.14E-07 1.35E-07 15.68% 13 12 8.33%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>apex4</td>
<td>20108 20271 -0.80% 9.77E-08 1.36E-07 28.14% 14 13 7.69%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bigkey</td>
<td>20665 16933 22.04% 5.31E-08 8.93E-08 40.54% 7 7 0.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>des</td>
<td>23813 25917 -8.12% 1.25E-07 1.14E-07 -9.42% 9 8 12.50%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>diffeq</td>
<td>13239 12802 3.41% 8.24E-08 9.58E-08 13.99% 8 8 0.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsip</td>
<td>19001 12788 48.58% 5.38E-08 8.03E-08 33.05% 7 6 16.67%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e64-4lut</td>
<td>3012 2857 5.43% 3.27E-08 4.18E-08 21.73% 9 8 12.50%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>elliptic</td>
<td>46719 42881 8.95% 1.28E-07 2.15E-07 40.31% 13 11 18.18%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex5p</td>
<td>18367 17403 5.54% 1.12E-07 1.26E-07 10.94% 15 13 15.38%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>frisc</td>
<td>58327 50022 16.60% 1.32E-07 1.78E-07 25.82% 15 12 25.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>misex3</td>
<td>20452 19390 5.48% 9.66E-08 1.04E-07 7.51% 12 11 9.09%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s298</td>
<td>21036 20306 3.59% 1.57E-07 2.06E-07 23.63% 9 7 28.57%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>seq</td>
<td>26775 25547 4.81% 9.40E-08 1.11E-07 15.65% 15 11 36.36%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spla</td>
<td>65561 61933 5.86% 1.63E-07 1.82E-07 10.54% 17 13 30.77%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tseng</td>
<td>8748 7909 10.61% 5.47E-08 7.31E-08 25.15% 8 6 33.33%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>8.38%</td>
<td>19.45%</td>
<td>17.15%</td>
</tr>
</tbody>
</table>

\(^7\) Open Sensitive bits for Net Timing Driven Placement  
\(^8\) Critical Path Timing for the Net Timing Driven placement routed using the timing driven router  
\(^9\) Number of routing Tracks (per channel) for the Net Timing driven placement and the Timing Driven router
We noticed an on average improvement 13.61% (for the case of path timing driven) and 8.39% (the case of net timing driven placement) in the open sensitive bits number when using the bounding box placement, however, this improvement in the open sensitive bits comes at a cost of increased delay of around 20% on average compared to the net timing driven placement (and 17.10% compared to the path timing driven placement), while improving in the number of tracks count per channel of on average around 17% over the net timing driven placement and 15.3% over the path timing driven placement.

### 5.1.2 Open Sensitive Placement (Open Sensitive Router) Comparison

For the comparison of the open sensitive router, we will compare between the bounding box placement using the timing driven router and the same placement using the open sensitive router.

We will be comparing the Open Sensitive router against the Timing Driven router as follows: Table 6 will compare the Bounding Box placement routed using the timing driven router (*BB TimingR*) against the Bounding Box placement routed using the Open Sensitive aware router (*BB OpenR*). Table 7 is the comparison of the Path timing driven placement routed using the timing driven router (*Path*) against the Bounding Box placement routed using the Open Sensitive aware router (*BB OpenR*). While in Table 8 we compare the Net timing driven placement routed using the timing driven router (*Net*) against the Bounding Box placement routed using the Open Sensitive aware router (*BB OpenR*).
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Open Sensitive Bits</th>
<th>Critical Path Timing</th>
<th>Number of Tracks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BB TimingR(^{10})</td>
<td>BB OpenR(^{11}) Δ</td>
<td>T_Crit BB TimingR(^{12})</td>
</tr>
<tr>
<td>alu4</td>
<td>19607</td>
<td>20635               -4.98%</td>
<td>1.04E-07</td>
</tr>
<tr>
<td>apex2</td>
<td>27498</td>
<td>26711               2.95%</td>
<td>9.59E-08</td>
</tr>
<tr>
<td>apex4</td>
<td>19897</td>
<td>18475               7.70%</td>
<td>9.64E-08</td>
</tr>
<tr>
<td>bigkey</td>
<td>17357</td>
<td>15711               10.48%</td>
<td>7.55E-08</td>
</tr>
<tr>
<td>des</td>
<td>22362</td>
<td>20792               7.55%</td>
<td>1.01E-07</td>
</tr>
<tr>
<td>diffeq</td>
<td>12439</td>
<td>12275               1.34%</td>
<td>8.67E-08</td>
</tr>
<tr>
<td>dsip</td>
<td>14013</td>
<td>11644               20.35%</td>
<td>8.05E-08</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>2824</td>
<td>2674                5.61%</td>
<td>3.00E-08</td>
</tr>
<tr>
<td>elliptic</td>
<td>40773</td>
<td>39890               2.21%</td>
<td>1.18E-07</td>
</tr>
<tr>
<td>ex5p</td>
<td>17472</td>
<td>17267               1.19%</td>
<td>8.31E-08</td>
</tr>
<tr>
<td>frisc</td>
<td>49613</td>
<td>49973               -0.72%</td>
<td>1.31E-07</td>
</tr>
<tr>
<td>misex3</td>
<td>19390</td>
<td>18945               2.35%</td>
<td>9.60E-08</td>
</tr>
<tr>
<td>s298</td>
<td>19069</td>
<td>16933               12.61%</td>
<td>1.38E-07</td>
</tr>
<tr>
<td>seq</td>
<td>26360</td>
<td>24776               6.39%</td>
<td>1.10E-07</td>
</tr>
<tr>
<td>spla</td>
<td>62239</td>
<td>60938               2.13%</td>
<td>2.19E-07</td>
</tr>
<tr>
<td>tseng</td>
<td>8269</td>
<td>7771                6.41%</td>
<td>5.70E-08</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6 Comparison of the Open Sensitive bits, Critical Path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Bounding box placement routed using the Timing Driven router, and using the Open Sensitive aware Router

\(^{10}\) Open Sensitive Bits for Bounding Box placement routed using the Timing Driven router

\(^{11}\) Open Sensitive Bits for Bounding Box placement routed using the Open Sensitive router

\(^{12}\) Critical Path Delay for Bounding Box placement routed using the Open Sensitive router

\(^{13}\) Critical Path Delay for Bounding Box placement routed using the Timing Driven router

\(^{14}\) Number of routing Tracks (per channel) for the Bounding box placement and the timing driven router

\(^{15}\) Number of routing Tracks (per channel) for the Bounding box placement and the Open Sensitive aware router
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Path <strong>16</strong></th>
<th>BB OpenR</th>
<th>Δ</th>
<th>T. Crit Path <strong>17</strong></th>
<th>BB OpenR</th>
<th>Δ</th>
<th>Tracks Path <strong>18</strong></th>
<th>Tracks BB OpenR</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>21593</td>
<td>20635</td>
<td>4.64%</td>
<td>1.07E-07</td>
<td>1.68E-07</td>
<td>36.22%</td>
<td>12</td>
<td>11</td>
<td>9.09%</td>
</tr>
<tr>
<td>apex2</td>
<td>32605</td>
<td>26711</td>
<td>22.07%</td>
<td>9.29E-08</td>
<td>1.72E-07</td>
<td>46.03%</td>
<td>15</td>
<td>12</td>
<td>25.00%</td>
</tr>
<tr>
<td>apex4</td>
<td>21444</td>
<td>18475</td>
<td>16.07%</td>
<td>1.09E-07</td>
<td>1.87E-07</td>
<td>41.46%</td>
<td>14</td>
<td>13</td>
<td>7.69%</td>
</tr>
<tr>
<td>bigkey</td>
<td>18841</td>
<td>15711</td>
<td>19.92%</td>
<td>7.55E-08</td>
<td>2.48E-07</td>
<td>69.57%</td>
<td>7</td>
<td>8</td>
<td>-12.50%</td>
</tr>
<tr>
<td>des</td>
<td>26244</td>
<td>20792</td>
<td>26.22%</td>
<td>1.01E-07</td>
<td>1.68E-07</td>
<td>39.96%</td>
<td>8</td>
<td>9</td>
<td>-11.11%</td>
</tr>
<tr>
<td>diffeq</td>
<td>13692</td>
<td>12275</td>
<td>11.54%</td>
<td>8.67E-08</td>
<td>1.11E-07</td>
<td>21.60%</td>
<td>8</td>
<td>7</td>
<td>14.29%</td>
</tr>
<tr>
<td>dsip</td>
<td>21026</td>
<td>11644</td>
<td>80.57%</td>
<td>8.68E-08</td>
<td>2.56E-07</td>
<td>66.12%</td>
<td>8</td>
<td>7</td>
<td>14.29%</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>2957</td>
<td>2674</td>
<td>10.58%</td>
<td>3.00E-08</td>
<td>5.43E-08</td>
<td>44.76%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td>elliptic</td>
<td>53380</td>
<td>39890</td>
<td>33.82%</td>
<td>1.36E-07</td>
<td>2.87E-07</td>
<td>52.67%</td>
<td>14</td>
<td>11</td>
<td>27.27%</td>
</tr>
<tr>
<td>ex5p</td>
<td>19067</td>
<td>17267</td>
<td>10.42%</td>
<td>8.31E-08</td>
<td>1.56E-07</td>
<td>46.68%</td>
<td>15</td>
<td>14</td>
<td>7.14%</td>
</tr>
<tr>
<td>frisc</td>
<td>58328</td>
<td>49973</td>
<td>16.72%</td>
<td>1.31E-07</td>
<td>2.93E-07</td>
<td>55.18%</td>
<td>15</td>
<td>14</td>
<td>7.14%</td>
</tr>
<tr>
<td>misex3</td>
<td>22091</td>
<td>18945</td>
<td>16.61%</td>
<td>8.87E-08</td>
<td>1.61E-07</td>
<td>44.90%</td>
<td>13</td>
<td>11</td>
<td>18.18%</td>
</tr>
<tr>
<td>s298</td>
<td>23741</td>
<td>16933</td>
<td>40.21%</td>
<td>1.26E-07</td>
<td>3.52E-07</td>
<td>64.15%</td>
<td>10</td>
<td>8</td>
<td>25.00%</td>
</tr>
<tr>
<td>seq</td>
<td>26746</td>
<td>24776</td>
<td>7.95%</td>
<td>1.10E-07</td>
<td>2.19E-07</td>
<td>49.92%</td>
<td>12</td>
<td>13</td>
<td>-7.69%</td>
</tr>
<tr>
<td>spla</td>
<td>63961</td>
<td>60938</td>
<td>4.96%</td>
<td>2.19E-07</td>
<td>2.42E-07</td>
<td>9.75%</td>
<td>15</td>
<td>15</td>
<td>0.00%</td>
</tr>
<tr>
<td>tseng</td>
<td>8724</td>
<td>7771</td>
<td>12.26%</td>
<td>5.70E-08</td>
<td>1.14E-07</td>
<td>50.05%</td>
<td>7</td>
<td>8</td>
<td>-12.50%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>20.91%</td>
<td></td>
<td></td>
<td>46.19%</td>
<td></td>
<td></td>
<td>6.96%</td>
</tr>
</tbody>
</table>

Table 7 Comparison of the Open Sensitive bits, Critical Path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path Timing Driven placement routed using the Timing Driven router, and the Bounding box placement using the Open Sensitive Aware router

---

**Superscript:**

16 Open Sensitive Bits for the Path Timing Driven placement routed using the Timing Driven router

17 Critical Path Timing for the Path Timing Driven placement routed using the Timing Driven router

18 Number of track (per channel) used for routing the Path Timing Driven placement routed using the timing driven router
Table 8 Comparison of the Open Sensitive bits, Critical Path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Net Timing Driven placement routed using the Timing Driven router, and the Bounding box placement using the Open Sensitive Aware router.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Net(^{19})</th>
<th>BB OpenR</th>
<th>NC</th>
<th>T(_{\text{Crit Net}}) (^{20})</th>
<th>T(_{\text{Crit BB OpenR}})</th>
<th>NC</th>
<th>Tracks Net(^{21})</th>
<th>Tracs BB OpenR</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>19439</td>
<td>20635</td>
<td>-5.80%</td>
<td>1.09E-07</td>
<td>1.68E-07</td>
<td>35.09%</td>
<td>12</td>
<td>11</td>
<td>9.09%</td>
</tr>
<tr>
<td>apex2</td>
<td>29247</td>
<td>26711</td>
<td>9.49%</td>
<td>1.13E-07</td>
<td>1.72E-07</td>
<td>34.07%</td>
<td>13</td>
<td>12</td>
<td>8.33%</td>
</tr>
<tr>
<td>apex4</td>
<td>19678</td>
<td>18475</td>
<td>6.51%</td>
<td>9.71E-08</td>
<td>1.87E-07</td>
<td>48.09%</td>
<td>14</td>
<td>13</td>
<td>7.69%</td>
</tr>
<tr>
<td>bigkey</td>
<td>20090</td>
<td>15711</td>
<td>27.87%</td>
<td>5.53E-08</td>
<td>2.48E-07</td>
<td>77.72%</td>
<td>7</td>
<td>8</td>
<td>-12.50%</td>
</tr>
<tr>
<td>des</td>
<td>24102</td>
<td>20792</td>
<td>15.92%</td>
<td>1.25E-07</td>
<td>1.68E-07</td>
<td>25.53%</td>
<td>9</td>
<td>9</td>
<td>0.00%</td>
</tr>
<tr>
<td>diffeq</td>
<td>13135</td>
<td>12275</td>
<td>7.01%</td>
<td>7.40E-08</td>
<td>1.11E-07</td>
<td>33.12%</td>
<td>8</td>
<td>7</td>
<td>14.29%</td>
</tr>
<tr>
<td>dsip</td>
<td>18979</td>
<td>11644</td>
<td>62.99%</td>
<td>5.29E-08</td>
<td>2.56E-07</td>
<td>79.34%</td>
<td>7</td>
<td>7</td>
<td>0.00%</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>2901</td>
<td>2674</td>
<td>8.49%</td>
<td>3.27E-08</td>
<td>5.43E-08</td>
<td>39.76%</td>
<td>9</td>
<td>8</td>
<td>12.50%</td>
</tr>
<tr>
<td>elliptic</td>
<td>46391</td>
<td>39890</td>
<td>16.30%</td>
<td>1.62E-07</td>
<td>2.87E-07</td>
<td>43.57%</td>
<td>13</td>
<td>11</td>
<td>18.18%</td>
</tr>
<tr>
<td>ex5p</td>
<td>18604</td>
<td>17267</td>
<td>7.74%</td>
<td>8.72E-08</td>
<td>1.56E-07</td>
<td>44.04%</td>
<td>15</td>
<td>14</td>
<td>7.14%</td>
</tr>
<tr>
<td>frisc</td>
<td>57207</td>
<td>49973</td>
<td>14.48%</td>
<td>1.32E-07</td>
<td>2.93E-07</td>
<td>54.84%</td>
<td>15</td>
<td>14</td>
<td>7.14%</td>
</tr>
<tr>
<td>misex3</td>
<td>20182</td>
<td>18945</td>
<td>6.53%</td>
<td>9.54E-08</td>
<td>1.61E-07</td>
<td>40.73%</td>
<td>12</td>
<td>11</td>
<td>9.09%</td>
</tr>
<tr>
<td>s298</td>
<td>20308</td>
<td>16933</td>
<td>19.93%</td>
<td>1.56E-07</td>
<td>3.52E-07</td>
<td>55.64%</td>
<td>9</td>
<td>8</td>
<td>12.50%</td>
</tr>
<tr>
<td>seq</td>
<td>26813</td>
<td>24776</td>
<td>8.22%</td>
<td>9.40E-08</td>
<td>2.19E-07</td>
<td>57.15%</td>
<td>15</td>
<td>13</td>
<td>15.38%</td>
</tr>
<tr>
<td>spla</td>
<td>66006</td>
<td>60938</td>
<td>8.32%</td>
<td>1.63E-07</td>
<td>2.42E-07</td>
<td>32.68%</td>
<td>17</td>
<td>15</td>
<td>13.33%</td>
</tr>
<tr>
<td>tseng</td>
<td>8697</td>
<td>7771</td>
<td>11.92%</td>
<td>5.47E-08</td>
<td>1.14E-07</td>
<td>52.11%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td><strong>14.12%</strong></td>
<td></td>
<td></td>
<td><strong>47.09%</strong></td>
<td></td>
<td></td>
<td><strong>7.64%</strong></td>
</tr>
</tbody>
</table>

\(^{19}\) Open Sensitive Bits for the Net Timing Driven placement routed using the Timing Driven router

\(^{20}\) Critical path timing (in seconds) for the Net Timing Driven placement routed using the Timing Driven router

\(^{21}\) Number of track (per channel) used for routing the Net Timing Driven placement routed using the Timing Driven router
We show the difference of the routing using the timing driven router (Fig. 31) and the opens sensitive router (Fig. 32) for the same path timing driven placement of circuit e64-4lut and net i_43_.

Figure 31 Circuit e64-4lut net 59 (i_43_) routed using Open Sensitive Aware Router

Figure 32 Circuit e64-4lut net 59 (i_43_) routed using Timing Driven Router
5.2 Short Sensitive Comparison

5.2.1 Short Sensitive Placement Comparison Using Timing Driven Router

We will compare our implementation of Short Sensitive Estimate 1 placement and Short Sensitive Estimate 2 placement with the Path timing driven placement; both placements routed using the timing driven router.

Then we will be comparing the run time of the Path timing driven placement with each of our Short Sensitive estimates, when running them on Teracluster.

The Teracluster is a computing cluster, consisting of 33 servers, all Dual processor, Dual-core, 2.0 GHz Intel Xeon EM64T processors with both 8GB and 16 GB memory configurations, over 16TB of Internal/External Raid protected storage, running Platform Rocks 4.11-2.0 with Red Hat Enterprise 4 (Linux 2.6.9-34.ELsmp, 64-bit kernel).

The runtime we are reporting is the job reported by the Teracluster job handler, Lava/LSF (Load Scheduler Facility), which will include both, the time required for both the placement and routing.

However, obtaining a better Short sensitive bit using the original placement algorithm (Bounding Box, Path Timing driven or Net Timing driven) should come at no surprise., since we are using a router that is not optimized for our goal. The results obtained using these placements vary largely from one run to another (since the placement is a random process, optimized for a certain goal), so we cannot achieve the full potential of our placement algorithm without using our short sensitive router (which optimizes for the short sensitive bits).
5.2.1.1 Short Sensitive Estimate 1 Placement Comparison Using Timing Driven Router

We noticed a slight improvement on the final count of the short sensitive bits (7.12% improvement, Table 10), however that came at an increase in the delay of the circuit (path timing driven placement is 24% faster in the critical path timing delay), and on average 9% increase in the number of tracks (per channel) used for routing the circuits. However there is a huge slowdown in the runtime required to place the circuit, that is due to the added time complexity of the $O(n \log n)$ on the original algorithm, where $n$ is the number of nets of the circuit.
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Path</th>
<th>Short1</th>
<th>Δ</th>
<th>T_Crit Path</th>
<th>T_Crit Short1</th>
<th>Δ</th>
<th>Tracks Path</th>
<th>Tracks Short1</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>51397</td>
<td>48708</td>
<td>5.52%</td>
<td>8.33E-08</td>
<td>1.17E-07</td>
<td>28.73%</td>
<td>12</td>
<td>13</td>
<td>8.33%</td>
</tr>
<tr>
<td>apex2</td>
<td>78203</td>
<td>72573</td>
<td>7.76%</td>
<td>9.29E-08</td>
<td>1.22E-07</td>
<td>23.66%</td>
<td>15</td>
<td>15</td>
<td>0.00%</td>
</tr>
<tr>
<td>apex4</td>
<td>54692</td>
<td>54404</td>
<td>0.53%</td>
<td>1.09E-07</td>
<td>1.27E-07</td>
<td>13.66%</td>
<td>14</td>
<td>14</td>
<td>0.00%</td>
</tr>
<tr>
<td>bigkey</td>
<td>40964</td>
<td>33574</td>
<td>22.01%</td>
<td>7.55E-08</td>
<td>9.93E-08</td>
<td>23.94%</td>
<td>7</td>
<td>9</td>
<td>28.57%</td>
</tr>
<tr>
<td>des</td>
<td>52769</td>
<td>47941</td>
<td>10.07%</td>
<td>1.01E-07</td>
<td>1.53E-07</td>
<td>33.99%</td>
<td>8</td>
<td>9</td>
<td>12.50%</td>
</tr>
<tr>
<td>diffeq</td>
<td>34678</td>
<td>35185</td>
<td>-1.44%</td>
<td>8.67E-08</td>
<td>8.75E-08</td>
<td>0.91%</td>
<td>8</td>
<td>9</td>
<td>12.50%</td>
</tr>
<tr>
<td>dsip</td>
<td>38526</td>
<td>30538</td>
<td>26.16%</td>
<td>8.68E-08</td>
<td>8.33E-08</td>
<td>-4.09%</td>
<td>8</td>
<td>7</td>
<td>-12.50%</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>7614</td>
<td>8056</td>
<td>-5.49%</td>
<td>3.45E-08</td>
<td>4.35E-08</td>
<td>20.76%</td>
<td>8</td>
<td>9</td>
<td>12.50%</td>
</tr>
<tr>
<td>elliptic</td>
<td>126397</td>
<td>112831</td>
<td>12.02%</td>
<td>1.36E-07</td>
<td>1.73E-07</td>
<td>21.20%</td>
<td>14</td>
<td>13</td>
<td>-7.14%</td>
</tr>
<tr>
<td>ex5p</td>
<td>48486</td>
<td>45551</td>
<td>6.44%</td>
<td>8.31E-08</td>
<td>1.21E-07</td>
<td>31.26%</td>
<td>15</td>
<td>15</td>
<td>0.00%</td>
</tr>
<tr>
<td>frisc</td>
<td>132054</td>
<td>119900</td>
<td>10.14%</td>
<td>1.31E-07</td>
<td>2.33E-07</td>
<td>43.70%</td>
<td>15</td>
<td>18</td>
<td>20.00%</td>
</tr>
<tr>
<td>misex3</td>
<td>56049</td>
<td>52388</td>
<td>6.99%</td>
<td>8.87E-08</td>
<td>1.12E-07</td>
<td>20.73%</td>
<td>13</td>
<td>14</td>
<td>7.69%</td>
</tr>
<tr>
<td>s298</td>
<td>58069</td>
<td>50881</td>
<td>14.13%</td>
<td>1.26E-07</td>
<td>2.09E-07</td>
<td>39.42%</td>
<td>10</td>
<td>11</td>
<td>10.00%</td>
</tr>
<tr>
<td>seq</td>
<td>70150</td>
<td>68736</td>
<td>2.06%</td>
<td>1.10E-07</td>
<td>1.29E-07</td>
<td>15.07%</td>
<td>12</td>
<td>14</td>
<td>16.67%</td>
</tr>
<tr>
<td>spla</td>
<td>153093</td>
<td>172540</td>
<td>-11.27%</td>
<td>1.10E-07</td>
<td>1.99E-07</td>
<td>44.89%</td>
<td>15</td>
<td>19</td>
<td>26.67%</td>
</tr>
<tr>
<td>tseng</td>
<td>22652</td>
<td>20915</td>
<td>8.31%</td>
<td>6.09E-08</td>
<td>8.18E-08</td>
<td>25.46%</td>
<td>7</td>
<td>8</td>
<td>14.29%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>7.12%</td>
<td></td>
<td></td>
<td>23.96%</td>
<td></td>
<td></td>
<td>9.38%</td>
</tr>
</tbody>
</table>

Table 9 Comparison of the Short Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path timing driven placement and the Short Estimate 1 placement

22 Short Sensitive Estimate 1 placement routed using the Timing Driven router
23 Critical Path delay for Short Sensitive Estimate 1 placement routed using the Timing Driven router
24 Number of track (per channel) used for routing the Short Sensitive Estimate 1 placement routed using the Timing Driven router
5.2.2 Short Sensitive Estimate 2 Placement Comparison Using Timing Driven Router

We noticed an improvement on the final Short Sensitive bits over the last case (Short Sensitive Estimate 1) up to 8.82% on average (Table 11) with a little better slow down (of on average 20.88% slowdown versus about 24% on average slow down for Short Sensitive Estimate1), that is because we are considering some routing information (estimating the routing path) during the placement. However the extra computation of the switch usage probability causes a huge slow down in the runtime.
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Path</th>
<th>Short 2</th>
<th>Δ</th>
<th>T_Crit Path</th>
<th>T_Crit Short2</th>
<th>Δ</th>
<th>Tracks Path</th>
<th>Tracks Short 2</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>51397</td>
<td>45618</td>
<td>12.67%</td>
<td>8.33E-08</td>
<td>1.15E-07</td>
<td>27.38%</td>
<td>11</td>
<td>11</td>
<td>0.00%</td>
</tr>
<tr>
<td>apex2</td>
<td>78203</td>
<td>68788</td>
<td>13.69%</td>
<td>9.29E-08</td>
<td>1.33E-07</td>
<td>30.35%</td>
<td>15</td>
<td>13</td>
<td>15.38%</td>
</tr>
<tr>
<td>apex4</td>
<td>54692</td>
<td>50445</td>
<td>8.42%</td>
<td>1.09E-07</td>
<td>1.33E-07</td>
<td>17.95%</td>
<td>14</td>
<td>13</td>
<td>7.69%</td>
</tr>
<tr>
<td>bigkey</td>
<td>40964</td>
<td>39834</td>
<td>2.84%</td>
<td>7.55E-08</td>
<td>9.27E-08</td>
<td>18.52%</td>
<td>7</td>
<td>7</td>
<td>0.00%</td>
</tr>
<tr>
<td>des</td>
<td>52769</td>
<td>51223</td>
<td>3.02%</td>
<td>1.01E-07</td>
<td>1.51E-07</td>
<td>33.40%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td>diffeq</td>
<td>34678</td>
<td>33005</td>
<td>5.07%</td>
<td>8.67E-08</td>
<td>9.24E-08</td>
<td>6.12%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td>dsip</td>
<td>38526</td>
<td>31427</td>
<td>22.59%</td>
<td>8.68E-08</td>
<td>1.05E-07</td>
<td>17.51%</td>
<td>8</td>
<td>7</td>
<td>14.29%</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>7614</td>
<td>7279</td>
<td>4.60%</td>
<td>3.45E-08</td>
<td>3.83E-08</td>
<td>10.05%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td>ex5p</td>
<td>48486</td>
<td>45931</td>
<td>5.56%</td>
<td>8.31E-08</td>
<td>1.04E-07</td>
<td>19.79%</td>
<td>15</td>
<td>14</td>
<td>7.14%</td>
</tr>
<tr>
<td>misex3</td>
<td>56049</td>
<td>50225</td>
<td>11.60%</td>
<td>8.87E-08</td>
<td>1.04E-07</td>
<td>14.89%</td>
<td>13</td>
<td>12</td>
<td>8.33%</td>
</tr>
<tr>
<td>s298</td>
<td>58069</td>
<td>49111</td>
<td>18.24%</td>
<td>1.26E-07</td>
<td>1.95E-07</td>
<td>35.22%</td>
<td>10</td>
<td>9</td>
<td>11.11%</td>
</tr>
<tr>
<td>seq</td>
<td>70150</td>
<td>67126</td>
<td>4.50%</td>
<td>1.10E-07</td>
<td>1.30E-07</td>
<td>15.79%</td>
<td>12</td>
<td>12</td>
<td>0.00%</td>
</tr>
<tr>
<td>tseng</td>
<td>22652</td>
<td>22234</td>
<td>1.88%</td>
<td>6.09E-08</td>
<td>8.07E-08</td>
<td>24.50%</td>
<td>7</td>
<td>7</td>
<td>0.00%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>8.82%</td>
<td></td>
<td></td>
<td></td>
<td>20.88%</td>
<td>4.92%</td>
<td></td>
</tr>
</tbody>
</table>

Table 10 Comparison of the Short Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path timing driven placement and the Short Estimate 2 placement.
5.2.3 Short Sensitive Placement (Short Sensitive Aware Router) Comparison

In this section will use the placement obtained from the last section and route it using our Short Sensitive aware placement. We will be comparing our Short Sensitive Estimate 1 placement routed using the Short Sensitive Aware router (Short1_SR) with the Path Driven placement routed using the Timing Driven router (Path Timing) in table 12. While in table 13 we will be comparing the Short Sensitive Estimate 2 placement routed using the Short Sensitive Aware router (Short2_SR) with the Path Driven placement routed using the Timing Driven router (Path Timing)

The router is well optimized that the runtime is not an issue compared to the original breadth first search (which our router is based on) or the timing driven router.

We see that Short Sensitive Estimate 1 does better than the Short Sensitive Estimate 2 when using the Short Sensitive aware router (on average 136% improvement on the short sensitive bits over path timing driven VS 58.28% for Estimate 2), however the critical path delay is better in the case of Estimate 2 (on average about 48% slowdown VS 45% slow down from the Path timing driven placement). In term of the number of Routing channel tracks number, the Estimate 2 is doing better, again, because we consider routing during the placement (2.27% on average increase in the number tracks for Estimate 2 VS ~18% increase for Estimate 1).
### Table 11: Comparison of the Short Sensitive bits, Critical Path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path timing driven placement (routed using the Timing Driven Router) and the Short Estimate 1 placement.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Path TimingR 25</th>
<th>Short1_SR 26</th>
<th>Δ</th>
<th>T_Crit Path TimingR 27</th>
<th>T_Crit Short1 SR 26</th>
<th>Δ</th>
<th>Tracks Path TimingR</th>
<th>Tracks Short1_SR</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>51397</td>
<td>17302</td>
<td>197.06%</td>
<td>8.33E-08</td>
<td>1.87E-07</td>
<td>55.48%</td>
<td>11</td>
<td>16</td>
<td>31.25%</td>
</tr>
<tr>
<td>apex2</td>
<td>78203</td>
<td>41611</td>
<td>87.94%</td>
<td>9.29E-08</td>
<td>1.67E-07</td>
<td>44.24%</td>
<td>15</td>
<td>16</td>
<td>6.25%</td>
</tr>
<tr>
<td>apex4</td>
<td>54692</td>
<td>36751</td>
<td>48.82%</td>
<td>1.09E-07</td>
<td>1.68E-07</td>
<td>34.68%</td>
<td>14</td>
<td>15</td>
<td>6.67%</td>
</tr>
<tr>
<td>bigkey</td>
<td>40964</td>
<td>12102</td>
<td>238.49%</td>
<td>7.55E-08</td>
<td>1.83E-07</td>
<td>58.83%</td>
<td>7</td>
<td>10</td>
<td>30.00%</td>
</tr>
<tr>
<td>des</td>
<td>52769</td>
<td>14738</td>
<td>258.05%</td>
<td>1.01E-07</td>
<td>2.18E-07</td>
<td>53.64%</td>
<td>8</td>
<td>12</td>
<td>33.33%</td>
</tr>
<tr>
<td>diffeq</td>
<td>30552</td>
<td>17553</td>
<td>74.06%</td>
<td>8.67E-08</td>
<td>1.29E-07</td>
<td>33.00%</td>
<td>8</td>
<td>11</td>
<td>27.27%</td>
</tr>
<tr>
<td>dsip</td>
<td>38526</td>
<td>6627</td>
<td>481.35%</td>
<td>8.68E-08</td>
<td>1.41E-07</td>
<td>38.38%</td>
<td>8</td>
<td>10</td>
<td>20.00%</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>7614</td>
<td>6192</td>
<td>22.97%</td>
<td>3.45E-08</td>
<td>5.17E-08</td>
<td>33.37%</td>
<td>8</td>
<td>9</td>
<td>11.11%</td>
</tr>
<tr>
<td>elliptic</td>
<td>126397</td>
<td>51148</td>
<td>147.12%</td>
<td>1.36E-07</td>
<td>2.42E-07</td>
<td>43.72%</td>
<td>14</td>
<td>16</td>
<td>12.50%</td>
</tr>
<tr>
<td>ex5p</td>
<td>48486</td>
<td>31241</td>
<td>55.20%</td>
<td>8.31E-08</td>
<td>1.51E-07</td>
<td>44.91%</td>
<td>15</td>
<td>16</td>
<td>6.25%</td>
</tr>
<tr>
<td>frisc</td>
<td>132054</td>
<td>38122</td>
<td>246.40%</td>
<td>1.31E-07</td>
<td>3.61E-07</td>
<td>63.66%</td>
<td>15</td>
<td>22</td>
<td>31.82%</td>
</tr>
<tr>
<td>misex3</td>
<td>56049</td>
<td>43881</td>
<td>27.73%</td>
<td>8.87E-08</td>
<td>1.64E-07</td>
<td>45.81%</td>
<td>13</td>
<td>13</td>
<td>0.00%</td>
</tr>
<tr>
<td>s298</td>
<td>45328</td>
<td>24894</td>
<td>82.08%</td>
<td>1.26E-07</td>
<td>3.72E-07</td>
<td>66.01%</td>
<td>10</td>
<td>12</td>
<td>16.67%</td>
</tr>
<tr>
<td>seq</td>
<td>70150</td>
<td>25376</td>
<td>176.44%</td>
<td>1.10E-07</td>
<td>1.71E-07</td>
<td>35.72%</td>
<td>12</td>
<td>19</td>
<td>36.84%</td>
</tr>
<tr>
<td>spla</td>
<td>153093</td>
<td>141683</td>
<td>8.05%</td>
<td>1.10E-07</td>
<td>2.94E-07</td>
<td>62.57%</td>
<td>15</td>
<td>20</td>
<td>25.00%</td>
</tr>
<tr>
<td>tseng</td>
<td>22652</td>
<td>18059</td>
<td>25.43%</td>
<td>6.09E-08</td>
<td>1.42E-07</td>
<td>56.98%</td>
<td>7</td>
<td>8</td>
<td>12.50%</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>136.07%</td>
<td></td>
<td></td>
<td>48.19%</td>
<td></td>
<td></td>
<td>17.93%</td>
</tr>
</tbody>
</table>

25 Short Sensitive bits for Path timing Driven placement routed using the Timing Driven router
26 Short Sensitive bits for the Short Sensitive Estimate 1 placement routed using the Short Sensitive Aware router.
28 Critical Path Timing delay for the Short Sensitive Estimate 1 placement routed using the Short Sensitive Aware router.
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Path TimingR</th>
<th>Short2_SR</th>
<th>( \Delta )</th>
<th>( T_{\text{Crit} \text{ Path}} )</th>
<th>( T_{\text{Crit} \text{ Short2_SR}} )</th>
<th>( \Delta )</th>
<th>Tracks Path</th>
<th>( \text{Tracks Short2_SR} )</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>51397</td>
<td>33088</td>
<td>55.33%</td>
<td>8.33E-08</td>
<td>1.66E-07</td>
<td>49.68%</td>
<td>11</td>
<td>11</td>
<td>0.00%</td>
</tr>
<tr>
<td>apex2</td>
<td>78203</td>
<td>49381</td>
<td>58.37%</td>
<td>9.29E-08</td>
<td>1.69E-07</td>
<td>45.15%</td>
<td>15</td>
<td>13</td>
<td>15.38%</td>
</tr>
<tr>
<td>apex4</td>
<td>54692</td>
<td>31470</td>
<td>73.79%</td>
<td>1.09E-07</td>
<td>2.19E-07</td>
<td>50.03%</td>
<td>14</td>
<td>14</td>
<td>0.00%</td>
</tr>
<tr>
<td>bigkey</td>
<td>40964</td>
<td>30111</td>
<td>36.50%</td>
<td>7.55E-08</td>
<td>1.50E-07</td>
<td>49.51%</td>
<td>7</td>
<td>7</td>
<td>0.00%</td>
</tr>
<tr>
<td>des</td>
<td>52769</td>
<td>30256</td>
<td>74.41%</td>
<td>1.01E-07</td>
<td>2.06E-07</td>
<td>50.94%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td>diffeq</td>
<td>30552</td>
<td>27669</td>
<td>10.42%</td>
<td>8.67E-08</td>
<td>1.27E-07</td>
<td>31.56%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td>dsp</td>
<td>38526</td>
<td>13770</td>
<td>179.78%</td>
<td>8.68E-08</td>
<td>2.13E-07</td>
<td>59.20%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>7614</td>
<td>6207</td>
<td>22.67%</td>
<td>3.45E-08</td>
<td>4.89E-08</td>
<td>29.52%</td>
<td>8</td>
<td>8</td>
<td>0.00%</td>
</tr>
<tr>
<td>ex5p</td>
<td>48486</td>
<td>30974</td>
<td>56.54%</td>
<td>8.31E-08</td>
<td>1.31E-07</td>
<td>36.57%</td>
<td>15</td>
<td>14</td>
<td>7.14%</td>
</tr>
<tr>
<td>misex3</td>
<td>56049</td>
<td>36510</td>
<td>53.52%</td>
<td>8.87E-08</td>
<td>1.56E-07</td>
<td>49.69%</td>
<td>13</td>
<td>12</td>
<td>8.33%</td>
</tr>
<tr>
<td>s298</td>
<td>45328</td>
<td>33807</td>
<td>34.08%</td>
<td>1.26E-07</td>
<td>3.62E-07</td>
<td>65.13%</td>
<td>10</td>
<td>11</td>
<td>11.11%</td>
</tr>
<tr>
<td>seq</td>
<td>70150</td>
<td>50879</td>
<td>37.88%</td>
<td>1.10E-07</td>
<td>1.75E-07</td>
<td>37.37%</td>
<td>12</td>
<td>12</td>
<td>0.00%</td>
</tr>
<tr>
<td>tseng</td>
<td>22652</td>
<td>13778</td>
<td>64.41%</td>
<td>6.09E-08</td>
<td>1.04E-07</td>
<td>41.50%</td>
<td>7</td>
<td>8</td>
<td>-12.50%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>58.28%</strong></td>
<td><strong>45.32%</strong></td>
<td><strong>2.27%</strong></td>
<td><strong>58.28%</strong></td>
<td><strong>45.32%</strong></td>
<td><strong>2.27%</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 12 Comparison of the Short Sensitive bits, Critical path delay (in sec) and the number of routing Tracks used for routing (for each channel) between the Path timing driven placement (routed using the Timing Driven Router) and the Short Estimate 2 placement.

29 Short Sensitive bits for the Short Sensitive Estimate 2 placement routed using the Short Sensitive Aware Router.
30 Critical Path Timing delay for the Short Sensitive Estimate 2 placement routed using the Short Sensitive Aware router.
31 Number of tracks (per channel) for the Short Sensitive Estimate 2 placement routed using the Short Sensitive Aware router.
We show the difference between the timing driven router (Fig. 33) and the short sensitive aware router (Fig. 34) of circuit e64-4lut net 163 (n_n_366) placed using the path timing driven placement, because the net we are showing is small net, the effect is seen shifting the routing tracks to ones with less possibility of short sensitive bit being formed due to an SEU, we see that the routing of the timing driven router is having 8 short sensitive bit for the small net we are showing, while for the short sensitive aware routing we have only 2 short sensitive bits.

Figure 33 Circuit e64-4lut net 163 (n_n366) routed using Timing Driven Router
5.3 Total Sensitive Bits

So far we only compared open sensitive bits and short sensitive bits separately, to show that our approach is working, and doing its goal in optimizing for the parameter of choice, however, to achieve reliability in the FPGA, we have to reduce both the open sensitive bits and the short sensitive bits, that is reducing the Total sensitive bits. Total sensitive bits is the sum of open sensitive bits and the short sensitive bits of a given placement routed using a given router.
For the sake of comparison we will compare each placement (Bounding Box, Net Timing Driven, Path Timing Driven, Short Sensitive Estimate 1 and Short Sensitive Estimate 2) each routed using Timing Driven router first and Short Sensitive aware router next, we will present the results side by side. We will compare in terms of the total sensitive bits (Table 14) and in terms of the critical path delay (Table 15).

Note that we didn’t use the Open Sensitive Aware router aside from the optimizing for Open Sensitive bits, one reason is that for a given circuit, the number of Short Sensitive bits is more than the number of Open sensitive bits (e.g. the case of Path Timing driven placement routed using Timing Driven router, on average there are 64112 Short Sensitive bits, while there are 27152 Open Sensitive bits), and combining the two factors (Open Sensitive bits, and Short Sensitive bits) in the router cost doesn’t improve much on either of them. We will be comparing the result for the circuits we could obtain the Short Estimate 2 placement for.

Fig. 35 shows a plot for the results obtained, it is the data set plotted for the values of the Total Sensitive bits versus the critical path delay, the trend line divide the data set into two parts, left of the trend line, and right of the trend line, the data point on the left represent the ones of interest, the achieve better Total sensitive bits optimization than with less Critical Path delay than the data points on the right side of the trend line, so these points represents the final candidates for the combination of placement and routing when we consider reliability (reducing the Total Sensitive bits), these candidates are: Short Sensitive Estimate 1 placement routed using the Short Sensitive bit Aware router, Bounding Box placement routed using the Short Sensitive Aware router, Path Timing Driven placement routed using the Short Sensitive Aware router and Finally, the Net Timing Driven placement routed using the timing driven router.
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Path TimingR</th>
<th>BB TimingR</th>
<th>BB ShortR</th>
<th>Path ShortR</th>
<th>Net TimingR</th>
<th>Net ShortR</th>
<th>Short1 TimingR</th>
<th>Short1 ShortR</th>
<th>Short2 TimingR</th>
<th>Short2 ShortR</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>72990</td>
<td>68623</td>
<td>56362</td>
<td>50963</td>
<td>62915</td>
<td>54768</td>
<td>71270</td>
<td>39075</td>
<td>65481</td>
<td>53094</td>
</tr>
<tr>
<td>apex2</td>
<td>110808</td>
<td>100084</td>
<td>77698</td>
<td>88937</td>
<td>99860</td>
<td>83248</td>
<td>105655</td>
<td>73706</td>
<td>98551</td>
<td>78621</td>
</tr>
<tr>
<td>apex4</td>
<td>76136</td>
<td>71654</td>
<td>53079</td>
<td>59961</td>
<td>66709</td>
<td>49742</td>
<td>76432</td>
<td>58876</td>
<td>70673</td>
<td>51664</td>
</tr>
<tr>
<td>bigkey</td>
<td>59805</td>
<td>52914</td>
<td>35323</td>
<td>50143</td>
<td>58067</td>
<td>62072</td>
<td>54595</td>
<td>32474</td>
<td>58978</td>
<td>49879</td>
</tr>
<tr>
<td>des</td>
<td>79013</td>
<td>76802</td>
<td>45963</td>
<td>56131</td>
<td>69393</td>
<td>53544</td>
<td>75904</td>
<td>42922</td>
<td>78896</td>
<td>57175</td>
</tr>
<tr>
<td>dffeeq</td>
<td>48370</td>
<td>42373</td>
<td>36485</td>
<td>31365</td>
<td>45470</td>
<td>38955</td>
<td>50484</td>
<td>33440</td>
<td>46808</td>
<td>42107</td>
</tr>
<tr>
<td>dsip</td>
<td>59552</td>
<td>42686</td>
<td>27328</td>
<td>29504</td>
<td>49368</td>
<td>46346</td>
<td>48123</td>
<td>23224</td>
<td>48505</td>
<td>30402</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>10571</td>
<td>10229</td>
<td>7683</td>
<td>8815</td>
<td>9766</td>
<td>8998</td>
<td>11038</td>
<td>9384</td>
<td>10157</td>
<td>9209</td>
</tr>
<tr>
<td>ex5p</td>
<td>67553</td>
<td>62047</td>
<td>46012</td>
<td>52189</td>
<td>64028</td>
<td>59118</td>
<td>64778</td>
<td>50591</td>
<td>64455</td>
<td>48959</td>
</tr>
<tr>
<td>misex3</td>
<td>78140</td>
<td>74444</td>
<td>50165</td>
<td>61892</td>
<td>70569</td>
<td>60324</td>
<td>75525</td>
<td>66735</td>
<td>71040</td>
<td>57097</td>
</tr>
<tr>
<td>s298</td>
<td>81810</td>
<td>71961</td>
<td>62352</td>
<td>48125</td>
<td>67112</td>
<td>51347</td>
<td>74566</td>
<td>47469</td>
<td>70147</td>
<td>54461</td>
</tr>
<tr>
<td>seq</td>
<td>96896</td>
<td>94930</td>
<td>71084</td>
<td>75849</td>
<td>87903</td>
<td>80273</td>
<td>98425</td>
<td>55943</td>
<td>93849</td>
<td>77278</td>
</tr>
<tr>
<td>tseng</td>
<td>31376</td>
<td>29730</td>
<td>22517</td>
<td>27212</td>
<td>25541</td>
<td>24404</td>
<td>30188</td>
<td>27996</td>
<td>31055</td>
<td>23042</td>
</tr>
<tr>
<td>Reduction</td>
<td>10.1%</td>
<td>50.3%</td>
<td>38.4%</td>
<td>12.7%</td>
<td>29.7%</td>
<td>4.3%</td>
<td>58.2%</td>
<td>7.6%</td>
<td>38.2%</td>
<td></td>
</tr>
</tbody>
</table>

Table 13 Total Sensitive bits for Bounding Box, Path Timing Driven, Net Timing driven, Short Sensitive Estimate 1 and Short Sensitive Estimate 2 routed using the Timing Driven router (TimingR) and the Short Sensitive Aware router
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Path TimingR</th>
<th>BB Timing R</th>
<th>BB ShortR</th>
<th>Path ShortR</th>
<th>Net TimingR</th>
<th>Net ShortR</th>
<th>Short1 TimingR</th>
<th>Short1 ShortR</th>
<th>Short2 TimingR</th>
<th>Short2 ShortR</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>8.33E-08</td>
<td>1.22E-07</td>
<td>1.78E-07</td>
<td>1.67E-07</td>
<td>1.09E-07</td>
<td>1.77E-07</td>
<td>1.17E-07</td>
<td>1.87E-07</td>
<td>1.15E-07</td>
<td>1.66E-07</td>
</tr>
<tr>
<td>apex2</td>
<td>9.29E-08</td>
<td>1.35E-07</td>
<td>1.93E-07</td>
<td>1.48E-07</td>
<td>1.13E-07</td>
<td>1.54E-07</td>
<td>1.22E-07</td>
<td>1.67E-07</td>
<td>1.33E-07</td>
<td>1.69E-07</td>
</tr>
<tr>
<td>apex4</td>
<td>1.09E-07</td>
<td>1.36E-07</td>
<td>1.73E-07</td>
<td>1.34E-07</td>
<td>9.71E-08</td>
<td>1.48E-07</td>
<td>1.27E-07</td>
<td>1.68E-07</td>
<td>1.33E-07</td>
<td>2.19E-07</td>
</tr>
<tr>
<td>bigkey</td>
<td>7.55E-08</td>
<td>8.93E-08</td>
<td>1.68E-07</td>
<td>1.34E-07</td>
<td>5.53E-08</td>
<td>1.55E-07</td>
<td>9.93E-08</td>
<td>1.83E-07</td>
<td>9.27E-08</td>
<td>2.14E-07</td>
</tr>
<tr>
<td>des</td>
<td>1.01E-07</td>
<td>1.14E-07</td>
<td>1.92E-07</td>
<td>1.83E-07</td>
<td>1.25E-07</td>
<td>1.64E-07</td>
<td>1.53E-07</td>
<td>2.18E-07</td>
<td>1.51E-07</td>
<td>2.06E-07</td>
</tr>
<tr>
<td>diffeq</td>
<td>8.67E-08</td>
<td>9.58E-08</td>
<td>1.40E-07</td>
<td>1.28E-07</td>
<td>7.40E-08</td>
<td>1.36E-07</td>
<td>8.75E-08</td>
<td>1.29E-07</td>
<td>9.24E-08</td>
<td>1.27E-07</td>
</tr>
<tr>
<td>dsip</td>
<td>8.68E-08</td>
<td>8.03E-08</td>
<td>1.62E-07</td>
<td>1.44E-07</td>
<td>5.29E-08</td>
<td>7.21E-08</td>
<td>8.33E-08</td>
<td>1.41E-07</td>
<td>1.05E-07</td>
<td>2.13E-07</td>
</tr>
<tr>
<td>e64-4lut</td>
<td>3.45E-08</td>
<td>4.18E-08</td>
<td>4.72E-08</td>
<td>4.21E-08</td>
<td>3.27E-08</td>
<td>4.75E-08</td>
<td>4.35E-08</td>
<td>5.17E-08</td>
<td>3.83E-08</td>
<td>4.89E-08</td>
</tr>
<tr>
<td>ex5p</td>
<td>8.31E-08</td>
<td>1.26E-07</td>
<td>1.65E-07</td>
<td>1.19E-07</td>
<td>8.72E-08</td>
<td>1.17E-07</td>
<td>1.21E-07</td>
<td>1.51E-07</td>
<td>1.04E-07</td>
<td>1.31E-07</td>
</tr>
<tr>
<td>misex3</td>
<td>8.87E-08</td>
<td>1.04E-07</td>
<td>1.67E-07</td>
<td>1.61E-07</td>
<td>9.54E-08</td>
<td>1.67E-07</td>
<td>1.12E-07</td>
<td>1.64E-07</td>
<td>1.04E-07</td>
<td>1.56E-07</td>
</tr>
<tr>
<td>s298</td>
<td>1.26E-07</td>
<td>2.06E-07</td>
<td>3.92E-07</td>
<td>2.65E-07</td>
<td>1.56E-07</td>
<td>2.61E-07</td>
<td>2.09E-07</td>
<td>3.72E-07</td>
<td>1.95E-07</td>
<td>5.61E-07</td>
</tr>
<tr>
<td>seq</td>
<td>1.10E-07</td>
<td>1.11E-07</td>
<td>1.86E-07</td>
<td>1.40E-07</td>
<td>9.40E-08</td>
<td>1.67E-07</td>
<td>1.29E-07</td>
<td>1.71E-07</td>
<td>1.30E-07</td>
<td>1.75E-07</td>
</tr>
<tr>
<td>tseng</td>
<td>6.09E-08</td>
<td>7.31E-08</td>
<td>1.33E-07</td>
<td>9.01E-08</td>
<td>5.47E-08</td>
<td>9.78E-08</td>
<td>8.18E-08</td>
<td>1.42E-07</td>
<td>8.07E-08</td>
<td>1.04E-07</td>
</tr>
<tr>
<td>Overhead</td>
<td>17.9%</td>
<td>47.3%</td>
<td>35.8%</td>
<td>-5.5%</td>
<td>34.7%</td>
<td>21.1%</td>
<td>46.2%</td>
<td>20.8%</td>
<td>47.4%</td>
<td></td>
</tr>
</tbody>
</table>

Table 14 Critical Path Timing for Bounding Box, Path Timing Driven, Net Timing driven, Short Sensitive Estimate 1 and Short Sensitive Estimate 2 routed using the Timing Driven router (TimingR) and the Short Sensitive Aware router.
Figure 35 Plotting the Data point Averages of Total Sensitive bits (Table 15) VS Average of Critical Path Delay (Table 16) for various placement and routing combinations.
Chapter 6

Conclusion

In this work we presented reliability-aware placement and routing approaches for designs to be mapped into SRAM-based FPGAs. Our approach was incorporated into a well known FPGA CAD tool, the VPR toolset. We implemented our modifications to the tool so that it accounts for reliability during the placement and routing. We ran the tool using different configurations on different circuits of the MCNC benchmark and compared the final placed and routed circuits in terms of sensitive bits, critical path delay and number of routing tracks (as a measure of the total area used). We were able to achieve 58.2% improvement on the number of total sensitive bits compared to the default configuration of the VPR tool (using the path timing driven placement routed using the timing driven router). However, the average increase in the critical path delay was 47%. The area overhead is 18.32% more routing track used by our algorithms.

We have proposed various SEU-aware cost components during placement and routing. The combinations (along with original cost components), parameters, and weights can be adjusted and fine-tuned to find the right balance of reliability, area, and delay. Also, different routing algorithms can be used to route different nets based on their time criticality (slack) and soft error vulnerability (the reliability constraints importance of the module they are used in).
References


