On-Chip Communication and Security in FPGAs

Shivukumar Basanagouda Patil

ON-CHIP COMMUNICATION AND SECURITY IN FPGAS

A Thesis Presented

by

SHIVUKUMAR BASANAGOUDA PATIL

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2018

Electrical and Computer Engineering

ON-CHIP COMMUNICATION AND SECURITY IN FPGAS

A Thesis Presented

by

SHIVUKUMAR BASANAGOUDA PATIL

ACKNOWLEDGMENTS

I would like to thank Prof. Russell Tessier and Prof. Daniel Holcomb for their guidance on this thesis. I would like to express my gratitude to Prof. Jun Yao for his time and for accepting to be on my thesis committee. I would also like to thank Chethan Ramesh, Tianqi Liu, George Provelengios, and Siva Nishok Dhanuskodi for their help while working on the project.

ABSTRACT

ON-CHIP COMMUNICATION AND SECURITY IN FPGAS

SEPTEMBER 2018

SHIVUKUMAR BASANAGOUDA PATIL

B.E., SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING, MYSORE

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Russell Tessier

Innovations in Field Programmable Gate Array (FPGA) manufacturing processes and architectural design have led to the development of extremely large FPGAs. There has also been a widespread adaptation of these large FPGAs in cloud infrastructures and data centers to accelerate search and machine learning applications. Two important topics related to FPGAs are addressed in this work: on-chip communication and security. On-chip communication is quickly becoming a bottleneck in today's large multi-million gate FPGAs. Hard Networks-on-Chip (NoC), made of fixed silicon, have been shown to provide low power, high speed, flexible on-chip communication. An iterative algorithm for routing pre-scheduled time-division-multiplexed paths in a hybrid NoC FPGA architecture is demonstrated in this thesis work. The routing algorithm is based on the well known Pathfinder algorithm, overcomes several limitations of a previous greedy implementation and successfully routes connections using a higher number of timeslots than greedy approaches. The new algorithm shows an average bandwidth improvement of 11% for unicast traffic and multicast traffic patterns.

Regarding on-chip FPGA security, a recent study on covert channel communication in Xilinx FPGA devices has shown information leaking from long interconnect wires into immediate neighboring wires. This information leakage can be used by an attacker in a multi-tenant FPGA cloud infrastructure to non-invasively steal secret information from an unsuspecting user design. It is demonstrated that the information leakage is also present in Intel SRAM FPGAs. Information leakage in Cyclone-IV E and Stratix-V FPGA devices is quantified and characterized with varying parameters, and across different routing elements of the FPGAs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
THESIS OUTLINE

This thesis work is based on two loosely connected topics: on-chip communication and security. Chapter 2 describes a routing algorithm based on the Pathfinder algorithm [20] to route time-division-multiplexed flits inside hybrid networks-on-chip (NoC). Chapter 2 also provides an introduction to NoC routing, earlier background work on NoC routing, and a hybrid NoC architecture. The chapter concludes with a description of the experimental setup and the NoC routing algorithm's results.

The major contributions of this work are as follows.

- The design of an iterative algorithm to route single source-destination time-division-multiplexed (TDM) paths for a hybrid NoC architecture. We have experimentally determined the Pathfinder algorithm's cost parameters to maximize the bandwidth utilization across different traffic patterns (Chapter 2).

- The single source-destination routing algorithm was extended to route multicast TDM traffic (Chapter 2).

- The algorithm was extended by analyzing the routing results of multiple traffic patterns and comparing the improvements with a previous implementation's [18] routing results. The routing results are simulated using a cycle accurate NoC simulator. Theoretical bandwidth and latency estimates derived from the routing results are experimentally confirmed using the NoC simulator (Section 2.5).

.

Chapter 3 describes a new security vulnerability in Intel FPGAs that allows an attacker to non-invasively steal secret information from another user design in a multi-tenant scenario in which multiple independent users have circuits on the same FPGA. The chapter includes experimental methodology for characterizing information leakage in Intel FPGAs. Section 3.6 analyzes several characterization results by varying FPGA parameters and operating conditions.

- It has been confirmed that there is information leakage from the long interconnect wires in the Intel SRAM FPGAs, namely Cyclone-IV and Stratix-V FPGAs. We have verified the information leakage across multiple copies of the FPGA boards. A CAD flow was devised to perform detailed manual routing for Intel FPGAs. An experimental flow was designed to characterize the information leakage using a ring oscillator (RO) counter along with measurement circuitry. (Chapter 3).

- We analyzed the amount of information leakage for different types of horizontal and vertical routing elements inside the FPGA, showed how the amount of information leakage varies at different locations on the chip, for different number of routing elements, and the amount of overlap between transmitter and receiver. (Section 3.6).

- We characterized the information leakage for static and dynamic transmitter patterns of different Hamming weights and frequency (Section 3.6).

CHAPTER 2
NETWORK-ON-CHIP ROUTING IN FPGAS

2.1 Introduction

The need for both hard and soft FPGA NoCs has been well documented and a number of NoC architectures have been developed. Packet-switched (PS) NoCs (similar to those used on multi-core microprocessors) are well-suited to bursty traffic with unpredictable destinations. Time division multiplexed (TDM) NoCs route stream-based traffic using a schedule derived at compile time. In general, TDM NoCs exhibit reduced latency and energy consumption versus packet-switched NoCs since router buffering can be largely eliminated. This savings does come at the cost of some per-router storage for the schedule memory. Recently, hybrid FPGA NoCs [13, 18] have been introduced that support both PS and TDM routing. Since TDM routing is performed at compile time and PS routing is performed at run time, care must be taken during TDM routing to provide low latency and high throughput for stream-based traffic without blocking additional PS traffic.

The selection of NoC channels and time slots for each TDM route requires an extensive spatial and temporal search even if routes are constrained to shortest paths. The lack of buffering for TDM routes requires time slots to be allocated sequentially. Additionally, the use of routing resources should be balanced across the network to allow for sufficient time slots for PS routing at run time. Previous work on TDM routing algorithms for FPGA NoCs [13, 18] generally focuses on algorithm speed rather than optimizing the amount of TDM bandwidth or balancing routing resources by minimizing channel congestion. Given the amount of compile time needed for

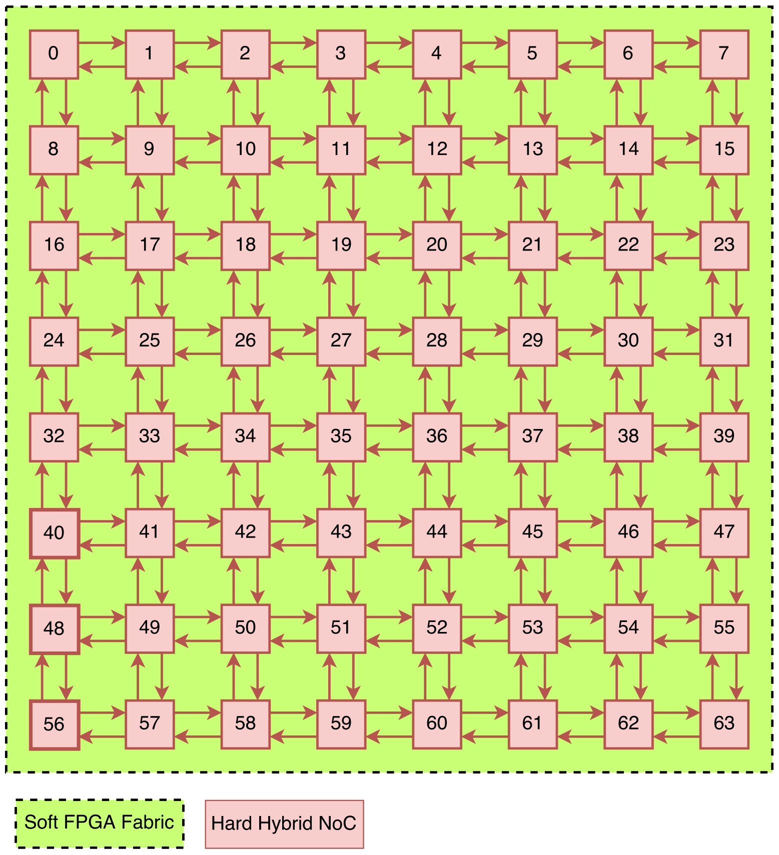


Figure 2.1: 8x8 NoC Mesh Topology [18]

physical design of the FPGA logic, allocating a few seconds of additional compile time to obtain higher-bandwidth TDM routes can be an effective goal.

Our new TDM routing approach for hybrid NoCs maximizes and balances TDM bandwidth by using a multi-iteration one-step routing algorithm. For each routed TDM flit, the algorithm performs a breadth-first search that simultaneously considers both physical routing channels and time slots. A key aspect of the algorithm is its use of a Pathfinder [20] approach to ripup and reroute. Wire and time-slot overuse in a specific iteration is reflected in a non-decreasing cost value that can be used to gently guide routes away from a resource during later iterations. Our TDM routing algorithm is applied to a hybrid FPGA NoC for five widely-used NoC routing patterns.

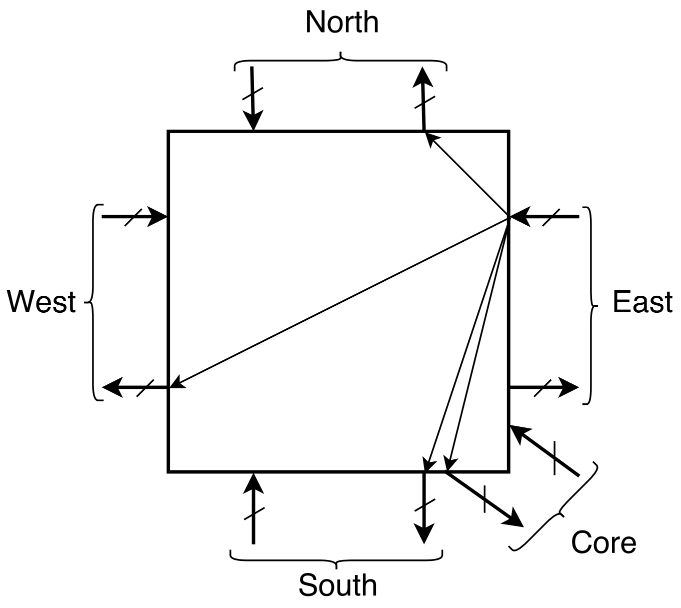


Figure 2.2: IO ports of a NoC router. Each input port can be connected to one or more output ports

2.1.1 NoC Architecture

The NoC routing experiments in this work are performed for the hybrid FPGA NoC architecture shown in Figure 2.1 [18]. The NoC network is implemented as hard silicon IP and is overlaid on the soft (programmable) FPGA fabric. The NoC network has two dimensions ($n = 2$), the X-dimension and the Y-dimension; along each dimension there are eight ($k = 8$) NoC routers. Thus, there are sixty four NoC routers ($size = k^n = 8^2 = 64$) in total, each router is referenced by a unique index number. Each NoC router can communicate in full-duplex mode with up to four of its adjacent routers. All routers in the NoC have the same number of timeslots (for our experimentation, eight).

Figure 2.2 shows the IO ports of a NoC router. The router has four interconnect ports along North, South, East, and West directions. The NoC router communicates with its adjacent NoC routers using the interconnect ports. The NoC also has a core port which interfaces the NoC router to the soft FPGA fabric. The core port

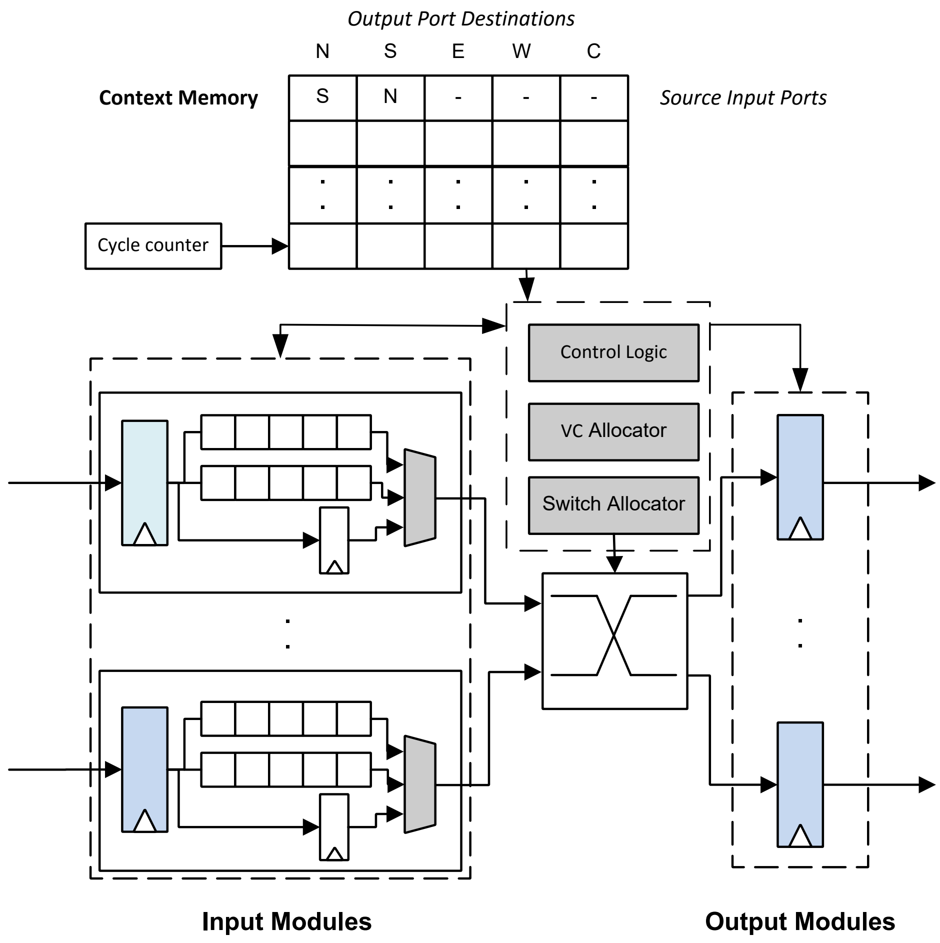


Figure 2.3: Router for Hybrid FPGA NoC [18]

has an asynchronous FIFO buffer that performs data width conversion and data rate conversion between the soft FPGA fabric and the NoC clock domains. Figure 2.3 shows the internal blocks of a typical hybrid NoC router [18]. The router contains a context memory for TDM schedule information that can configure the crossbar to connect a specific source port to each output port (e.g. S, N, ..). Output ports with a – in the context memory on a specific cycle instead accept PS data. If a TDM connection is allocated and no input data is available, PS data may be forwarded instead. Each input module contains two sets of FIFOs that serve as PS buffers and a bypass register for TDM data. Packet-switched routing is performed using dimensionordered routing (DOR). Since the TDM schedule length is short (eight time slots), effective routing heuristics are needed to efficiently use available bandwidth.

2.2 Background

FPGA NoCs have several characteristics that differentiate them from multi-core microprocessor NoCs. Since FPGA cores are expected to be simple, transmitted data values are expected to arrive in order at the receiver. This issue generally results in shortest-path routing for TDM communication and communication protocols with predictable paths (e.g. DOR) for PS routing [2].

Most previous compile-time TDM approaches first identify the routing channels used by paths and then select routing time slots. This two-step approach is easy to implement but can lead to suboptimal results. For example, Lu and Jantsch [19] use a depth-first search to select routing paths followed by scheduling. In Carle *et al.* [4], TDM routes are limited to DOR to reduce route search complexity. A previous hybrid FPGA router [13] can support simultaneous TDM and PS routing although all traffic is limited to DOR patterns. Multiple TDM routing approaches have combined channel with time slot selection. These algorithms greedily identify feasible solutions. Kapre *et al.* [14] attempt to schedule routes using the first available time slot. Multi-iteration ripup and reroute is mentioned but not implemented. Shpiner *et al.* [26] use a single- iteration random-greedy scheduling algorithm to determine route time slots. Evain and Diguet [6] use a space-time route allocation algorithm to minimize the TDM schedule. In contrast, our approach combines path and time slot selection in a single pass. TDM routes are not restricted to DOR; they can take any available shortest path from source to destination. Although our TDM routing algorithm could be applied to any FPGA NoC that supports TDM, it is optimized for hybrid NoCs with both TDM and packet-switching.

Liu *et al.* [18] described a one-step greedy shortest path algorithm for TDM routing. The algorithm selects intermediate nodes that have the least congestion along the shortest path from source to destination. Once a routing resource is assigned to a TDM net, it is permanently allocated to that TDM net and is not changed when routing subsequent TDM nets. This activity leads to a net ordering issue where the nets routed first will always have a higher priority claim on the routing resources. If the initial routing path selection is not optimal, the overall routability of the design will be affected. Figure 2.4 illustrates the limitations of the greedy shortest path algorithm. Assume that the Path1 is routed before Path2. Since Path1 does not experience any congestion, it can select a routing path which conflicts with the routing path of Path2. Our algorithm overcomes the net ordering issue using Pathfinder's iterative rip-up and reroute technique. After the first iteration, the conflicting path's nodes are assigned a higher cost. Hence, during the rip-up and reroute phase of the algorithm the router will select a different, less congested routing path for Path1.

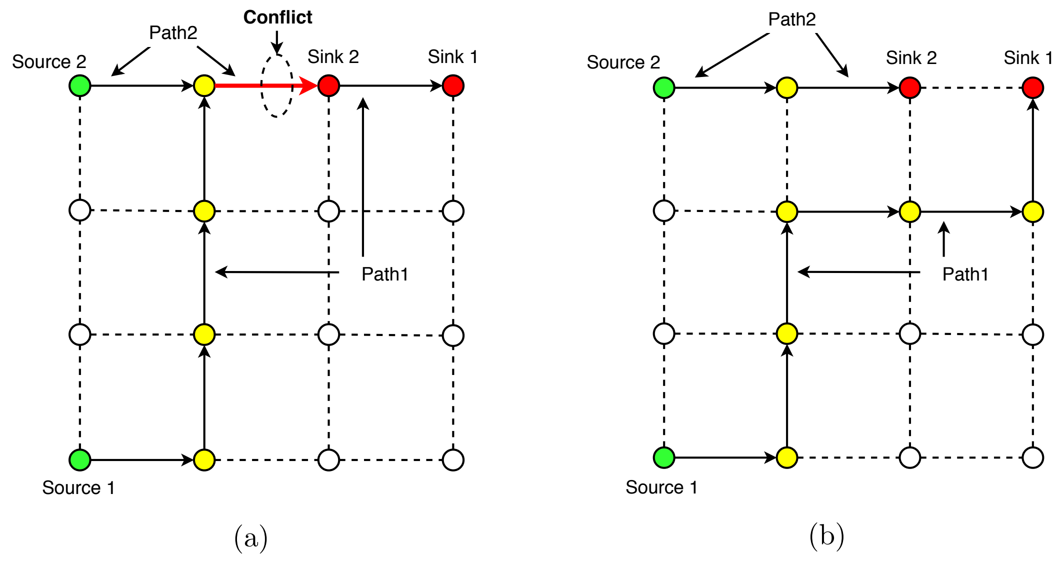


Figure 2.4: (a) Shortest Path algorithm [18] has resulted in conflicts. (b) Pathfinder routing results after rip-up and rerouting.

2.3 Routing Algorithm Implementation

Our multi-iteration routing algorithm [21] attempts to maximize TDM routing up to the allocation requested by the user at compile time. Unused routing time slots can be used for packet-switched routing. Our algorithm consists of three parts illustrated in Algorithms 1, 2, and 3. Each FPGA source component (e.g. soft processor, IP core) communicates with one or more destination components forming a *connection*. Each connection supports one or more periodic data transmissions (e.g. flits). To provide comparisons to PS routing, four flits are considered a packet.

```
  Data: List of multi-fanout connections (input to output)
  Result: List of routing paths ($S$) for every connection
  /* Try all starting time slots */
1 foreach $i = 0$ to num_slots do
2     for $j = i$, $j <$ num_slots, j++ do
3         unrouted_set = $perform\_set\_route$(unrouted_set, j)
4         if no shared (channel, time slot) then
5             break
6         end
7     end
8 end
```

Algorithm 1: Traffic path algorithm [21]

```
  Data: Set of unlocked connections, start slot j
  Result: List of routing paths ($S$) with no (channel, slot) pair overlaps
  Result: Set of connections with (channel, slot) pair overlap
1 $Pathfinder\_ts$($set$, j)
2 Lock all connections with paths with no (channel, slot) pairs to overlap
3 $set$ = $set$ - locked connections
```

Algorithm 2: Route a set of unlocked connections $perform\_set\_route$

Algorithm 1 shows the top-level loop of data communication scheduling. An attempt is made to schedule a route for each connection beginning at a starting time slot at the source router. After paths consisting of (inter-router channel, timeslot) assignments are made for each connection, a check for route success is performed. A successful schedule includes no inter-router channel, time slot pairs that are shared by multiple connections. If unsuccessful, paths for connections without overlap are locked and another set of routing attempts for the remaining unrouted connections are performed beginning at the next starting time slot (e.g. starting time slot+1). The process terminates when all connections have been successfully scheduled with no overlap or no schedule can be found for them. In the latter case, unrouted connections are routed at run-time using packet switching.

```
Data: List of connections (source, destination, and starting time slot)
Result: List of routing paths ($S$) for each connection of channel, time slot pairs
1  while shared (channel, slot) pair exists and iteration $\leq$ max_iter do
2      for all connections $i$ starting at time slot $i.slot$ do
3          Rip up existing connection path $P_i$
4          $P_i \leftarrow$ (source, i.slot)
5          Initialize expansion_list = new MinHeap()
6          while all destinations $d_{ij}$ not found do
7              Remove lowest cost node m with time slot $j$ from expansion_list
8              while fanouts $n$ of node $m$ at time slot
9              (j + 1) $mod$ max_slot on shortest path exist do
10                 Add n to expansion_list at cost $c_n + C_i$
11             end
12             for all nodes $n$ in path from $d_{ij}$ to source do
13                 Update $c_n$
14                 Add $n$ to $P_i$
15             end
16         end
17     end
18 end
```

Algorithm 3: Pathfinder_ts algorithm including time slots

Algorithm 2 illustrates the use of our Pathfinder_ts algorithm that considers both spatial (channels) and temporal (timeslots) information. One application of Pathfinder_ts is performed per starting time slot. Connections that are successfully routed are locked.

Algorithm 3 forms the heart of our route scheduling approach. Each source-destination connection i is built from a series of channel, time slot pairs along a shortest path. Multi-fanout connections are supported and multiple iterations of rip-up and reroute are performed. As a route proceeds from source to destination, the cost of a new channel, time slot pair is considered. The lowest cost pair is selected at each step to create the lowest-cost scheduled path with cost C_i.

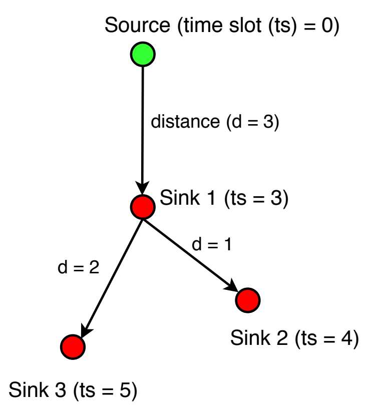
$$c_n(ts) = (1 + congestion_{ts} \times p_{fac})(1 + history_{ts} \times h_{fac}) \tag{2.1}$$

The Pathfinder cost function [3] in 2.1 determines the cost of using an adjacent node (channel, time slot pair) in Algorithm 3. Congestion cost ($congestion_{ts}$) of a node is the number of routing paths that are currently sharing it. History cost ($history_{ts}$) is a non-decreasing value which is increased by the amount of congestion cost at the end of each iteration. Values p_{fac} and h_{fac} are congestion cost and history cost multiplication factors, respectively. They are set to constant values of 1.6 and 0.2, respectively, as determined via experimentation.

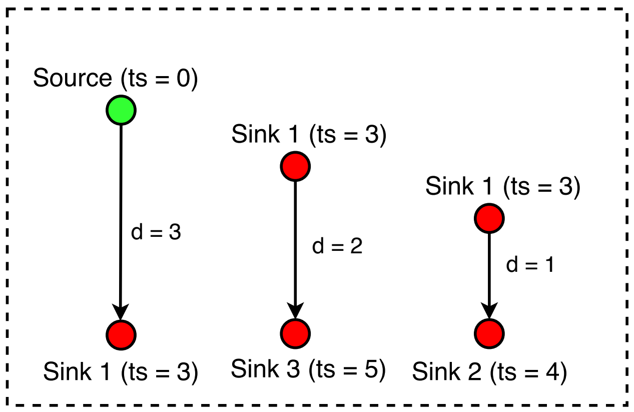
2.3.1 Multicast Routing

In multicast routing, a packet from a single source is transmitted to multiple destinations. In PS multicast routing, a separate copy of the packet is sent from the source to each destination. In TDM multicast routing, only a single copy of the TDM flit is inserted from the core port of the driving source node. At all other non-terminal sink nodes, the TDM flits are forwarded to multiple output ports to achieve multicasting; this makes multicast routing more efficient in TDM-routing. TDM multicast routing can be treated as a multi-fanout net consisting of a single source node and multiple destination nodes.

Algorithm 2 was modified to route multi-fanout connections. The multi-fanout $perform\_set\_route$ routine is shown in Algorithm 4. The algorithm decomposes the multi-fanout connections into a list of single source-destination (unicast) connections along with their corresponding starting time slots. The unicast connections are then routed using the Pathfinder_ts algorithm described in Algorithm 3. Figure 2.5 illustrates the process. A minimum spanning tree is constructed using all the nodes of a multi-fanout connection. The source node made the root of the MST. Edge weights of



(a)



(b)

Figure 2.5: Routing a multi-fanout connection with one source node and three sink nodes. (a) A MST is constructed from the multi-fanout connection nodes. The edge weights correspond to Manhattan distance between the nodes. The starting timeslot number at the source node is 0. Assuming a delay of one clock cycle per unit Manhattan distance, the timeslot number at the sink nodes is calculated. (b) The MST is decomposed into three single source-destination connections. These single source-destination connections are routing using the Pathfinder_ts algorithm (3)

the MST correspond to Manhattan distance between the nodes. The starting timeslot at each sink node is calculated using the starting timeslot of the parent node and the Manhattan distance between the nodes. A unicast connection is created for each edge of the MST. After routing the unicast connections using the Pathfinder_ts algorithm, a multi-fanout connection is added to the locked set if all the edges of the MST are routed with no overlap. The TDM flits are inserted once from the core port of the source node (router). At the branching nodes, the TDM flit is simply forwarded (copied) to multiple output ports.

2.4 Experimental Procedure

A comparison of our new bandwidth reservation algorithm over a previously-published congestion avoidance shortest-path algorithm [18] was performed using the following traffic patterns: Bit-Reverse (source = x, destination = $\textit{bit-reversed}(x)$),

```
Data: Set of unlocked multi-fanout connections, start slot j
Result: List of routing paths ($S$) with no (channel, slot) pair overlaps
Result: Set of connections with (channel, slot) pair overlap
1 foreach multi_fanout_connection in unlocked connection set do
2     Create a minimum spanning tree (MST) of the multi_fanout_connection
3     Add each edge of the MST along a unicast connection to set.
4 end
5 $Pathfinder\_ts(set$, j)
6 Lock all connections whose MST edges are routed with no (channel, slot) pairs to
    overlap
7 $set = set$ - locked connections
```
Algorithm 4: Route a set of unlocked multi-fanout connections $perform\_set\_route$

Tornado (source = (x, y), destination = $(x + \frac{k}{2} - 1 \mod k, y + \frac{k}{2} - 1 \mod k)$), Transpose (source = (x, y), destination = (y, x)), 2-Side (source-destination on parallel terminal edges of the NoC), and 4-Side (source-destination along the perimeter of the NoC). An open-source cycle accurate NoC simulator, called BookSim [11], was used to simulate the NoC, measure latency, and verify the theoretical bandwidth estimates. BookSim was updated to include the algorithms described in the previous section. Earlier work [18] was leveraged to generate TDM and PS traffic for simulation. For every traffic pattern, detailed routing was performed using the bandwidth reservation algorithm. Simulations were performed using PS-only, TDM-only, and hybrid (PS & TDM) routing. A 64 node, 8 × 8, 2D mesh NoC model was used for simulation. PS flits were routed using X-Y (dimension-order) routing. Each packet is 512 bits in size, and divided into four 128-bit flits. There are eight timeslots available for each router, giving each context memory eight entries. Table 2.1 lists all the NoC parameters of our experiments.

2.5 Results and Analysis

2.5.1 NoC Routing Results

Table 2.2 lists the routing results of the Pathfinder based traffic pattern algorithm (Algorithm 1) and shortest path algorithm [18] for the five different traffic patterns.

Topology	64-node, 8 x 8, 2D Node
Technology	45nm at 1.1V, 1.0 GHz
Routing	X-Y Routing
Channel width	128 bits
Packet size	4-flits
Context memory	8 entries
Virtual Channels	2/port
Buffer size per VC	10 flit depth

Table 2.1: Network simulation parameters of Hybrid NoC

Traffic pattern	Shortest Path					Pathfinder				
	Routed/Total	Min	Max	Mean	σ	Routed/Total	Min	Max	Mean	σ
Bit-Reverse	272/512	1	8	4.25	1.82	281/512	3	8	4.39	1.62
Tornado	141/512	1	5	2.20	0.75	161/512	2	3	2.51	0.53
Transpose	288/512	2	8	4.50	1.88	288/512	3	8	4.50	1.73
2-Side	91/128	5	7	5.68	0.68	127/128	7	8	7.93	0.24
4-Side	104/128	5	8	6.50	0.86	128/128	8	8	8.00	0.00

Table 2.2: TDM routing results of Shortest Path and Pathfinder algorithms for different single-source, single-destination traffic patterns.

For each traffic pattern, several routing results are reported. The Bit-Reverse, Tornado, and Transpose traffic patterns have 64 connections, therefore there are 512 (64 $\times$ 8 time slots) possible TDM connections. In contrast, 2-Side and 4-Side are low congestion traffic patterns with 16 connections, i.e. 128 (16 $\times$ 8 timeslots) TDM connections. The first column of the each algorithm's result shows the number of TDM connections successfully routed out of the maximum available TDM connections for each one of the traffic patterns. The Min and Max columns indicate the minimum and the maximum number of timeslots routed for any connection in a traffic pattern. The Mean and σ columns show the average and standard deviation of the number of timeslots routed for all the connections in a traffic pattern.

From Table 2.2, it is clear that the Pathfinder_ts based routing algorithm routes a higher number of timeslots than the greedy shortest path algorithm [18] for all traffic patterns. A higher number of timeslots directly translates to higher bandwidth as shown in Table 2.3. On average, our Pathfinder based algorithm achieves 11 % higher

Traffic pattern	Shortest Path (%)	Pathfinder (%)
Bit-Reverse	53.12	54.88
Tornado	27.53	31.44
Transpose	56.25	56.25
2-Side	71.09	99.21
4-Side	81.25	100.0
Average	57.84	68.35

Table 2.3: Fraction (in %) of total bandwidth routed by shortest path and Pathfinder algorithms.

bandwidth compared to the shortest path algorithm [18] for single-destination traffic patterns. Also, in the Pathfinder's results, a higher Min and lower standard deviation means a better distribution of the available timeslots among the different connections of the traffic pattern.

For Bit-Reverse and Tornado traffic patterns, the shortest path algorithm is limited by the net ordering issue, and for some connections, the shortest path algorithm is able to route only a single timeslot. The effects of the net ordering issue on the routing results are more pronounced in the low congestion 2-Side and 4-Side traffic pattern results. Both 2-Side and 4-Side traffic patterns are fully routable designs, which means that there is a feasible path from every source node to its sink node. From the bandwidth results shown in Table 2.3, the Pathfinder algorithm achieves about 29% higher bandwidth for the 2-Side traffic pattern, and about 20% higher bandwidth for the 4-Side traffic pattern. These results indicate the advantages of the one-step rip up and reroute approach in the Pathfinder algorithm.

The latency of the Transpose, Tornado, and Bit-Reverse traffic patterns for the hybrid NoC model were generated using the BookSim software. The average network latency results of PS routing, and TDM routing using the shortest path algorithm and Pathfinder algorithms are plotted in Figure 2.6. As evident by the bandwidth results in Table 2.3, the Pathfinder algorithm has the lowest latency at higher packet insertion rates for all traffic patterns when compared to the shortest path algorithm.

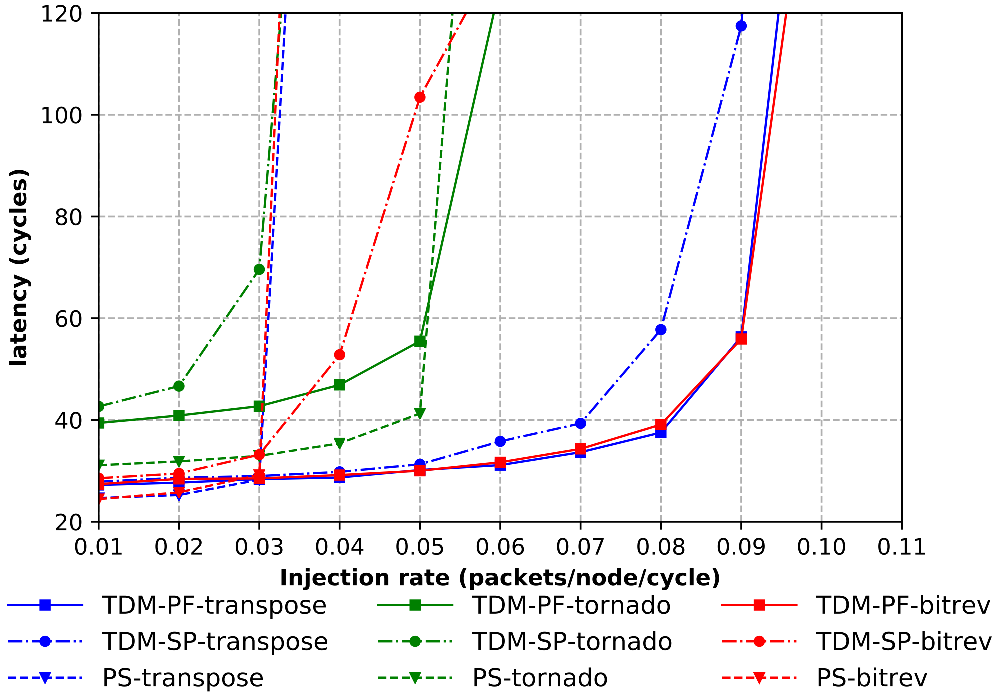


Figure 2.6: Average packet latency for transpose, tornado and bit-reverse traffic pattern using only TDM or PS routing. Note: SP (Shortest Path), PF (Pathfinder).

As there are more scheduled TDM timeslots in the Pathfinder's routing, the TDM flits wait for a shorter time for the next available timeslot in the core port.

Interestingly, at lower packet insertion rates, PS routing performs better than Pathfinder algorithm because the PS flits routing don't have to wait at the core port for a scheduled timeslot. As the packet insertion rate increases, the PS routing quickly gets limited by DOR, and the latency increases sharply at higher packet insertion rates. For the Bit-Reverse and Tornado traffic patterns, although the shortest path algorithm's average number of timeslots routed per connection is very close to the Pathfinder's value (Table 2.2), it still performs poorly at higher packet insertion rates. This is due to the fact that for some of the connections the minimum number of timeslots routed is just one in the shortest path algorithm (Table 2.2). Therefore, these bandwidth starved connections get saturated at relatively lower packet insertion rates and sharply increase the average latency of the entire network. PS routing

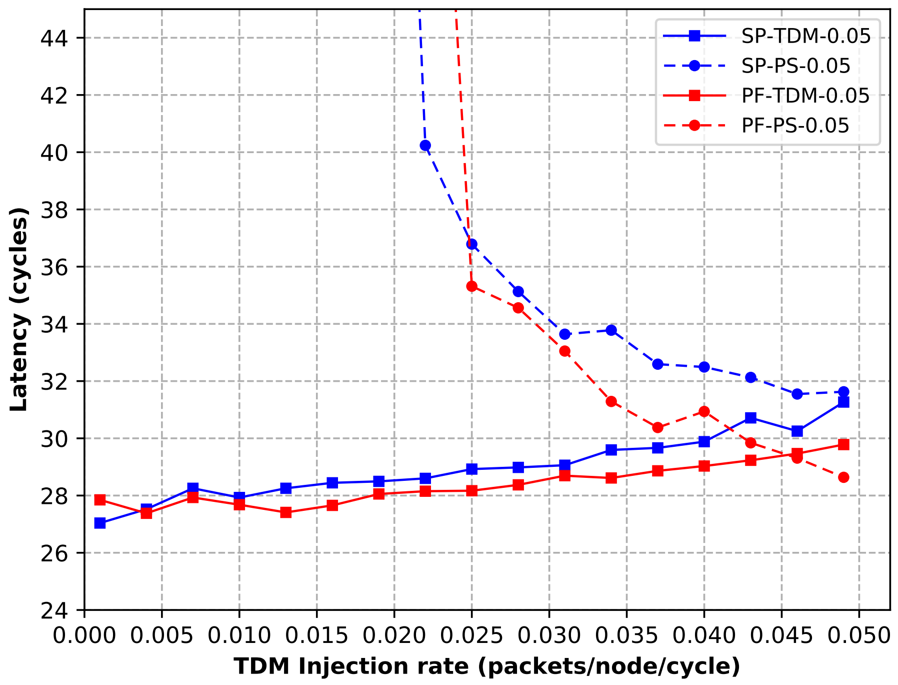


Figure 2.7: Average NoC latency for different amounts of TDM traffic for transpose under fixed total packet injection rate of 0.05 packets/node/cycle. Note that PS traffic decreases as TDM traffic increases.

routing performs better than Pathfinder routing for the Tornado traffic pattern since Tornado is well suited for DOR, and PS flits do not need to wait for the scheduled timeslots in the core port.

Figure 2.7 shows latency results for the Transpose traffic consisting of both TDM and PS packets. The total injection rate is fixed at 0.05 packets/node/cycle, and the TDM injection rate is increased gradually. Initially, as PS traffic is high, it has higher latency. As the TDM injection rate is increased, the PS traffic reduces, and subsequently, PS latency decreases. As indicated in Table 2.2, the Pathfinder algorithm has a lower standard deviation compared to the shortest path algorithm, indicating better distribution of timeslots throughout the network. Hence, the Pathfinder algorithm results in lower latency for both PS and TDM traffic.

Traffic pattern	Shortest Path				Pathfinder			
	Routed/Total	Bandwidth(%)	Mean	σ	Routed/Total	Bandwidth (%)	Mean	σ
4-Destination Multicast	84/128	65.62	5.25	1.14	99/128	77.34	6.18	0.91

Table 2.4: Multicast routing results of the shortest path and Pathfinder algorithms. There are 16 four-destination multicast connections in the traffic pattern

2.5.2 Multi-cast Routing Results

Table 2.4 compares the multicast routing results using the Pathfinder and shortest path algorithms. The multi-cast traffic pattern [18] consists of 16 four-destination multicast connections. The four destination nodes are chosen such that they are separated from the source by a minimum distance of three nodes. Similar to the unicast routing results, the Pathfinder algorithm overcomes the net ordering issue and shows about 12% improvement in overall bandwidth.

2.6 Conclusion

In this chapter, a heuristic routing algorithm that is tuned for routing traffic in hybrid FPGA NoCs is presented. This multi-iteration router balances routed traffic across routing resources to reduce congestion. Multiple rip-up and reroute iterations are used to enhance low-latency, time-scheduled communication. Substantial data stream bandwidth and latency improvement for a collection of five benchmarks is shown using our new routing approach.

CHAPTER 3

CHARACTERIZATION OF INFORMATION LEAKAGE IN INTEL FPGAS

3.1 Introduction

In recent years, there has been an increasing interest in the use of FPGAs in cloud based infrastructure to accelerate machine learning and big data applications. Amazon EC2 F1 [1] and Microsoft Project Catapult [5] projects have deployed Xilinx Ultrascale and Altera Stratix-V FPGAs, respectively in their server infrastructures. Although these platforms support a single user at this time, a multi-tenant use scenario seems inevitable in the near future. A recent study [8, 9] on several Xilinx devices has discovered that a long interconnect wire induces a different amount of delay on its adjacent wires depending on whether it is transmitting logic '1' or logic '0'. This information can be used by attackers in a multi-tenant scenario to non-invasively snoop information from a user design. Figure 3.1 illustrates what happens when an attacker and a user design share adjacent interconnect wires in a routing channel. As shown in [8], the attacker can snoop information from the user design by measuring the delay change in the long wires attached to the attacker's circuits.

This raises serious security concerns in a multi-tenant scenario where a single FPGA will be shared by multiple untrustworthy users. The existing work is extended by first proving that the information leakage is indeed present in multiple Intel FPGA device families. Following this, the delay change is measured and characterized at different FPGA parameters and operating conditions. Experiments are repeated several times on multiple boards to validate the correctness of the results. Based on the

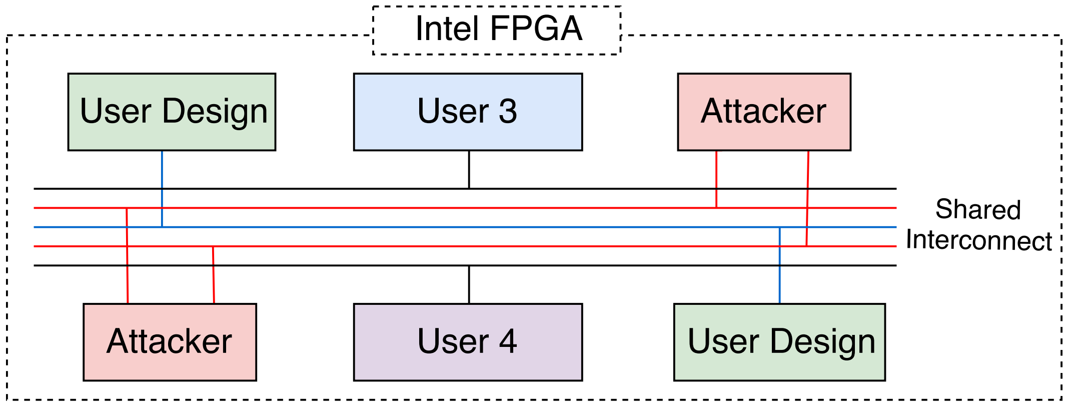


Figure 3.1: A multi-tenant FPGA attack scenario [8]. A single FPGA is shared by multiple users and an attacker can snoop information from another user by routing his interconnect wires close to the user's wires.

characterization results, vulnerabilities and limitations of information leakage are reported. These results can be used to formulate attacks and countermeasures.

3.2 Side Channel Information Leakage in FPGAs

FPGAs, like ASICs, leak information in the form of temperature, electromagnetic emanations, and power in side channels. If an attacker has physical access to the FPGA, he can snoop this information leakage from the side channels to steal secret information from the user design. Even though side channel information leakage has been studied by Ji *et al.* [27], Daniel *et al.* [30], and Iakymchuk *et al.* [10] for establishing covert channel communication, these findings focus on physical design manipulation. In an FPGA cloud based environment, most of these side channel attacks can be eliminated by restricting physical access to the side channels of the FPGAs. However, Schellenberg *et al.* [7] and Zhao *et al.* [29] have demonstrated the possibility of remote power side channel attacks inside FPGAs using on-chip sensors.

Krieg *et al.* [16] showed a technique to leak information from FPGAs by deliberately inserting malicious hardware Trojans in the device during the manufacturing process. Kelly *et al.* [15] and Lecomte *et al.* [17] showed that the hardware Trojans can be detected using on-chip monitoring techniques by analyzing the electrical activities of the device. Hardware Trojans can be less effective in FPGAs than ASICs as the design functionality is not known during the manufacturing process.

In a recent study, Giechaskiel *et al.* [8] discovered that a long interconnect wire induces different amounts of delay into its adjacent wires when it is carrying a logic '1' or a logic '0'. This delay change, although small, can be measured using a ring oscillator (RO) and a counter circuit. The period of the ring oscillator adjacent to a transmitter decreases when the transmitter is sending logic '1' compared to when the transmitter is sending logic '0'. As shown in Figure 3.2, the logic '1' and logic '0' values of the transmitter can be directly inferred from the ring oscillator's count value. The authors proved that the information leakage is present in multiple Xilinx FPGA device families and at different locations on the chip. The authors performed a comprehensive characterization of the amount of information leakage with different varying parameters for Xilinx FPGA devices.

Although interesting, the authors leave several questions unanswered, the root cause of information leakage is not analyzed, and the authors do not attempt to characterize the information leakage in Intel SRAM FPGAs. Since the routing architecture of Intel FPGAs is different from Xilinx FPGAs, the information leakage characterization of Intel FPGAs is crucial to identify security vulnerabilities. The authors' work is extended by first proving that information leakage is indeed present in Intel FPGAs. The problem is approached from an attacker's perspective, using only the information available in the public domain. A CAD tool flow is established to perform detailed manual routing for Intel FPGAs. The adjacency information between the interconnect wires is experimentally determined. Following this, a comprehensive characterization of the amount of information leakage in Cyclone-IV E and Stratix-V Intel FPGA devices is performed. From the characterization results, vulnerabilities

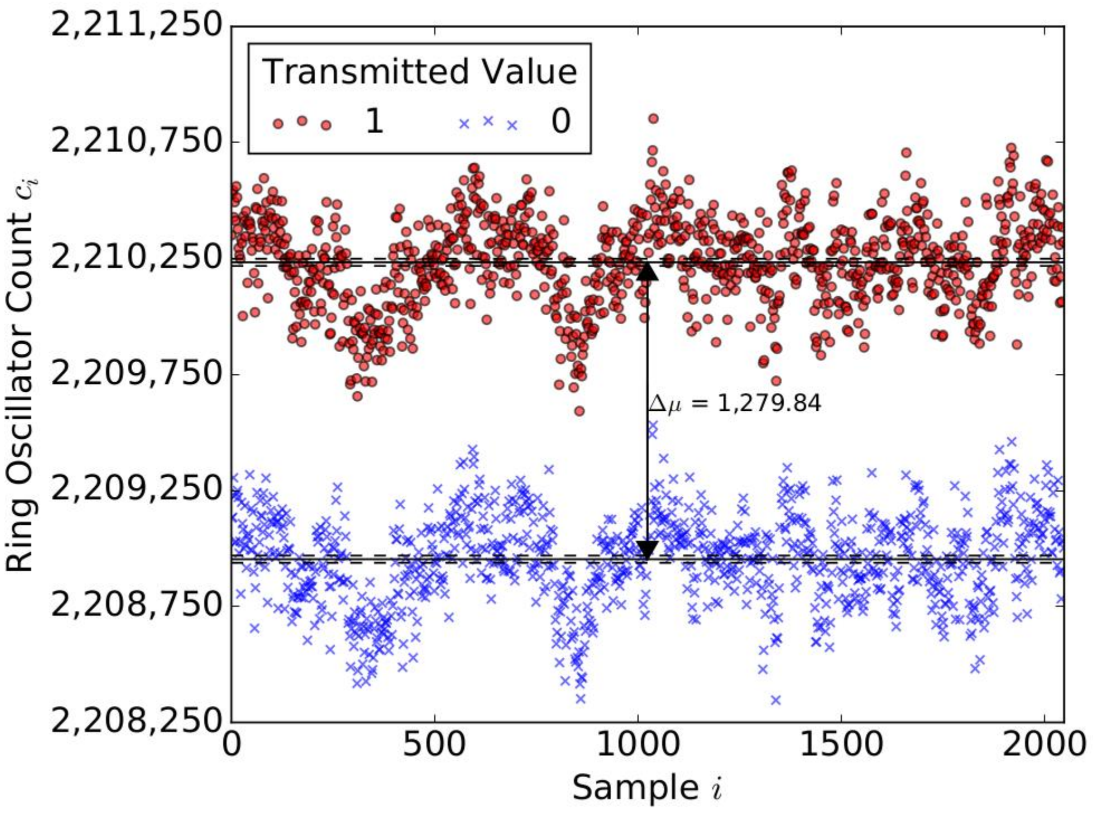


Figure 3.2: Ring oscillator count when an adjacent transmitter is carrying logic '1' or logic '0' in a Xilinx Virtex-5 XUPV5-LX110T (ML509) FPGA [8].

and limitations are identified. The characterization results can be further used to reduce the likelihood of an attack.

3.3 Approach for Characterizing Information Leakage

It is important to understand the routing architecture of Intel SRAM FPGAs to characterize the information leakage. The next sub-section briefly describes the generic routing architecture of Intel SRAM FPGAs (e.g. Stratix-V and Cyclone-IV families) . Following this, a CAD tool flow and an experimental setup is presented for characterizing the information leakage.

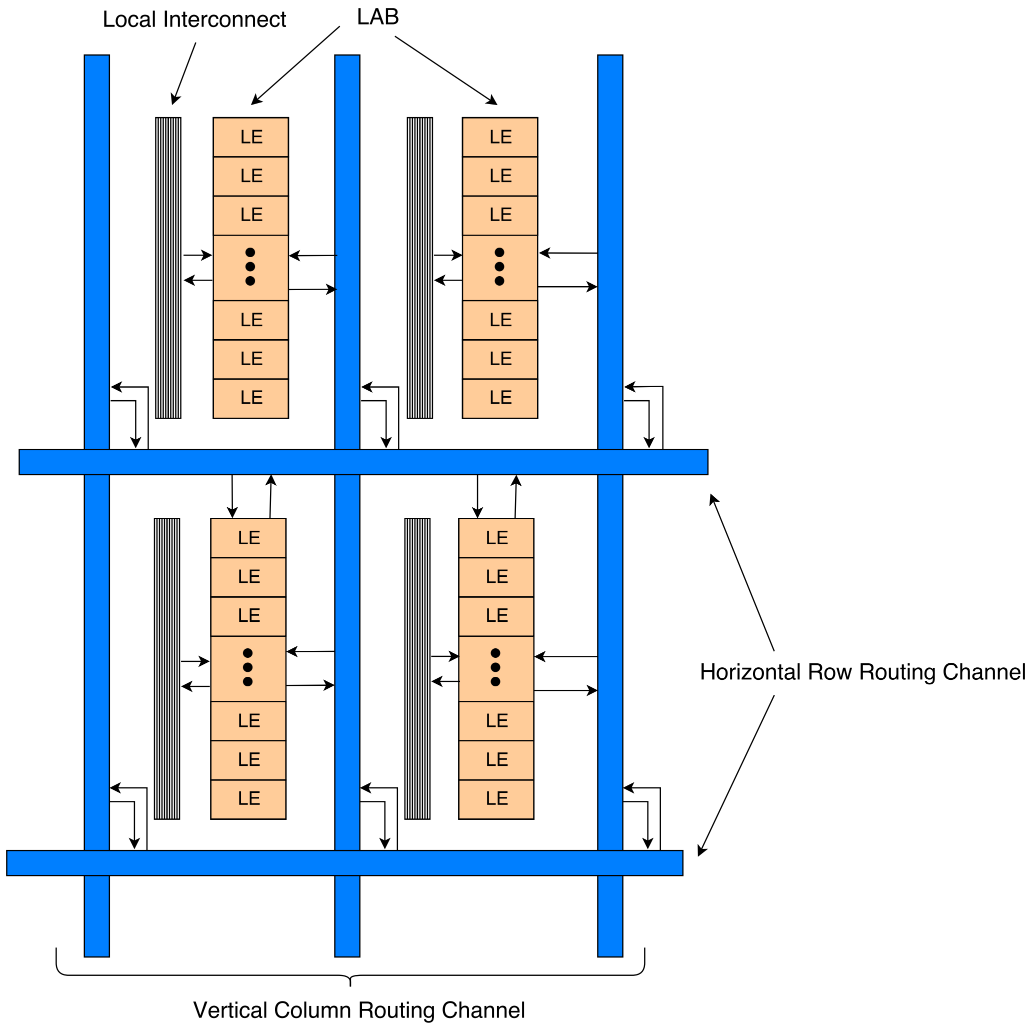

Local Interconnect
LAB
LE
Horizontal Row Routing Channel
Vertical Column Routing Channel

Figure 3.3: Routing architecture of Intel Cyclone-IV and Stratix-V FPGAs [23, 24].

3.3.1 Routing Architecture of Intel SRAM FPGAs

A generic routing architecture of a typical Intel SRAM FPGA, such as the Cyclone-IV, is illustrated in Figure 3.3. The global routing resources inside the Intel SRAM FPGAs consist of Row (horizontal) and Column (vertical) routing elements. The Row and Column routing channels interconnect the logic elements (LEs) from one logic array block (LAB) to another LAB. The column routing channel consists of C4, C8, C14, and C16 individual routing elements which can span up to 4, 8, 14, and 16 vertical LABs respectively. Similarly, the row routing channel has R3, R4, R6, and

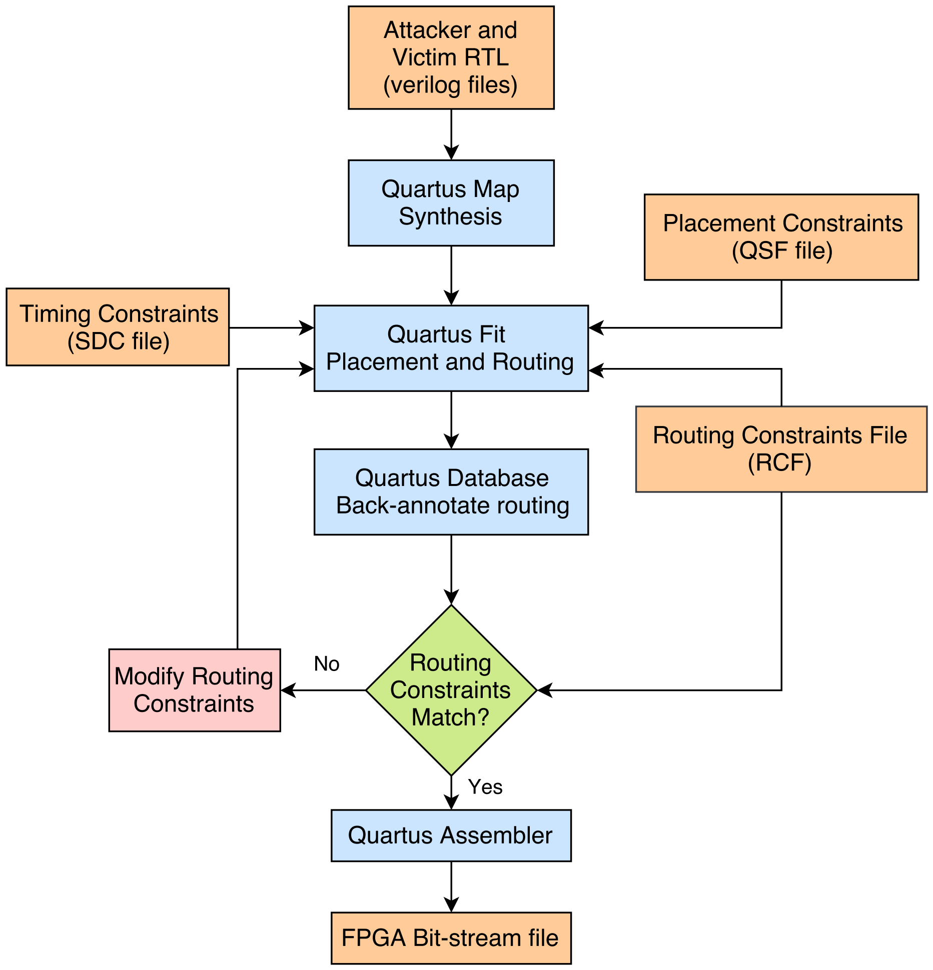

Attacker and Victim RTL (verilog files)
Quartus Map Synthesis
Placement Constraints (QSF file)
Timing Constraints (SDC file)
Quartus Fit Placement and Routing
Routing Constraints File (RCF)
Quartus Database Back-annotate routing
Modify Routing Constraints
No
Routing Constraints Match?
Yes
Quartus Assembler
FPGA Bit-stream file

Figure 3.4: Intel FPGA CAD tool flow used for characterization experiments.

R24 routing elements which span 3, 4, 6, and 24 horizontal LABs respectively. The presence of a routing resource and the number of routing resources is device dependent. For example, Cyclone-IV E FPGAs have 96 C4 and 16 C16 routing elements in the column routing channel. The relatively short routing elements (C4, R3/R4/R6) are the most abundant routing resources in Stratix-V and Cyclone-IV FPGA devices. The R3/R4/R6 and C4 routing elements can be directly driven by logic elements, while the longer R24 and C16/C14 routing elements can be only driven by R4/C4 routing elements.

3.3.2 CAD Flow

The information leakage experiments require precise placement and routing of the attacker's (receiver) circuit and the victim's circuit (transmitter). Quartus Prime Standard Edition 17.1 software is used for characterizing Stratix-V (5SGXEA7K2F40C2) and Cyclone-IV E (EP4CE115F29C7) FPGA devices. The Quartus Prime software includes all the tools required to synthesize, place and route, generate a bit-stream, and test the FPGA designs. The CAD tool flow for characterization experiments is shown in Figure 3.4. The measurement design along with the attacker's and the victim's circuits are specified in the System Verilog RTL (Register-Transfer Level) hardware description language. The RTL is synthesized using the Quartus Map (`quartus_map`) tool. The Placement and Routing tool, Quartus Fit (`quartus_fit`), is guided by placement constraints in the Quartus Settings File (QSF) and the routing constraints in the Routing Constraints File (RCF) to achieve the desired manual placement and routing of the attacker and the victim's circuits.

To guarantee that the Quartus Fit tool has honored the routing constraints, the routing results are back-annotated using the Quartus Database Tool (`quartus_cdb`), and manually compared against the user specified routing constraints. If the routing constraints match then the FPGA bit-stream is generated using the Quartus Assembler tool (`quartus_asm`).

3.3.3 Decoding the Routing Graph of Intel SRAM FPGAs

To perform precise routing as mentioned in the previous section, the attacker should have detailed device level information about the underlying routing architecture of the FPGA. The Quartus Prime tool includes an `advanced_device` TCL package which provides detailed information about the availability of different types of routing resources in a FPGA device. However, it does not provide information about fan-in and fan-out connections of the routing elements. By finding out the le-

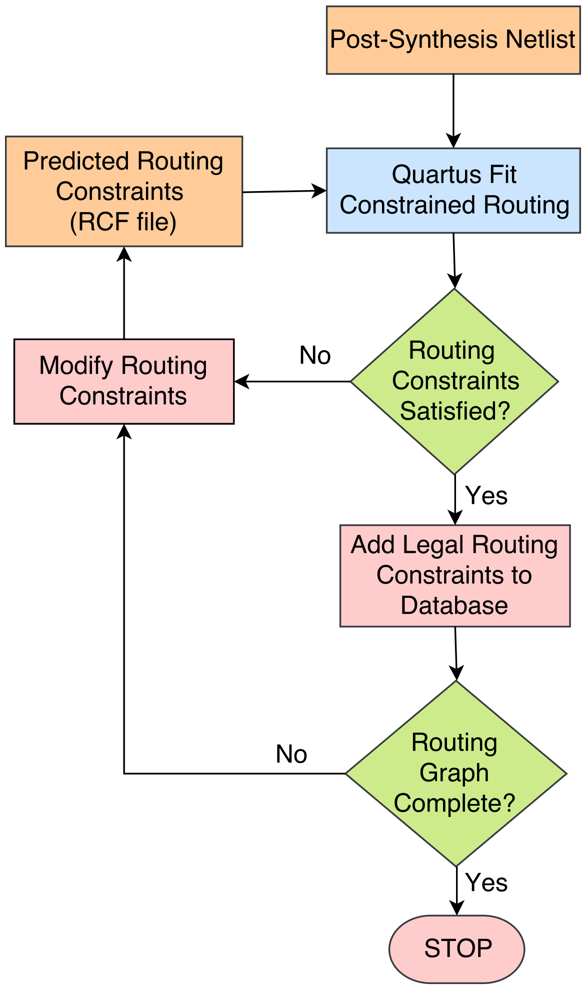

Post-Synthesis Netlist
Predicted Routing Constraints (RCF file)
Quartus Fit Constrained Routing
Modify Routing Constraints
No
Routing Constraints Satisfied?
Yes
Add Legal Routing Constraints to Database
No
Routing Graph Complete?
Yes
STOP

Figure 3.5: Tool flow to determine legal fan-in and fan-out connections of routing elements.

gal fan-in and fan-out of the routing elements, the attacker can reconstruct the entire routing graph of the FPGA device.

Typically, for security reasons, the FPGA vendors do not make the routing architecture public. However, it is still possible for an attacker to uncover the underlying routing architecture by performing a set of routing experiments and then reverse engineering auto routing results. Figure 3.5 illustrates this process. A post-synthesized netlist consisting of a single source logic cell (LCELL) connected to a single destination LCELL is used for the experimentation. The LCELLs are manually placed using the placement constraints specified in the Quartus Settings File (QSF) of the project. To find out the legal fan-outs from the source LCELL, all the routing elements in the nearby locations of source LCELL are iteratively searched. For each routing element near the source LCELL, the fan-out connection from the LCELL to the routing element is specified in the routing constraint file (RCF). Following this, constrained routing is performed using the Quartus Fit tool. If there is a legal route between the LCELL and the routing element, the Quartus Fit tool will honor the routing constraints, otherwise they are ignored. The legal routing constraints which are accepted by the Quartus Fit tool are added to a database. Since there are a finite number of routing elements near the LCELLs of a LAB, all the legal connections can be determined in a short amount of time. Consistent with the highly regular structure of the FPGAs, it has been discovered that the decoded connection information of the LAB is valid across many LABs at different locations on the chip.

3.3.4 Determining Adjacent Interconnect Wires

To perform characterization experiments, the attacker's long wire or wires should be adjacent to the victim's long wire. Since this information is not publicly available, it is derived experimentally. Figure 3.6 illustrates the process.

One of the long wires in the channel is connected to the ring oscillator, which serves as the attacker (receiver), and all the remaining wires are victim (transmitter) wires which can transmit either a logic zero or one. A one-hot test pattern is driven through the transmitters in which a single transmitter is driving logic one, and all the remaining transmitters are tied to logic zero. If a transmitter adjacent to the ring oscillator's long wire is driven to logic one, it will cause a noticeable change in the ring oscillator's frequency. In all other scenarios there will not be any significant changes in the ring oscillator's frequency. This experiment can be repeated by changing the position of the ring oscillator to determine the adjacency information of all the routing elements in the channel.

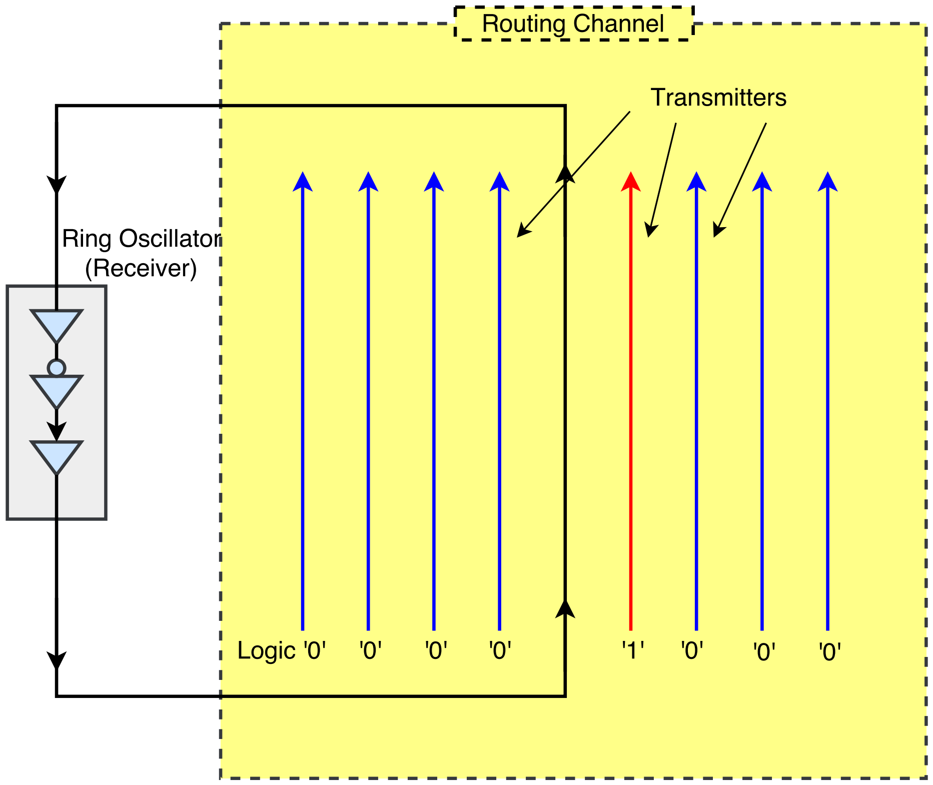


Figure 3.6: Approach for determining transmitter wires that are adjacent to a receiver.

3.4 Experimental Setup

Figure 3.7 shows the block diagram of the test setup used to characterize the information leakage from the long interconnect wires. The *transmitter* is implemented using one or more long interconnect wires. The test pattern generator (Figure 3.7) consists of a 32-bit circular serial shift register. When the test pattern generator is *ON* (enabled), it serially shifts logic values from the 32-bit register into the transmitter. By programming different values into the 32-bit test pattern generator's register, different types of static or dynamic test patterns can be sent through the transmitter. The test pattern generator assigns a logic '0' to transmitter when it is *OFF* (disabled). Similar to [8], the receiver is implemented as a three-stage ring oscillator (RO) with one inverter and two buffers. One of the wires of the RO is adjacent to the long wires of the transmitter.

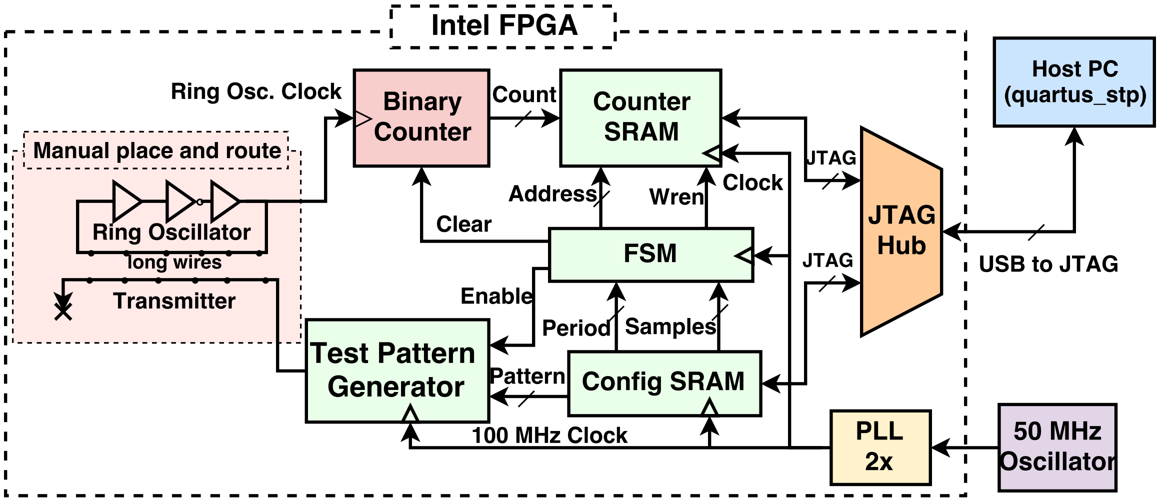


Figure 3.7: Experimental setup for information leakage characterization [25].

A binary counter measures the RO's frequency by incrementing a 32-bit count value at every positive edge of the ring oscillator clock for a fixed time duration (measurement period). After the measurement period, the count values are sampled and stored into an on-chip SRAM memory. A SignalTap II JTAG interface is used to read the stored count values. The design has a JTAG-accessible configuration memory and an FSM which coordinate test pattern generation, counting, and sampling of counter values. Unless otherwise noted, all circuitry, except the transmitter and receiver, are auto-placed and routed by Quartus Prime v17.1.

3.5 Experimental Parameters

The information leakage was measured for two different Intel FPGA family boards, Cyclone-IV and Stratix-V. For the Cyclone-IV FPGA device experiments, the DE2-115 FPGA board shown in Figure 3.8 was chosen. The board features a Cyclone-IV E FPGA device with part number EP4CE115F29C7. The device is manufactured in 65-nm process. For the Stratix-V experiments, an Altera Stratix-V GX development kit consisting of Stratix-V GX FPGA with the part number 5SGXEA7K2F40C2 was chosen (see Figure 3.9). Stratix-V FPGAs are manufactured in 28-nm process. All

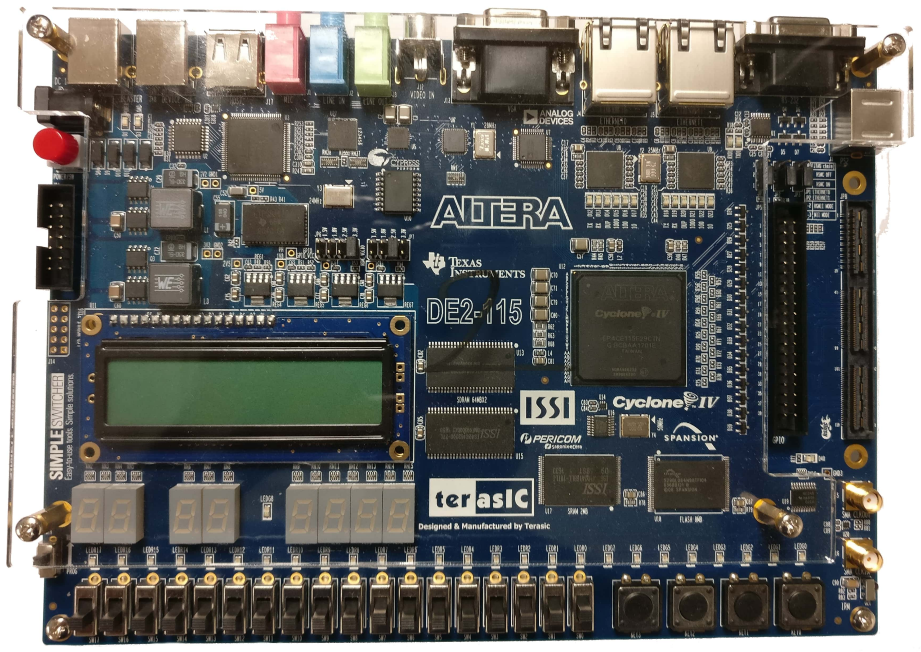


Figure 3.8: A DE2-115 board contains a Cyclone-IV E EP4CE115F29C7 FPGA [28].

the experiments were performed on two copies of the DE2-115 and Stratix-V FPGA boards. Except for heat chamber experiments, all the other experiments are performed at room temperature without any external temperature regulating equipment. Information leakage was tested at nine different locations on the FPGA.

The ring oscillator's frequency change was measured using the experimental setup [25] described in Section 3.4. The measurement period is set to 21ms, and 1024 samples were collected for each experiment. Each experiment was repeated five times for consistency. Unless otherwise stated, in all experiments the transmitter and the receiver consists of a single long routing element. The ring oscillator's frequency changes with time due to changes in temperature, voltage, and other operating parameters. Therefore, the relative change in the ring oscillator's count (ΔRC) metric described by Giechaskiel *et al.* [8] is used for is measuring the information leakage as shown in the following equation.

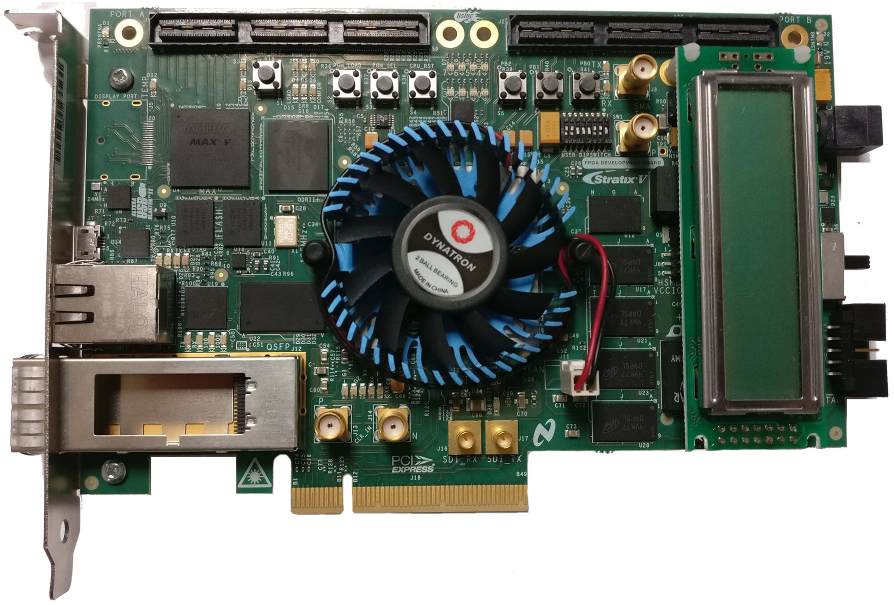


Figure 3.9: A Stratix-V GX development kit board contains a 5SGXEA7K2F40C2 FPGA [22].

$$\Delta RC = \frac{C^{ON} - C^{OFF}}{C^{ON}} \tag{3.1}$$

C^{ON} is ring oscillator's count when the test pattern generator is enabled (ON), and the transmitter is carrying the serial value shifted out by the 32-bit test pattern register. C^{OFF} is the ring oscillator's count when the test pattern generator is disabled (OFF), and the transmitter is carrying logic '0'. The transmitter alternates between the ON phase and the OFF phase for every 21ms measurement duration. Unless otherwise mentioned, the transmitter carries logic '1' during ON phase, and logic '0' during the OFF phase.

3.6 Results and Analysis

The experimental results are briefly summarized below. A detailed discussion of each experiment is provided in the following subsections.

- **Location Independence**: The information leakage was characterized at nine different locations for the Cyclone-IV and Stratix-V FPGAs. It has been proved

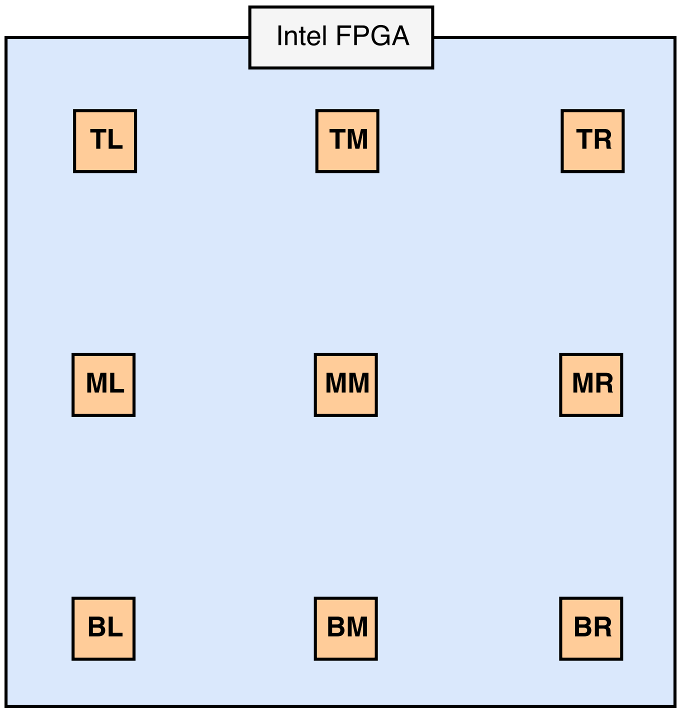


Figure 3.10: Information leakage was verified at different locations on Stratix-V and Cyclone-IV FPGAs. (T=Top, M=Middle, B=Bottom, L=Left, R=Right)

that the information leakage is present at all nine locations across multiple routing elements in Cyclone-IV and Stratix-V FPGAs. The information leakage is strong in the shorter C4 routing elements.

- **Transmitter/Receiver Length**: The information leakage increases linearly with the length of transmitter/receiver in C4 routing elements in Cyclone-IV and Stratix-V FPGAs. The information leakage is weak in longer C14/C16 routing elements.
- **Transmitter Duty Cycle**: The information leakage increases linearly with the duty cycle of the transmitter.
- **Transmitter Frequency**: In the majority of the experiments, the information leakage decreases with the transmitter frequency. However, at some transmitter frequencies the ring oscillator shows phase locking behavior with the transmit-

ter. When the ring oscillator is locked with the transmitter, the ring oscillator's count value remains almost constant between different trials.

- **Measurement Duration**: The information leakage value (ΔRC) is more noisy at short measurement durations because the ring oscillator counter has less time to capture the changes in the ring oscillator's frequency.
- **Multiple Transmitters**: The information leakage is strong when a transmitter is immediately adjacent to the receiver. The information leakage is negligible at higher adjacency levels.
- **Impact of Temperature**: As the operating temperature is increased, the ring oscillator's frequency decreases because the electron mobility decreases with temperature. In Cyclone-IV FPGAs it has been observed that the ring oscillator's count value becomes more stable with an increase in operating temperature.

3.6.1 Location Independence

To test if the information leakage is present at all locations on the chip, information leakage is measured at nine distinct chip locations, as illustrated in Figure 3.10. The measurement circuit described in Section 3.4 along with a single transmitter and a single receiver is used for the experiment. The transmitter and receiver are each implemented using a single global routing element. They are manually placed and routed, and a separate bit-stream file is generated for each location. For each FPGA device, several types of routing elements are characterized. The transmitter sends a static logic '1' for 21ms, and then transmits logic '0' for the next 21ms.

Figure 3.11 shows the average ΔRC value on two identical Cyclone-IV FPGA boards. The routing elements C4, R4, and C16 are characterized. These results confirm that the information leakage is present at almost all locations on the chip.

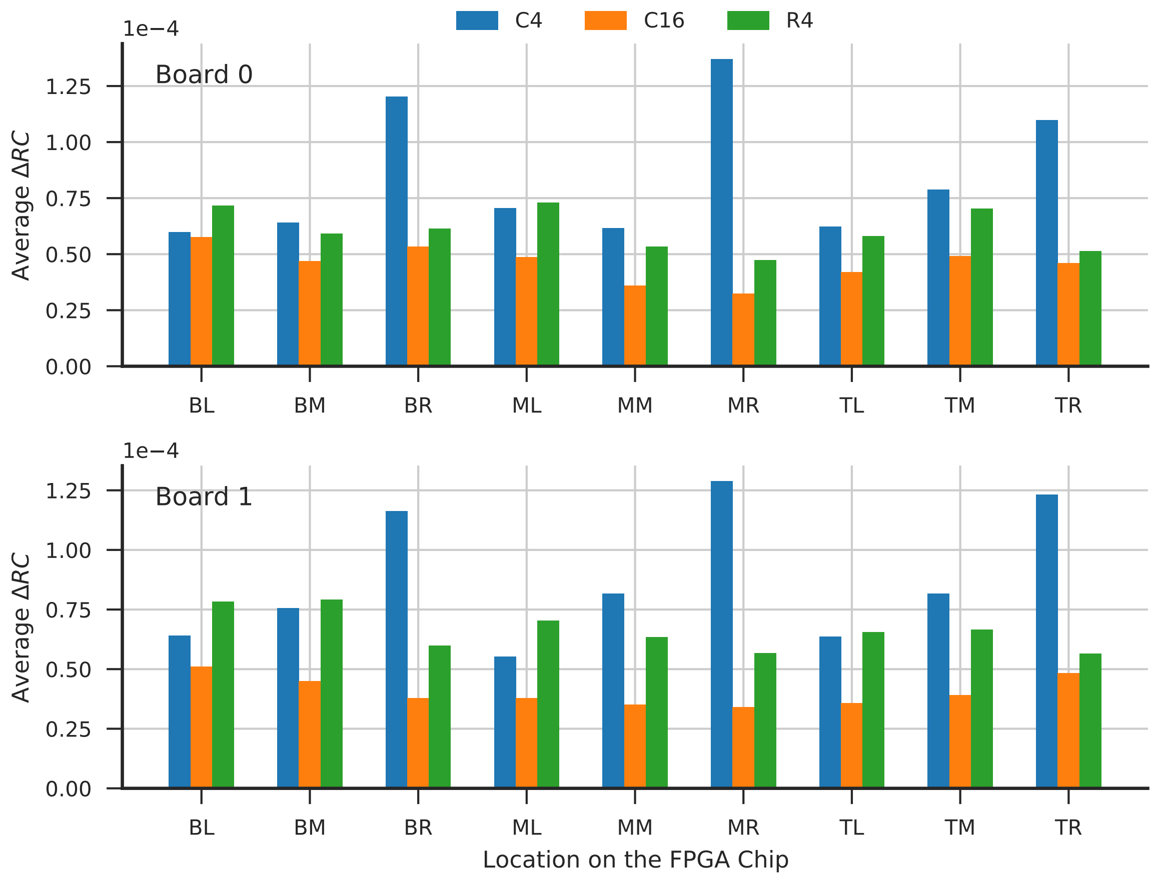


Figure 3.11: Relative count difference of the ring oscillator due to values of the adjacent transmitter (see Eq. 3.1). For this experiment, both the transmitter and the receiver include a single global routing element. Across various locations and FPGAs, driving a 1 onto the transmitter causes a receiver speed-up of the order of 0.01% (1 part per 10,000).

Leakage is higher in the shorter C4 and R4 routing elements, which constitute the majority of the global routing resources in the FPGA device. Although there are some variations in the ΔRC value at different locations on the chip, the ΔRC values at a given location are nearly identical across Board 0 and Board 1. Therefore, the variation in ΔRC at different locations can be attributed to the changes in the FPGA silicon layout, and not to manufacturing or silicon variations in general. Figure 3.12 shows information leakage at different locations on the Stratix-V FPGA device. Four different routing elements are characterized, C4, C14, R3, and R6. The results are similar to the one observed in the Cyclone-IV device. This confirms that the

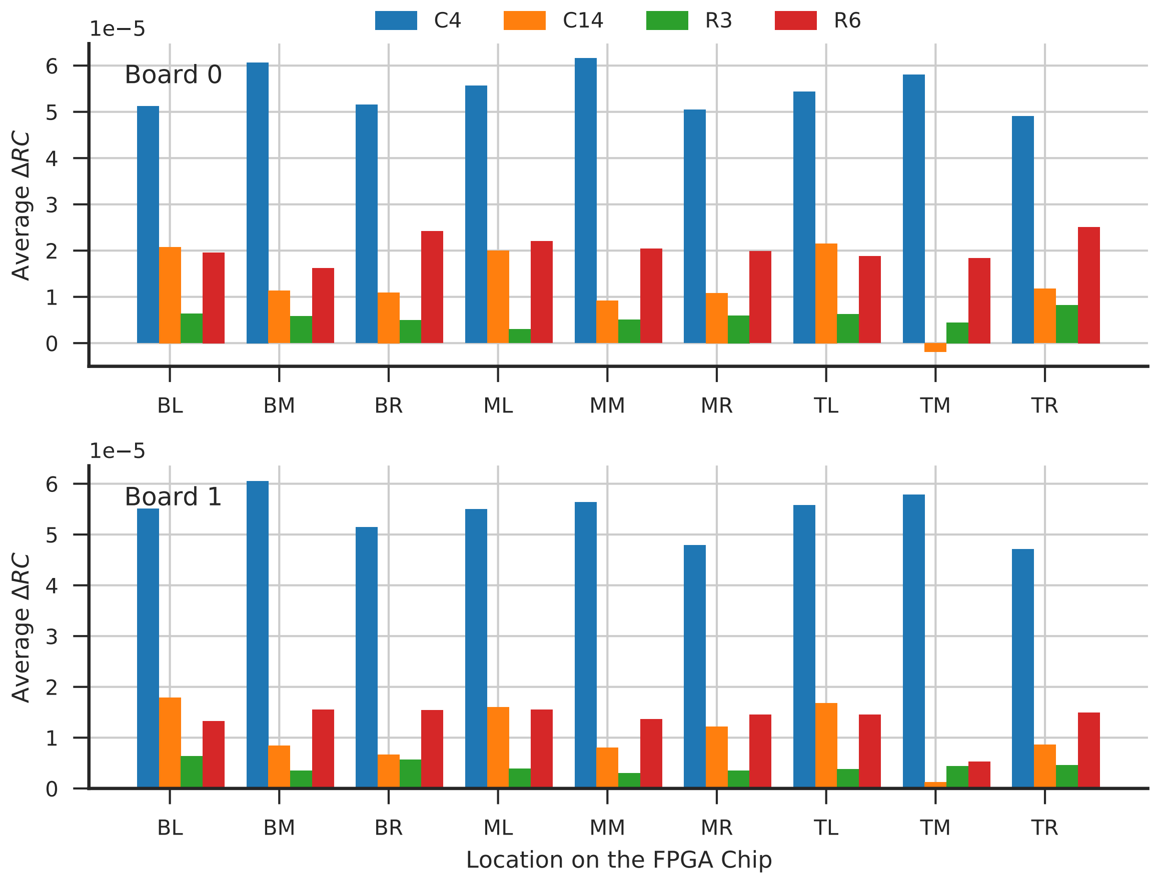


Figure 3.12: Repeating the experiment from Figure 3.11 on Stratix-V devices on two identical boards produces comparable results.

information leakage phenomenon is present in multiple Intel FPGA device families. The average ΔRC value in Stratix-V FPGA device is almost half the value observed in the Cyclone-IV FPGA device. The C4 routing element has the highest ΔRC value compared to other routing elements. The ΔRC values at a given location are identical in Board 0 and Board 1. Although the relative count difference value is small, (less than 0.01%), it is still measurable. This small change in the ring oscillator frequency is sufficient to perform side channel attacks similar to differential power analysis as demonstrated by Ramesh *et al.* [25]. The information leakage may be due to small changes in the ring oscillator's driver's voltage when the transmitter is carrying logic '1'.

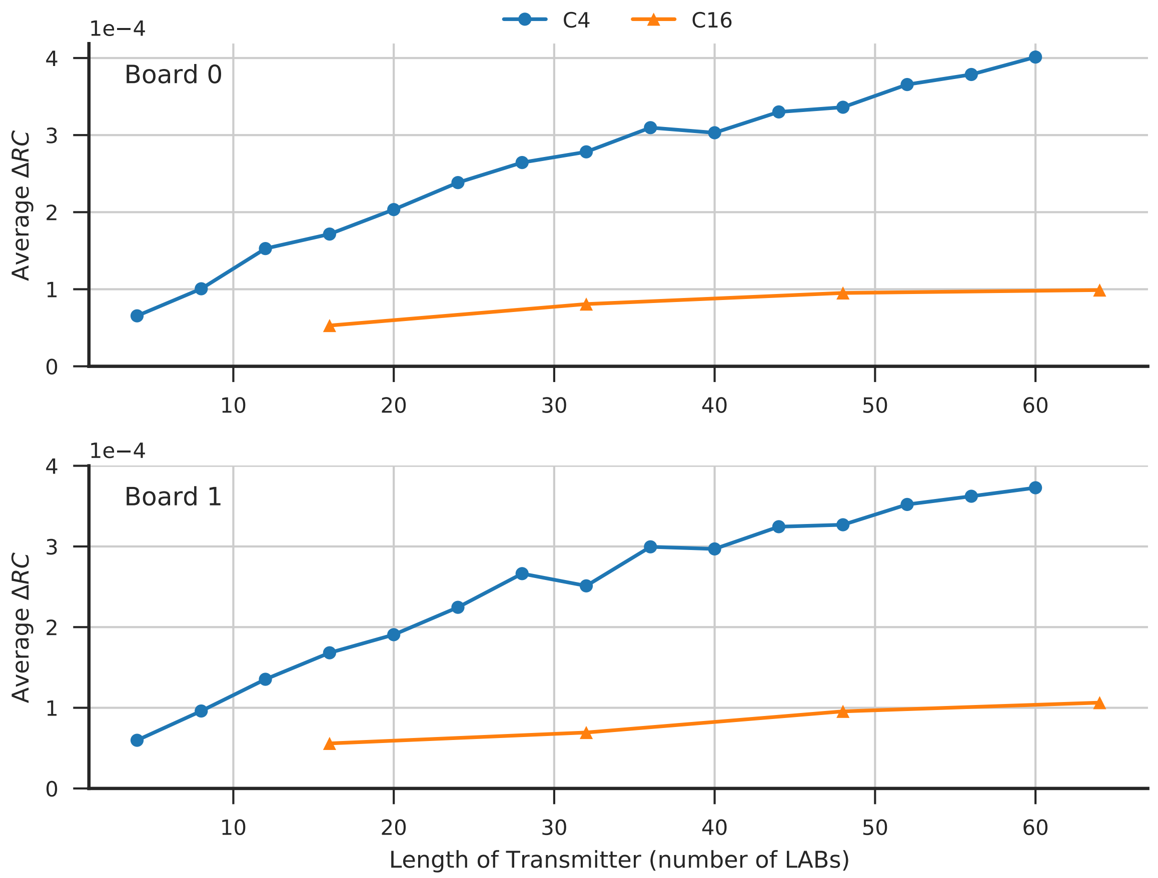


Figure 3.13: Relative count differences increase with different transmitter/receiver lengths (in terms of number of LABs). The experiment uses a transmitter/receiver pair at the bottom left of the Cyclone-IV EP4CE115F29 FPGA.

3.6.2 Length of Transmitter/Receiver

In this experiment the effect of transmitter and receiver length on the relative count difference is analyzed. The transmitter's and receiver's lengths are increased by stacking multiple routing elements on top of each other. Apart from the transmitter/receiver length, the experimental setup is same as in the location independence experiment. The transmitter and receiver are placed in the bottom left location on the chip. In the Cyclone-IV and Stratix-V FPGAs, the column routing elements can be vertically stacked on top of each other, i.e. a routing element can be connected to another routing element of the same index at the endpoint. For example, the routing element `C4:X5Y4Z0I0` can be connected to `C4:X5Y8Z0I0` in a Cyclone-IV device.

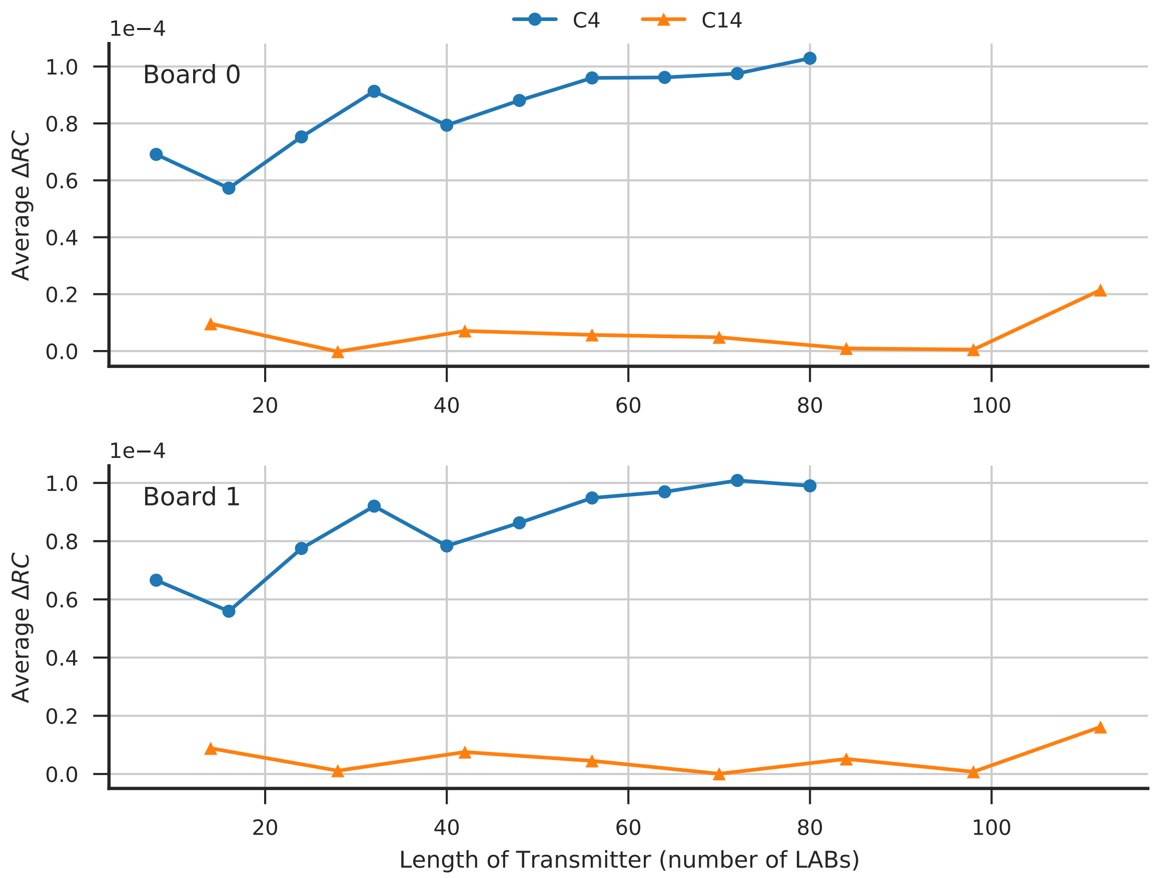


Figure 3.14: Impact of transmitter/receiver length on relative count difference in a Stratix-V 5SGXEA7K2F40C2 FPGA. The relative count differences in the Stratix-V FPGA are much lower compared to the Cyclone-IV FPGA (Figure 3.13). For the C14 routing element, ΔRC remains relatively flat.

The routing element `C4:X5Y4Z0I0` with index `I0` spans vertically from `Y=4` till `Y=8`, at `Y=8`, it can be connected to another routing element (`C4:X5Y4Z0I0`) with index `I0`. However, the horizontal routing elements are rotated as they move through the switch box, therefore, they cannot be stacked on top of each other. Therefore, in this experiment, only the column routing elements, C4, C14, and C16 are considered.

Figure 3.13 shows the average relative count difference for a Cyclone-IV FPGA. For the C4 routing elements, the average ΔRC value increases linearly with the transmitter/receiver length. However, for C16 routing elements, the rate of increase is much slower. Figure 3.14 shows the relative count change with the transmit-

Test pattern (32-bit)	Hamming Weight	Duty Cycle (%)	Frequency (MHz)
0x0000000F	4	12.5	3.125
0x000000FF	8	25.0	3.125
0x00000FFF	12	37.5	3.125
0x0000FFFF	16	50.0	3.125
0x000FFFFF	20	62.5	3.125
0x00FFFFFF	24	75.0	3.125
0x0FFFFFFF	28	87.5	3.125
0xFFFFFFFF	32	100.0	3.125

Table 3.1: Dynamic transmitter test patterns with the same frequency and increasing Hamming weights. The system clock is 100MHz.

ter/receiver length in the Stratix-V FPGA device. There is a small increase in relative count difference with the transmitter/receiver length for the C4 routing elements. However, for the C14 routing elements, the transmitter/receiver length doesn't have a significant impact on the relative count difference. The number of long routing elements (C14/C16) in the column routing channel is much less compared to the shorter C4 routing elements. It is possible that the longer routing elements are sparsely placed with a higher degree of spacing or isolation to reduce delay and coupling, which might explain the weaker information leakage in long interconnect wires.

3.6.3 Transmitter Duty Cycle

In this experiment, the impact of transmitter duty cycle (on time) on the information leakage is analyzed. The test pattern generator has a 32-bit register which can be programmed to transmit different test pattern signals through the transmitter. The test pattern generator circuit serializes the 32-bit pattern register by shifting one bit out on each clock cycle starting from the least significant bit into the transmitter wire. Table 3.1 lists different test patterns along with their Hamming weight, duty cycle, and frequency. The frequency of all test patterns is set to 3.125 MHz. Apart from the test pattern generation, all other experimental parameters are the same as

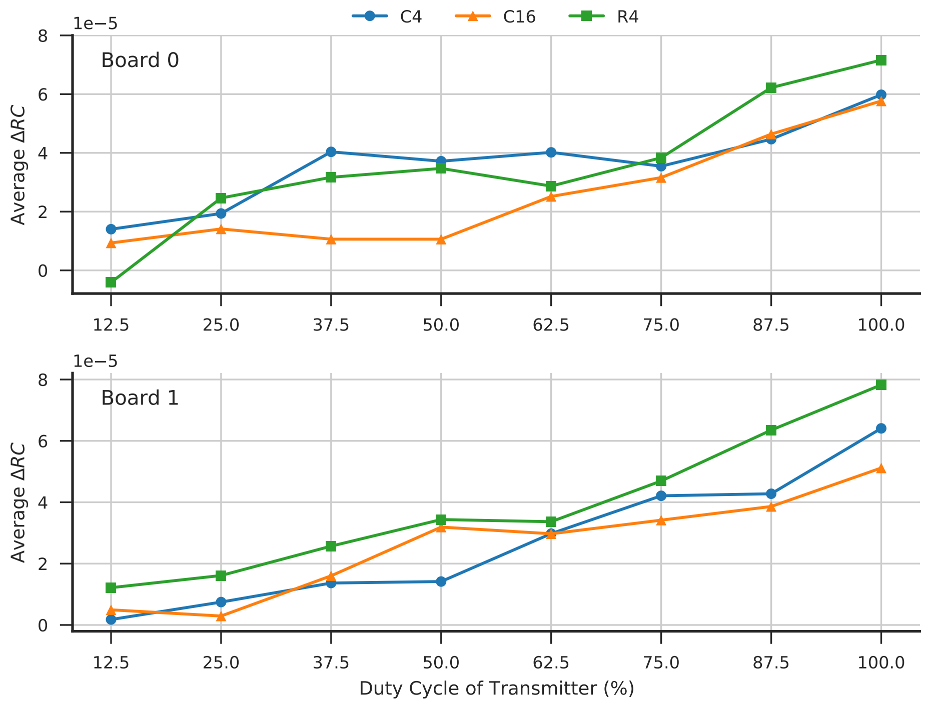


Figure 3.15: Impact of transmitter duty cycle on information leakage in a Cyclone-IV FPGA. The average ΔRC value increases with an increase in the duty cycle of transmitter.

the ones used in the location independence test (Section 3.6.1). The transmitter and the receiver are placed in the bottom left location on the chip.

Figure 3.15 and Figure 3.16 show the impact of transmitter duty cycle on the relative count difference in Cyclone-IV FPGA and Stratix-V FPGA devices, respectively. As the duty cycle of the transmitter is increased, the time duration of the transmitter carrying logic ‘1’ is increased. This increases the ring oscillator’s frequency count value (C^{ON}). Therefore, the average ΔRC value increases with the duty cycle. This observation is true for all routing elements across both FPGA boards. It can be observed that the C4 routing element has higher ΔRC compared to all other routing elements in the Stratix-V FPGA.

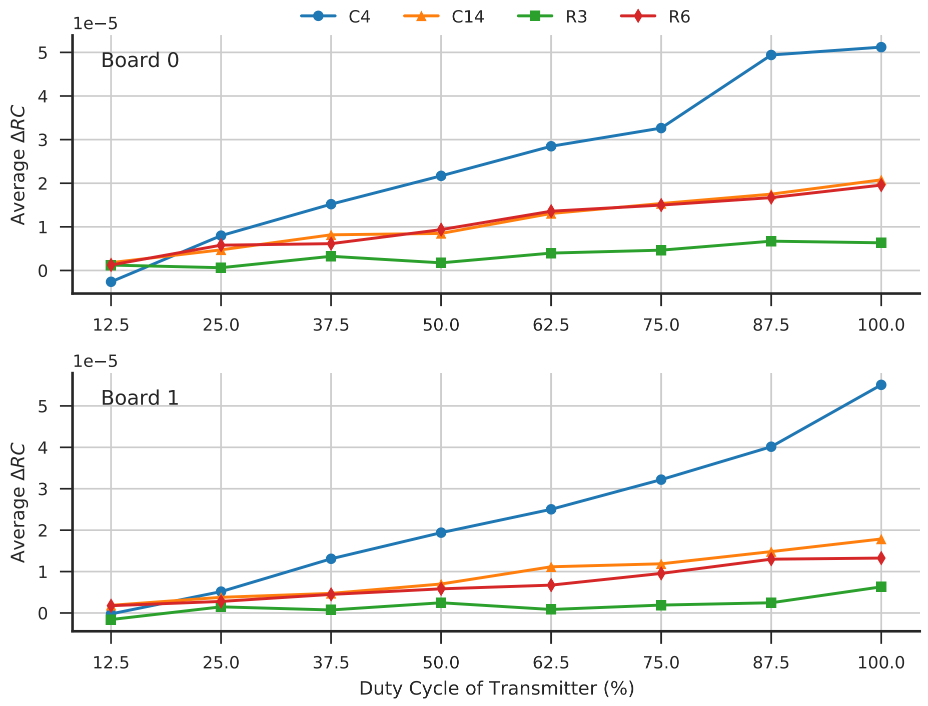


Figure 3.16: Repeating the experiment described in Figure 3.15 for a Stratix-V FPGA device. Similar to the Cyclone-IV, the ΔRC value increases with the duty cycle of the transmitter.

3.6.4 Transmitter Frequency

One of the main claims in the Xilinx characterization results [8] is that the relative count difference is only dependent on the duty cycle of the transmitter signal, and not on the switching rate of the transmitter. This claim is tested in Intel FPGAs by programming the test pattern generator register with values listed in Table 3.2. The test patterns have different frequency but fixed Hamming weight and duty cycle. The experimental setup is similar to the one used in the transmitter duty cycle experiment (Section 3.6.3).

Figure 3.17 shows the relative counter difference in the Cyclone-IV FPGAs. From the plots, it is clear that the ring oscillator's frequency is affected by the transmitter signal's frequency. In general, the ΔRC value decreases with an increase in transmitter frequency in the Cyclone-IV FPGA.

Test pattern (32-bit)	Hamming Weight	Duty Cycle (%)	Frequency (MHz)
`0xFFFF0000`	16	50.0	3.125
`0xFF00FF00`	16	50.0	6.250
`0xF0F0F0F0`	16	50.0	12.500
`0xC0C0C0C0`	16	50.0	25.000
`0xAAAAAAAA`	16	50.0	50.000

Table 3.2: ΔRC for dynamic transmitter test patterns with the same Hamming weight and increasing frequencies. The system clock is 100MHz.

Figure 3.18 shows the variation of ΔRC in Stratix-V FPGA devices. From the location independence experiments it is clear that information leakage is weak in R3, C14, and R6 routing elements. Therefore, the R3, C14, and the R6 routing elements are less sensitive to frequency changes in the transmitter. Since the information leakage is strong in the C4 routing element, the ΔRC value changes sharply with the transmitter's frequency. The variation in the ΔRC value for C4 appears to be random.

3.6.5 Measurement Duration

Figures 3.19 and 3.20 show the relative count difference for increasing measurement periods in Cyclone-IV and Stratix-V FPGAs, respectively. The experimental setup is the same as the one used in the location independence experiment. The transmitter and the receiver are placed in the bottom left location of the FPGA chip. For smaller measurement periods, the relative count difference is noisy, as the ring counter lacks sufficient cycles to count the frequency change. For higher measurement periods, the ΔRC value is more stable.

3.6.6 Multiple Transmitters

Table 3.3 compares the ΔRC value when the transmitters are placed at different adjacency levels from the receivers in the Cyclone-IV FPGA. The experimental setup is similar to the one used in location independence experiment. The transmitter and

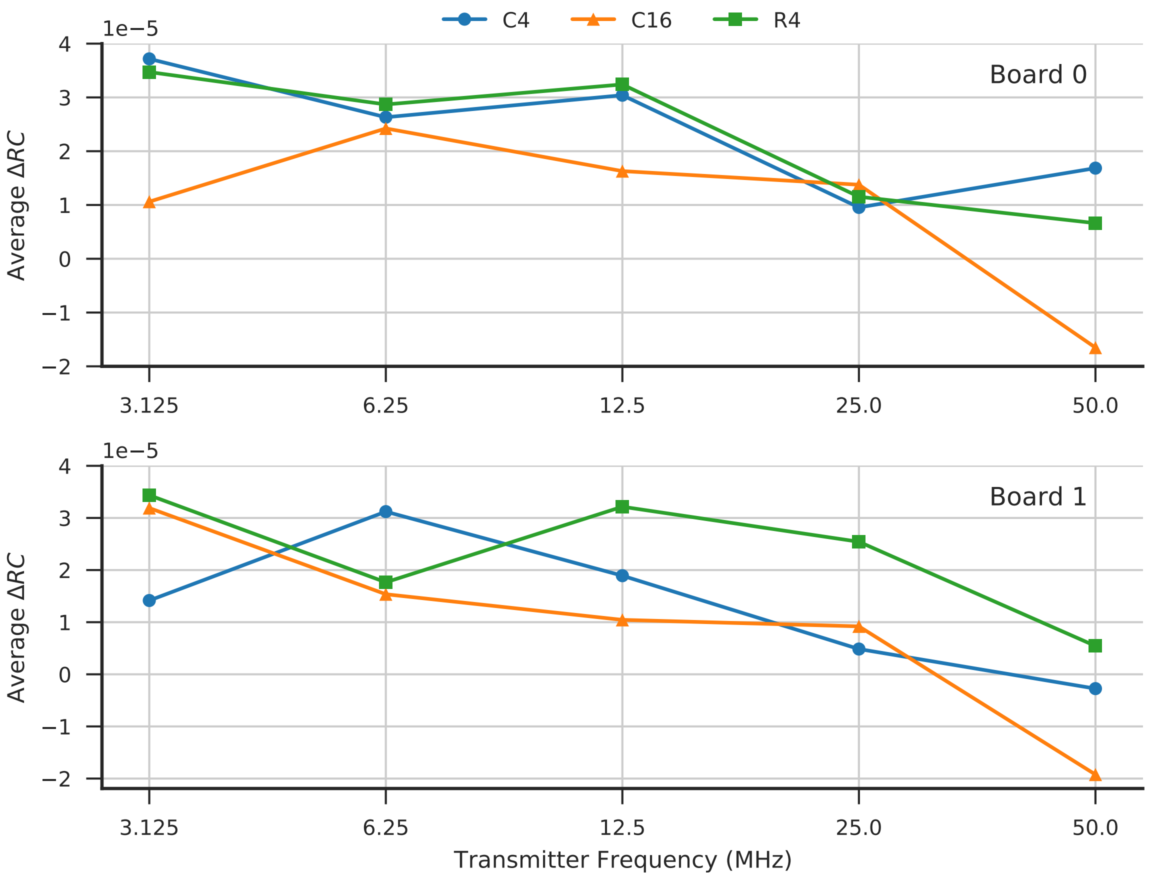


Figure 3.17: Impact of transmitter frequency on the relative count difference (ΔRC) in a Cyclone-IV FPGA. The ΔRC value decreases with an increase in transmitter frequency.

the receiver are placed in the bottom left location. The transmitters on either side of the receiver induce almost identical delay changes in the receiver. If both the left and the right transmitters are active, an additive effect is incurred on the delay change of the receiver. When a transmitter is at one or more adjacency level away from the receiver, there is no noticeable change in the ΔRC value.

3.6.7 Impact of Temperature on the Ring Oscillator's Frequency

The ring oscillator's frequency variations with temperature are characterized using the experimental setup shown in Figure 3.21. Test boards containing an Intel FPGA were placed inside a TestEquity 115A heating chamber. The ring oscillator's

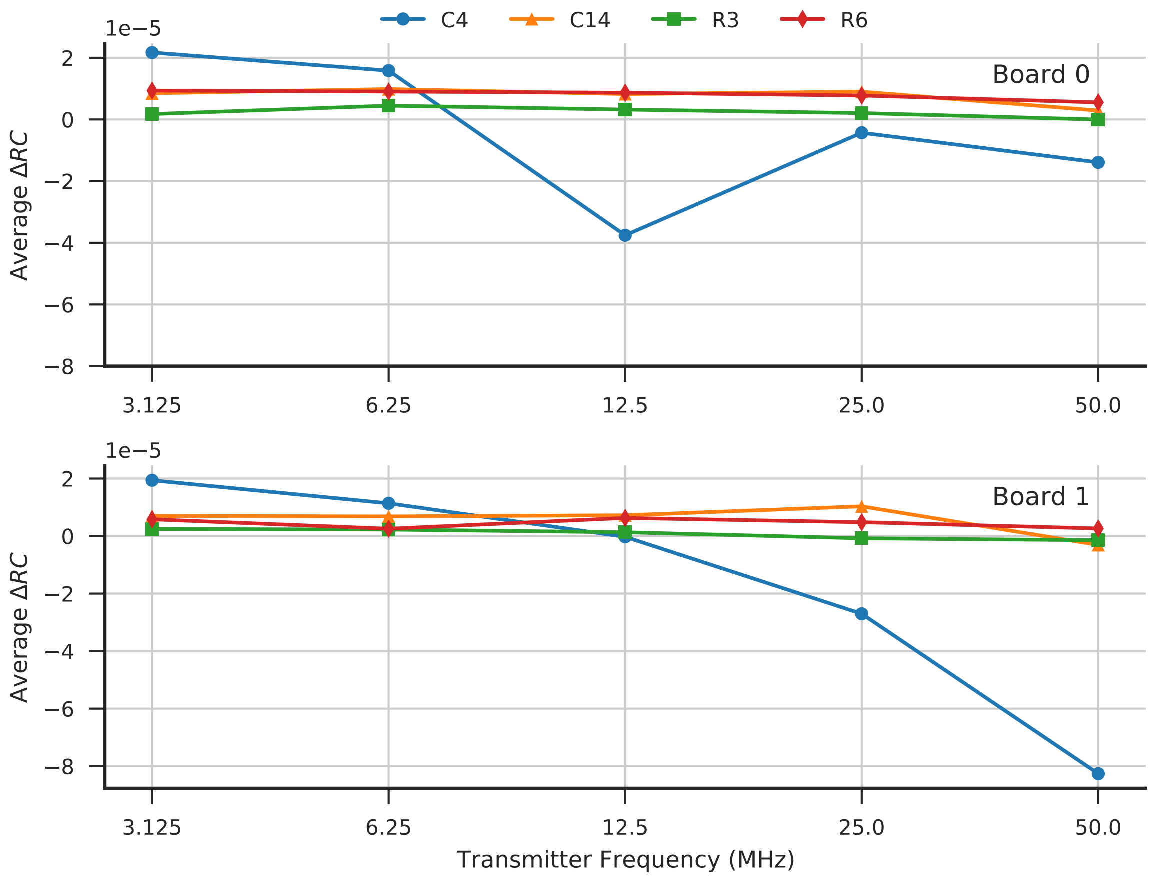


Figure 3.18: Relative count difference for different transmitter frequencies in Stratix-V FPGAs. The ΔRC value changes with the transmitter frequency. The ΔRC for C4 routing elements is inconsistent across boards. The relative change almost remains flat for other types of routing elements (C14, R3, and R6).

frequency was measured at temperatures from 25°C to 65°C. The experimental setup for characterization is the same as the one described in the location independence section 3.6.1. The FPGA boards were preheated for 10 minutes before running each experiment. This allowed the FPGA boards to be in thermal equilibrium with the heating chamber.

Figures 3.22 and 3.23 show the relative count difference change with temperature in Cyclone-IV and Stratix-V FPGAs. Since the ΔRC is a normalized value, it shows no correlation with respect to temperature changes in either FPGA device. Figures 3.24 and 3.25 show the ring oscillator's absolute count value at increasing temperature

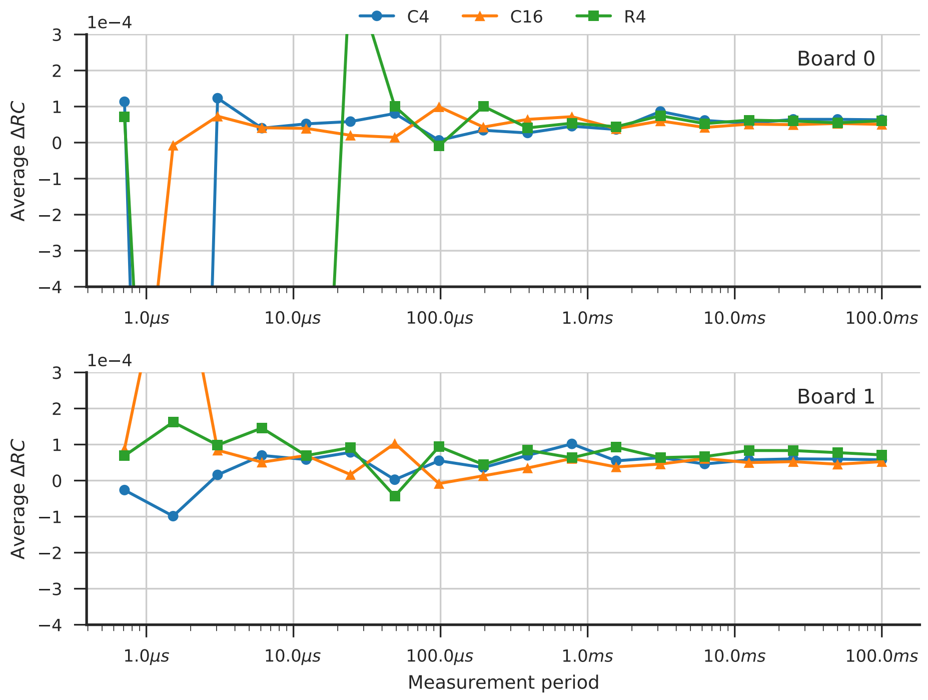


Figure 3.19: Relative frequency count with respect to measurement period in a Cyclone-IV FPGA device.

values when the transmitter is always carrying a logic ‘0’. In both FPGA devices, there is a small decrease in the ring oscillator’s count with temperature. This is expected behavior since electron mobility decreases with temperature, thus increasing the delay of the ring oscillator.

It has been discovered that the ring oscillator’s count value becomes less noisy at higher temperatures in Cyclone-IV FPGAs. Figure 3.26 shows the ring oscillator’s count when the transmitter is sending logic ‘1’ and logic ‘0’ at 25°C and 65°C. It can be observed that at 65°C the ring oscillator’s count value is more stable. To confirm this phenomenon, the standard deviation of the ΔRC value was plotted at increasing temperatures when the transmitter is carrying logic ‘0’. Figures 3.27 and 3.28 show the standard deviation of ΔRC with temperature. In the Cyclone-IV FPGA devices, standard deviation decreases linearly with temperature. This phenomenon can be theorized as follows. The FPGA die temperature is high due to

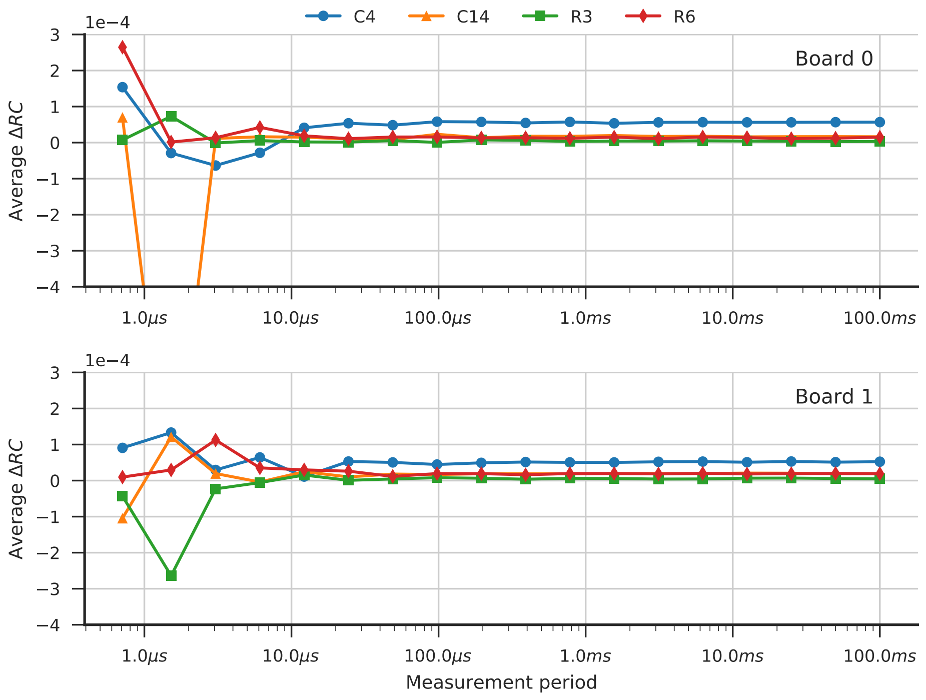


Figure 3.20: Average ΔRC value at increasing measurement periods in a Stratix-V device. The ΔRC value is more stable at higher measurement periods as the counter has sufficient time period to capture the ring oscillator's frequency change.

internal power dissipation. At lower operating temperatures, there is a large difference between the FPGA die temperature and the heating chamber temperature. These temperature fluctuations might cause the random changes in the ring oscillator's frequency at lower temperatures. As the heating chamber temperature is increased, the temperature gap between the FPGA die and the chamber decreases, this reduces the temperature fluctuations in the FPGA die, and thereby, reducing the variations in the ring oscillator's frequency. However, in Stratix-V FPGA devices, the standard deviation remains almost flat with respect to temperature in all types of routing elements. It can be theorized that in Stratix-V FPGAs, the FPGA die always remains at higher temperatures due to large internal power dissipation, and the temperature inside the die is less affected by the heating chamber temperature.

At some transmitter frequencies, the ring oscillator shows a type of locking behavior with the transmitter. Figure 3.29 shows the locking behavior for 12.5 MHz

Configuration	ΔRC (1e-4)
0 0 1 RO 0 0 0	0.965
0 0 0 RO 1 0 0	1.002
0 0 1 RO 1 0 0	2.000
1 1 0 RO 0 1 1	-0.041
0 0 0 RO 0 0 0	–

Table 3.3: Relative count results for different transmitter configurations in the Cyclone-IV FPGA. The transmitters and receivers have a length of two C4 wires. The final row is the baseline configuration against which the ΔRC values of the other configurations are evaluated.

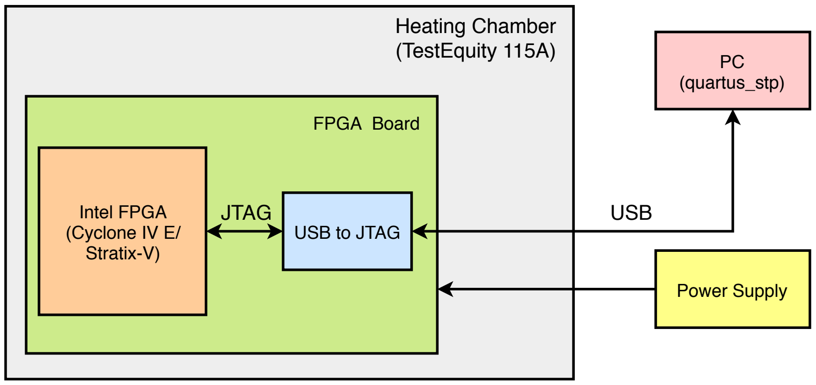


Figure 3.21: Experimental setup for characterizing information leakage at different operating temperatures.

transmitter frequency in Stratix-V FPGA board 1. The transmitter/receiver consist of four C4 routing elements located in the bottom left location on the FPGA chip. The experiment was conducted at 25°C temperature. From Figure 3.29, it can be observed that the ring oscillator's count values are locked around 2,887,500 when the transmitter is sending a toggling signal of 12.5 MHz. In contrast, the ring oscillator's count values are noisy when the transmitter is sending logic '0'. The ring oscillator's count value 2,887,500 corresponds to exactly 11 (2887500 / 21ms measurement duration / 12.5 MHz TX frequency) times the transmitter frequency. This means that the

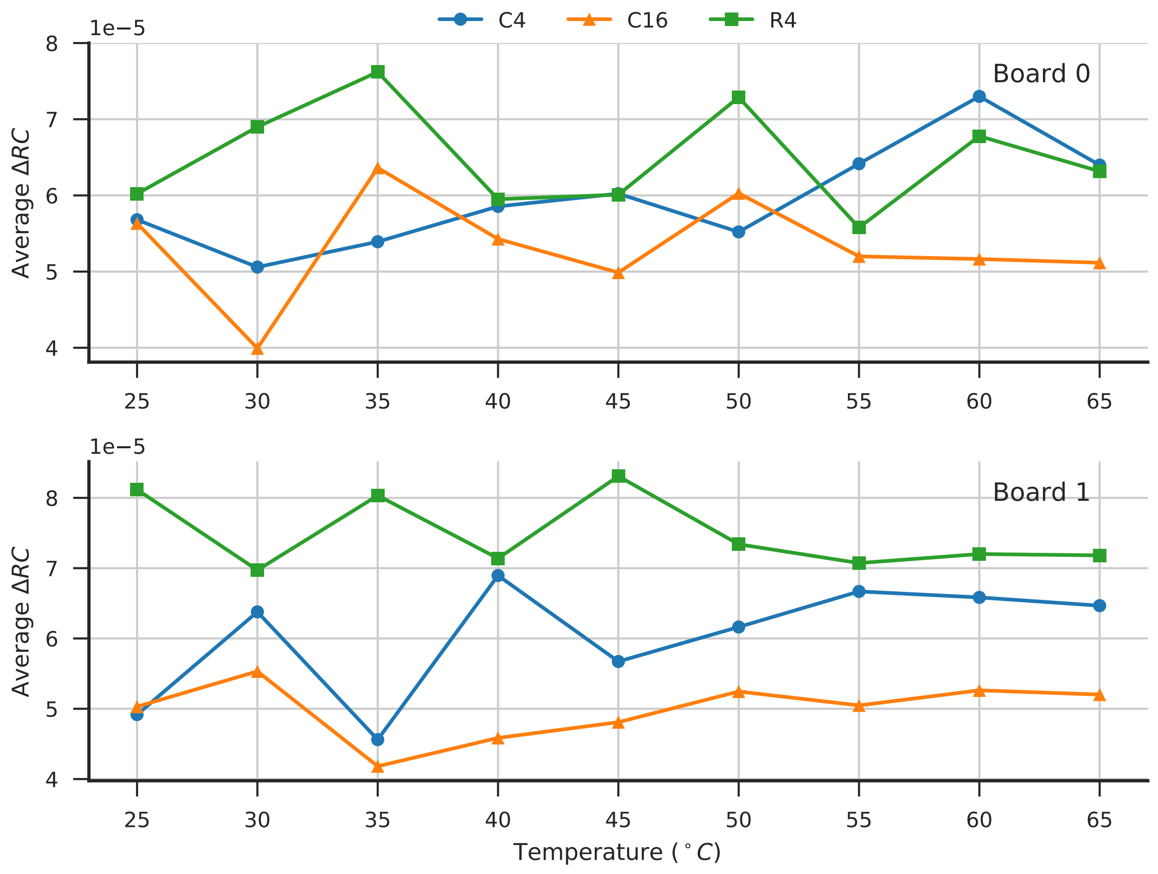


Figure 3.22: The average ΔRC shows no correlation with temperature in a Cyclone-IV FPGA.

ring oscillator (RO) oscillates exactly 11 times for each oscillation of the transmitter (TX) wire. To check if locking happens at other ring oscillator count values, the ring oscillator's frequency (count) is changed by increasing the temperature from 25°C to 65°C. The ring oscillator's count values at increasing temperatures are shown in Figure 3.30. The locking ring oscillator's count values are marked on the y-axis. The ring oscillator's frequency (count) decreases with temperature. However, the transmitter (TX) stays at 12.5 MHz as it is regulated by the clock.

Initially, at 25°C, the ring oscillator's count is locked around 2,887,500. As the temperature is increased, the ring oscillator loses the lock around 30°C temperature. The ring oscillator regains the lock for a short duration at six other ring oscillator count values (Figure 3.30). Figure 3.31 is a magnified version of Figure 3.30 showing

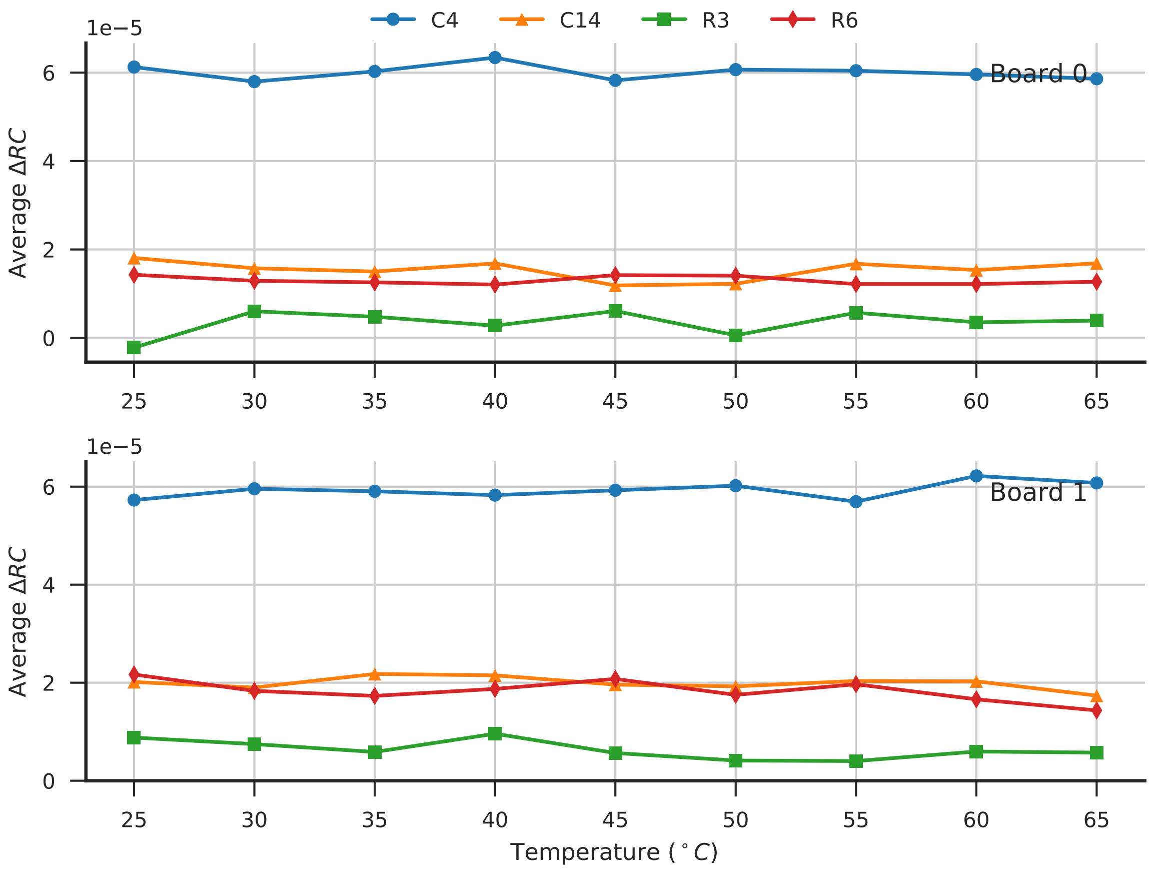


Figure 3.23: There is no change in the average ΔRC with temperature in a Stratix-V FPGA.

the ring oscillator locking around a 2,782,500 count value. In this case, the ring oscillator has exactly 53 oscillations in the time that the transmitter has 5 oscillations. Table 3.4 lists all the ring oscillator's count values at which the locking behavior was observed along with the temperature span duration for which the ring oscillator was locked. From Table 3.4, it can be observed that the locking behavior is strong when the transmitter's frequency is an integer multiple (11) of the ring oscillator's frequency. The locking behavior is also observed at non-integer multiples of the transmitter's frequency. In these cases the ring oscillator and the transmitters are periodic to each other. The locking behavior might be due to coupling between the transmitter and the receiver.

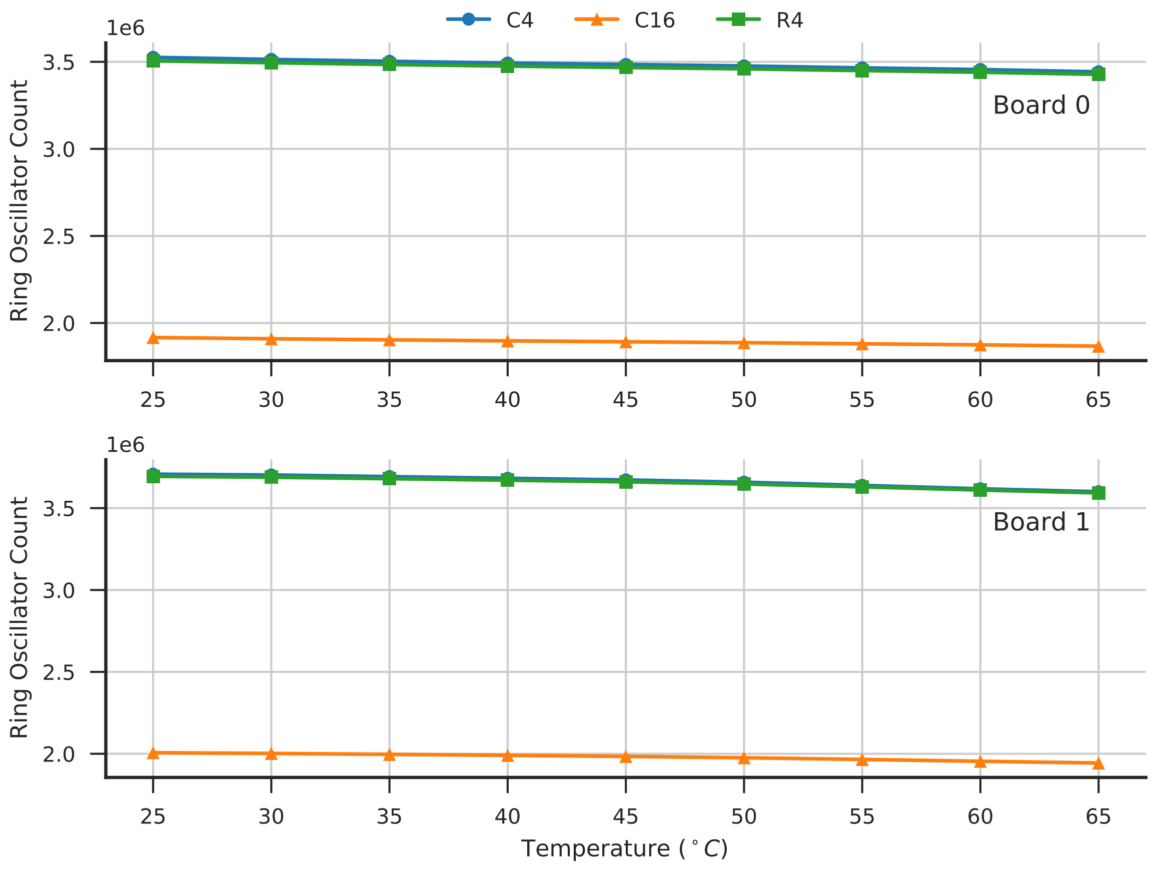


Figure 3.24: The ring oscillator's count decreases with temperature in a Cyclone-IV FPGA. The electron mobility decreases at higher temperatures, thereby reducing the ring oscillator's frequency.

3.7 Limitations

The information leakage observed in this study is only present between the immediately adjacent interconnect wires. If an attacker does not have access to the target user design, it would very be difficult to locate the transmitter's interconnect wires in the FPGA device. The information leakage is quite noisy, and the ring oscillator's frequency changes with temperature, voltage, and other operating parameters. It is difficult to statically predict the logic level of the transmitter using the absolute ring oscillator count value. A successful attack [25] requires sophisticated statistical analysis on large number of measurements. It has been shown that the ΔRC values become less predictable with increased transmitter frequency. Therefore, it might be

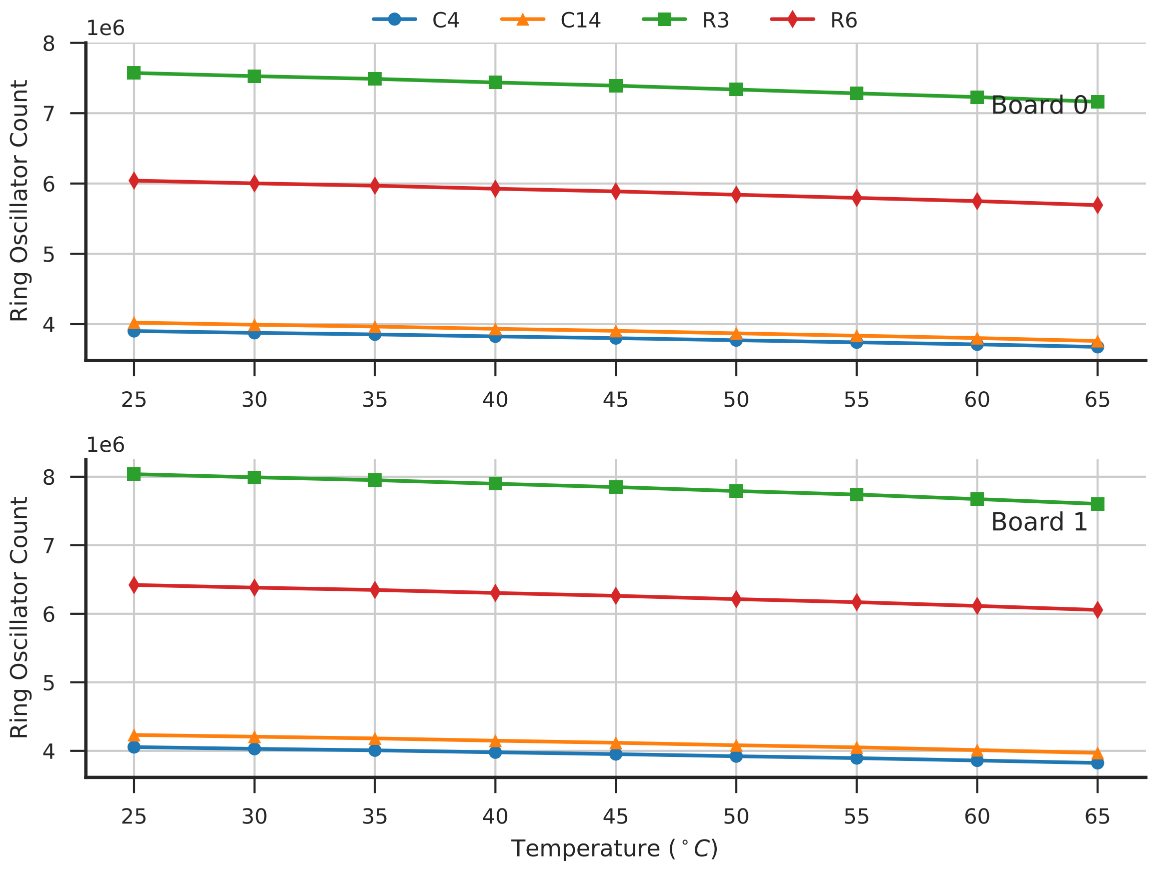


Figure 3.25: The experiment illustrated in Figure 3.24 repeated for Stratix-V FPGAs. There is a small decrease in the ring oscillator's frequency with temperature.

difficult to attack user designs operating at higher frequencies, although work at the University of Massachusetts is continuing in this area.

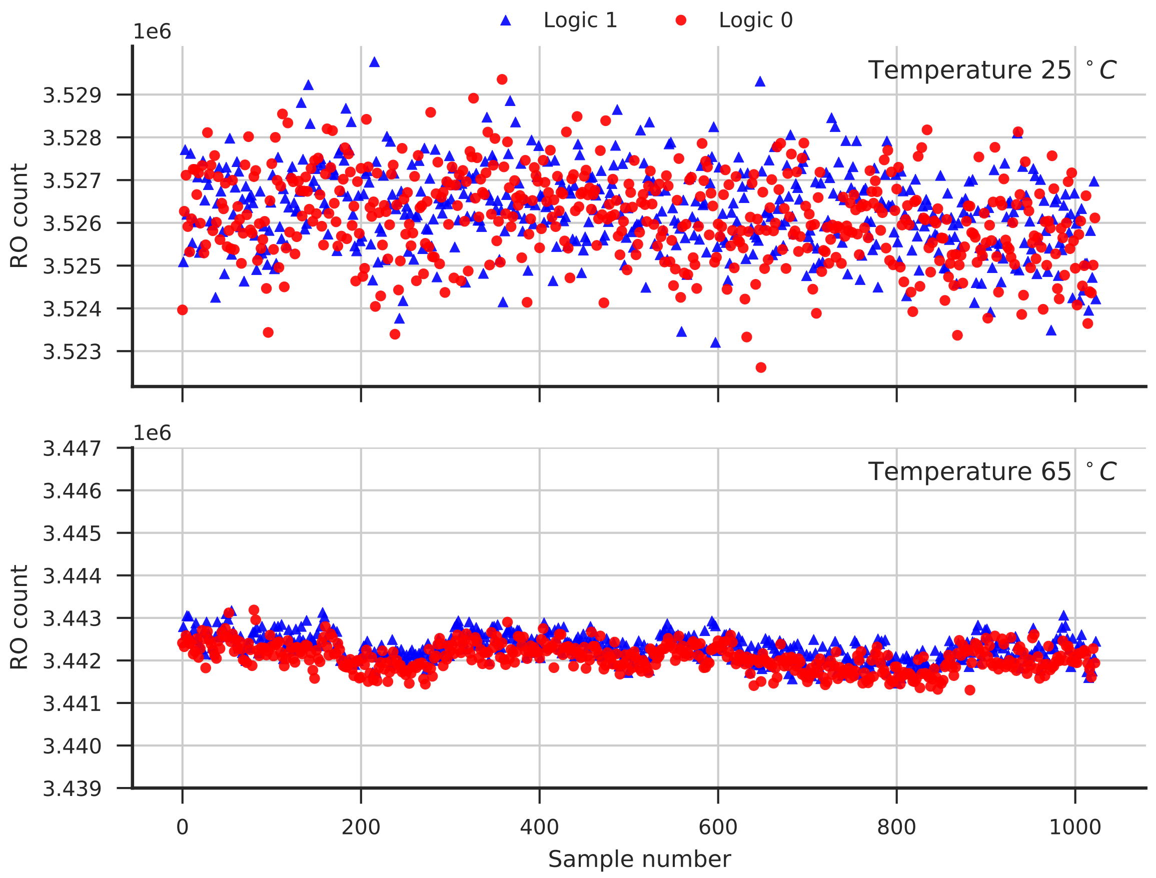


Figure 3.26: Ring oscillator count in the Cyclone-IV FPGA at 25°C and 65°C when the transmitter is sending logic '1' and logic '0'. The ring oscillator's count is more stable at 65°C.

RO count in 21ms	RO frequency (MHz)	Ratio of RO frequency to TX frequency (12.5 MHz)	Temperature span over which RO remains locked ($^\circ C$)
2,887,500	137.5000	11/1	4.20
2,822,000	134.3810	≈ 43/4	0.20
2,812,500	133.9286	75/7	0.18
2,800,000	133.3333	32/3	0.18
2,782,500	132.5000	53/5	0.58
2,756,250	131.2500	21/2	0.15
2,737,500	130.3571	73/7	0.18

Table 3.4: Ratio of ring oscillator frequency to transmitter frequency at different ring oscillator count values.

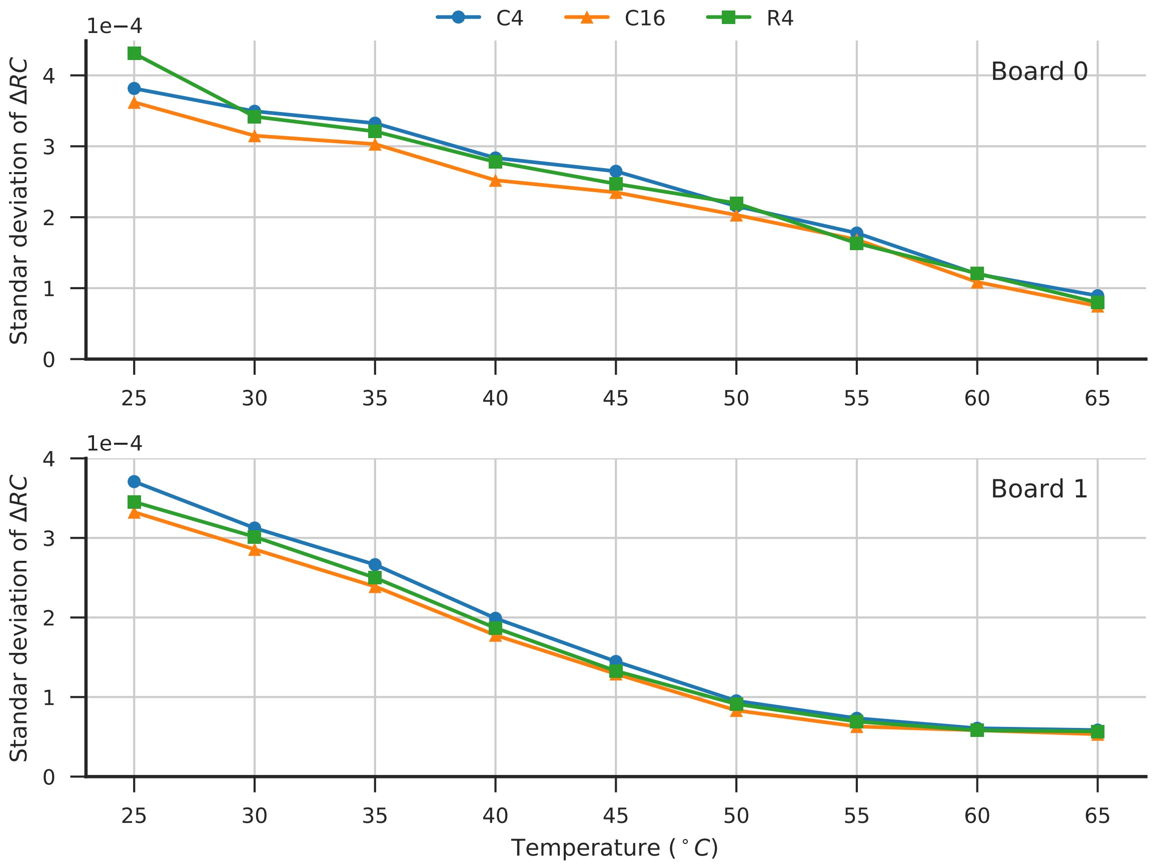


Figure 3.27: The standard deviation of the ΔRC decreases with temperature in two Cyclone-IV FPGA devices.

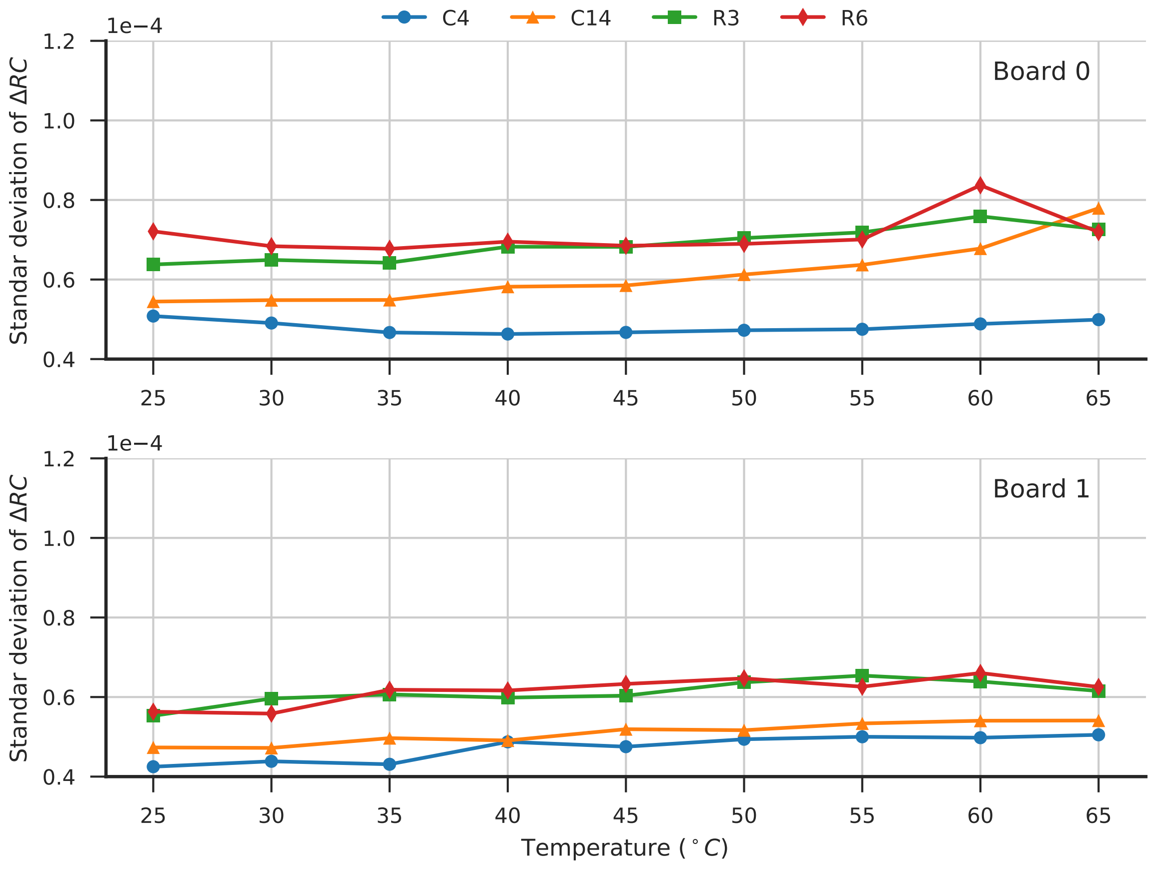


Figure 3.28: The standard deviation in ΔRC is unaffected by temperature changes in two Stratix-V FPGA devices.

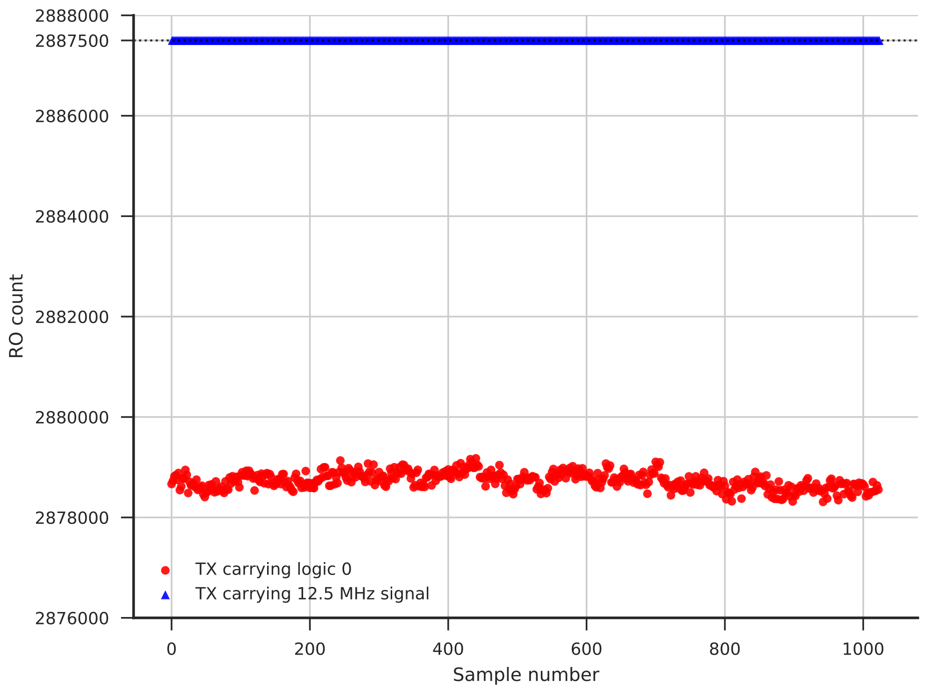


Figure 3.29: Ring oscillator locking at 12.5 MHz in a Stratix-V FPGA. The transmitter/receiver are four C4 routing elements long. The ring oscillator's count values are not noisy when the transmitter is carrying a 12.5 MHz frequency signal.

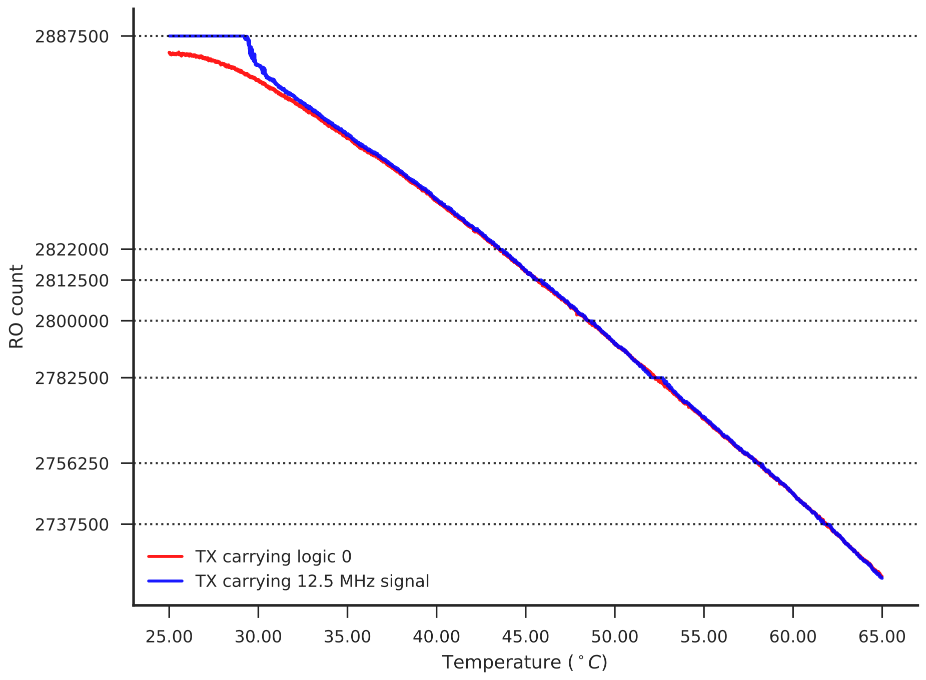


Figure 3.30: The ring oscillator's frequency is changed by increasing the operating temperature from 25°C to 65°C. The ring oscillator shows locking behavior at seven different ring oscillator count values.

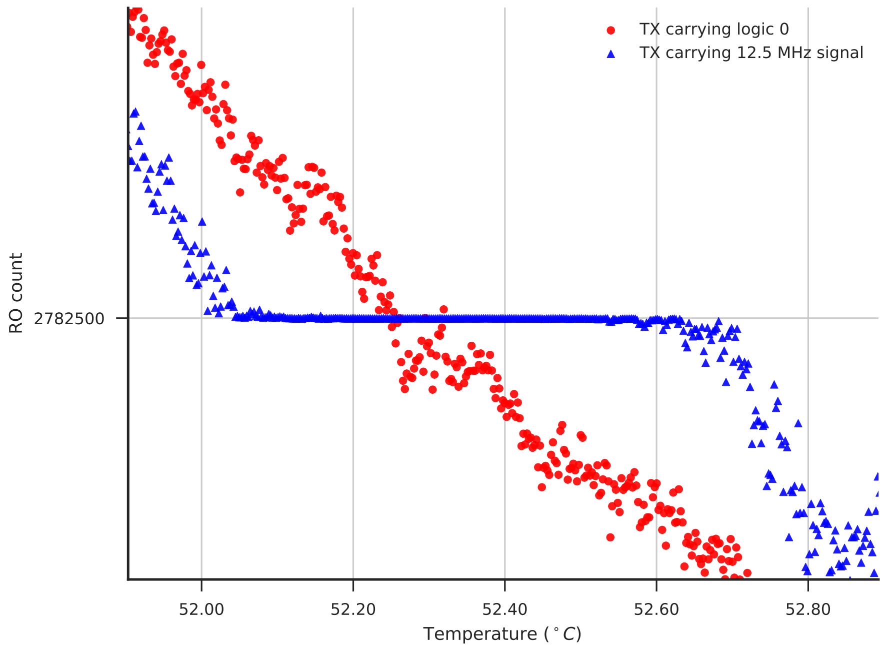


Figure 3.31: Figure 3.30 zoomed in around the 2,782,500 RO count value. The ring oscillator's count is highly stable when the transmitter is sending a 12.5 MHz signal.

CHAPTER 4
CONCLUSION

This thesis work addressed two important challenges in modern FPGAs: on-chip communication and security. For the on-chip communication topic, a new iterative routing algorithm based on the Pathfinder algorithm is developed to route TDM flits in a hybrid FPGA NoC architecture. The proposed routing algorithm showed an 11% improvement in bandwidth for unicast and multicast traffic patterns. As FPGAs continue to increase in size and complexity, providing high speed, low power, scalable interconnect will be a challenge. FPGA NoCs are expected to be one of the major architectural enhancements in next generation FPGA devices. The design of efficient NoC routing algorithms is crucial to extract the best possible performance from FPGA NoCs. Since most industrial FPGA CAD tools already use the Pathfinder algorithm, our routing algorithm can be easily integrated into existing CAD tools. Future work could consider optimizing the routing for multi-fanout connections by using Steiner Trees [12].

On the security aspect, a new vulnerability is identified in Intel Stratix-V and Cyclone-IV FPGAs. It has been confirmed that there is information leakage from long interconnect wires in these devices. It has been discovered that information leakage occurs at almost all locations on the FPGA chip, and across several types of routing elements. The impact of transmitter length, switching rate, and duty cycle on the information leakage has been characterized. It has been shown that information leakage is dominant in the nearest adjacent routing elements. The effect of temperature on the ring oscillator's frequency and the information leakage has also been characterized. Recently, the use of FPGAs in cloud infrastructures has gained significant attention. With increased interest in big data, machine learning, and multimedia applications, one can only expect this trend to grow in the future. Although current FPGA cloud infrastructures do not support multi-tenancy, there are strong incentives for providing multi-tenancy in the near future. A multi-tenant FPGA cloud infrastructure could dramatically reduce the cost of FPGAs for clients by efficiently sharing FPGA resources between multiple users. Therefore, it is crucial to identify and isolate security threats before deploying FPGA multi-tenancy in cloud infrastructures.

A successful attack has already been demonstrated [25] on an AES core using our characterization results. Future work could consider identifying attacks on other types of applications. New countermeasures should be developed to isolate user designs from these security threats. The countermeasures could be architectural changes or silicon layout improvements in new FPGAs. For existing FPGA devices, new CAD flows could be explored that mitigate the likelihood of an attack.

BIBLIOGRAPHY

[1] Amazon F1 web site. `https://aws.amazon.com/ec2/instance-types/f1/`.

[2] Abdelfattah, Mohamed S., and Betz, Vaughn. Networks-on-Chip for FPGAs: Hard, Soft or Mixed? *ACM Trans. Reconfigurable Technol. Syst. 7*, 3 (Sept. 2014), 20:1–20:22.

[3] Betz, V., and Rose, J. Directional bias and non-uniformity in FPGA global routing architectures. In *Proceedings of International Conference on Computer Aided Design* (Nov 1996), pp. 652–659.

[4] Carle, Thomas, Djemal, Manel, Potop-Butucaru, Dumitru, and De Simone, Robert. Static mapping of real-time applications onto massively parallel processor arrays. In *14th International Conference on Application of Concurrency to System Design* (Hammamet, Tunisia, June 2014), Proceedings ACSD 2014.

[5] Caulfield, Adrian M., Chung, Eric S., Putnam, Andrew, Angepat, Hari, Fowers, Jeremy, Haselman, Michael, Heil, Stephen, Humphrey, Matt, Kaur, Puneet, Kim, Joo-Young, Lo, Daniel, Massengill, Todd, Papamichael, Kalin Ovtcharov Michael, Woods, Lisa, Lanka, Sitaram, Chiou, Derek, and Burger, Doug. A cloud-scale acceleration architecture. In *ACM/IEEE International Symposium on Microarchitecture* (2016).

[6] Evain, Samuel, and Diguet, Jean-Philippe. Efficient Space-time NoC Path Allocation Based on Mutual Exclusion and Pre-reservation. In *Proceedings of the 17th ACM Great Lakes Symposium on VLSI* (New York, NY, USA, 2007), GLSVLSI '07, ACM, pp. 457–460.

[7] F. Schellenberg et al. An Inside Job: Remote Power Analysis Attacks on FPGAs. In *DATE* (Mar. 2018).

[8] Giechaskiel, Ilias, Rassmussen, Kasper B., and Eguro, Ken. A Robust Covert Channel on FPGAs Based on Long Wire Delays. *CoRR abs/1611.08882v2* (2017).

[9] Giechaskiel, Ilias, Rassmussen, Kasper B., and Eguro, Ken. Leaky Wires: Information Leakage and Covert Communication Between FPGA Long Wires. In *AsiaCCS* (June 2018).

[10] Iakymchuk, T., Nikodem, M., and Kpa, K. Temperature-based covert channel in FPGA systems. In *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)* (June 2011), pp. 1–7.

[11] Jiang, N., Balfour, J., Becker, D. U., Towles, B., Dally, W. J., Michelogiannakis, G., and Kim, J. A detailed and flexible cycle-accurate Network-on-Chip simulator. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (April 2013), pp. 86–96.

[12] Kahng, A. B., and Robins, G. A new class of iterative Steiner tree heuristics with good performance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 11*, 7 (Jul 1992), 893–902.

[13] Kapre, N. Marathon: Statically-Scheduled Conflict-Free Routing on FPGA Overlay NoCs. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)* (May 2016), pp. 156–163.

[14] Kapre, N., Mehta, N., deLorimier, M., Rubin, R., Barnor, H., Wilson, M. J., Wrighton, M., and DeHon, A. Packet Switched vs. Time Multiplexed FPGA Overlay Networks. In *2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (April 2006), pp. 205–216.

[15] Kelly, Shane, Zhang, Xuehui, Tehranipoor, Mohammed, and Ferraiuolo, Andrew. Detecting Hardware Trojans Using On-chip Sensors in an ASIC Design. *J. Electron. Test. 31*, 1 (Feb. 2015), 11–26.

[16] Krieg, C., Wolf, C., and Jantsch, A. Malicious LUT: A stealthy FPGA Trojan injected and triggered by the design flow. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (Nov 2016), pp. 1–8.

[17] Lecomte, M., Fournier, J. J. A., and Maurine, P. Thoroughly analyzing the use of ring oscillators for on-chip hardware trojan detection. In *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)* (Dec 2015), pp. 1–6.

[18] Liu, Tianqi, Dumpala, N. K., and Tessier, R. Hybrid hard NoCs for efficient FPGA communication. In *2016 International Conference on Field-Programmable Technology (FPT)* (Dec 2016), pp. 157–164.

[19] Lu, Z., and Jantsch, A. TDM Virtual-Circuit Configuration for Network-on-Chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16*, 8 (Aug 2008), 1021–1034.

[20] McMurchie, L., and Ebeling, C. PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs. In *Third International ACM Symposium on Field-Programmable Gate Arrays* (1995), pp. 111–117.

[21] Patil, Shivukumar B., Liu, Tianqi, and Tessier, Russell. A Bandwidth-Optimized Routing Algorithm for Hybrid FPGA Networks-on-Chip. In *International Symposium on Field-Programmable Custom Computing Machines* (Boulder, United States, Apr. 2018), pp. 1–4.

[22] PSG, Intel. Stratix V GX FPGA Development Kit. `https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-sv-gx-host.html`.

[23] PSG, Intel. Cyclone IV Device Handbook. `https://www.altera.com/en_US/pdfs/literature/hb/cyclone-iv/cyclone4-handbook.pdf`, 2010.

[24] PSG, Intel. Stratix V Device Handbook. `https://www.altera.com/en_US/pdfs/literature/hb/stratix-v/stratix5_handbook.pdf`, 2010.

[25] Ramesh, Chethan, Patil, Shivukumar B., Dhanuskodi, Siva Nishok, Provelengios, George, Pillement, Sébastien, Holcomb, Daniel, and Tessier, Russell. FPGA Side Channel Attacks without Physical Access. In *International Symposium on Field-Programmable Custom Computing Machines* (Boulder, United States, Apr. 2018), pp. 1–8.

[26] Shpiner, A., Kantor, E., Li, P., Cidon, I., and Keslassy, I. On the Capacity of Bufferless Networks-on-Chip. *IEEE Transactions on Parallel and Distributed Systems 26*, 2 (Feb 2015), 492–506.

[27] Sun, Ji, Bittner, Ray, and Eguro, Ken. FPGA Side-Channel Receivers. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (2011), pp. 267–276.

[28] Terasic. Altera DE2-115 Development and Education Board. `https://www.altera.com/solutions/partners/partner-profile/terasic-inc-/board/altera-de2-115-development-and-education-board.html`.

[29] Zhao, Mark, and Suh, G. Edward. FPGA-Based Remote Power Side-Channel Attacks. In *IEEE Symp. Security and Privacy* (May 2018), pp. 805–820.

[30] Ziener, Daniel, Baueregger, Florian, and Teich, Jurgen. Using the Power Side Channel of FPGAs for Communication. In *IEEE International Symposium on Field-Programmable Custom Computing Machines* (2010), pp. 1–8.