The Pennsylvania State University

The Graduate School

A TOOLCHAIN FOR ON-CHIP THERMAL MANAGEMENT OF FPGA DESIGNS

A Thesis in

Computer Science and Engineering

by

Rishab Gulati

© 2021 Rishab Gulati

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2021

The thesis of Rishab Gulati was reviewed and approved by the following:

Vijaykrishnan Narayanan
Distinguished Professor of Computer Science and Engineering
Thesis Advisor

Swaroop Ghosh
Associate Professor of Electrical Engineering

Chitaranjan Das
Distinguished Professor of Computer Science and Engineering
Department Head of Computer Science and Engineering

ABSTRACT

The FPGA industry has been growing at a staggering rate these past few years. Some of the new emerging technologies such as Artificial Intelligence (AI) and Machine Learning (ML) have fueled the FPGA market's growth. At the same time, the size of transistors has been shrinking, which results in ever-increasing chip density. Given this, the size and the complexity of designs synthesized on FPGAs has been increasing as well. These large and complex FPGA designs make it tougher to manage the on-chip thermal behavior, such as flattening the chip temperature. The industry has been calling for solutions related to a more efficient design process for managing the thermal behavior of FPGAs, and this proposed work provides precisely that. This work introduces an FPGA toolchain that would aid in managing the on-chip thermal behavior by working as a programmable heat sink for validation and testing. This toolchain consists of two components – an FPGA Heater design and a Pseudo Heat Sensor. The sole purpose of this FPGA heater design is to increase the on-chip temperature, with customizable parameters such as the clock frequency and the chip area utilization, controlling the amount of heat generated. On the other hand, the heat sensors are deployed on different parts of the FPGA chip, sensing the increase in temperature at those specific locations. This toolchain works directly on the fully implemented FPGA designs and thus provides a very accurate understanding of the on-chip thermal behavior of those designs.

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES vii

ACKNOWLEDGEMENTS viii

Chapter 1 Introduction 1

- 1.1 Motivation 1
  - 1.1.1 Motivation for the FPGA Heater Design 2
  - 1.1.2 Motivation for the Pseudo Heat Sensor 2
- 1.2 Contributions 3
- 1.3 Outline 4

Chapter 2 Background 5

- 2.1 Field Programmable Gate Array (FPGA) 5
- 2.2 Linear Feedback Shift Register (LFSR) as a Pseudo-Random Number Generator 6
- 2.3 Ring Oscillators as Temperature Sensors 7

Chapter 3 Design and Architecture of the System 8

- 3.1 FPGA Heater Design Overview 8
  - 3.1.1 LFSR based Heater Block 9
  - 3.1.2 Vendor Specific IPs 11
  - 3.1.3 Top-Level Heater Design 12
- 3.2 Pseudo Heat Sensor Design Overview 14
  - 3.2.1 Ring Oscillator based Frequency Generator 15
  - 3.2.2 Frequency Counter 17
  - 3.2.3 Top Level Sensor Design 18
- 3.3 FPGA and the Processor Interconnect 22
- 3.4 Overall Top Level Architecture 23

Chapter 4 Experimental Setup and Results 25

- 4.1 Environmental Setup 25
- 4.2 FPGA Heater Setup and Results 26
  - 4.2.1 Temperature Sensor Results 27

4.2.2 FPGA Heater Design Bitstreams 28
4.2.3 FPGA Heater Design Results 29
4.3 Pseudo Heat Sensor Setup and Results 33
4.3.1 Pseudo Heat Sensor Initial Simulations 34
4.3.2 Overall FPGA Design Toolchain Bitstreams 35
4.3.3 Overall FPGA Design Toolchain Results 37

Chapter 5 Conclusion 40

Bibliography 42

LIST OF FIGURES

Figure **2.1**: A Sample 5-bit LFSR Circuit 6

Figure **3.1**: The Architecture of a Single LFSR based Heater Block 10
Figure **3.2**: Top-Level FPGA Heater Design 12
Figure **3.3**: FPGA Heater 100MHz/78% Design Implemented on the FPGA 14
Figure **3.4**: Ring Oscillator-based Frequency Generator Design 16
Figure **3.5**: Top Level Block Diagram of the Frequency Counter Design 18
Figure **3.6**: Top Level Pseudo Heat Sensor Block 20
Figure **3.7**: Four Heat Sensors implemented on the FPGA 21
Figure **3.8**: Top Level Architecture of the FPGA Design Toolchain 23

Figure **4.1**: FPGA Heater Result at 100MHz 30
Figure **4.2**: FPGA Heater Result at 200MHz 30
Figure **4.3**: FPGA Heater Result at 300MHz 31
Figure **4.4**: FPGA Heater Result at 450MHz 31
Figure **4.5**: Results of the Four Pseudo Heat Sensors for the 100MHz/60% Heater Design 37
Figure **4.6**: Results of the Four Pseudo Heat Sensors for the 450MHz/25% Heater Design 38

LIST OF TABLES

Table **4.1**: FPGA Heater Bitstreams 28

Table **4.2**: Pseudo Heat Sensors and FPGA Heater Designs Bitstreams 36

ACKNOWLEDGEMENTS

I want to extend my deepest gratitude to my advisor, Dr. Vijaykrishnan Narayanan, for all the support he has shown towards me throughout my time as a graduate student. Throughout the thick and thin, he was always there to support me and always helped me move forward. He was generous enough to consider my particular research interests and provided me with appropriate directions to work. He will always be an inspiration for me. Any student would be fortunate to be advised by him. I would also like to thank my other committee member, Dr. Swaroop Ghosh, for his insightful comments on my work. I have also learned a lot from him during my time as his student. I want to give special regards to Dr. Oren Gall, whom I have known for more than four years as a teaching assistant. I very much appreciate all the advice that I received from him throughout this time. I am also very grateful for working as a co-op at Viavi Solutions, where I was introduced to similar work and problems faced in the industry. I also want to thank all the members of MDL, as well as our department staff, including the IT staff, for helping me troubleshoot all the technical issues that I experienced, especially during this time of working remotely. Lastly, my deepest gratitude to my parents, family, friends, and teachers who shaped me into what I am today.

This work was supported in part by the National Science Foundation (NSF). The views and findings expressed herein are solely those of the author and do not reflect the opinion or position of the sponsor.

Chapter 1

Introduction

1.1 Motivation

In the current age where we are surrounded by digital devices, the market of FPGAs has been on a growth [1]. The use of FPGAs in today's world has evolved from mere prototyping devices to high-speed heterogenous processors integrated into several domains such as military, aerospace, medicine etc. [2]. In addition to this, the constant technological advancements in the semiconductor industry is leading to smaller feature sizes, which in turn is resulting to an ever-increasing chip density. The implementation of these large and complex digital designs on FPGAs is not the most straightforward, and often leads to non-ideal behavior such as thermal hotspots. Such non-ideal behavior can also lead to the introduction of other issues, such as timing failures in FPGA designs. The motivation for this work comes directly from the need in the industry, calling for efficient solutions to manage the on-chip thermal behavior of FPGA designs. One major advantage of this work is that it introduces us to such a solution that is fully ready to be deployed in the industry.

1.1.1 Motivation for the FPGA Heater Design

Nowadays, as the feature size of the technology nodes has been shrinking, which is allowing an increase in the chip logic density. An advantage of this is that logic designers have been able to fit more and more logic in the same chip. However, this also leads to some disadvantages, such as a poor thermal behavior of the chip. Suppose that we are given a large FPGA design that could be divided into grids experiencing different thermal hotspots. This nature of the design could lead to some timing failures on the FPGA. The FPGA heater design was developed precisely to assist in managing this issue. It has the ability to be placed onto such large design and manage the thermal properties of those grids. This would enable us to create a uniform thermal nature throughout the chip, hence flattening the temperature variations over the existing thermal hotspots in those grids.

1.1.2 Motivation for the Pseudo Heat Sensor

One of the most common ways for detecting the temperature on an FPGA chip as seen in the industry is using the temperature sensor block present on the chip. For detecting the heat generated by the heater design, this temperature sensor that is present at the center of the FPGA chip was used as the first method. The issue with this temperature sensor is the location. In most cases, it is present at the very center of the chip, and if the aim is to analyze the thermal properties of large FPGA designs consisting of several thermal hotspots throughout the chip, this temperature sensor will not be able to provide accurate results. This is where the heat sensor included in this toolchain helps in providing a more

accurate understanding of the thermal properties of such designs. These heat sensors could be manually placed by the logic designers at various locations on the chip during the placement phase of the FPGA design build process and will provide a more accurate understanding of the thermal behavior across the various hotspots than the vendor temperature sensor placed at the very center of the chip.

1.2 Contributions

This thesis makes the following key contributions –

- An FPGA toolchain that would aid in managing the on-chip thermal behavior on fully implemented designs by working as a programmable heat sink for validation and testing.

- This toolchain consists of mainly two components – an FPGA Heater and a Pseudo Heat Sensor.

- The sole purpose of the FPGA Heater design is to increase the on-chip ambient temperature. It consists of fully customizable parameters such as the clock frequency and the chip area utilization (logic cells) that control the amount of heat generated.

- The Pseudo Heat Sensor senses the increasing temperature. These sensors are deployed on different parts of the FPGA chip.

1.3 Outline

This thesis is organized as follows. In chapter 2, we discuss the background needed to effectively comprehend this work and some related work. It consists of some background on FPGAs and on some other technical aspects related to the implemented designs. Following this, chapter 3 dives deep into the design and architecture of this toolchain, consisting of the FPGA Heater design, the Pseudo Heat Sensor design, and the overall top-level architecture including the interface between the processor and the programmable logic (FPGA). Chapter 4 discusses the experimental setup and results, covering the environmental setup that was used to perform the work, the heater results showing the various temperature readings taken from multiple bitstreams, and finally the heat sensor results showing the skewed sensor readings taken from different parts of the chip with varying temperature. Finally, Chapter 5 summarizes all the work that was accomplished.

Chapter 2

Background

2.1 Field Programmable Gate Array (FPGA)

FPGA stands for Field Programmable Gate Array. It is a reconfigurable IC which can be reconfigured after manufacturing. Hence, it is “field programmable”. In general, it consists of these three major components – CLBs (configurable logic blocks), I/O blocks, Switching/Routing blocks and Block RAMs.

The reconfigurable feature of an FPGA comes from the CLB, which in general consists of LUTs (Look-up table) and flip flops. We will make use of these CLBs to implement the heater design and increase their utilization on the FPGA chip, and also for implementing the sensor design. The I/O blocks, as the name suggests allows us to interact with external connections. The switching blocks enables us to connect the various CLBs. Finally, the block RAMs are the memory blocks used for storing large amounts of data in the FPGA. Some of the purposes for using BRAMs in FPGA designs are for creating FIFOs, large look up table data, crossing clock domains using those FIFOs, and in general just storing large amount of data. In the recent advancements of FPGAs, there are additional complex memory blocks present on some large FPGA chips, such as Ultra RAMs.

2.2 Linear Feedback Shift Register (LFSR) as a Pseudo-Random Number Generator

A Linear Feedback Shift Register (LFSR) is a very well-known digital circuit used in many applications such as cryptography, built-in self-testing (BIST) techniques, and some other broadcasting and communications applications. The very basic application that will be the focus of this work is the working of a LFSR as a Pseudo-Random Number Generator.

They come in various forms, such as Fibonacci LFSRs, Galois LFSRs, and Xorshift LFSRs, but a basic LFSR in its simplest form is a chain of registers (shift register) with a linear function of “taps” acting as the feedback into the input data of the LFSR. Figure 2.1 is a sample circuit design of a 5-bit LSFR –

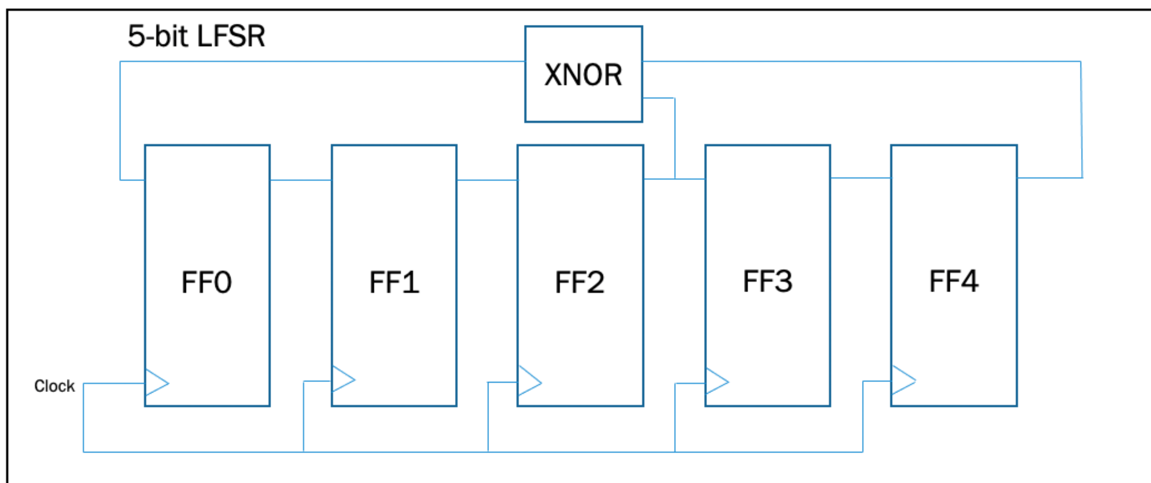

Figure 2.1: A Sample 5-bit LFSR Circuit

As seen in Figure 2.1, this LFSR consists of 5 bits, with the “taps” being at outputs of FF2 and FF4 that are passing through the XNOR logic gate to the first bit. The outputs of

this circuit generate “pseudo-random” numbers. The term “pseudo” comes from the fact that they would be random, but that sequence would repeat after a while. The actual heater design that was implemented as part of the toolchain consists of a similar chain of registers, but the number of registers is 128.

2.3 Ring Oscillators as Temperature Sensors

Ring Oscillators is also a very well-known digital circuit. It is a chain of inverters, with the last inverter in the chain acting as the feedback into the first inverter, completing the chain. This chain of inverters only must consist of odd number of inverters. The output that a ring oscillator generates is a periodic square wave, like a clock signal. Now, the frequency of this output depends on two factors – the number of inverters in the ring oscillator, and the propagation delays of the logic blocks in which the ring oscillator is synthesized on an FPGA, for our application. The higher the number of stages in the ring oscillator, and the higher the propagation delay through each of those stages, the lower will be the frequency of the output signal. What makes this design interesting is that with just some fine tuning, these ring oscillators can be used as heat sensors, which is exactly how they are used for our application. These ring oscillators can provide us with a sense of “pseudo” temperature. The way this could happen is if the frequency of the output generated by a ring oscillator is observed over time as the temperature of the chip increases. As the heat increases, the performance of a ring oscillator will degrade. The frequency will show skewed behavior. This is the exact application for which ring oscillators are used for this FPGA design toolchain.

Chapter 3

Design and Architecture of the System

In this chapter, the proposed architecture of the FPGA toolchain for on-chip thermal management, including the FPGA Heater and the Pseudo Heat Sensor is described in detail.

3.1 FPGA Heater Design Overview

The first component of the toolchain is the FPGA Heater design. The sole purpose of this component was to come up with a design which would generate heat on the FPGA chip. In order to accomplish this, the technique used was to create a design that would consist of N number of 128-bit Linear Feedback Shift Register (LFSR) instantiations, all running simultaneously. The LFSR design is a very well-known pseudo-random number generator. This would utilize the resources (specifically registers) on the FPGA chip, continue to generate pseudo-random numbers using the registers on the FPGA, and this constant logic execution with a high utilization and high clock frequency would generate heat on the FPGA. This design provides the user with two fully customizable parameters. The first one is used for controlling the overall utilization of this heater design block on the chip. More specifically, it controls the instantiations of the 128-bit LFSR circuit, which in turn controls the number of registers being used on the chip, and hence the overall utilization. The second parameter is used for controlling the clock frequency used

for the design. Higher the overall utilization and clock frequency of the heater block, higher the temperature generated. Note that from this design, again, there is nothing to do with the actual logic of the design, which is the pseudo-random number generation. Instead, this logic is being used as a heat generator. The constant execution of thousands of these registers performing pseudo random number generation would induce a high-power draw from the FPGA chip. That, in turn would generate heat. In order to measure the temperature, the on-chip temperature sensor present at the center of the Xilinx FPGA that was used as part of the setup was utilized.

One thing to note is that, generally with these digital circuits, we explore several low power techniques, but instead, for this specific component, we are exploring precisely how to draw the most power out of this design. Overall, the heater design consists of the LFSR based heater block, and some other vendor specific IPs including the Xilinx's System Monitor for temperature reading. The following sections describe the heater block in detail, as well as the vendor specific IPs used.

3.1.1 LFSR based Heater Block

This component is at the core of the top-level heater design. A single heater block consists of a basic 128-bit LFSR circuit. It is very similar to the sample 5-bit LFSR circuit shown in Figure 2.1, but the difference being that the taps are present at bits 127, 125, 100 and 98 passing through the XNOR logic. For the heat generation to work effectively, a large LFSR design that constantly generates pseudo random numbers is required. This single heater block consisting of a 128-bit LFSR needs to be instantiated

hundreds of times in order to effectively generate the heat on the chip. Hence, a single 128-bit LFSR was chosen as the basic heater block. This heater design consists of a customizable parameter “count” that would instantiate this 128-bit LFSR design block “count” number of times. This is what controlled the utilization of the chip area, utilized with this pseudo random number generating logic running simultaneously. This utilization, for instance, could be just 10% with 100 instantiations of the 128-bit LFSR design and could go up to 96% with 800 such instantiations. It also consisted of an enable bit to enable/disable the heat generating logic. Besides this, the FPGA implementation required to use certain Xilinx Vivado synthesis attributes such as “DONT_TOUCH” in order to not let the synthesis tool remove the duplicate logic of LFSR registers instantiated hundreds of times. The architecture of a single LFSR based heater block is shown in the circuit in Figure 3.1 –

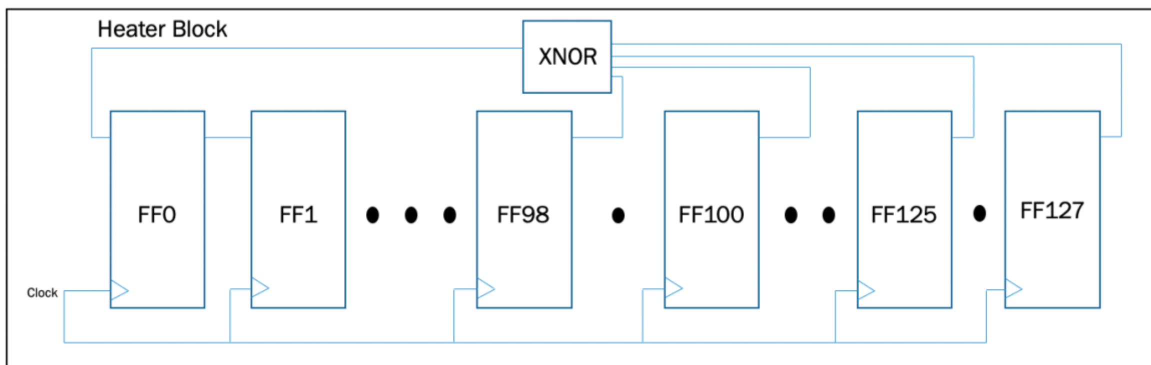

Figure 3.1: The Architecture of a Single LFSR based Heater Block

This block in Figure 3.1 is instantiated hundreds of times in order to increase the chip area utilization, and hence this circuit summarizes the overall top-level heater generation logic.

3.1.2 Vendor Specific IPs

There are two Xilinx's IPs that were used in the top-level heater design – Xilinx's XADC IP that includes the on-chip temperature sensor, and the Mixed-Mode Clock Manager (MMCM) module.

The XADC IP provides an easy interface to the on-chip ADC for reading FPGA temperature and on chip voltages. For the heater design's use case, only the FPGA temperature sensor was activated and utilized through this IP. In order to interface with this IP, there are several options such as AXI4-Lite, Dynamic Reconfiguration Port (DRP), and AXI4-Stream. The DRP interface was used with this IP given the simplicity of the operation. This temperature sensor was mainly used for validating the heater block.

The MMCM module is a very commonly used Xilinx IP which provides the ability to generate multiple clocks with defined phase and frequency relationships to a given input clock. The given input clock source that was used for the heater block is a 100MHz clock pin provided on the FPGA chip that was used as part of the setup. The MMCM was then used to generate more clock outputs with the frequencies of 200MHz, 300MHz and 450MHz in addition to the already existing 100MHz clock. At any given time, the heater block was using one of these clocks generated by the MMCM.

3.1.3 Top-Level Heater Design

The overall top-level heater design consists of a core heater block that is responsible for generating the heat using hundreds of 128-bit LFSR circuit, some Xilinx IPs such as the XADC temperature sensor and the MMCM for multiple clocks generation, and a processor interconnect design to add the communication functionality between the PS (ZYNQ processor) and the PL (FPGA) in order to interface and add the read/write ability to/from the FPGA. This was done by simply adding the ZYNQ processor to the design and the AXI interconnect. Most of this was auto generated using the Vivado tool. This processor interconnect design was later modified to accommodate the entire circuit design for the toolchain including the heat sensor, as described later. The top-level heater design architecture is shown in Figure 3.2 below –

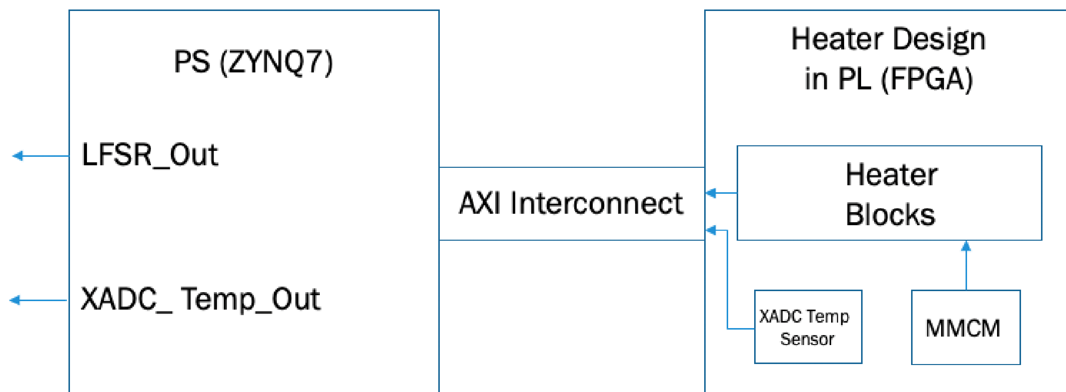

Figure 3.2: Top-Level FPGA Heater Design

As seen in the architecture in Figure 3.2, as part of the Programmable Logic (PL) in the FPGA, the top-level heater design consists of hundreds of heater blocks, which are taking

the clock input as one of the clocks generated by the MMCM. It also includes an instantiation of the XADC temperature sensor which would read the chip temperature. There also exists a Processing System (PS) which is the ZYNQ7, that is the software used for communicating with the FPGA to read the LFSR data and the XADC temperature output data. The communication between the ZYNQ7 and the FPGA is performed through the AXI interface. The LFSR output would provide the pseudo-random numbers that are generated from the heater blocks. The XADC temperature output is driven directly from the XADC temperature sensor IP.

This top-level heater design was fully implemented on an FPGA, and there were several bitstreams generated for testing. The difference between each of these bitstreams is the FPGA resource utilization and the clock frequency used to drive the heat generation logic, which are the two customizable parameters. As a sample, Figure 3.3 is a design implemented on the FPGA that is running on a 100MHz clock with 78% utilization –

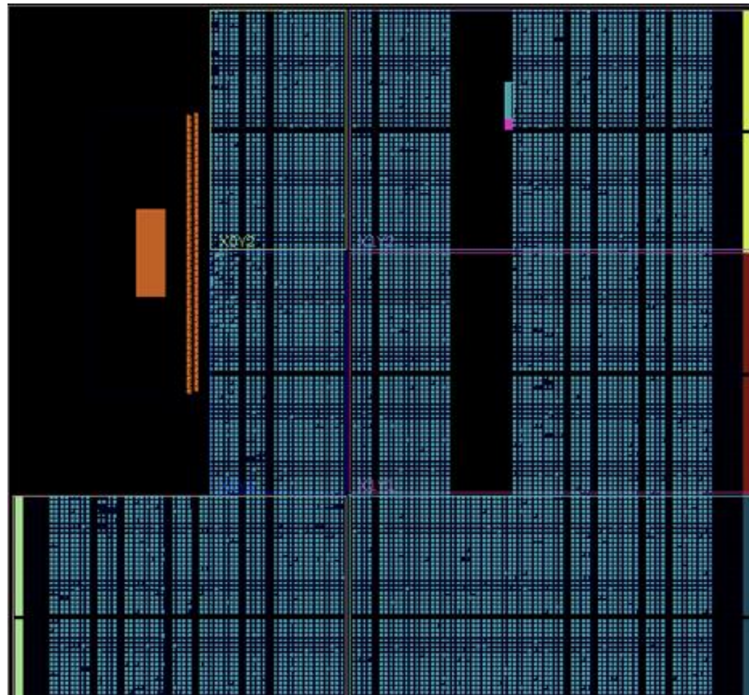

Figure 3.3: FPGA Heater 100MHz/78% Design Implemented on the FPGA

3.2 Pseudo Heat Sensor Design Overview

The Pseudo Heat Sensor is the second major component of this FPGA toolchain and is responsible for sensing the heat generated by the other major component, the FPGA Heater. It is made up of two main designs – a frequency generator and a frequency counter. The frequency generator is made up of a basic ring oscillator design and generates a clock signal of a fixed frequency. The frequency counter is a complex counter design that monitors the performance of the frequency generator by taking in the generated clock signal from it as the input and computes the actual frequency value of that signal in Hertz. The intended behavior of this Pseudo Heat Sensor is that as it is working independently of the FPGA Heater design, and as the amount of heat generated

by it is increasing, the performance of the ring oscillator-based frequency generator should be degrading, that is, the frequency value computed by the frequency counter should be getting skewed.

There could be multiple number of Pseudo Heat Sensor blocks placed throughout the FPGA chip, and that is completely up to the user configuring the toolchain. This toolchain would give the best performance in terms of managing the thermal behavior of the FPGA chip if the sensors are placed near or around the thermal hotspots on the chip. The term “pseudo” comes from the fact that there is no direct computation being performed to figure out the actual temperature of the hotspots on the chip, but instead, the amount by which the frequency generators get degraded, or in other words, have their frequency skewed, is exactly what would describe how hot the chip is becoming.

3.2.1 Ring Oscillator based Frequency Generator

A Ring Oscillator is one of the most fundamental designs in digital circuits, consisting of a chain of odd number of inverters. This chain of inverters generates an output signal which oscillates between a HIGH and a LOW voltage value, exactly like a clock signal. This output signal is taken at the end of the chain of inverters, and that is fed back onto the very start of the chain. The behavior of this oscillating signal is caused due to the NOT gates inverting the input at every stage in this chain, and the odd number causing the final output to be the opposite to what it was before at every iteration. The frequency of this oscillating signal mainly depends on the delay of each of these logic cells in the FPGA at every stage of the chain, and the number of these stages in the ring oscillator. In

the design, an appropriate number of the ring oscillator stages was chosen based on the observation made by experimenting with different odd number of stages. The exact chosen number of stages in this ring oscillator-based frequency generator design is seven, although, the number of stages is parametrized in the design, and the user is able to configure this. One addition to this basic design of a 7-staged ring oscillator is that it is gated by an enable signal. This additional logic serves us two purposes. The first being that it can simply control the execution of the heat sensor, that is, it acts as an ON/OFF switch for the heat sensor in the sense that if the enable is de-asserted, there is no frequency generated, and hence, no heat sensed. The other purpose being that since this design is achieved as a purely combinational logic in the FPGA, that is, it makes exclusive use of look-up tables, it aids in initializing the ring oscillator to start with ‘0’ as the input to the very first inverter. Unlike the flip-flops in the FPGA, as this is not a sequential logic design, these logic cells consisting of the ring oscillator logic will not be initialized to ‘0’ at the start of the FPGA programming, unless this additional logic of gated enable is added to the start of the ring oscillator chain. The circuit design of this ring oscillator-based frequency generator is shown in Figure 3.4 –

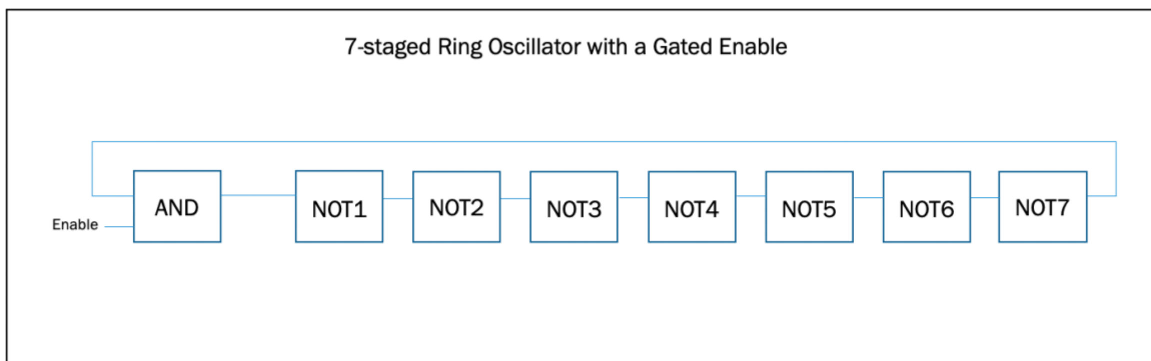

Figure 3.4: Ring Oscillator-based Frequency Generator Design

This ring oscillator-based frequency generator design serves as a very efficient way for monitoring the behavior of a constantly generated oscillating signal's frequency as the heat of the FPGA chip is increasing, and hence allows us to monitor the thermal behavior of the locations where these ring oscillators are placed, ideally at the thermal hotspots. For this specific application of sensing the heat generated, in order to use a clock-like oscillating signal, an external oscillator or a Phase Locked Loop (PLL) generating a clock externally in an FPGA cannot be used, as that signal's frequency will have no effect from the increasing heat generated. Note that for the FPGA Heater design, a clock from a MMCM was indeed used, as that indeed served the purpose well for that application. However, for this application of a Heat Sensor, an internal logic generating a clock signal making use of internal logic cells of the FPGA that are sensitive to external environmental factors such as degrading thermal effects caused by the FPGA Heater.

3.2.2 Frequency Counter

The overall responsibility of this Frequency Counter design is to figure out the frequency of a given periodic square wave input, such as a clock. In this case, the input that is being fed into this design is the output signal generated by the Ring Oscillator based Frequency Generator design. The Frequency Counter computes the frequency of this signal, and this is required to keep a track of the degrading thermal effects caused by the FPGA Heater.

The design mainly consists of two synchronous counters. The first counter is responsible for providing a sense of time. This is achieved by running the counter on a 100MHz clock and counting till 1,000,000 cycles, hence, indicating that 1 second has

passed. On the other hand, the second counter is responsible for counting the cycles of the input periodic square wave coming from the ring oscillator. The output of this second counter is captured exactly when 1 second has passed, that is, when the first counter hits 1,000,000 cycles. This captured value of the number of cycles counted by the second counter is exactly the frequency of that signal, in Hertz. This is mainly based on the definition of the frequency of a wave. As both counters are interacting with each other every second, potentially running on different frequencies, there is a synchronizer required to cross the clock domains of these two counters. The block level design of the Frequency Counter is shown in Figure 3.5 –

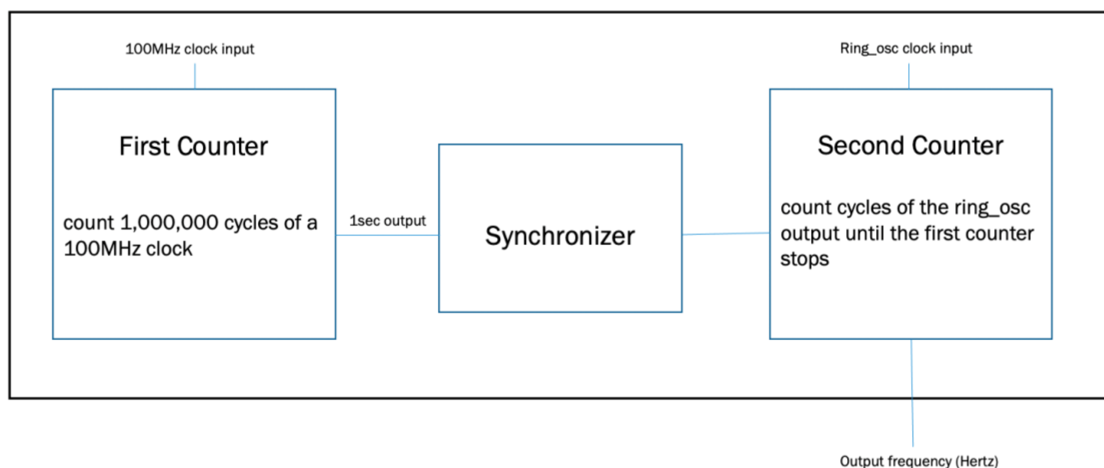

Figure 3.5: Top Level Block Diagram of the Frequency Counter Design

3.2.3 Top-Level Sensor Design

The overall top level sensor design consists of two major components - the Ring Oscillator based Frequency Generator design and the Frequency Counter design, all working towards a common goal that is to aid in observing the degrading thermal effects

on the FPGA chip caused by the FPGA Heater design. The ring oscillator-based frequency generator uses a 7-staged ring oscillator, that is gated by an enable signal controlling the functioning of the design. The number of stages in this design is also fully customizable. In the study, it was determined that the number of stages as 7 is appropriate for generating and analyzing the sample periodic square wave under increasing thermal effects. This generated periodic square wave is fed into the Frequency Counter, which makes use of two synchronous counters and generated the output as the frequency of the ring oscillator's periodic square wave in Hertz. The 100MHz clock as the input for the Frequency Counter design is used from one of the FPGA's clock pins. There also exists a double flop-based synchronizer in the Frequency Counter design. There also exists some processor interface logic to test this Pseudo Heat Sensor block individually, which is eventually integrated along with the FPGA Heater's processor interface logic to create a single processor interface block for the entire design, that is discussed in the next section. This entire block's design generates a Pseudo Heat Sensor, that can be placed at any location on the chip. The intention is to place these sensors around the FPGA chip's thermal hotspots to get the most accurate understanding of the thermal behavior of a design. The understanding would be that these sensors would determine the heat generated by those thermal hotspots on the chip, and that would help in determining exactly how and which hotspots should be worked on to remove the inconsistent thermal behavior of those different hotspots in a large FPGA design that consists of such inconsistent thermal hotspots.

Given below is a block level diagram of a single Pseudo Heat Sensor block shown in Figure 3.6 and a fully implemented design consisting of four Pseudo Heat Sensors

placed on various locations of the Xilinx FPGA chip that was used in the development shown in Figure 3.7. The number of sensors to generate and the location to place them at is fully customizable by the user.

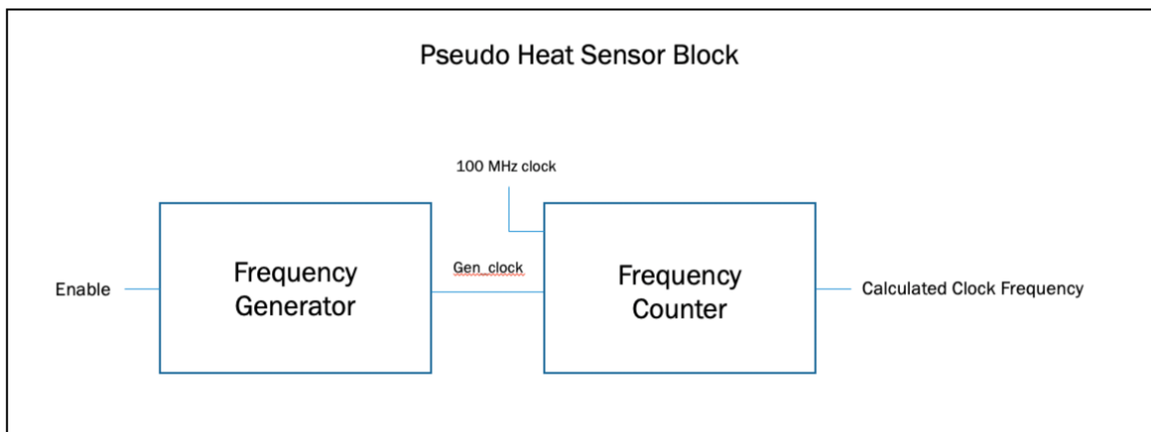

Figure 3.6: Top Level Pseudo Heat Sensor Block

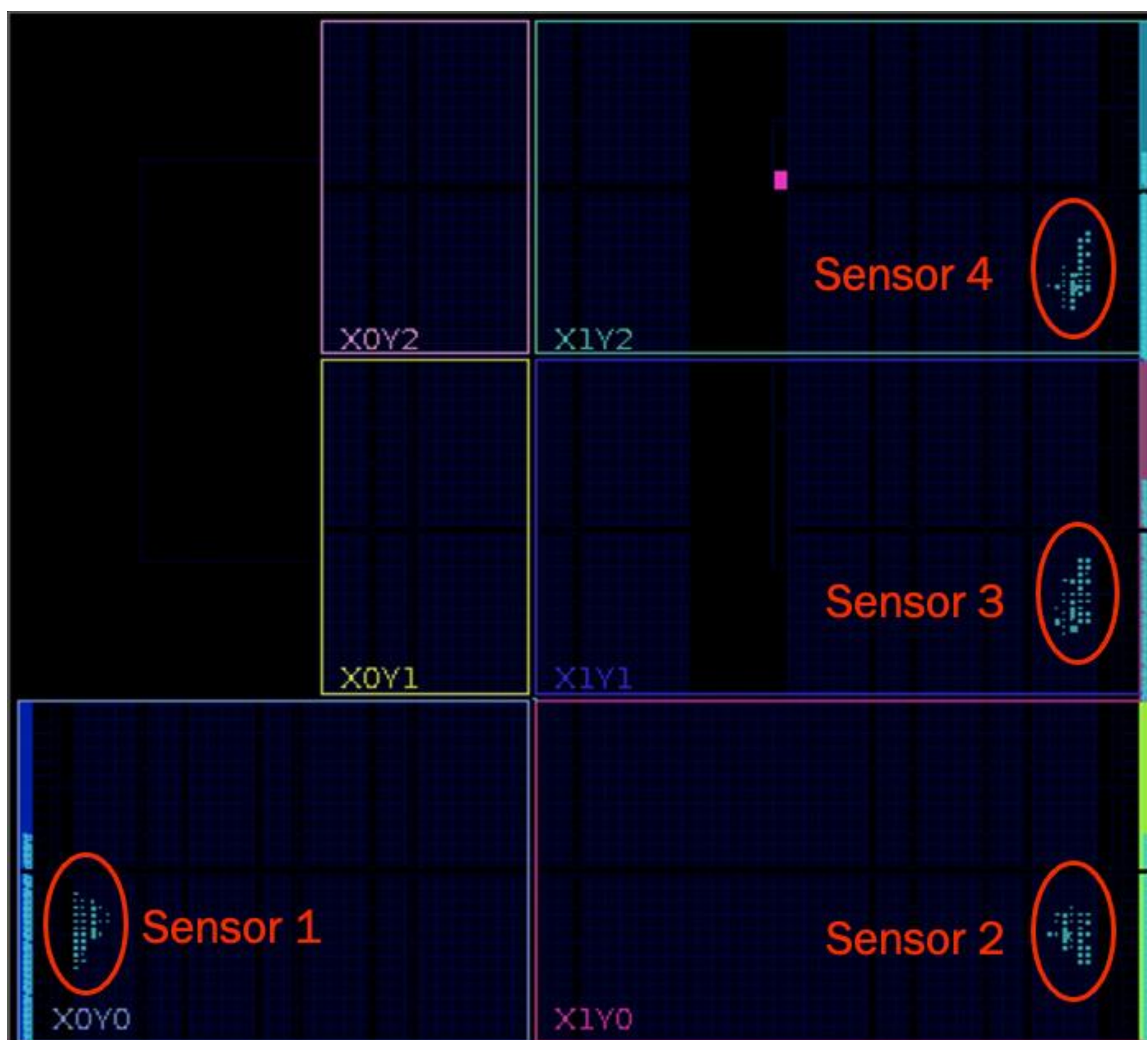

Figure 3.7: Four Heat Sensors implemented on the FPGA

3.3 FPGA and the Processor Interconnect

Apart from the core logic design of this FPGA toolchain for thermal management, another important aspect of the entire design process is the processor interface between the Programmable Logic (FPGA) and the Processing System (ZYNQ7). The board that was used to evaluate this FPGA design toolchain is Xilinx Zedboard, that consists of the ZYNQ SoC, containing the FPGA and the ARM processors. The data that is being generated by the logic on the FPGA, such as the Pseudo Heat Sensor readings and the outputs from the FPGA Heater needs to be passed on to the ZYNQ7 Processing System using a communication protocol. The specific communication protocol used for this exact purpose is the AXI Lite interface offered through the Xilinx suite of IPs. The data that is being received by or sent to the FPGA will be mapped to the memory blocks on the ARM core. The ARM core, which is the Processing System, acts as the Master in this interface. This means that all the reads/writes will be initiated by the Processing System. The FPGA acts in the Slave mode in this interface. The number of registers required, and the data widths were set up accordingly.

Note that the Processor Interconnect aspect of this FPGA design toolchain is very specific to how the testing was performed while designing the toolchain. It is not a critical part of the toolchain and should be specific to the user deploying this design toolchain. The performance of the core logic of the design toolchain consisting of the FPGA Heater and the Pseudo Heat Sensor does not depend on the AXI Lite interface used here, but only used to test the core logic being programmed on the Zedboard.

3.4 Overall Top-Level Architecture

As different blocks of this FPGA design toolchain have been discussed, they all come together to form the overall top-level architecture. The core design blocks are the FPGA Heater, and the Pseudo Heat Sensor, working independently to achieve a common goal. The top-level architecture also consists of the Processor Interconnect driven by the AXI protocol. The block diagram of the overall top-level architecture of this FPGA design toolchain is shown in Figure 3.8 below –

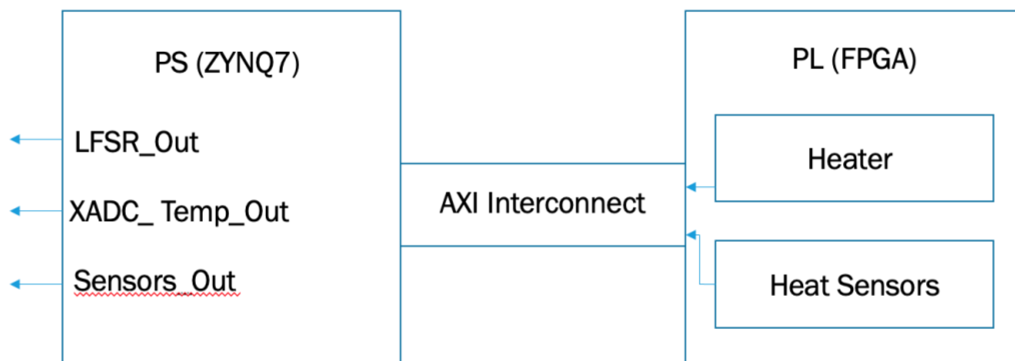

Figure 3.8: Top Level Architecture of the FPGA Design Toolchain

As seen in Figure 3.8, the Programmable Logic (PL) component, which consists of the core FPGA design logic for the overall design toolchain, consists of the FPGA Heater block and multiple instantiations of the Pseudo Heat Sensors. At a very high level, the heater block consists of hundreds of 128-bit LFSRs running constantly to generate heat on the FPGA chip. The heat sensors are deployed around the thermal hotspots on the FPGA chip, and these individual sensors are continuously computing the frequency of the

constantly generated periodic square waves from the ring oscillators. The sole aim of the heater block is to increase the heat generated on the FPGA chip, and that is controlled by the overall utilization of the heater registers on the chip which is a fully customizable parameter. The aim of the sensors is to sense the fluctuating frequency of the ring oscillator-generated clock around the area of the chip where the sensor is deployed, and hence providing with a sense of pseudo temperature of that thermal hotspot.

Another component is the ZYNQ7 Processing System (PS) that is responsible for initiating the reads/writes from/to the FPGA to control the functionality of the core design logic. It is making use of the AXI protocol for this exact interface. The three readings that are being monitored for the working of this design toolchain are the LFSR output, the XADC temperature output, and the individual sensors output. The LFSR output contains the pseudo-random values generated by a single 128-bit LFSR block in the FPGA Heater. This output ensures the working of the LFSR. To ensure the working of the FPGA Heater in the testing phase, the output from the XADC IP instantiated in the FPGA Heater design is taken, that gives a direct temperature value of the chip from the temperature sensor present at the center of the Xilinx FPGA chip used for development. This ensures the increasing heat of the FPGA chip caused by the FPGA Heater. Finally, to ensure the working of the Pseudo Heat Sensors, the reading from each of the sensors is taken, and this reading is precisely the frequency value from the Frequency Counter block in the Pseudo Heat Sensor design. This reading assists in keeping a track of the skewed frequency value from those heat sensors deployed at the thermal hotspots.

Chapter 4

Experimental Setup and Results

In this chapter, the environmental setup that was used to experiment with the functioning of the FPGA design toolchain, and the results that were gathered from each and every component of the design toolchain are described in detail.

4.1 Environmental Setup

The very first step taken in order to plan the design of this FPGA thermal management toolchain is to decide which FPGA development board to choose to implement this design. The aim was to choose an FPGA development board that consists of an SoC including the FPGA acting as the programmable logic chip, as well as the ARM cores acting as the processing system. Another requirement was to ensure that the chosen FPGA chip consists of an on-chip temperature sensor in order to validate the functioning of the FPGA Heater design. Xilinx's Zedboard Development Board matched all these requirements, and hence was chosen for the development of this design toolchain.

Given the chosen board was Xilinx's Zedboard, the toolflow for developing the core design consisted of Xilinx's Vivado Suite and Xilinx's SDK. The Register Transfer Level (RTL) design was written using SystemVerilog, and the software used for interfacing was written in C. The overall environment used for all these tools was Linux. The Zedboard was connected to the Linux development PC using a serial port, and hence, a tool called Minicom was used to perform the serial communication with the board.

4.2 FPGA Heater Setup and Results

The first major component of this FPGA design toolchain that was tested on the Zedboard development board was the FPGA Heater design. Starting from the initial architecture definition to the RTL design, to pre-synthesis simulations that were mainly focused on the working of the LFSRs and were fully successful, and finally to the synthesis and implementation of the overall heater design on the FPGA, there were several bitstreams that were generated and programmed in order to test the functionality FPGA Heater on the actual FPGA chip. These several design bitstreams consisted of different clock frequencies for the heat generating logic and different chip area utilization to observe how the temperature changes with these different design parameters programmed to the Zedboard. Note that all the temperature readings are in Celsius. This was the main test to verify the functionality of the overall FPGA Heater design.

As a secondary test, another tool was used to verify the initial temperature of the FPGA chip as the baseline. This other tool is Matlab's FPGA Data Capture that directly interfaces with the on-chip temperature sensor using the XADC IP. It only requires a JTAG connection to the board. It was used to program the FPGA without any Heater logic, but just the logic responsible for interfacing with the XADC IP. This would provide us with the stable temperature value of the idle FPGA chip without any logic being executed on it. This baseline initial temperature value would then be compared against the actual FPGA Heater bitstreams programmed, to observes exactly by how much the temperature increases, if it does.

4.2.1 Temperature Sensor Results

The very first step to verify the functionality of the overall FPGA Heater design was to test the Xilinx's on-chip temperature sensor present on the FPGA chip. This test would provide us with the baseline initial temperature of the chip used to compare against when the actual heater logic is being executed. In order to confirm the working of the temperature sensor, two implementations were used and compared. The first one is the primary one, that is using a bitstream consisting of the heater logic fully disabled and consists of the processor interface logic using the AXI interconnect. This method also uses the serial UART connection to program the FPGA to read the temperature value and makes use of Xilinx SDK tool to achieve this. The second method is using Matlab's FPGA Data Capture tool which can capture the data from the temperature sensor using just the JTAG connection to the board. This method also included no heater logic when programmed.

The FPGA on the Xilinx Zedboard development board was programmed using both these implementations. Both the implementations gave a stable temperature reading of around 36 degrees Celsius on average with no heater logic running on the FPGA. Specifically, the Matlab's FPGA Data Capture implementation gave a temperature reading of 35 degrees Celsius, whereas the primary implementation using the Xilinx SDK consisting of the AXI interconnect logic gave a reading of 37 degrees, having some extra minimal heat due to the AXI interconnect logic overhead. The heater designs consisted of the AXI logic along with it, and hence their temperature readings were only taken using the direct serial connection through Xilinx SDK. We will keep this temperature of 36

degrees Celsius as the baseline initial temperature with no logic running on the FPGA to observe how different heater designs with varying utilization and clock frequencies affect the temperature of the chip.

4.2.2 FPGA Heater Design Bitstreams

As explained earlier, there were several FPGA Heater design bitstreams that were generated to perform the testing. These bitstreams varied from each other based on two customizable design parameters – clock frequency of the heat generating logic, and the chip area utilization. The clock used in the heat generating logic is for the hundreds of LFSRs constantly being executed. Higher the clock frequency, higher the heat generation is expected. The chip area utilization is increased by increasing the number of instantiations of the heater blocks on the FPGA, hence increasing the registers responsible for the working of the LFSR. Table 4.1 below shows a list of design bitstreams that were generated for testing –

Table 4.1: FPGA Heater Bitstreams

No. of Bitstreams	Clock Frequency	Chip Area Utilization
Bitstream 1	100MHz	36%
Bitstream 2	100MHz	78%
Bitstream 3	100MHz	96%
Bitstream 4	200MHz	10%
Bitstream 5	200MHz	36%
Bitstream 6	200MHz	60%
Bitstream 7	300MHz	36%
Bitstream 8	300MHz	50%
Bitstream 9	450MHz	10%
Bitstream 10	450MHz	25%

As seen in Table 4.1, in total there were ten bitstreams generated for programming the FPGA to verify the functionality of the FPGA Heater. The clock frequency used varied from as low as 100MHz to as high as 450MHz. The total chip area utilization varied from as low as 10% to as high as 96%.

4.2.3 FPGA Heater Design Results

The several bitstreams that were generated to verify the functionality of the FPGA Heater were programmed one by one on the Zedboard and the results of the chip temperature were observed. The heater logic was enabled at the start of the FPGA programming, and the temperature readings were taken throughout a span of 30 seconds, except for one design bitstream that was running at 300MHz that was considered as the most heat generating design, for which the temperature reading was observed throughout 1 minute. Several graphs are shown below to depict the results from various FPGA Heater design bitstreams programmed on the Zedboard –

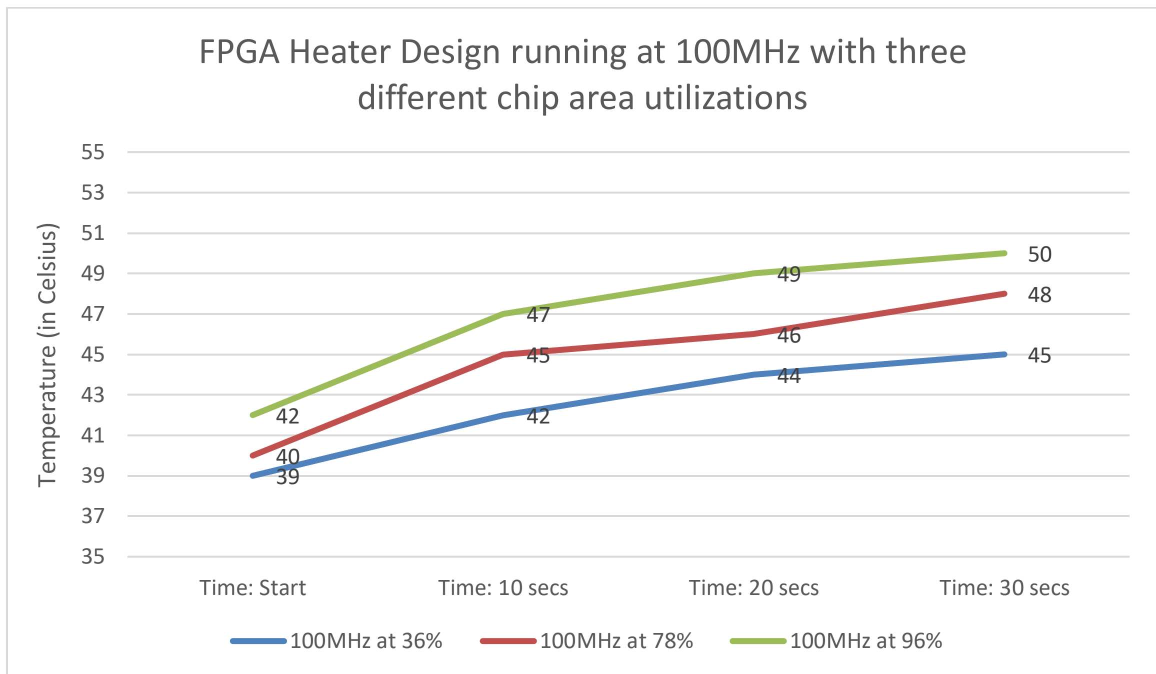

Figure 4.1: FPGA Heater Result at 100MHz

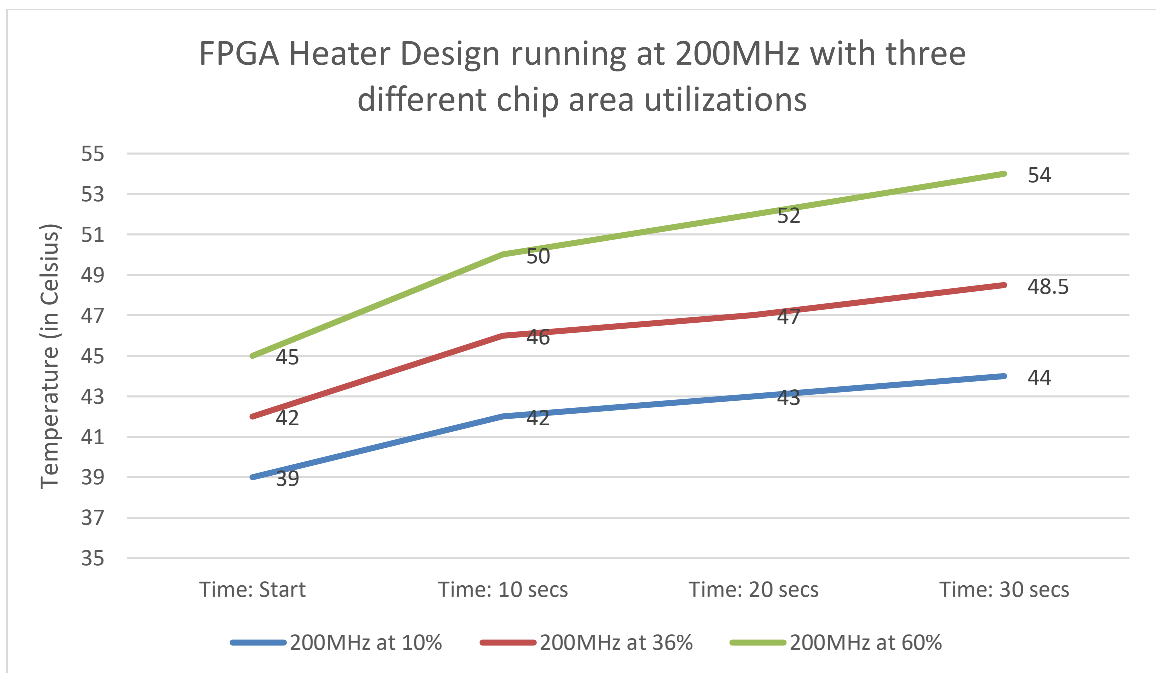

Figure 4.2: FPGA Heater Result at 200MHz

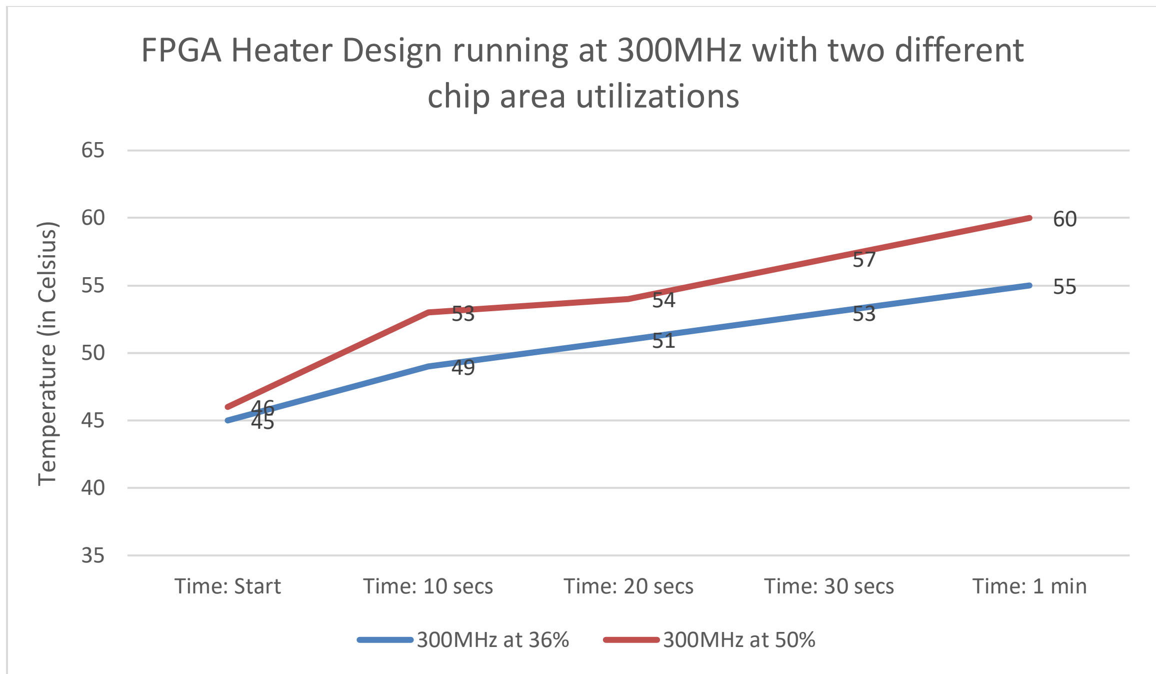

Figure 4.3: FPGA Heater Result at 300MHz

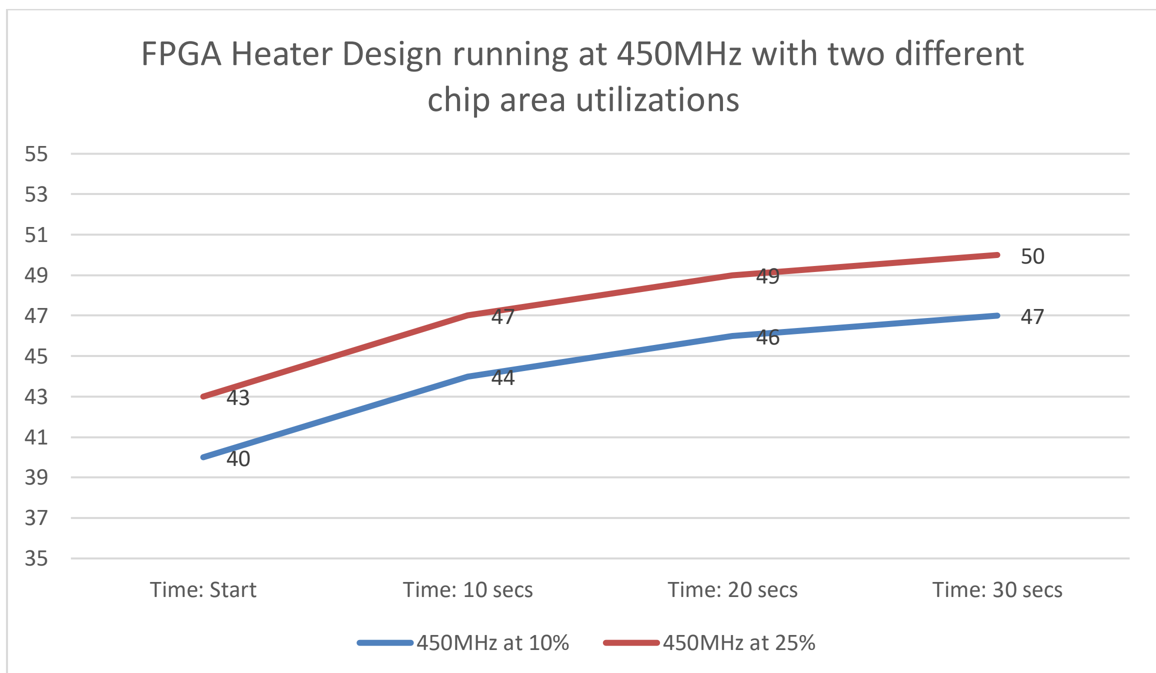

Figure 4.4: FPGA Heater Result at 450MHz

Through these results, it is confirmed that as the chip area logic utilization and the clock frequency increased, the heat generated, and the temperature increased. The design that resulted in the highest temperature reading from these experiments was the heater design running at 300MHz frequency at a 50% chip area utilization. The highest temperature reading from this design was 60 degrees Celsius right after executing it for 1 minute, that is a 66% increase in chip temperature. Note that due to some tool related limitations, programming designs running at very high frequencies (200MHz and above) and very high utilization (60% and above) were not possible at the time. This was due to the Xilinx SDK tool not being able to program the ARM cores at extreme design parameters (such as 300MHz running with 96% utilization). If an alternative method is used for the processor interconnect, this FPGA design toolchain should be able to produce even higher chip temperatures when programmed at extreme design parameters.

Some interesting observations were that for a heater design running at 100MHz, the highest temperature recorded was 50 degrees Celsius, but that required a 96% of chip area utilization. The same heater result of 50 degree Celsius or higher could be recorded by running the same heater design at 200MHz with 36% to 60% utilization, or 300MHz with only 36% utilization or potentially even lesser, or directly at 450MHz at only 25% chip area utilization. All these observations fully verified the working of the FPGA Heater design in this toolchain.

In order to physically confirm the chip temperature, the FPGA chip did indeed feel hot just by touching it from the bottom of the board when showing a reading of 55 degrees or higher. This just added an additional confirmation of the working of the heat generating logic.

4.3 Pseudo Heat Sensor Setup and Results

The second and the final major component that was tested on the Zedboard development board was the Pseudo Heat Sensor design. In this case, there were four heat sensors placed at four different locations on the FPGA chip as shown in Figure 3.7, with the FPGA Heater logic running constantly to increase the heat on the chip. Then, the readings from each of these four sensors is taken and observed over time. The reading consists of the frequency value computed from the frequency counter design that is taking in the generated clock signal output from the ring oscillator in the Pseudo Heat Sensor logic.

There are several different bitstreams programmed, just like the testing performed for the FPGA Heater with these bitstreams varying degrading thermal effect on the chip by varying the heater design parameters, that are the clock frequency and the chip area utilization. However, in this case of testing the Pseudo Heat Sensor, in each of these bitstreams, the number of sensors which is four, and their location on the FPGA chip remains the same, unlike in the testing of FPGA Heater where there were absolutely no heat sensors deployed on the FPGA chip. There would be four temperature data points taken over time, and the skewed sensor readings would be observed at those intervals. The initial reading that is the very first reading from the sensors is taken, and that would be around when the chip is least hot. As time passes by and the thermal conditions are further degraded, the computed frequency values from the sensors will be compared against the initial value to observe the exact amount by which the computed value has deviated, if any.

One thing to note is that, in this case, the heater logic was additionally configured to cause more heat around some heat sensors and less heat around the other heat sensors. This would provide the sensors with a sense of thermal hotspots around some heat sensors, and minimal degraded thermal effects around the other heat sensors. The expectation would be to observe a higher skewed frequency result from the heat sensors near the thermal hotspots, and a lesser skewed frequency result from the sensors far away from the thermal hotspots. These observations are shown in the next sections.

4.3.1 Pseudo Heat Sensor Initial Simulations

At the very start of the verification phase, the first step performed was a pre-synthesis simulation to verify the functionality of the frequency counter design. This design was by far the most complex design in this entire FPGA design toolchain, and it was crucial to be very sure of the working of this block. The other design block in the overall Pseudo Heat Sensor design is the ring oscillator-based frequency generator but verifying the working of a ring oscillator using the look-up tables (LUTs) on an FPGA is not possible during pre-synthesis simulations as the delays in the LUTs cannot be observed in pre-synthesis simulations, and hence, this part was left to be verified once the design is fully implemented and programmed on an actual FPGA chip. For the pre-synthesis simulation of the frequency counter design, a clock signal having a known frequency of 50MHz was sent as the input signal to this block, and once the timer counter in this design hit 1 second, the computed frequency value from the other counter was indeed an exact value of 50MHz. To perform some further testing, some more inputs of different frequencies

were also used, and the computed frequency was confirmed to be correct every time. This verified the working of the frequency counter design at least in simulations.

4.3.2 Overall FPGA Design Toolchain Bitstreams

In total, there were two bitstreams programmed on the FPGA in order to test the overall functionality of the Pseudo Heat Sensors along with the FPGA Heater executing to degrade the thermal effects on the FPGA chip. These two bitstreams varied in the form of some sensors placed near more heater logic blocks and some sensors placed near less heater logic blocks. The locations with more heater logic blocks will act as the thermal hotspots, and the other locations will act as experiencing less thermal degradation effects. This is controlled mainly by the chip area utilization of the heater logic blocks. But the two bitstreams also differ in the clock frequency at which the heater blocks are being executed. The location and the number of the sensors, which is four, that are deployed on the chip remains the same for both the bitstreams. Table 4.2 summarizes the two bitstreams, and the results gathered from programming those bitstreams will be discussed in the section following that.

Table 4.2: Pseudo Heat Sensors and FPGA Heater Designs Bitstreams

No. of Bitstreams	Heater Clock Frequency	Total Heater Chip Area Utilization	Expected Thermal Effects on the Sensors
Bitstream 1	100MHz	60%	Sensor 1 - High Sensor 2 - Low Sensor 3 - Medium Sensor 4 - High
Bitstream 2	450MHz	25%	Sensor 1 - High Sensor 2 - Medium Sensor 3 - Low Sensor 4 - High

As seen in Table 4.2, there are two bitstreams that were generated to test this entire FPGA design toolchain for thermal management. It consists of the four Pseudo Heat Sensors deployed on the chip at various locations. The fourth column in the table describes the expected thermal effects on the sensors. This is controlled by the number of heater logic blocks placed near them. An expected thermal effect on the sensor is high if the sensor is placed near a high number of heater logic blocks out of the total heater chip area utilization, and hence that area acts as a thermal hotspot for the sensor. Similarly, the expected thermal effect on the sensor will be low if there are a smaller number of heater blocks placed near the sensor. The two bitstreams differ in the clock frequency as that would reflect in how gradually or quickly the sensor outputs are getting skewed as the rate of the heater effect differs among them.

4.3.3 Overall FPGA Design Toolchain Results

The two bitstreams that are described in the previous section were programmed on to the Zedboard one by one, and specifically the results from the four sensors deployed on the chip were gathered while the FPGA Heater logic was being executed which was causing an increase in the chip temperature. The readings are recorded at four intervals of the temperatures, and the skewed behavior of the calculated frequency values from the heat sensors is observed. There are multiple graphs following this, that depict this behavior –

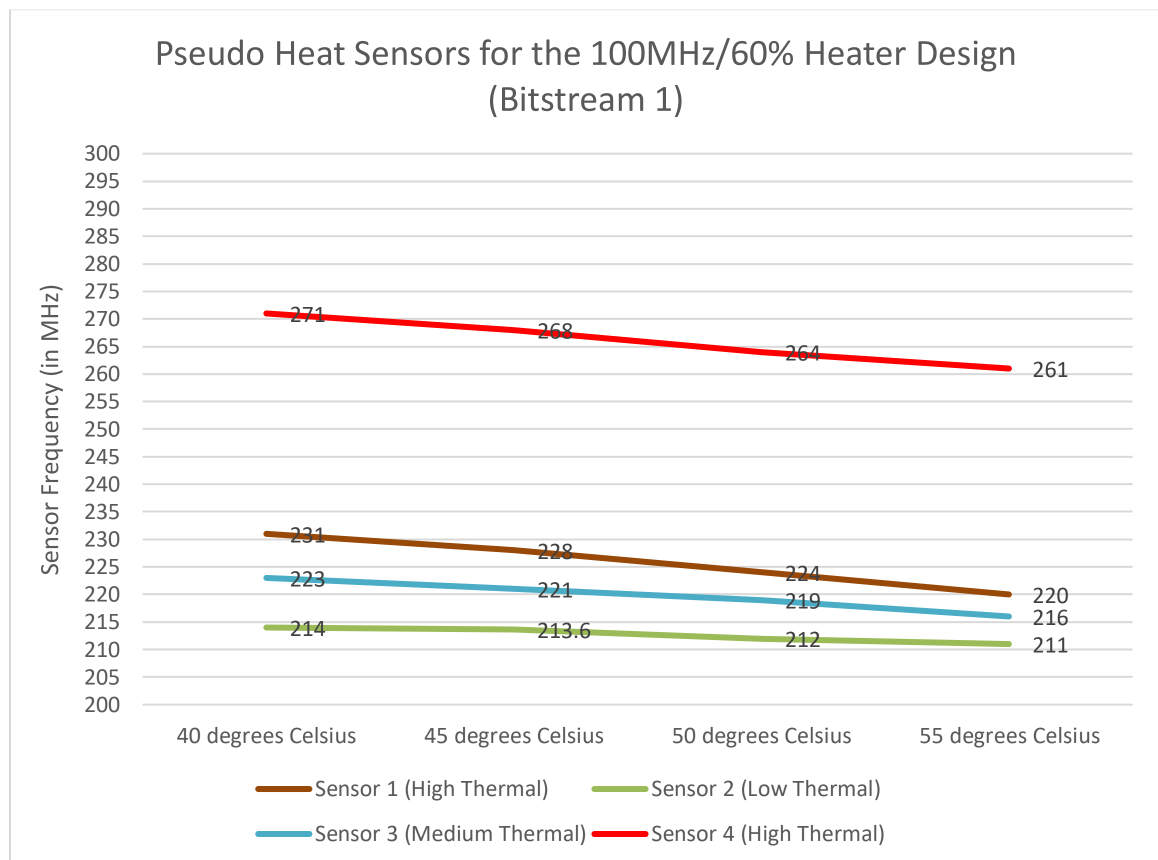

Figure 4.5: Results of the Four Pseudo Heat Sensors for the 100MHz/60% Heater Design

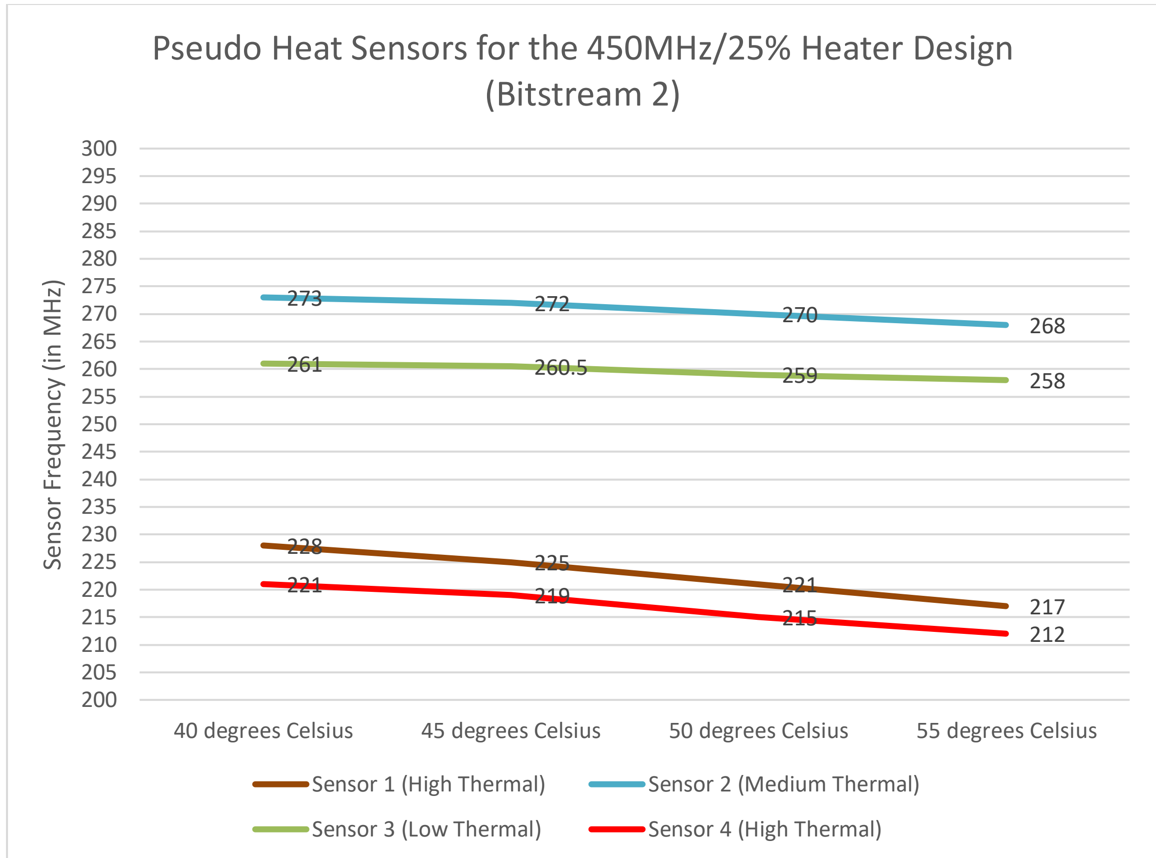

Figure 4.6: Results of the Four Pseudo Heat Sensors for the 450MHz/25% Heater Design

As observed in Figure 4.5 and Figure 4.6, the general trend of the Pseudo Heat Sensor output frequency is that it starts to decline as the chip temperature increases, which is the expected result. The frequency generated by the ring oscillator depends on the number of stages, and the logic block propagation delays. As the sensor is placed closer to a thermal hotspot, the effects of delay are further worsened. The rate by which the frequency is decreased depends on how close or far away the sensors are placed from the thermal hotspots. In these tests, the sensors placed near a high number of heater logic blocks

(High Thermal) show a higher rate of the decline in frequency, than the sensors that are placed near a smaller number of heater logic blocks (Medium or Low Thermal). This result also matches the expectations. The worst behavior by a sensor was seen in the sensors in the High Thermal regions, and their frequency degraded by around 11 MHz as the chip temperature reached 55 degrees Celsius. That is an approximately 5% of decline in the ring oscillator performance. As the temperature of the chip is even further increased, this decline is expected to rise further.

Note that there was no method available while testing to find the real temperature around the sensors that were deployed on the FPGA chip. This is the job of these sensors to provide us with a sense of “pseudo” temperature by observing the rate of the decline in calculated frequency from the sensors. Sensors placed near thermal hotspots on the chip will show much drastic skewed behavior in frequency. However, in order to test, as seen on the x-axis of the charts depicting these results, the temperature values shown are taken from the Xilinx’s on-chip temperature sensor that is placed at the center of the FPGA chip. This is the exact reading taken from the temperature output of the FPGA Heater. This was taken to observe the behavior of the sensor as the overall chip temperature increased. Additionally, note that different sensors show different frequency ranges. This is due to the way the designs are implemented on the FPGAs, at least from Xilinx tools. Different sensors implemented on the chip will vary based on different routing methods as decided by the implementation tool. To observe the overall behavior of the chip’s temperature instead of localized behavior, a higher number of sensors can be deployed on the chip and an average of the readings can be calculated, which would provide us with a sense of pseudo temperature of the overall chip.

Chapter 5

Conclusion

In conclusion, this project described an FPGA design toolchain for managing the on-chip thermal behavior. It consists of two core designs as part of the toolchain – an FPGA Heater and a Pseudo Heat Sensor. The FPGA Heater design provides us with two customizable parameters – the clock frequency and the chip utilization. These parameters can be used to control the amount of heat needed to be produced, and the rate at which that needs to happen, if required. This part of the toolchain can be immediately utilized in the industry as a programmable heat sink. The results of FPGA Heater showed that the out of all the tests performed, the design that was running at a 300MHz of clock frequency and a 50% of chip area utilization reached a peak of 60 degrees Celsius after executing the design for 1 minute. This is a 66% increase in chip temperature caused mainly by the FPGA Heater logic. On the other hand, the Pseudo Heat Sensors were designed and used to sense the temperature at parts of the chip where these sensors were deployed. The results showed that the sensors placed closest to the thermal hotspots simulated by placing a high number of heater logic blocks around them showed the most decline in the calculated frequency. The decline in the frequency was around 11MHz, which is a 5% decline, over a chip temperature rise from 40 degrees Celsius to 55 degrees Celsius. The number of sensors deployed on the chip and their locations is fully customizable. Finally, suppose there exists a large FPGA design that could be divided into grids, with each grid experiencing different thermal behavior, these Pseudo Heat

Sensors could provide us with an insight into this exact behavior, and the FPGA Heater logic blocks can be customized and accordingly placed to help in achieving an overall flattening of the chip temperature across those grids.

BIBLIOGRAPHY

[1] S. Donthi and R. L. Haggard, "A survey of dynamically reconfigurable FPGA devices," Proceedings of the 35th Southeastern Symposium on System Theory, 2003., 2003, pp. 422-426, doi: 10.1109/SSST.2003.1194605.

[2] Balas M.M. (2014) A New Generation of Biomedical Equipment Based on FPGA. Arguments and Facts. In: Iantovics B., Kountchev R. (eds) Advanced Intelligent Computational Technologies and Decision Support Systems. Studies in Computational Intelligence, vol 486. Springer, Cham. https://doi.org/10.1007/978-3-319-00467-9_11

[3] *Efficient Shift Registers, LFSR Counters, And Long Pseudo-Random Sequence Generators - Xilinx.com*. Xilinx Inc.; 1996. https://www.xilinx.com/support/documentation/application_notes/xapp052.pdf. Accessed December 22, 2020.

[4] *Mixed-Mode Clock Manager (MMCM) Module (V1.00a) - Xilinx.com*. Xilinx Inc., 24 June 2009, https://www.xilinx.com/content/dam/xilinx/support/documentation/ip_documentation/mmcm_module.pdf. Accessed December 28, 2020.

[5] *7 Series Fpgas and Zynq-7000 SoC XADC Dual 12-Bit ... - Xilinx.com*. Xilinx Inc., 23 July 2018, https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf. Accessed January 4, 2021