Performance Variability Analysis of the OpenSPARC Architecture on FPGAs

Richard Wiley
Advisor: Prof. Puneet Gupta

***Abstract*—As the trend towards increasing semiconductor miniaturization continues, maintaining uniform performance of integrated circuits in manufacturing becomes difficult. FPGA design software takes this variability along with other factors such as temperature into consideration when determining timing constraints. This paper explores the variability of the maximum achievable frequency of the OpenSPARC T1 architecture implemented on two Xilinx FPGA boards at multiple temperatures.**

***Index Terms*—OpenSPARC, FPGA, Virtex 5, Performance.**

I. Introduction

The continual push towards smaller manufacturing processes for integrated circuits improves both power consumption and performance. Smaller transistors physically require less energy to operate, which lowers power consumption as well as heat dissipation. Lower heat dissipation per transistor allows for more transistors to be placed in close proximity at higher frequencies without producing excessive amounts of heat. The reduced physical size of smaller transistors allows for shorter distances between transistors, which also increases the potential frequency of a chip.

Modern manufacturing processes can produce transistors that are tens of nanometers in size. New limitations that are introduced to the manufacturing process at these minute scales make it increasingly difficult to reliably produce uniformly performing chips [1]. The result is that two chips of identical design may have very different performance characteristics – one may produce more heat, or have a higher reliable operating frequency.

The manufacturing process is not the only factor relevant in determining the performance characteristics of a chip. Environmental variables such as temperature and humidity may affect performance [2]. Additionally, factors such as a chip's supply voltage may vary within a certain range. Even with these uncertainties, a manufacturer must guarantee customers that a given chip will operate at a certain frequency. Manufacturers determine this value by finding the highest frequency that a chip can operate at given the worst case scenario of manufacturing imperfections, environmental variables, and power supply fluctuation. The result of this practice is that most chips are capable of performing beyond their documented specifications [3].

This research project examines the maximum potential operating frequencies of the OpenSPARC T1 microprocessor architecture implemented on two Xilinx FPGA boards. FPGAs (Field Programmable Gate Arrays) are integrated circuits that can easily be reconfigured by the user with a HDL (Hardware Design Language). Our specific project uses Xilinx's XUPV5-LX110T Development System which is based around the Virtex 5 FPGA. We implement the open source OpenSPARC processor design on our FPGAs, which allows us to boot a full operating system on our development boards. Our experiment focuses on the variability in maximum achievable frequency across multiple chips and across a range of frequencies and temperatures. We use two development boards to observe the difference between chips and a temperature chamber to control the boards' operating temperatures.

II. Background

A. Hardware

We use Xilinx's XUPV5-LX110T Development System which implements Xilinx's Virtex 5 FPGA. The development system is a PCB that contains the FPGA chip, a DDR2 DIMM, a clock generator chip, I/O peripherals, a programming port for modifying the FPGA configuration, and other devices that are not relevant to our project. To use the system, a processor configuration is first downloaded to the board over the JTAG programming interface. A ramdisk image is then loaded into the board's memory over the JTAG cable. At this point, the FPGA's design is implemented and the program to run on the device is loaded in memory; the system can now execute the program. For our project, we use the serial port for I/O with the system.

B. Software

We use the Xilinx Platform Studio (XPS) to modify the FPGA design, place and route the design, and output the design in a format which can be downloaded to the development board. Xilinx's FPGA Editor software allows us to make modifications to the FPGA design after the place and route has occurred. Without this software, we could not change the operating frequency of our FPGA design without XPS recalculating the placement and routing of the design, making performance comparisons between designs impossible. We use Xilinx's iMPACT software to download our packaged designs to the FPGA board, then Xilinx Microprocessor Debugger (XMD) to download and run software on the system. The Xilinx Timing Analyzer helps us locate which modules are responsible for the timing constraints that limit the maximum

frequency of the processor. For temperature monitoring, we use the Analyzer program which is part of the Xilinx LabTools suite. With this software we can monitor the FPGA's on-chip temperature sensor in real time. Finally, we use the minicom program which is responsible for I/O with the board via the serial connection.

C. FPGA Configuration

We use Sun Microsystem's OpenSPARC architecture. The open source design includes a prebuilt Xilinx EDK (Embedded Development Kit) project which we can modify with XPS software and download to our Virtex 5 boards.

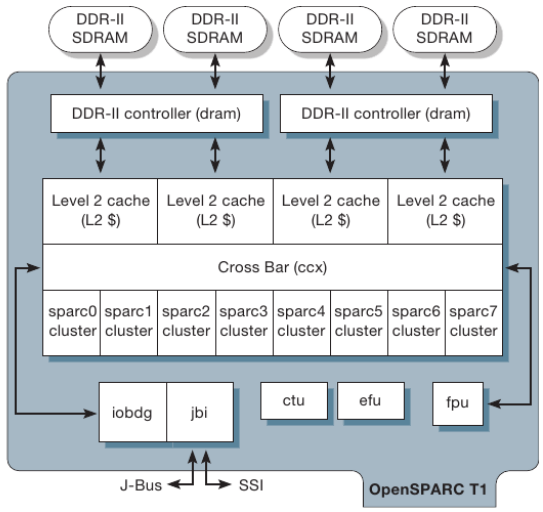


Fig. 1. OpenSPARC T1 Architecture

The Xilinx EDK project has been tailored specifically for our XUPV5-LX110T board. The particular instantiation of the design we use includes one SPARC core. To support the SPARC core, the design also includes a Xilinx Microblaze processor to handle the memory subsystem as well as communication with the board's various peripherals.

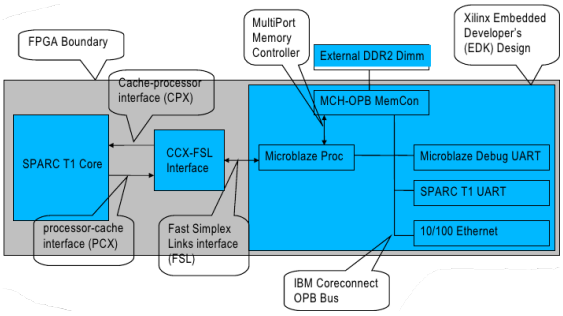


Fig. 2. Virtex 5 OpenSPARC implementation

The Virtex 5 FPGA is too small to contain a SPARC core, L2 cache, a memory controller, and a peripheral controller. The Virtex 5 implementation addresses this problem by integrating the microblaze controller at the OpenSPARC's crossbar (Fig 2). This design allows the Microblaze controller to handle access to L2 cache and memory. In the interest of space, the Microblaze controller does not implement an L2 cache, but still provides an interface to the SPARC core that handles L2 cache access. This separation of subsystems allows us to change the frequency of the SPARC core without affecting the frequency of the memory or peripherals.

III. Testing

We needed to find a way to concretely test the successful operation of our FPGA boards across a range of frequencies and temperatures. This required a way to change frequencies without repeating the place and route process, an environment in which temperature could be measured and manipulated, and software that would fully utilize the FPGA design.

A. Frequency Modification and Verification

The Virtex 5 implementation of the OpenSPARC processor uses Xilinx's Clock Generator IP (Intellectual Property) to manipulate the 100 MHz clock chip integrated on the XUPV5-LX110T. In this design, the clock generator uses either one or two PLLs(Phase Locked Loops) to generate frequencies for the SPARC core, the Microblaze controller, the DDR2 memory, and the board peripherals. A PLL uses an input clock, an integer multiplier, and an integer divider to produce output frequencies. The PLLs in the Xilinx IP can output up to 16 frequencies per PLL, with a global multiplier for all outputs and a divider for each output. The PLLs have additional restrictions on the ranges of the multiplier and dividers.

Xilinx's XPS software provides a simple interface for making modifications to the Clock Generator IP (Fig 3). The clock generator configuration let us specify the frequency of the outputs as well as which output frequencies to bind to which PLL. We bound the SPARC core frequency to PLL1 and all other outputs to PLL0. This separation of outputs allows us a wider range of multipliers to apply to the SPARC core without being constrained by the frequency requirements of other components. Both PLL inputs come from the board's 100 MHz clock chip. Additionally, we made modifications to the project's MHS (Microprocessor Hardware Specification) file and the UCF (User Constraints File) to route the SPARC core frequency to an output pin on the board. We verified the SPARC core frequency by probing this output pin with an oscilloscope.

The XPS software has a 'Generate Bitstream' button that takes the high level design, passes through several intermediate stages, and finally produces a bitstream which can be downloaded to the FPGA. The first step in this process is synthesis, which uses the project HDL to create an intermediate NGC file. Design implementation takes the NGC file, adds user constraints to the intermediate product, maps the design, places and routes the design, then finally generates the programming file.

Our goal was to change the frequency of the design after the placement and routing of the design had occurred. If we had just changed the frequency in the clock generator configuration, the place and route tools would change the layout of the design. Xilinx's FPGA editor let us modify the

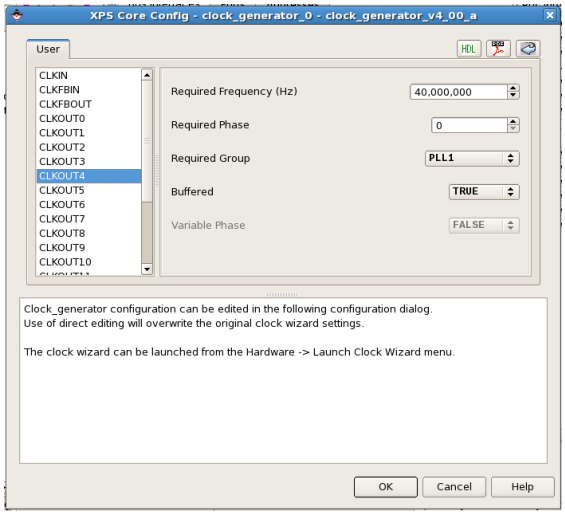


Fig. 3. Clock Generator Configuration in XPS

output of the place and route process, the NCD (Native Circuit Description) file. This file already contains the physical layout of the design and only goes through bitstream generation before it gets downloaded to the board.

The FGPA Editor program uses the NCD and constraint files to produce a bit file. Once FPGA Editor created our bit file, We used a simple bash script that automates the necessary steps that the iMPACT software takes to download the bit file to the FPGA. After the design was downloaded to the board, we used a tcl script we wrote to make the XMD software download various selected programs to the board's DRAM.

We used the FPGA Editor program to change the parameters of the PLL which generated the SPARC core's frequency (Fig 4). We used a Java program to generate the PLL parameters that would let us change the output frequency at the highest resolution possible. With a design that was placed and routed with a 40MHz SPARC core, we found that in the core could operate up to 75 MHz. We tested the boards over a range of 60 MHz to 76.9 MHz.

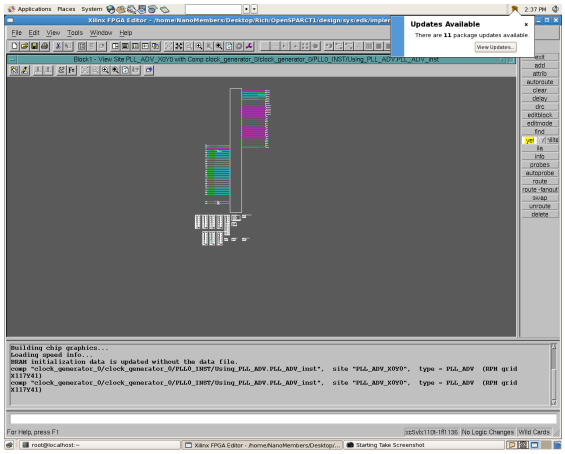


Fig. 4. Viewing the PLL in FPGA Editor

B. Temperature control

We used a Styrofoam box as a temperature chamber and a 57 watt light bulb connected to a dimmer to modulate temperature. The Virtex 5 chip has an integrated temperature sensor which we used to measure temperature, through Xilinx's LabTools Analyzer program. We performed cycles of tests at operating temperatures of 70°C, 80°C, 90°C, and 100°C. We determined a stable 'operating temperature' by running a 'helloworld' program on the system and then letting the program settle in an infinite loop upon completion. We would then use the chamber and light bulb to stabilize the board at the desired temperature. Between test runs, we would let the board run in an infinite loop for 30 minutes to allow the board to return to the base temperature, as the chip temperature may have risen during the course of the test. We were not able to find a way to record the temperature of the chips throughout the duration of the tests, which lowers the resolution of our temperature testing measurements. Potential errors in measurements are discussed in detail in a later section.

C. Testing Software

We performed tests on the boards by running a basic 'helloworld' program and running the OpenSolaris operating system. The 'helloworld' program was written in C on a Linux machine and then cross-compiled to run on the SPARC architecture. The OpenSolaris ramdisk image is included with the OpenSPARC project download. The main differences between the two programs are the length of time it takes to run them, and the degree to which they stress our SPARC core. The 'helloworld' program can be loaded to DRAM and run within 2 minutes. The OpenSolaris operating system takes over one hour to load to DRAM and complete the boot sequence. The operating system is a far more comprehensive test of our SPARC core than our 'helloworld' program, which essentially just outputs one string from memory.

D. Methodology

We tested both boards with the 'helloworld' program and the OpenSolaris operating system. When testing the 'helloworld' program, we conducted 3 test runs per frequency per temperature. A failure in any 3 of the runs was recorded as a failure for that test. Due to the lengthy duration of the OpenSolaris tests, we performed one boot of the OS per frequency per temperature. We wrote a bash script that, for a given temperature environment, would load each frequency and attempt to boot the OS. We would then manually modify the temperature environment and rerun the testing script. We considered a test passed if the program produced the expected output.

IV. Experimental Results

TABLE I
MAX FREQUENCY - 'HELLOWORLD'

Temp (C)	Board 1 (MHz)	Board 2 (MHz)
70	72.7	75
80	72.7	72.7
90	72.7	72.7
100	72.7	72.7

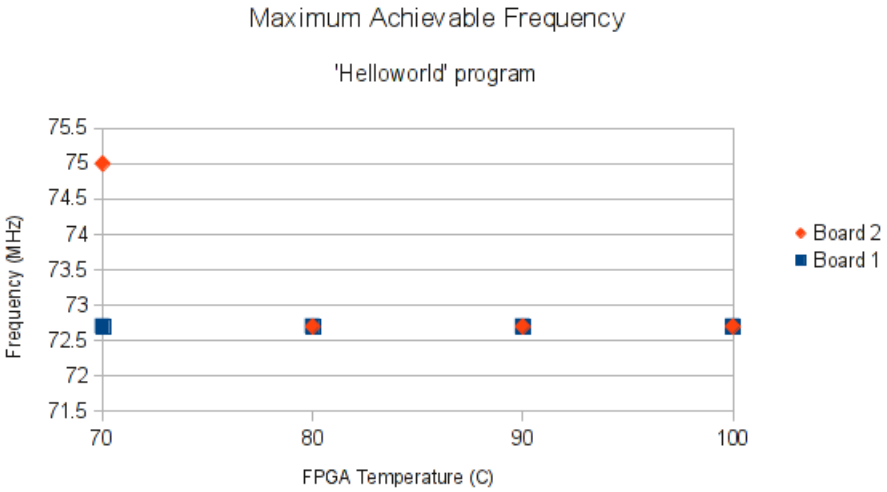
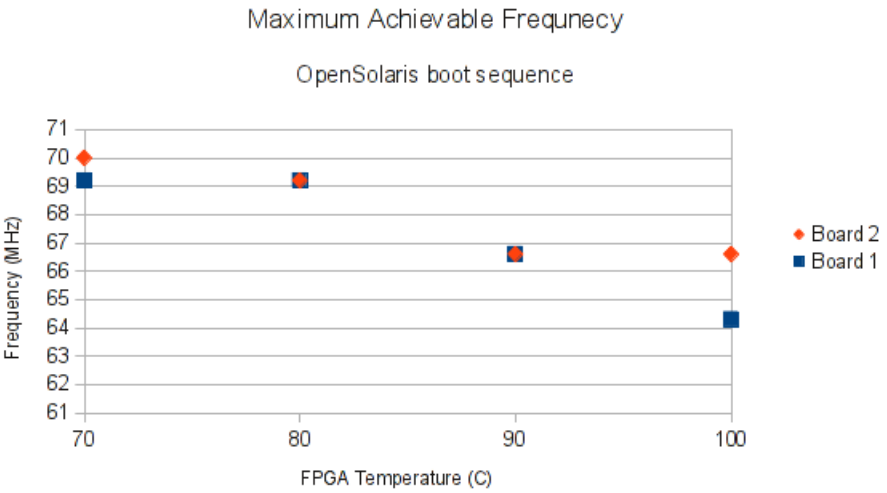


TABLE II
MAX FREQUENCY - OPENSOLARIS

Temp (C)	Board 1 (MHz)	Board 2 (MHz)
70	69.2	70
80	69.2	69.2
90	66.6	66.6
100	64.3	66.6



V. Analysis

We saw very little correlation between temperature and speed while running the 'helloworld' program and a much higher correlation between temperature and speed while running OpenSolaris. Additionally, we can conclude that board 2 had a higher maximum operating frequency than board 1.

Our 'helloworld' test shows almost no correlation between rising temperatures and decreasing maximum frequency. In fact, at the resolution of frequency that we had available, board 1's max frequency did not change at all for any of the temperatures. Board 1 was capable of running at 75 MHz in a room temperature environment, then settled at 72.7 MHz for the rest of the temperature environments. I think this low variance was mostly due to two factors – the low resolution of our frequency testing and the simplicity of the program.

The resolution of our frequency tests was bounded by the capabilities of the PLLs in the Virtex 5's clock generator. As such, when a test failed at a given frequency, the next step down could be a 2.5 MHz difference. In our 'helloworld' test, Board 2's maximum frequency falls one frequency increment after 70°C and remains there for the duration of the testing. The board's maximum frequency, however, could be falling in .5 MHz increments, and we would only detect this derivative between two test points. Similarly, Board 1 could show slight variations in frequency between temperatures that were too small for our experiment to detect.

The 'helloworld' program's simplicity could also affect our results. The program uses few instructions that may not leverage some parts of our SPARC core, meaning that at a frequency at which these unused components fail, the program will still successfully execute. Our timing analysis of the design showed that the Ethernet controller and parts of the memory subsystem had the least 'slack' in them – meaning that they would be the first to fail if the design was pushed past specification. This program does not use the Ethernet controller at all, and accesses very little memory.

Our OpenSolaris tests had a higher frequency variability across temperatures. It is interesting to note that all of the 'helloworld' tests resulted in a higher max frequency than any of the OpenSolaris tests. This implies that the SPARC core is not entirely stable at the 'helloworld' frequencies, but is just stable enough to run the very basic program. The OpenSolaris tests show much more conclusive data that describe the difference between our boards. At two temperatures out of four, Board 2 has a higher achievable max frequency, as it did at 70°C in our 'helloworld' program. The two data points at which the boards have the same max frequency is likely due to the low resolution of our frequency testing. Both boards, as expected, consistently showed a drop in maximum frequency as temperature increases.

VI. Potential Error Sources

The relatively crude method we used for temperature modulation and measurement added a high degree of uncertainty to our results. This potential error is the main reason that we tested temperatures in large increments of 10°C. The largest problem with our method is that we did not have data that showed the temperature of the board throughout each test. We could not automatically record the temperature readings during the OpenSolaris tests because the temperature monitoring software needed to be manually reset after each downloaded bitstream. We know that the temperature rose several degrees Celsius during some tests, but we do not have thorough quantitative data from which to derive conclusive results. Additionally, the temperature sensor only recorded the temperature at the location of the sensor – the temperature of the rest of the SPARC topography is unknown.

The low resolution of our frequency testing is also likely responsible for less accurate data. At the resolution we had available, Board 1 and Board 2 often had the exact same maximum frequency at some temperatures, while having large differences at others. Additionally, each board's maximum frequency sometimes seemed to stabilize at one frequency across two or three temperatures, only to fall at the next temperature point.

VII. Future Work

The best way to improve the success of this experiment would be to increase the accuracy of our measurements. This applies to both our temperature measurements and the resolution of our frequency measurements. Perhaps the best way to correct our temperature measurements would be to record the temperature of the chip at the time of failure, instead of the temperature at which the test was initiated. An increase of available frequencies within our frequency testing range would also benefit this experiment. As the design stands, the available speeds are limited by the divider and multiplier of the PLL. We could, however, make the input of the SPARC core's PLL come from the design's second PLL rather than the 100 MHz clock chip. Having a range of input frequencies rather than just one increases the PLL's possible resolution to less than 1 MHz.

We could also gather more conclusive results if we knew exactly which modules of the SPARC core were failing. This could be accomplished by running specialized software that heavily stresses some modules while not stressing others. For example, a program that simply performs read and write operations on a large array would specifically stress the memory subsystem. Another program might stress the ALU with a small dataset, to avoid stressing the memory subsystem. These programs could help identify the SPARC core module that is limiting the overall operating frequency.

VIII. Conclusion

The goal of our experiment was to observe the effect that increasing temperature has on modern microprocessors'

maximum achievable operating frequency. Additionally, we tested multiple chips expecting that our results would show the variability that exists in modern microprocessor manufacturing. Testing with multiple programs further allowed us to examine the behavior of the boards in various test conditions.

While we would have liked to have used more precise measurement techniques, our results did show a correlation between higher temperatures and lower maximum achievable speeds. Our results also revealed that one of our boards is capable of running faster, with either program and over a range of temperatures.

ACKNOWLEDGMENT

I would like to thank my research partner Steven for being of great help throughout this project. I would also like to thank Jagannath for getting the entire software environment set up, as well as for getting me started with the project.

REFERENCES

[1] Pete Sedcole, Justin S. Wong, and Peter Y.K. "Characterisation of FPGA Clock Variability," IEEE Computer Society Annual Symposium on VLSI, Montpellier, 2008.

[2] Phillip H. Jones, Young H. Cho, John W. Lockwood. "Dynamically Optimizing FPGA Applications by Monitoring Temperature and Workloads," 6th International Conference on Embedded Systems, Bangalore, 2007.

[3] Zolotov, V., Visweswariah, C., Jinjun Xiong. "Voltage binning under process variation". *Computer-Aided Design - Digest of Technical Papers*, p. 425, Nov 2009.