Leon3 Processor Variability Emulator for Delay Variability Impact on Performance

Master Project Report

Nan Lyu
Dept. Electrical Engineering
University of California, Los Angeles
lvnanucla@g.ucla.edu
Advisor: Prof. Puneet Gupta

***Abstract*—Modern processors exhibit increasing variations in performance due to complicated manufacture process as well as long term usage. [1] That makes a processor emulator useful to study delay impact on performance when executing various tasks. This paper illustrates the implementation of a Leon 3 processor delay variability emulator based on Altera DE2-115 FPGA board, as well as several emulation results.**

Keywords—Leon3 processor, Altera FPGA, Delay Variability Emulation, CPU Performance

I. INTRODUCTION

Performance of a processor is usually different from what is labeled on the product due to hardware variabilities. The processor can either be overdesigned to guarantee to perform well in the worst case, or can have a decreased performance due to environmental influence or aging problem. Thus a processor delay variability emulator becomes useful to study the performance under several circumstances with different tasks, and can hopefully provide a platform to further study hardware-aware software development.

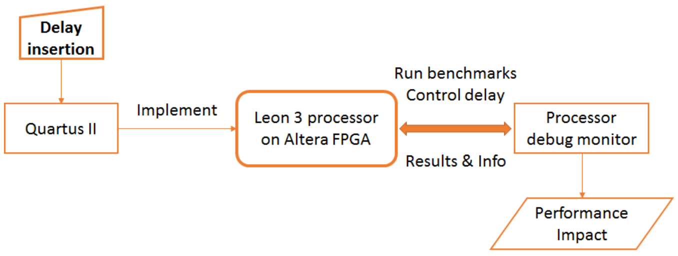

Figure 1. Framework of Leon3 processor emulator

In the project, a delay variability emulator is developed for a SPARC-V8 ISA based Leon 3 processor which is implemented on Altera De2-115 FPGA board. The emulator uses GRMON debug monitor to realize processor simulation, and can be used to conduct different types of experiments about delay variability impact on processor performance. The framework of the emulator is illustrated in Figure 1.

The delay insertion flow in this emulator is different from what is done previous based on a Xilinx FPGA board [2]. The advantage and disadvantage of the new method will be explained in this report.

The following of the report is organized as follow. Section II introduces the delay insertion flow and methodology; section III introduces the configuration of the Leon3 processor and how the emulator evaluates processor performance; section IV illustrates several experimental results; and section V concludes the project.

II. DELAY INSERTION AND VARIABILITY CONTROL

A. Delay insertion flow

The Leon 3 processor is the combination of multiple soft IP cores which are written in VHDL. The implementation flow of Leon 3 processor on Altera FPGA board is shown in Figure 2.

There are two methods of inserting the delay based on the implementation flow. The first one is to insert the delay after mapping and before fitting. In this method, after the processor is analyzed and mapped, and the critical paths of the processor can be selected according to a post-mapping timing analysis. The delay can be inserted into these critical paths by manually modifying the mapping output netlist which can be further used as the input of an automated place and route in Quartus software. This is a straightforward method, however, there are some disadvantages. Firstly, this method varies among different FPGA developing software, and it requires the software to provide an interface for users to be able to change the flow, in this case to play back the changes made in the fitting stage. In addition, inserting the delay based on critical paths does not give a clear look in the architecture level; and it also requires lots of work to insert the delay in a different unit in the processor.

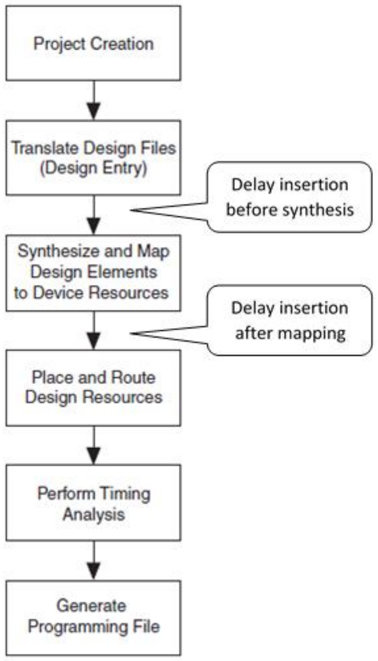
Project Creation
Translate Design Files (Design Entry)
Delay insertion before synthesis
Synthesize and Map Design Elements to Device Resources
Delay insertion after mapping
Place and Route Design Resources
Perform Timing Analysis
Generate Programming File

Figure 2. Processor implementation flow

The second method to insert the delay, on the other hand, is to directly insert them in the design phase in multiple units of the processor, that is, to add the delay modules and the control modules in the design source code. Since now the combined 'new' processor is a complete design, the rest of the flow can be finished by Quartus software. This method does not depend on the software interfaces, and it requires less work to insert the delay in different parts of the processor. Therefore, the method has strong portability and flexibility, and it makes it easier to study the performance impact of delay insertion in various units of the processor. The second method is selected for the advantages above in this project.

B. *The delay element module*

The delay element is implemented as a series of Altera Quartus predefined module *lcell*, which is simply one logic cell on FPGA with output value equals to input value. Each such logic cell will cause about 0.4 ns delay. A total number of 80 logic cells are used in one delay element, and a 7-bit selection vector controls the number of activated delay cells in the module. The structure of a delay element is shown in Figure 3. The post-fit simulation shows that the module can provide a range of 10 - 42 ns delay. The base delay of the module is 10 ns even when the select value is 0.

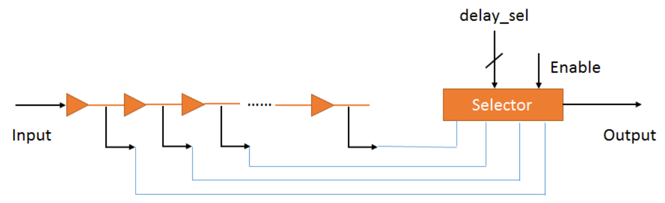

Figure 3. Delay element

C. *The general purpose register (GPREG)*

The general purpose register is a soft IP core provided with the Leon 3 processor. [3] The core takes in the value of a 32-bit register that is mapped to a memory address, and the value is propagated to an output vector.

D. *Controlling multiple delay elements*

The method to control delay elements in multiple paths is illustrated in Figure 4. The delay controller controls the enable bit of each delay element, while the GPREG controls the delay value as well as the delay controller. By writing certain value to the address that maps the GPREG, all the delay elements in each paths can be dynamically controlled.

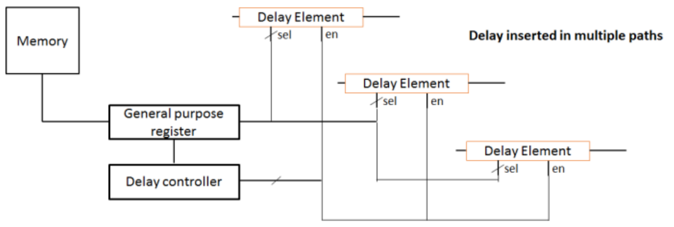

Figure 4. Controlling multiple delay elements

E. *Combine the delay element, control module and GPREG with the Leon 3 processor*

In order to insert the delay in the design stage, the three modules are combined with the Leon 3 processor in VHDL source file. Since GPREG is a regular IP core which is in the Leon 3 library, a direct instantiation works well through connecting the GPREG with a specific memory address. However, a direct instantiation of delay element will be optimized away by the software for it did nothing but to add unnecessary delay in the design. To prevent this, several steps needs to be done. Firstly, the delay elements and the controllers should be saved as post-fit LogicLock hard blocks through Quartus. Secondly, the delay elements and the controllers should be instantiated in the top level design of Leon 3 as empty wrapper modules which is connected with GPREG, and these instantiations needs to be set as 'design partitions' in Quartus. Finally, the hard blocks should be imported into those design partitions, and a full compilation should be executed for the Leon 3 processor.

There are two ways to choose the delay insertion paths. From hardware perspective, a post-mapping timing analysis of the original Leon3 can provide the critical paths with the smallest setup timing slack, and inserting the delay in those paths will cause setup timing violation. From the architectural perspective, specific units of the processor can be selected, and inserting the delay can force the paths in that unit to cause a setup timing violation. In this project, 16 delay elements are inserted in memory control module based on the first method, and 8 delay elements are inserted into different stages of the 7-stage pipeline of the processor. Figure 5 illustrate the location of the delay insertion.

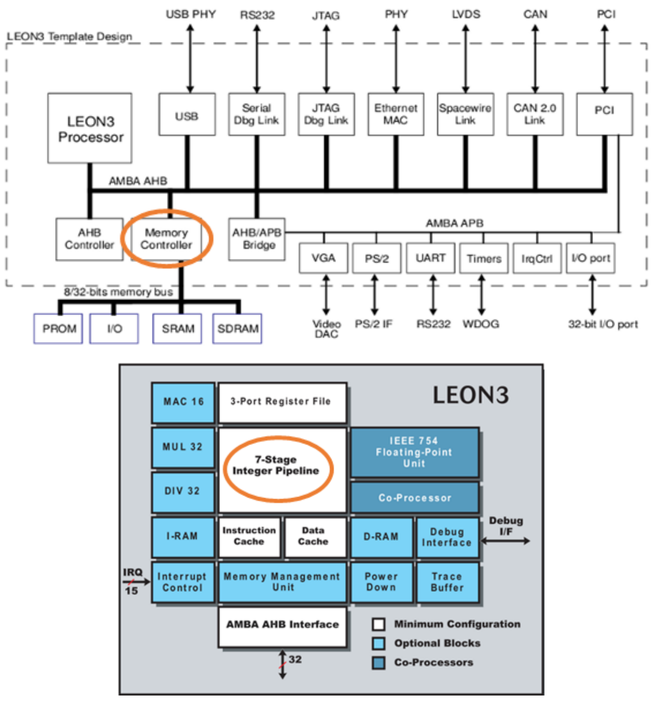

Figure 5. Delay insertion locations

The way to connect the delay element ports into the critical paths is a little tricky. Since the memory controller and the pipeline are instantiated many levels down from the top level design of Leon3, two ports have to be reserved for each of the delay element at each level of design, with one input port and one output port for the delay element. That is to say, a total of 48 ports should be created in top level Leon 3, with 32 ports going down to memory controller and 16 ports going down to pipeline through multiple levels. Notice that the input port of the delay element should be connect with the source signal in the memory controller or the pipeline module, while all signals which uses the value of the source signal should be connected with the output port of the delay element.

After the full compilation, the chip planner shows the fitting result of the Leon3 processor on the FPGA chip in Figure 6. Since the inserted delay are hard blocks scattered on the chip, there exists wiring delays in addition to the initial 10 ns base delay of the delay element. The extra delay causes the highest clock rate which the processor can work with decreasing from 90MHz to 70MHz, but this brings little influence to the focus of this emulator.

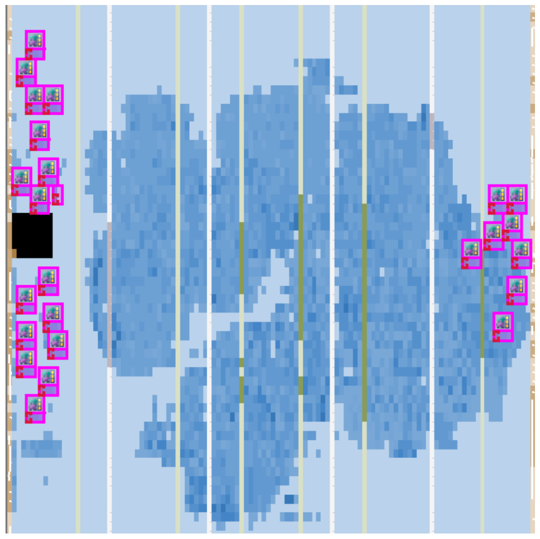

Figure 6. Chip planner results of the new processor

III. Leon 3 Processor Configuration and Performance Evaluation

A. *Leon 3 processor configuration*

In this project, a high performance configuration is used for the Leon 3 processor. The processor is set with debug support unit, floating point unit, separate instruction/data cache, memory management unit with TLB, and branch prediction unit with a predict-taken policy. [4] In addition, the working clock rate can be configured around a base clock rate of 50 MHz. The configuration will only be effective after full compilation.

B. *Performance evaluation of the Leon3 processor*

Four typical benchmarks are selected in this case, including Stanford, Dhrystone, Coremark and Sream, and they need to be compiled by sparc-gcc cross compiler in order to run on Leon 3 processor. [5] Debug monitor GRMON provides an interface to perform loading and running executables on FPGA, as well as to write or read memory value (which can be used to control delay elements). [6] The debug monitor can also run batch scripts, which make it possible for automatically running multiple programs and changing delay element control value on FPGA.

IV. Results

Three types of experiments are conducted using the Leon 3 delay emulator.

A. *Performance impact of different delay paths on different benchmarks, when delay is inserted in memory control module*

The critical paths in the same module may be different when benchmark is changed. In this experiment, four types of typical benchmarks are used. To study the impact of different delay paths in different benchmarks, a three dimensional sweep is done using scripts, with one of sixteen critical path enabled at a time, processor clock rate sweeping from 30 MHz to 70 MHz, and delay value sweeping from 0 to 80. If the delay value is too large for the processor to work under certain clock rate, the program which runs on the processor will end up with errors. Error types and details are listed in Table 1.

Table 1. Error types in experiment type 1

Error types	Details
Illegal instruction	Unknown opcode or 'unimp'
Memory address not aligned	'ld' & 'st' instructions, caused by wrong address in instruction
Data store error	Write buffer error
Hang	Keep outputting unreadable code
Instruction/Data access error	Error during instruction fetch and data load

Based on the sweeping results in Figure 7, the critical paths which has a big influence impact on performance varies among the four benchmarks. In addition, the impact of different paths in a single benchmark also have obvious difference.

B. *Performance impact of transient delay insertion, when delay is inserted in memory control module*

To randomly insert delay when benchmark is running on Leon 3, a new thread has to be created in the benchmark. The sparc-gcc cross compiler support library *lpthread*, which provides a way to change delay value for a transient time randomly by writing to the memory connected with GPREG. The program will fail with some probability with transient delay, and that probability will increase as the transient time interval grows. The result is shown in Figure 8 when running Stanford benchmark under 60 MHz.

C. *Performance impact of delay insertion, when delay is inserted in different units of the CPU*

The delay elements are also inserted in three stages in the processor pipeline: fetch stage, decode stage and execution stage. The fetch stage delay, which creates error in branch unit, will cause the program jumping among instruction segments and eventually ends with errors. The delay in decode and execution stage will cause data store error due to wrong memory address computed.

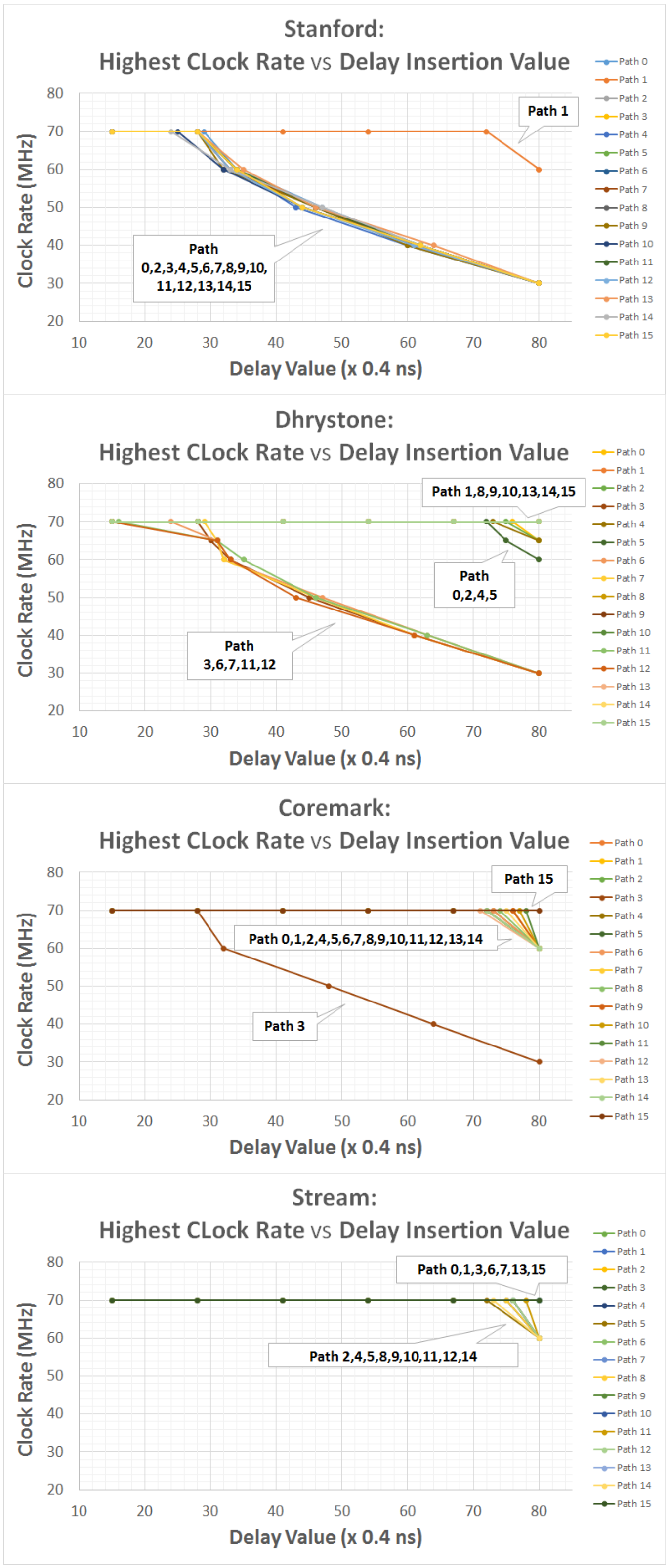

Figure 7. Experiment results: type A

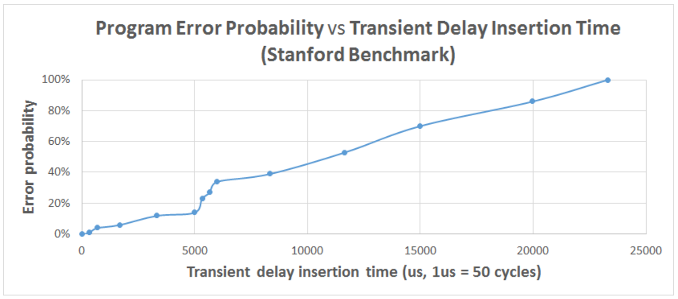

Figure 8. Experiment results: type B

D. *Further thoughts on performance impact*

The performance of a processor can be calculated based on equation (1).

*Execution time = Instruction count * CPI * Cycle Time (1)*

Experiment type A, B and C studied the delay impact of cycle time, which causes error when clock rate increases. Although it is not usual for hardware to influence instruction count, inserting the delay into speculation units of the processor, such as branch predictor or prefetching unit, can actually increase the instruction count by making those units work in a wrong way. Unfortunately, Leon3 processor is a basic and simple processor with limited speculation: it adopts a static branch prediction (just predict always taken) and it does not include a prefetching unit. Thus experimental results cannot be obtained.

V. Conclusion

An Altera FPGA based Leon 3 processor delay variability emulator is developed in the project. The emulator uses a portable and flexible method to implement controlled delay insertion, and several experiment results about delay insertion impact on performance have been obtained and analyzed.

With further support from debug monitor on performance detection, such as a detailed full program trace, the impact of the delay insertion can be analyzed in a more accurate way. In more sophisticated processors with speculation schemes, the delay insertion can influence instruction count in addition to cycle time.

References

[1] Puneet Gupta, et al, "Underdesigned and Opportunistic Computing in Presence of Hardware Variability," IEEE transactions on computer-aided design of integrated circuits and systems, vol. 32, no. 1, January 2013.

[2] Abhishek Bhatia, "VarLEON: FPGA Based Processor Variability Emulator for Variation Aware Software," 2014.

[3] Cobham website, "GRLIB IP Core User's Manual," Janurary 2016.

[4] Cobham website, "Configuration and Development Guide," Janurary 2016.

[5] Cobham website, "BCC User's Manual," December 2015.

[6] Cobham website, "GRMON2 User's Manual," May 2016.