Leon 3 Processor
Delay Variability Emulator

Nan Lyu
Project Background and Motivation

Modern microelectronics have increasing variations in performance, power and reliability.

Causes:

- Manufacturing:
  Material properties and scaling of semiconductor devices
  Worsen as critical dimension shrinks to atomic scale
Project Background and Motivation

Modern microelectronics have increasing variations in performance, power and reliability.

Causes:

• Manufacturing
• Environment

VDD: 10% floating; Temperature: -30 °C ~ 175 °C
Delay influenced by resistance and driving strength
Project Background and Motivation

Modern microelectronics have increasing variations in performance, power and reliability.

Causes:
• Manufacturing
• Environment
• Aging: Wear-out, performance decrease as usage time increases
Project Background and Motivation

The variation causes problems to both the hardware and software designer

Hardware designer
  Must estimate the variation conservatively
  Over design to guarantee performance

Software designer
  Assumes the performance limit as labeled
  A waste of performance
Project Background and Motivation

Aim of the project

To develop a delay variability emulator for Leon 3 processor:
- Emulate variability through delay insertion
- Show the impact of delay variability on performance
- Hopefully can be used for hardware-aware software development experiments
Framework

- **Delay insertion**
- **Quartus II**
- **Implement**
- **Leon 3 processor on Altera FPGA**
- **Run benchmarks**
- **Control delay**
- **Results & Info**
- **Processor debug monitor**
- **Performance Impact**
Leon 3 processor: Basic info

- Developed by COBHAM GAISLER
- Free licensed and open source embedded processor
- Consist of multiple soft IP cores (VHDL)
- Based on SPARC architecture (not popular now)
Leon 3 processor: Architecture

Configuration:

- Enable DSU and FPU
- Include separate cache
- Enable MMU (TLB)
- Enable branch prediction
Delay insertion and control
Delay insertion flow

Insert delay in design source file
✓ Portability
✓ Flexibility
✓ Clear look of location inserted
✗ Some extra delay

Insert delay after mapping in netlist
✓ Small extra delay
✗ Vary in different software
✗ Difficult to implement
✗ Poor flexibility
Delay element

Icell: one logic cell causes ~0.4 ns delay
Delay controller and General purpose register

Delay inserted in multiple paths
Choosing delay insertion paths

**Hardware perspective**
- Post-mapping timing analysis
- Select several paths with smallest timing slack (setup time)
- Choose a signal in the path

Better for highest working clock rate

**Architecture perspective**
- Select a unit
- Choose a signal inside the unit

Better for delay impact in different units
Choosing delay insertion paths

Hardware perspective

Architecture perspective
Combine delay with original Leon 3 design

**Constraint**
- GPREG should be instantiated in top level design (uses memory bus in top level)

**Methodology**
- Instantiate wrapper modules (empty) for delay elements and delay controllers in top level, set as design partitions, and connect with GPREG.
- Create ports in modules in each level from the top until reaching the desired insertion unit (two ports for each delay path).
- Connect specific signal with the ports, and connect ports with delay elements.
- Import hard macro (logic lock region) into the design partition.

**Flexibility**
- Manually create all the ports needed for several units at one time.
- The only thing to do is to change the port connection in top level.
Combine delay with original Leon 3 design

Delay scattered on the chip plan, which causes an extra base delay in the delay paths, and that may lead to a decrease in performance even if delay is disabled.
Performance evaluation

GRMON: Leon 3 processor debug monitor

• Processor Info
• Load program and run
• Standard I/O
• Write/Read memory
• Trace buffer
Performance evaluation

Benchmarks (use sparc-gcc cross compiler)

• **Stanford**: A suite of benchmarks that are relatively short, both in program size and execution time. (Non-floating & Floating)

• **Dhrystone**: A synthetic computing benchmark program intended to be representative of system (integer) programming

• **Coremark**: A benchmark that aims to measure the performance of CPU used in embedded systems

• **Stream**: A simple synthetic benchmark program that measures sustainable memory bandwidth
Results

Type 1: Performance impact of different delay paths on different benchmarks, when delay is inserted in memory control module.

Created automated batch and python script for running benchmark 16*80*5 times and extracting results.
Results

Error Type:

• **Illegal instruction** (‘unimp’ or unknown opcode)

• **Memory address not aligned** (‘ld’ and ‘st’, caused by incorrect instruction content)

• **Data store error** (write buffer error)

• **Hang** (output unreadable code)

• **Instruction/Data access exception** (Error during instruction fetch and data load)
Results

Stanford:
Highest Clock Rate vs Delay Insertion Value

Dhrystone:
Highest Clock Rate vs Delay Insertion Value

Coremark:
Highest Clock Rate vs Delay Insertion Value

Stream:
Highest Clock Rate vs Delay Insertion Value

Path 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Path 0, 1, 3, 6, 7, 13, 15

Path 2, 4, 5, 8, 9, 10, 11, 12, 14
Results

*Type 2: Performance impact of transient delay insertion, when delay is inserted in memory control module*

Create a new thread in Stanford benchmark to randomly insert delay for a short period in parallel with the benchmark main thread.

Connect GPREG output with LEDs to show delay value. LEDs will blink during the program runtime if the program fortunately exits normally; otherwise the LEDs will stuck at light on with a program error.
Results

Benchmark is run 100 times for each data point
Results

Type 3: Performance impact of delay insertion, when delay is inserted in 7-stage pipeline

Fetch stage (branch controller)
Error type: Hang for a long time, then end up with error

Execution stage
Error type: Data store error

Decode stage
Error type: Data store error
Results

Further thoughts..
Execution time = $IC \times CPI \times CT$

Delay insertion can have influence on:

**Cycle time**: A high clock frequency can lead to program running incorrectly

**Instruction count**: Mess up the speculation units in processor to increase instruction count (branch prediction, prefetching, ..)

- Speculation units needed (Leon 3 is simple)
- Detailed program trace needed for analysis (not available in GRMON)
Conclusion

What has been done
• Implemented the delay variability emulation flow for Leon 3 processor
• Obtained and analyzed several performance experiments results

What can be done further
• Insert delay in more units
• Use a more sophisticated processor
• With more support from GRMON (may need a paid license), precise impact can be studied from detailed program trace (trace buffer only 4KB now)
Thank you