Safe Overclocking of Tightly Coupled CGRAs and Processor Arrays using Razor

Alex Brant, Ameer Abdelhadi, Douglas Sim, Shao Lin Tang, Michael Yue, Guy Lemieux

© 2012 Guy Lemieux
Overclocking… is it safe?

- Clock frequency determined by 2 things:
  - CAD timing analysis (timing margins)
  - speed binning of actual wafers + chips (variation)

- Can you go faster?
  - Yes, if your chips are fast
  - Yes, if your data is not “worst-case”, eg carry propagation
  - Yes, if you do not want “safe” timing margin guardbands
Overclocking... is it safe?

3.8GHz  4.2 V

3.818GHz !!!  4.210V
Overclocking… is it safe?

• How fast is too fast?
  – Blows up
  – Fails to POST
  – Fails to boot
  – Blue screen of death
  – Random crashes
  – Data errors in documents and spreadsheet

• When these problems go away, is it safe?
Overclocking... is it safe?

• Root cause: timing errors
  – Problem 1: can we detect them?
    • Yes, e.g. using Razor

  – Problem 2: can we correct them?
    • Yes, using Razor with feed-forward pipelines
      – Pipeline must be ‘replayed’, input data ‘unfetched’
    • Not possible with general sequential logic
      – Need ‘spare’ cycles to ‘unfetch’ input data
      – Cyclic dependencies make this difficult
Overclocking… is it safe?

• If not for general logic, what about…

  – Traditional CPU pipelines?
  
  Yes:
  • Feed-forward, correctable by Razor-replay

  – Multi-core CPUs?
  
  Yes:
  • Each CPU is a traditional pipeline
  • Loosely coupled
    – Other CPUs tolerate race conditions
Overclocking… is it safe?

• If not for general logic, what about…

  – Ambric-style processor arrays?

    Yes:
    • Like multi-core CPUs
    • Loosely coupled
      – Neighbour CPUs tolerate uncertainty of arrival time

  – Tightly coupled processor arrays or CGRAs?

    No: neighbour CPUs cannot tolerate delays!
Main Contribution

• Extends Razor error correction to...
  – tightly coupled processor arrays, CGRAs
  – time-multiplexed FPGAs/CGRAs

Tightly coupled means...

• Pre-scheduled communication
  – Data must be present during cycle X

• No “data presence” indicators / handshakes
Razor FF in Pipeline

 clk

 clk + delay

 Main Flip-Flop

 Shadow Latch

 Error_L

 comparator
Razor FF in Pipeline
How can we overclock?

10 + 9 + 8 = 27ps clock
How can we overclock?

$9 + 8 = 17\text{ps clock, most of the time}$
How can we overclock?

9+8 = 17ps clock, most of the time
How can we overclock?

9+8 = 17ps clock, most of the time
How can we overclock?

$9 + 8 = 17 \text{ps clock, most of the time}$
How can we overclock?

$9 + 8 = 17\text{ps clock, most of the time}$
How can we overclock?

9+8 = 17ps clock, most of the time
Array Architecture

Tightly coupled communication:
- can be FIFO-based or ‘mailbox’-based

Fully bypassed:
- write on cycle X
- read on cycle X+1, ie address provided cycle X
Add “Razor” to Block RAM
Processor Error Detection
(for East direction only)

Incoming stall (from N,S,W): produces Outgoing E stall

Processor memory error: Causes stall

Early warning signal: prevents incoming data
Processor Error Detection
(all four directions)
Writes entering error region....
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)
Razor Stall Propagation (1D)

3 Errors Detected
2 Stalls to Correct
# STALLS < # ERRORS

Might be scalable!!
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)
Razor Stall Propagation (2D)

3 Errors Detected
2 Stalls to Correct
# STALLS < # ERRORS

Might be scalable!!
Manual experiment: 
# stalls vs # errors

A graph showing the relationship between the number of stalls and the number of errors, with errors increasing as stalls increase.
Monte Carlo Simulations…
Stalls vs Errors (N x N array)
Recovered Utilization (N x N)
Experimental Results...
Experimental Results

• Build Processor Array on FPGA…
  – 2 x 2 array in silicon (running)
  – 3 x 3 array in simulation (verification)

• Static critical path always through ALU + communication channels
  – Typically multiplier, but only if used (!)
  – Depends upon values being multiplied

• Overclock system
  – $F_{\text{max}}$ depends upon multiplier use, data values
Place & Route Results

• Area analysis for processor
  – 2,958 ALMs + 304 Regs (baseline)
  – 3,082 ALMs + 517 Regs (with Razor)

• Static timing analysis for array
  – 90 MHz (baseline)
  – 88 MHz (with Razor)

• Overhead is very low (4% ALMs)
Methodology (baseline)

• Run once: circuit at low speed
  – Record correct output vectors

• For increasingly higher clock speeds
  – Run circuit with input test vectors
  – Fail on first error

• Remember highest clock speed
Results (baseline)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Static Timing (CAD)</th>
<th>Overclocked (1st error)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>90 MHz</td>
<td>135 MHz</td>
</tr>
<tr>
<td>Mean</td>
<td>90 MHz</td>
<td>121 MHz</td>
</tr>
<tr>
<td>Wang</td>
<td>90 MHz</td>
<td>131 MHz</td>
</tr>
<tr>
<td>PR</td>
<td>90 MHz</td>
<td>136 MHz</td>
</tr>
<tr>
<td>average</td>
<td>90 MHz</td>
<td>130.4 MHz</td>
</tr>
</tbody>
</table>

- Processor arrays can be overclocked
  - Amount depends on application + data + chip
- But is it safe?
  - Our “test jig” tested results offline to find errors
  
Unsafe: baseline cannot detect errors
Methodology (Razor)

• Run once: circuit at low speed
  – Record correct output vectors

• For increasingly higher clock speeds
  – For increasingly higher shadow FF delay
    • Run circuit
    • Record # errors, # corrected errors, # stalls

• Remember highest throughput
## Results (baseline)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Static Timing (CAD)</th>
<th>Overclocked (runs past 1st error)</th>
<th>Stall Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>88 MHz</td>
<td>163 MHz</td>
<td>5.0%</td>
</tr>
<tr>
<td>Mean</td>
<td>88 MHz</td>
<td>144 MHz</td>
<td>1.3%</td>
</tr>
<tr>
<td>Wang</td>
<td>88 MHz</td>
<td>147 MHz</td>
<td>0.7%</td>
</tr>
<tr>
<td>PR</td>
<td>88 MHz</td>
<td>145 MHz</td>
<td>1.7%</td>
</tr>
<tr>
<td>average</td>
<td>88 MHz</td>
<td>149.4 MHz</td>
<td>2.0%</td>
</tr>
</tbody>
</table>

• Processor arrays can be overclocked
  – Even higher rates past 1st error
  – Errors require stalls to correct, lowers thru-put
  – Stop increasing Fmax after thru-put peaks
## Results (new Razor)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Static Timing (CAD)</th>
<th>Overclocked (runs past 1\textsuperscript{st} error)</th>
<th>Stall Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>88 MHz</td>
<td>163 MHz</td>
<td>5.0%</td>
</tr>
<tr>
<td>Mean</td>
<td>88 MHz</td>
<td>144 MHz</td>
<td>1.3%</td>
</tr>
<tr>
<td>Wang</td>
<td>88 MHz</td>
<td>147 MHz</td>
<td>0.7%</td>
</tr>
<tr>
<td>PR</td>
<td>88 MHz</td>
<td>145 MHz</td>
<td>1.7%</td>
</tr>
<tr>
<td>average</td>
<td>88 MHz</td>
<td>149.4 MHz</td>
<td>2.0%</td>
</tr>
</tbody>
</table>

- But is it safe?
  - Safe! Razor detects and corrects errors
  - Our “test jig” tested results offline to verify the errors were corrected
**Results (Comparison)**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Baseline STA (safe)</th>
<th>Razor-Corrected Effective Throughput</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>90 MHz</td>
<td>155 MHz</td>
<td>1.72 x</td>
</tr>
<tr>
<td>Mean</td>
<td>90 MHz</td>
<td>142 MHz</td>
<td>1.58 x</td>
</tr>
<tr>
<td>Wang</td>
<td>90 MHz</td>
<td>146 MHz</td>
<td>1.62 x</td>
</tr>
<tr>
<td>PR</td>
<td>90 MHz</td>
<td>143 MHz</td>
<td>1.59 x</td>
</tr>
<tr>
<td>average</td>
<td>90 MHz</td>
<td>146.5 MHz</td>
<td>1.63 x</td>
</tr>
</tbody>
</table>

- Processor arrays can be **safely** overclocked
  - 63% higher throughput
- Low area cost (+4% ALMs)
Observations / Notes

• Time-multiplexed CGRAs/FPGAs can also benefit
  – Just reserve 1-2 clock cycles in the time-mux schedule for error recovery

• Loosely coupled processor arrays can be overclocked locally
  – Just add Razor to each processor
  – No need to propagate stall signals; automatically done through data presence indicators
Summary / Conclusions

• Processor arrays can be safely overclocked
  – Even with very tightly scheduled communication

• Processor arrays are scalable
  – Errors produce stall wavefronts
  – Several wavefronts merge into a single stall cycle

• Throughput increased 63% on average
  – Speedup depends upon benchmark, data values