

Hardening an academic core for Industrial Use

Roger Espasa

SemiDynamics

2020.06.29

A few words about SemiDynamics

- Small Startup, European based
- Previous project: 7nm Machine Learning Accelerator
 - Started from an academic core, then heavily adapted it to target needs
- Currently developing a RISC-V OOO core with a vector unit for the upcoming European Processor Initiative accelerator
 - Started from an academic core
 - Added to it the RISC-V vector extension (still WIP)
 - Heavily modified the Load/Store unit
 - Extensive re-verification

Academic cores are a great starting point...

...for startups
...for big companies
...for fast prototyping

...used Rocket (UC Berkeley)
...used Ariane (ETH)
...very happy with both

Yet, there's a bit of work before
hitting volume...

(still, you're much better off starting from these academic cores!)

Source Code

- Rocket: Chisel(Scala)
 - Authors claim big productivity gains
- Ariane: System Verilog
 - Better matches industry tools
- In general, very good coding styles
 - Clear, easy to ready code
 - In Chisel case, you need to grasp Chisel, of course! 😊
- Good Assertions and coverage points present in code
- (personal preference) flip flops should be hidden inside macro
 - ``RST_EN_FF (clk, d, q, en, rst)`
 - Instead of “always_ff @(posedge clk or negedge)”
 - Forces “state” out of FSMs....

Verification

- Academic cores successfully boot Linux
 - i.e., They start at pretty good quality
- Yet you'll want to invest in additional verification
 - Additional Coverage points & Assertions
 - Additional Randoms
 - U-S-M transitions, Interrupts
 - Compressed instructions corner cases

A taste of issues found on FP logic

- minimum negative value must set invalid bit in fcvt
- Underflow when result is denormal
- sticky bit incorrectly set on exact SQRT, rounding caused a ulp difference.
- rounding issues on fcvt for signed conversions on maximum positive
- Underflow should only be set if result is also inexact
- NV flag was incorrectly computed for infinite cases
- 0/0 incorrectly set the DZ flag
- issue with the sticky bit and the ulp for SP values
- FCVT instructions didn't take infinities as special values
- UF and OF were swapped in case the result is below the minimal value
- Sticky bit was set to 0 in case of underflow resulting in incorrect NX bit
- ... and another 20....

Load/Store Unit

- Not all academic cores support coherency; most are AXI based
- “Miss pipeline” typically under-optimized; opportunities for improving
 - From blocking to non-blocking
 - From N-stage FSM to 1-stage FSM
 - Hit-under-miss?
 - Hit-under-fill?
 - How many misses?
 - Write-back?
- Re-pipeline to match your SRAM array technology
 - Off-the-shelf code may or may not match your needs
- For 512b read-out, may need to reconsider how ways are read/flopped
- After optimization, need to re-do debug support
- Change PMA to match your SoC Memory Map
- Cache management ops missing (this is a RISC-V definition gap!)

Low-level clock gating

- Commercial tools very good at inferring clock gating
- But they can always use a bit of help
 - Expose flip-flop “enable” clearly
 - Regional clock gates are not that obvious for tools
- Most ROI comes from fine-grain enables
 - “turn off this block when X && Y && Z
- Large module gating good for standby
 - Turn off all “core” / “front-end” / “lsu”
 - Needs hand intervention

SoC glue

- Both Rocket and PULP come with lots of goodies
 - PLIC, CLINT, AXI Masters, Slaves, Debug Module,
 - Synchronizers, level shifters...
- Yet Likely your SoC requires something special
 - Lots of cores ?
 - Distribution of debug signals
 - Inter-core signaling (events? IPI? Other?)
 - Special NoC
- Errors
 - i.e., What to do on an AXI device error? – will require re-plumbing
 - If you have ECC, how/where/when do you report sbe and dbe?
- Fuses
- Secure Boot

Summary

- Overall, academic cores are an **excellent starting point**
 - Boot Linux
 - Well written
 - Well supported
- You'll obviously have to add work to go into production
 - Additional verification to make sure corner cases covered
 - Depending on target market, improve cache subsystem performance
 - Depending on power targets, improve clock gating
 - Add SoC dependent features
- **Strong recommendation to start from an academic core!**