Raven: A 28nm RISC-V Vector Processor with Integrated Switched-Capacitor DC-DC Converters and Adaptive Clocking

Yunsup Lee, Brian Zimmer, Andrew Waterman, Alberto Puggelli, Jaehwa Kwak, Ruzica Jevtic, Ben Keller, Stevo Bailey, Milovan Blagojevic, Pi-Feng Chiu, Henry Cook, Rimas Avizienis, Brian Richards, Elad Alon, Borivoje Nikolic, Krste Asanovic

University of California, Berkeley
Motivation

- Energy efficiency constrains everything
  - SoCs are designed with an increasing number of voltage domains for better power management
  - Dynamic voltage and frequency scaling (DVFS) maximizes energy efficiency while meeting performance constraints

- Off-chip conversion
  - Few voltage domains
  - Costly off-chip components
  - Slow mode transitions

- On-chip conversion
  - Many domains
  - No off-chip components
  - Fast transitions
iPhone 6

Front Side

Voltage Regulators

Back Side

Primary Power Management IC

Secondary Power Management IC

Credits: iFixit
Raven Project Goals

Build a microprocessor that is:

- **Energy-efficient**
  - Fine-grained DVFS
  - High conversion efficiency

- **Low-cost**
  - Entirely on-chip converter
  - Low area overhead
Talk Outline

- Motivation/Raven Project Goals
- On-Chip Switched-Capacitor DC-DC Converters
- Raven3 Chip Architecture
- Raven3 Implementation
- Raven3 Evaluation
- RISC-V Chip Building at UC Berkeley
- Summary

For more details on the Switched-Capacitor DC-DC Converters, please take a look at HC23 tutorial “Fully Integrated Switched-Capacitor DC-DC Conversion” by Elad Alon
Switched-Capacitor (SC) DC-DC Converters

- 0.6V + Vout=0.6V

- 0.4V + Vout=0.4V

+ 0.6V - Vout=0.4V

+ 0.4V - Vout=0.6V
SC DC-DC Converters Cont’d

- Partition capacitor and switches into many “unit cells” for better modularity
- Keeps the custom design modular
- Makes it easier to floorplan SC DC-DC converter
Traditional Approach: Interleaved Switching

- Switch one unit cell at a time to smooth out voltage ripple

  ![Diagram of interleaved switching](image)

- Pros
  - Voltage ripple at the output is suppressed
  - Great for digital designs with fixed frequency clocks

- Cons
  - Each unit cell charge shares with other unit cells
  - Causes an efficiency loss beyond typical switching losses
Raven’s Approach: Simultaneous Switching

- Switch all unit cells simultaneously when $V_{out}$ reaches a lower bound $V_{ref}$

- Pros
  - Simplifies the design
  - No charge sharing losses
  - Better energy efficiency

- Cons
  - Need to deal with big ripple on voltage output

- Add an adaptive clock generator so that clock frequency tracks the voltage ripple
Self-Adjusting Clock Generator

Block Diagram

- Replica tracks critical path with voltage ripple
- Controller quantizes clock edge

(Tunable Replica Path)

Adaptive system clock

Tunable Replica Path

Controller

Select closest edge

DLL

2Ghz input, 16 phases

V_{out}
Reconfigurable SC Converters for DVFS

- Simple lower bound control

- Operational Waveform
RISC-V is a new, open, and completely free general-purpose instruction set architecture (ISA) developed at UC Berkeley starting in 2010.

RISC-V is simple and a clean-slate design.
- The base (enough to boot Linux and run modern software stack) has less than 50 instructions.

RISC-V is modular and has been designed to be flexible and extensible.
- Better integrate accelerators with host cores.

RISC-V software stack:
- GNU tools (GCC/Binutils/glibc/newlib/GDB), LLVM/Clang, Linux, Yocto (OpenJDK, Python, Scala).

Checkout [http://riscv.org](http://riscv.org) for more details.
Rocket Scalar Core

- 64-bit 5-stage single-issue in-order pipeline
- Design minimizes impact of long clock-to-output delays of SRAMs
- 64-entry BTB, 256-entry BHT, 2-entry RAS
- MMU supports page-based virtual memory
- IEEE 754-2008-compliant FPU
- Supports SP, DP Fused-Multiply-Add (FMA) with HW support for subnormals
### ARM Cortex-A5 vs. RISC-V Rocket

<table>
<thead>
<tr>
<th>Category</th>
<th>ARM Cortex-A5</th>
<th>RISC-V Rocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA</td>
<td>32-bit ARM v7</td>
<td>64-bit RISC-V v2</td>
</tr>
<tr>
<td>Architecture</td>
<td>Single-Issue In-Order</td>
<td>Single-Issue In-Order 5-stage</td>
</tr>
<tr>
<td>Performance</td>
<td>1.57 DMIPS/MHz</td>
<td>1.72 DMIPS/MHz</td>
</tr>
<tr>
<td>Process</td>
<td>TSMC 40GPLUS</td>
<td>TSMC 40GPLUS</td>
</tr>
<tr>
<td>Area w/o Caches</td>
<td>0.27 mm²</td>
<td>0.14 mm²</td>
</tr>
<tr>
<td>Area with 16K Caches</td>
<td>0.53 mm²</td>
<td>0.39 mm²</td>
</tr>
<tr>
<td>Area Efficiency</td>
<td>2.96 DMIPS/MHz/mm²</td>
<td>4.41 DMIPS/MHz/mm²</td>
</tr>
<tr>
<td>Frequency</td>
<td>&gt;1GHz</td>
<td>&gt;1GHz</td>
</tr>
<tr>
<td>Dynamic Power</td>
<td>&lt;0.08 mW/MHz</td>
<td>0.034 mW/MHz</td>
</tr>
</tbody>
</table>

- **PPA reporting conditions**
  - 85% utilization, use Dhrystone for benchmark, frequency/power at TT 0.9V 25C, all regular Vt transistors
  - 10% higher in DMIPS/MHz, 49% more area-efficient
Hwacha Vector Accelerator
1.3mm X 1.8mm Raven3 Chip
Fabricated in ST 28nm FDSOI
(Fully Depleted Silicon-on-Insulator)
Chip area: 2.34 mm²

- Single-Core RISC-V Processor with a Vector Accelerator
  Core area: 1.19mm²

- Integrated Switched-Capacitor DC-DC Converter
  Converter area: 0.19mm²
  Area overhead: 16%

- Adaptive Clock Generator

- 122 pins total
  60 for signals
  62 for power/ground

- 971 MHz
  34 GFLOPS/W w/o Converter
  26 GFLOPS/W w/ Converter
Raven3 Test setup

- Daughterboard: Chip-on-board
- Motherboard: Programmable supplies
- Host: Zedboard (ZYNQ FPGA, network accessible)
Raven3 Voltage Measurements

SC-DCDC Modes
- Blue: 1V
- Green: 1.8V 1/2
- Red: 1V 2/3
- Cyan: 1V 1/2

Measured Output Voltage vs. Time (us)

<20ns transition

Time (ns)
Measuring Efficiency

- Traditional efficiency measurement is not appropriate for our design
  1. Difficult to measure on-chip current and voltage
  2. Cannot measure adaptive clock overhead

\[
P_{\text{in}} = V_{\text{in}} \times I_{\text{in}} \quad P_{\text{out}} = V_{\text{out}} \times I_{\text{out}}
\]

Efficiency = \( \frac{P_{\text{out}}}{P_{\text{in}}} \)
Raven3 Achieves >80% System Efficiency

- System efficiency = Ratio of energy required to finish same workload in same time with the converter versus without the converter.

![Measured system efficiency diagram](chart.png)
Raven3 Achieves 34 Double-Precision GFLOPS/W

- Double-precision matrix-multiplication running on vector unit used as energy-efficiency metric
  - 34 GFLOPS/W without DC-DC converter
  - 26 GFLOPS/W with DC-DC converter
  - Note, GFLOPS/W is inverse of nJ/FLOP
Five 28nm & Six 45nm RISC-V Chips Taped Out So Far
Agile Hardware Development Methodology

- **“Tape-in”:** Designs that could be taped out
  - LVS clean & DRC sane
  - Pass RTL/gate-level simulation and timing

- **Fully scripted ASIC flow**
  - RTL change to chip <1d
  - Get early feedback
  - Automatic nightly regressions
  - Identify source of subtle bugs

- **Check longer programs on FPGA**

- **Iterate quickly on RTL with using the C++ emulator (Checkout chisel.eecs.berkeley.edu for more details)**
Summary

- 28nm Raven3 processor features:
  - Fine-grained, wide-range DVFS (20ns, 0.45-1V)
  - Entirely on-chip voltage conversion
  - High system efficiency (>80%)
  - Extreme energy efficiency (34/26 GFLOPS/W)

- Key enablers
  - RISC-V, a simple, yet powerful ISA (free and open!)
  - Agile development, build hardware like software
  - Chisel, lets designers do more things with less effort

- RISC-V core generators and software tools open-sourced at [http://riscv.org](http://riscv.org)

- Energy-efficient, low-cost on-chip DC/DC converters are buildable!
Acknowledgements

- Funding: BWRC members, ASPIRE members, DARPA PERFECT Award Number HR0011-12-2-0016, Intel ARO, AMD, GRC, Marie Curie FP7, NSF GRFP, NVIDIA Fellowship
- Fabrication donation by STMicroelectronics
- Tom Burd, James Dunn, Olivier Thomas, and Andrei Vladimirescu
SUPPLEMENTAL SLIDES
"Rocket Chip" SoC Generator
Who should use the Rocket Chip Generator?

1. Change Parameters
2. Develop New Accelerators
3. Develop Own RISC-V Core
4. Develop Own Device
- Chisel is a new HDL embedded in Scala
  - Rely on good software engineering practices such as abstraction, reuse, object oriented programming, functional programming
  - Build hardware like software
- Chisel is not a high-level synthesis tool
  - It is a program that writes RTL
Functional Programming 101

- (1, 2, 3) map { n => n+1 }
  Result: (2, 3, 4)

- (1, 2, 3) zip (a, b, c)
  Result: ((1, a), (2, b), (3, c))

- ((1, a), (2, b), (3, c)) map { case (left, right) => left }
  Result: (1, 2, 3)

- (1, 2, 3) foreach { n => print(n) }
  Result: “123”

- for (x <- 1 to 3; y <- 1 to 3) yield (x, y)
  Result: ((1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3))
class AHBXbar(nMasters: Int, nSlaves: Int) extends Module {
  val io = new Bundle {
    val masters = Vec.fill(nMasters){new AHBMasterIO}.flip
    val slaves = Vec.fill(nSlaves){new AHBSlaveIO}.flip
  }

  val buses = List.fill(nMasters){ Module(new AHBBus(nSlaves)) }
  val muxes = List.fill(nSlaves){ Module(new AHBSlaveMux(nMasters)) }

(buses.map(b => b.io.master) zip io.masters) foreach { case (b, m) => b <> m }
(muxes.map(m => m.io.out) zip io.slaves) foreach { case (x, s) => x <> s }
for (m <- 0 until nMasters; s <- 0 until nSlaves) yield {
  buses(m).io.slaves(s) <> muxes(s).io.ins(m)
}
- TileLinkIO consists of Acquire, Probe, Release, Grant, Finish
- UncachedTileLinkIO consists of Acquire, Grant, Finish
- Convertors for TileLinkIO/UncachedTileLinkIO in uncore library
- Rocket sends coprocessor instruction via the Cmd interface
- Accelerator responds through Resp interface
- Accelerator sends memory requests to L1D$ via CacheIO
- busy bit for fences
- IRQ, S, exception bit used for virtualization
- UncachedTileLinkIO for instruction cache on accelerator
- PTWIO for page-table walker ports on accelerator
After a 2-cycle initial startup latency, the banked RF is effectively able to read out 2 operands/cycle.
Our Physical Design Flow

Chisel Source Code

Chisel

RTL Code (Verilog)

Synthesis
Place-and-Route

Hierarchical SYN & PNR
UPF-based MV SYN & PNR

Gate-level Netlist

Formality
Formal Verification

PrimeTime/StarRC
Static Timing Analysis

VCS Post-PNR
Gate-level Simulation

Signed-Off Design
What’s Different about RISC-V?

- **Simple**
  - Far smaller than other commercial ISAs
- **Clean-slate design**
  - Clear separation between user and privileged ISA
  - Avoids µarchitecture or technology-dependent features
- **A modular ISA**
  - Small standard base ISA
  - Multiple standard extensions
- **Designed for extensibility/specialization**
  - Variable-length instruction encoding
  - Vast opcode space available for instruction-set extensions
- **Stable**
  - Base and standard extensions are frozen
  - Additions via optional extensions, not new versions
- **RISC-V** is smallest ISA for 32- and 64-bit addresses
  - Average 34% smaller for RV32C, 42% smaller for RV64C
RISC-V Documentation, Software Ecosystem

- **Documentation**
  - RISC-V User-Level ISA Specification V2
  - RISC-V Compressed ISA Specification V1.7
  - RISC-V Privileged-Level ISA Specification V1.7

- **Modern Software Stack**
  - GCC/Binutils/glibc/newlib/GDB
  - LLVM/Clang
  - Linux
  - Yocto (OpenJDK, Python, Scala, …)

- **RISC-V Software Implementation**
  - ANGEL JS ISA Simulator
  - Spike ISA Simulator
  - QEMU

- **Checkout** [http://riscv.org](http://riscv.org) for more details
**RISC-V Core Generators from UC Berkeley**

- **Z-scale**: Family of Tiny Cores
  - Similar class: ARM Cortex M0/M0+/M3/M4
  - Integrates with AHB-Lite interconnect
  - Open-Sourced

- **Rocket**: Family of In-order Cores
  - Currently 64-bit single-issue only
  - Plans to work on dual-issue, 32-bit options
  - Similar class: ARM Cortex A5/A7/A53
  - Will integrate with AXI4 interconnect
  - Open-Sourced

- **BOOM**: Family of Out-of-Order Cores
  - Supports 64-bit single-, dual-, quad-issue
  - Similar class: ARM Cortex A9/A15/A57
  - Will integrate with AXI4 interconnect
CoreMark Scores

Checkout Chris Celio’s “BOOM: Berkeley Out-of-Order Machine” talk from 2nd RISC-V Workshop for more details
Glossary

- DVFS: Dynamic Voltage and Frequency Scaling
- SC: Switched Capacitor
- DLL: Delay-Locked Loop
- LDO: Low-Dropout Regulator
- FDSOI: Fully Depleted Silicon-on-Insulator
- FMA: Fused-Multiply-Add
- BTB: Branch Target Buffer
- BHT: Branch History Table
- RAS: Return Address Stack