Opportunities for computer architecture research with open-source hardware

The case of RISC-V

Olivier Sentieys

Univ. Rennes, Inria, Irisa, Taran

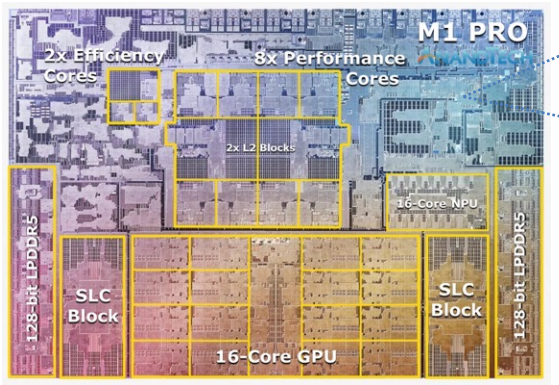
olivier.sentieys@inria.fr

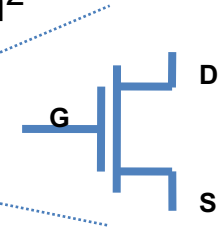
Key Messages

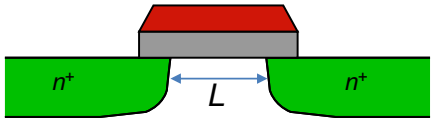
- This talk is about computer architecture
  - A key enabler for research and innovation in many fields of CS
  - Do we still need computer architecture (and compiler) research?
- Is Moore's law really ending?
  - No more free lunch
  - A new golden age for computer architecture
- RISC-V: a standardized, free and open ISA
  - The Linux of hardware?
- RISC-V and open hardware
  - Key enablers for computer architecture research

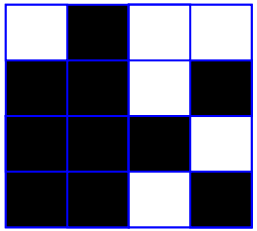
Silicon Technology Evolution

- Now several billions or transistors!
  - Apple M1: 33 B.Tr, 5nm, 2.5cm^2







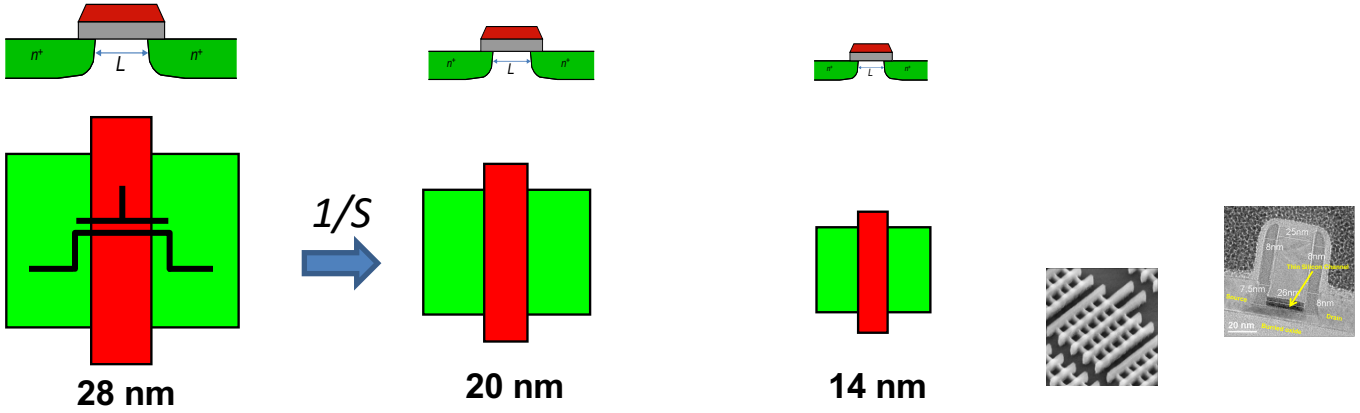


Dark Silicon

A short interlude on Dark Silicon

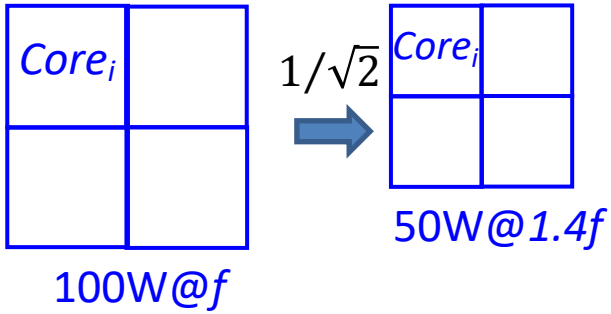
and the several walls of computer architectures

Technology Scaling



Classical (Dennard's) scaling

Device count	S^2
Device frequency	S
Capacitance, Vdd	1/S
Device power	$1/S^2$
Utilization	**1**



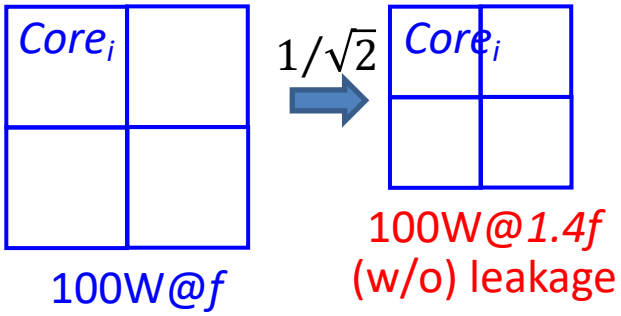
End of Dennard's Scaling

- No more free lunch! Energy efficiency is not scaling along with integration capacity

(since ~2016, 28nm)

Leakage limited scaling

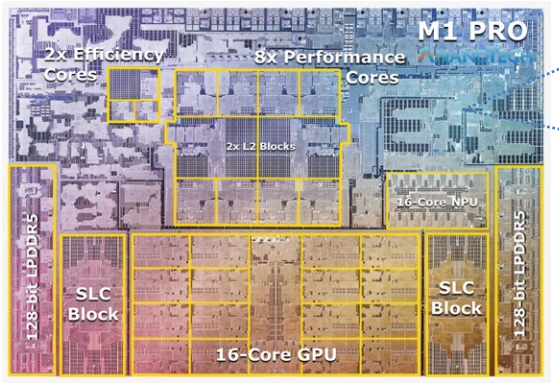
Device count	S^2
Device frequency	~S
Device power (cap)	1/S
Device power (V_{dd})	**~1**
Utilization	**$1/S^2$**

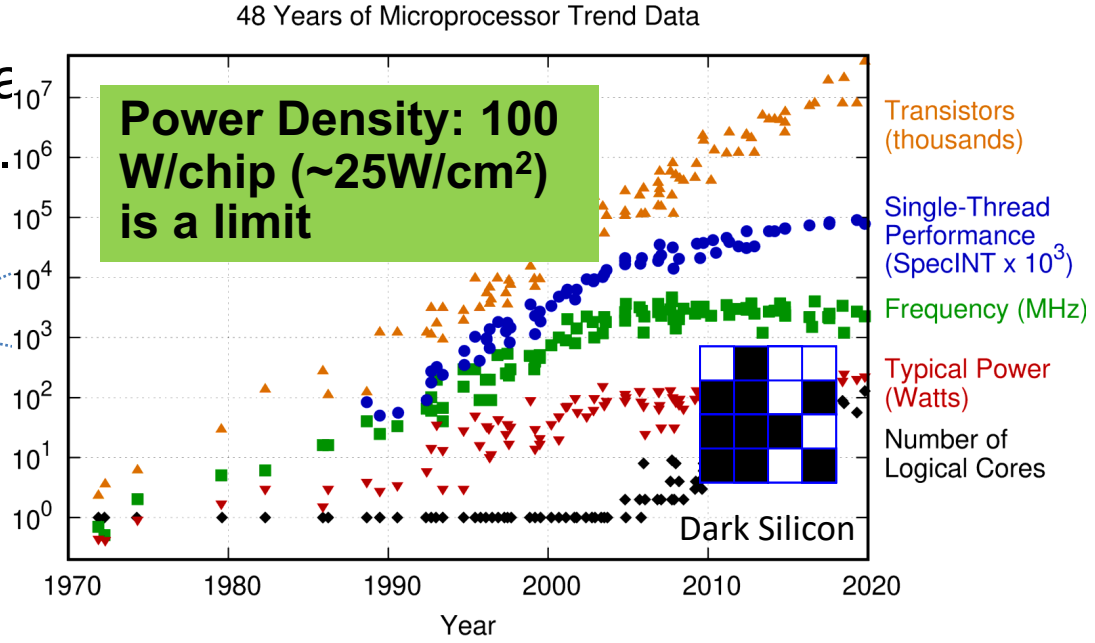


More Computing Power?
More Electrical Power...
or Domain-Specific Computers

Silicon Technology Evolution

- Now several billions or tra
  - Apple M1: 33 B.Tr, 5nm, 2.





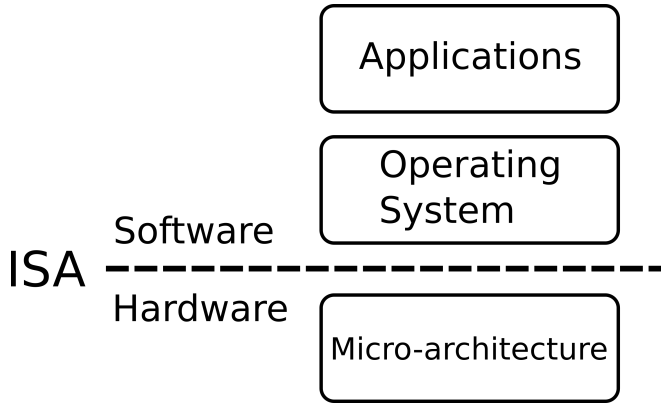
- No more free lunch!
  - No more scaling of performance and energy with technology
  - End of technology (Moore's law) scaling is near

Back to RISC-V

Instruction Set Architecture

Instruction Set Architecture (ISA)

- How does the compiler know what instructions the chip understands?
- Instruction Set Architecture
  - abstract model of a computer
  - interface between HW and SW
  - Examples: x86 (Intel/AMD) and ARM
    - Both are proprietary and need commercial licensing



- Not only defines instructions, but also
  - supported data types, registers (number, size), size of address space, address translation mechanisms, memory management, etc.

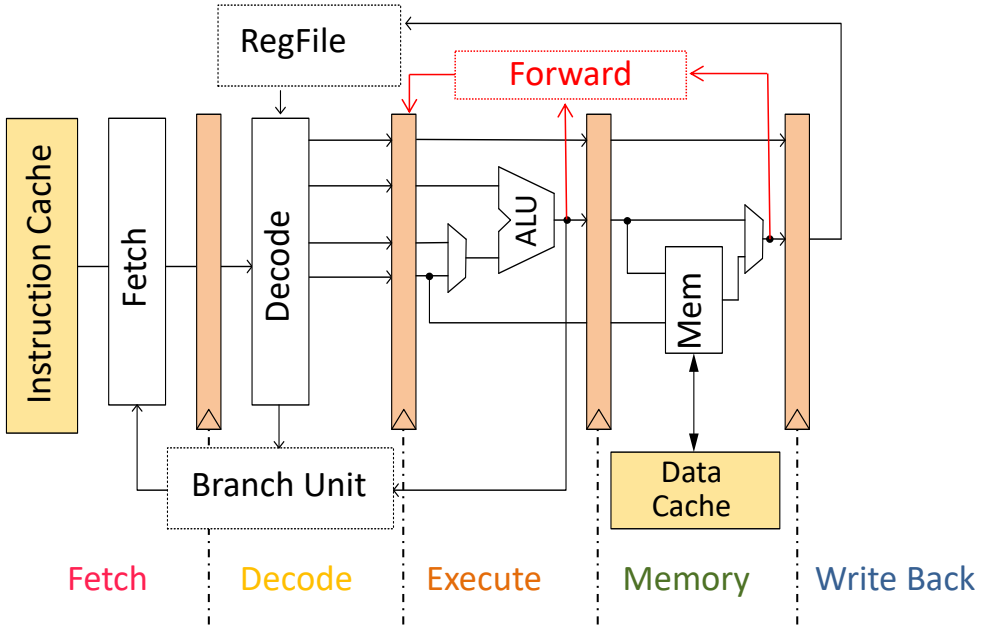
Instruction Set Architecture (ISA)

- Agreeing on the ISA gets you lots of good software:
  - Tools that speak the language
  - Compilers, assemblers, debuggers
  - Emulators, simulators
  - Documentation, programmer's guides, application notes...

An ISA is a Standard

Instruction Set Architecture (ISA)

- ≠ microarchitecture, which is the set of processor design techniques used, in a particular processor, to implement the instruction set



How many ISAs does a chip speak?

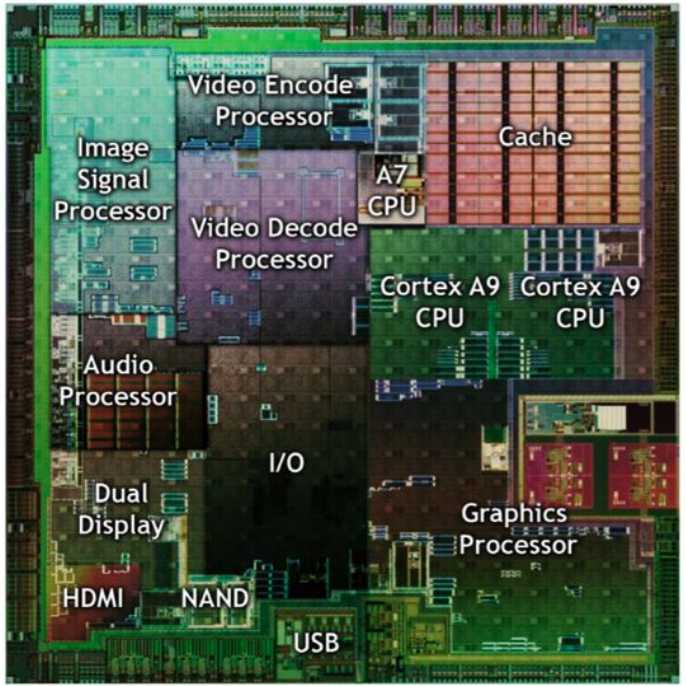
- > dozen ISAs on some SoCs – each with unique SW stack
  - Application processor (e.g. ARM)
  - Graphics proc.
  - Image proc.
  - Audio proc.
  - Radio proc.
  - Security proc.
  - …

NVIDIA Tegra SoC

SoC: System on a Chip

How many ISAs does a chip speak?

- Do we need all these different ISAs?
- Must they be proprietary?
- Must they keep disappearing?



NVIDIA Tegra SoC

IP: Intellectual Property

RISC-V: a Free and Open ISA

- “new instruction set architecture (ISA) that was originally designed to support computer architecture research and education and is now set to become a standard open architecture for industry”

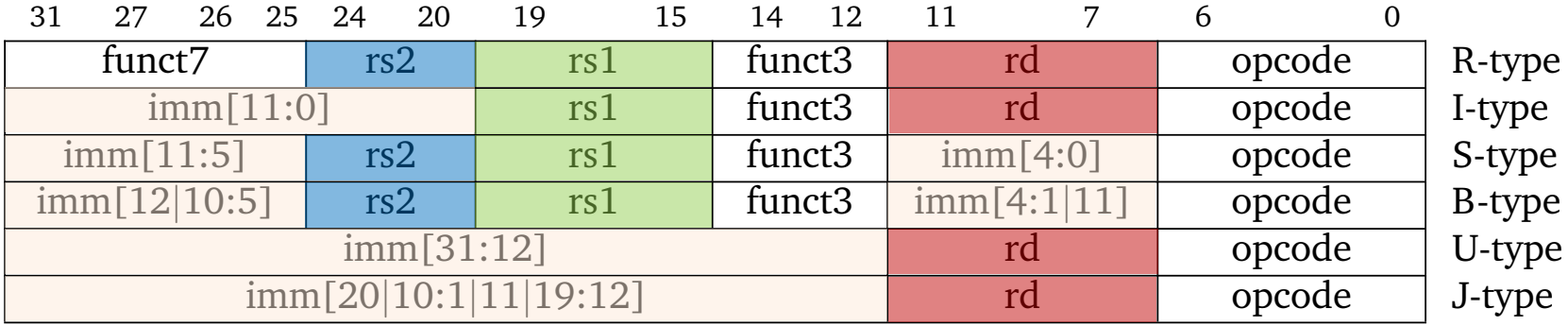
- More than 12 billions RISC-V cores deployed for profit!
  - expected to double this year and in 2023

RISC-V: a Free and Open ISA

- RISC project originally leaded by David Patterson at Berkeley during the 80s
  - MIPS at Standford in the same years
- RISC-V is the 5th generation of RISC ISA (2010)
- May 2014, released frozen base user spec
- RISC-V Foundation created to maintain it as a free and open standard

- RISC-V: the instruction set for everything?

RISC-V Standard Base ISA (RV32I)

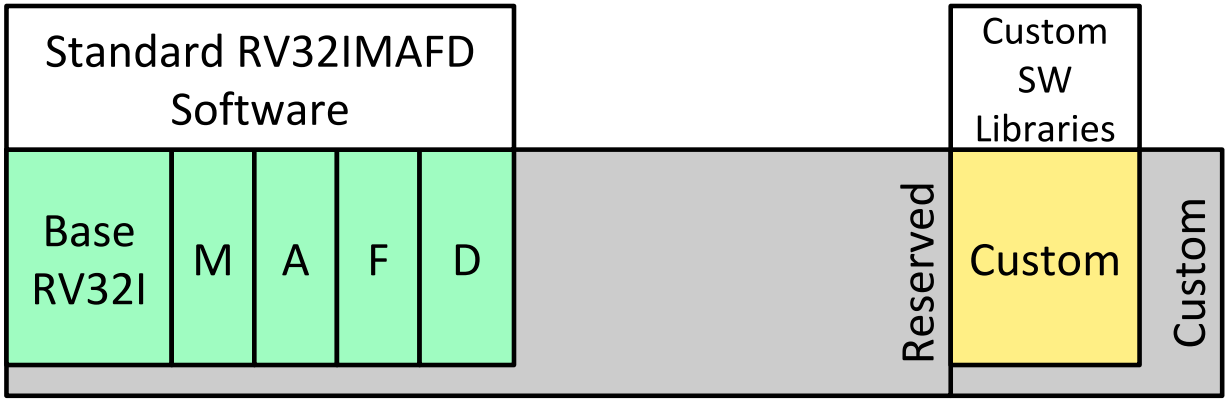


- 32-but fixed-width, naturally aligned instructions
- 31 registers x1-x31, plus x0 zero reg.
- rd/rs1/rs2 in fixed location, no implicit registers
- load/store, integer arithmetic, branch/jump
- Control and Status Registers (CSRs)

Inst	Name	FMT	Opcode	funct3	funct7	Description (C)	Note
add	ADD	R	0110011	0x0	0x00	rd = rs1 + rs2	
sub	SUB	R	0110011	0x0	0x20	rd = rs1 - rs2	
xor	XOR	R	0110011	0x4	0x00	rd = rs1 ^ rs2	
or	OR	R	0110011	0x6	0x00	rd = rs1 | rs2	
and	AND	R	0110011	0x7	0x00	rd = rs1 & rs2	
sll	Shift Left Logical	R	0110011	0x1	0x00	rd = rs1 << rs2	
srl	Shift Right Logical	R	0110011	0x5	0x00	rd = rs1 >> rs2	
sra	Shift Right Arith*	R	0110011	0x5	0x20	rd = rs1 >> rs2	msb-extends
slt	Set Less Than	R	0110011	0x2	0x00	rd = (rs1 < rs2)?1:0	
sltu	Set Less Than (U)	R	0110011	0x3	0x00	rd = (rs1 < rs2)?1:0	zero-extends
addi	ADD Immediate	I	0010011	0x0		rd = rs1 + imm	
xori	XOR Immediate	I	0010011	0x4		rd = rs1 ^ imm	
ori	OR Immediate	I	0010011	0x6		rd = rs1 | imm	
andi	AND Immediate	I	0010011	0x7		rd = rs1 & imm	
slli	Shift Left Logical Imm	I	0010011	0x1	imm[5:11]=0x00	rd = rs1 << imm[0:4]	
srli	Shift Right Logical Imm	I	0010011	0x5	imm[5:11]=0x00	rd = rs1 >> imm[0:4]	
srai	Shift Right Arith Imm	I	0010011	0x5	imm[5:11]=0x20	rd = rs1 >> imm[0:4]	msb-extends
slti	Set Less Than Imm	I	0010011	0x2		rd = (rs1 < imm)?1:0	
sltiu	Set Less Than Imm (U)	I	0010011	0x3		rd = (rs1 < imm)?1:0	zero-extends
lb	Load Byte	I	0000011	0x0		rd = M[rs1+imm][0:7]	
lh	Load Half	I	0000011	0x1		rd = M[rs1+imm][0:15]	
lw	Load Word	I	0000011	0x2		rd = M[rs1+imm][0:31]	
lbu	Load Byte (U)	I	0000011	0x4		rd = M[rs1+imm][0:7]	zero-extends
lhu	Load Half (U)	I	0000011	0x5		rd = M[rs1+imm][0:15]	zero-extends
sb	Store Byte	S	0100011	0x0		M[rs1+imm][0:7] = rs2[0:7]	
sh	Store Half	S	0100011	0x1		M[rs1+imm][0:15] = rs2[0:15]	
sw	Store Word	S	0100011	0x2		M[rs1+imm][0:31] = rs2[0:31]	
beq	Branch ==	B	1100011	0x0		if(rs1 == rs2) PC += imm	
bne	Branch !=	B	1100011	0x1		if(rs1 != rs2) PC += imm	
blt	Branch <	B	1100011	0x4		if(rs1 < rs2) PC += imm	
bge	Branch $\geq$	B	1100011	0x5		if(rs1 >= rs2) PC += imm	
bltu	Branch < (U)	B	1100011	0x6		if(rs1 < rs2) PC += imm	zero-extends
bgeu	Branch $\geq$ (U)	B	1100011	0x7		if(rs1 >= rs2) PC += imm	zero-extends
jal	Jump And Link	J	1101111			rd = PC+4; PC += imm	
jalr	Jump And Link Reg	I	1100111	0x0		rd = PC+4; PC = rs1 + imm	
lui	Load Upper Imm	U	0110111			rd = imm << 12	
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)	
ecall	Environment Call	I	1110011	0x0	imm=0x0	Transfer control to OS	
ebreak	Environment Break	I	1110011	0x0	imm=0x1	Transfer control to debugger	

RISC-V ISA Extensions

- Multiply and divide RV32M
- Atomic RV32A
- Float/Double RV32F/D
  - adds f0-f31 regs., fp CSR
- Bit (B), Vector (V), ...
- CSR, Fence, ECALL instructions
- RV32C Compressed Extension
- RV32E (16 registers)
- 64-bit ISA (RV64I)
- 128-bit ISA (RV128I)



What's Different About RISC-V?

- Simple
  - Smaller than other commercial ISAs
- Clean-slate design
  - Clear separation between user and privileged ISA
  - Avoids microarchitecture or technology-dependent features
- A modular ISA designed for extensibility/specialization
  - Small standard base ISA, with multiple standard extensions
  - Sparse and variable-length instruction encoding for vast opcode space
- Stable
  - Base and standard extensions are frozen
  - Additions via optional extensions, not new versions
- Community designed
  - With leading industry/academic experts and software developers

Why a new ISA?

- Frozen standards are stable forever
- Designed to be extensible
  - Microcontroller to supercomputer
- RISC-V Foundation now controls standard: riscv.org
  - Over 400 members: companies, universities and more
- Freedom to leverage open-source implementations
  - BOOM, Rocket, PULP, and many more
- Open-source compilers and operating systems

Why a new ISA? (Industry and World)

- Companies like Nvidia and Western Digital will ship millions of devices with RISC-V
- Avoid ARM licensing fees

- European sovereignty
  - European Processor Initiative
  - EU roadmap for open source hardware, software and RISC-V Technologies
  - EU Chip Act
- Strong interest from chipmakers in China

RISC-V Ecosystem

Open-source software:
Gcc, binutils, glibc, Linux, BSD, LLVM, QEMU, FreeRTOS, ZephyrOS, LiteOS, SylixOS, …

Commercial software:
Lauterbach, Segger, IAR, Micrium, ExpressLogic, Ashling, Imperas, AntMicro, …

Software

RISC-V Foundation

ISA specification | Golden Model | Compliance

Hardware

Open-source cores:
Rocket, BOOM, RI5CY, Ariane, PicoRV32, Piccolo, SCR1, Swerv, Hummingbird, …

Commercial core providers:
Andes, Bluespec, Cloudbear, Codasip, Cortus, C-Sky, Nuclei, SiFive, Syntacore, …

Inhouse cores:
Nvidia, WDC, Alibaba, ...



Open-source Hardware

Open-source Hardware

- Hardware whose **design** is made **publicly available** so that anyone can **study**, **modify**, **distribute**, **make**, and **sell** the design or hardware based on that design

RISC-V == open-source hardware?

- No!
- But it allows open-source HW
- Or closed source, or a mixture

Solderpad Hardware License 2.1

- A permissive open hardware license
- Based on, and acts as an exception to, Apache-2.0
- SPDX-License-Identifier: Apache-2.0 WITH SHL-2.1
- Covers physical hardware as well as open silicon and gateware
- Modifies, clarifies and extends various Apache definitions, and the scope of rights to explicitly cover hardware
- Not specifically OSI approved, but we know it falls within the OSI definition of “open source” because any licensee can treat as plain Apache-2.0
- http://solderpad.org/licenses/SHL-2.1/

MOORCROFTS
CORPORATE LAW

Examples of Open-source RISC-V Cores

- Rocket Chip/SoC Generator 🚀 (UC Berkeley)
- Berkeley Out-of-Order Machine (BOOM)

- Pulp platform, Ariane (ETHZ)
- CV32E4x, CV32Ax, CV64Ax

- VexRiscv: 32-bit Linux Capable RISC-V CPU

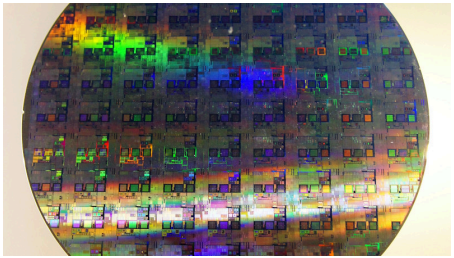
- And many more to build an SoC

Open Source Hardware: What Else is Needed?

- Put it on open-source FPGAs?
  - SymbiFlow (now F4PGA) a fully open source toolchain for the development of FPGAs
    - open-source or commercial FPGAs

F4PGA

- Design your chip only with open-source CAD tools?
  - Yosys/OpenROAD

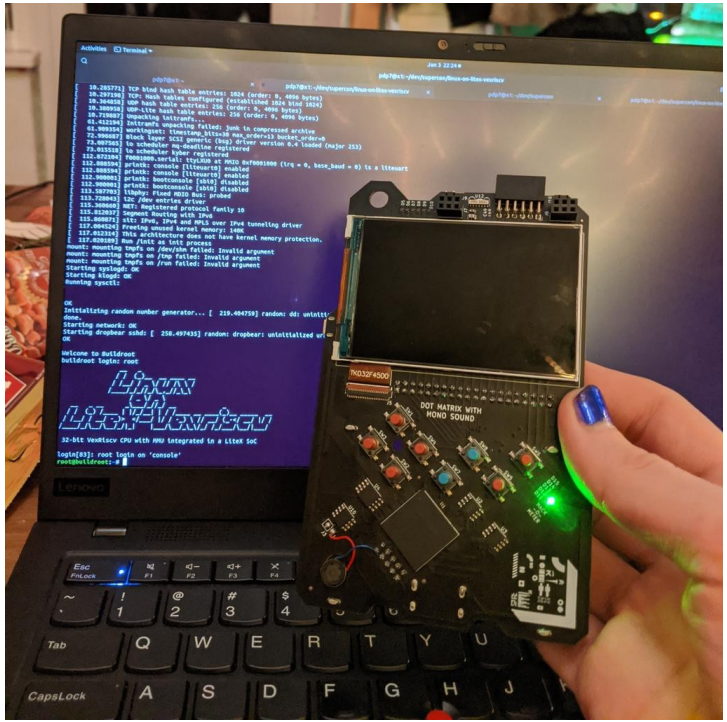


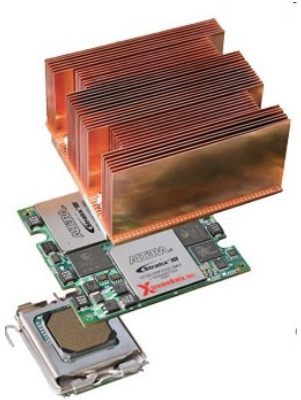
- Build your chip in an open-source fab?

skywater

More Cool Things: FPGA

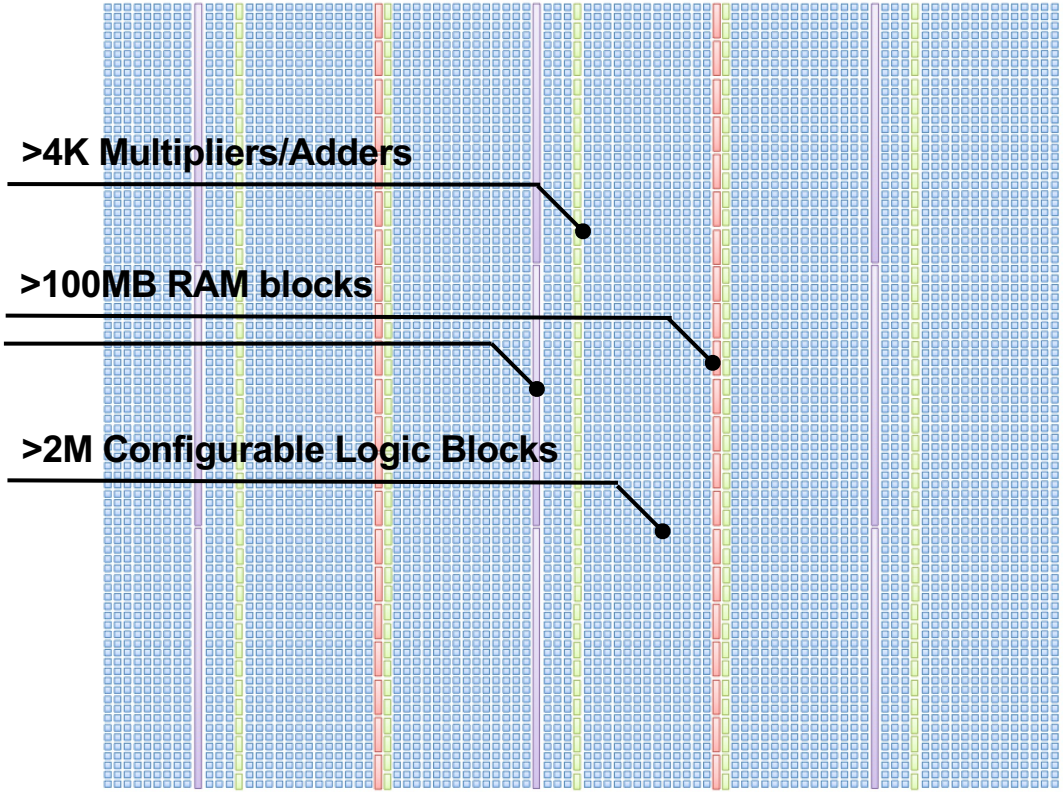
- Cool thing for computer architect is tuning the core
- But
  - using simulator is hard and slow
  - it takes weeks/months/money to build a new chip
- FPGA is a piece of hardware that can be programmed (configured)

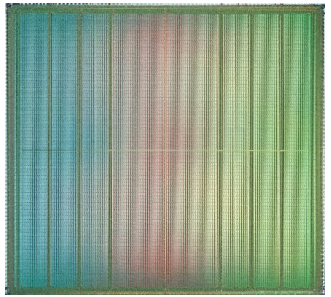




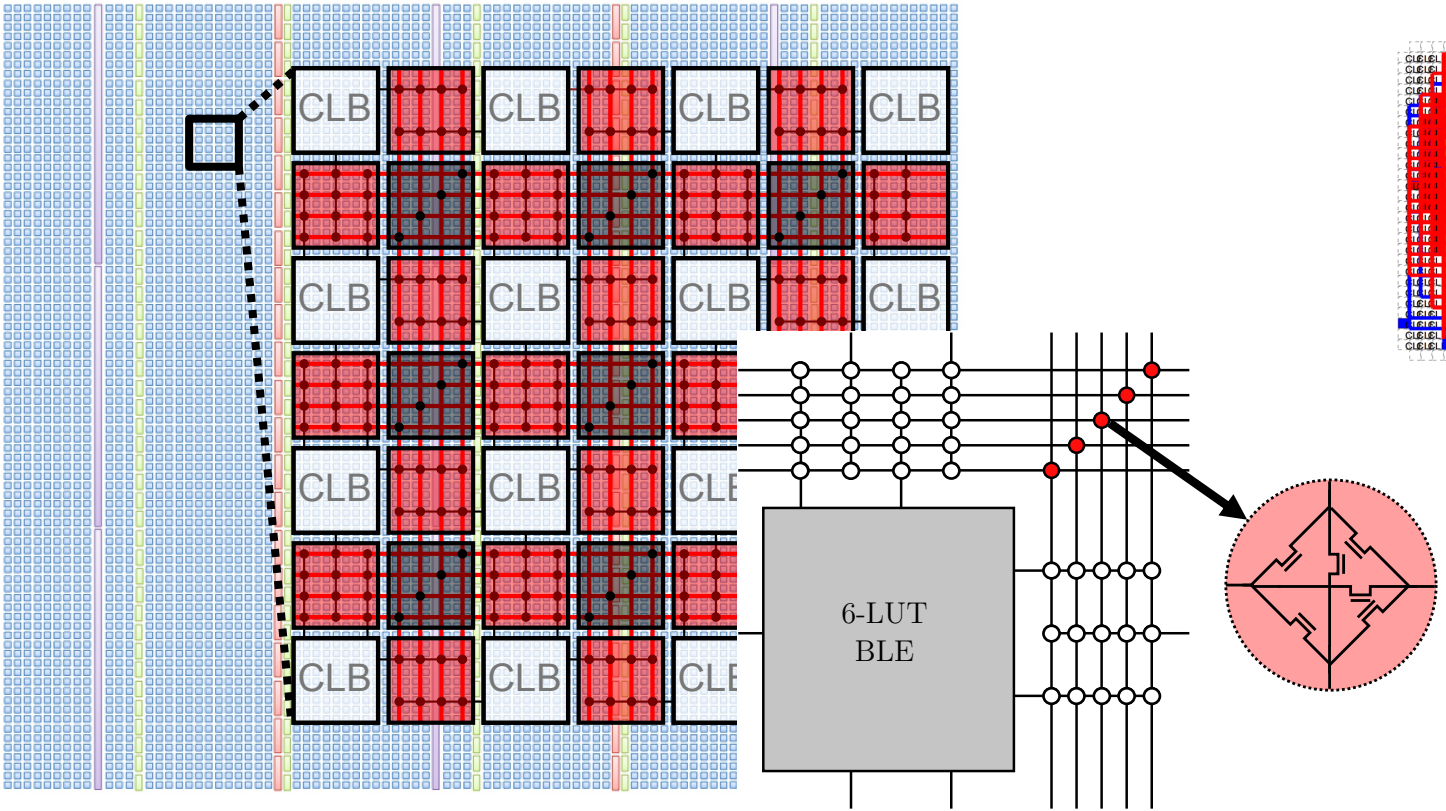
A short interlude on FPGA!

Field Programmable Gate Array (FPGA)

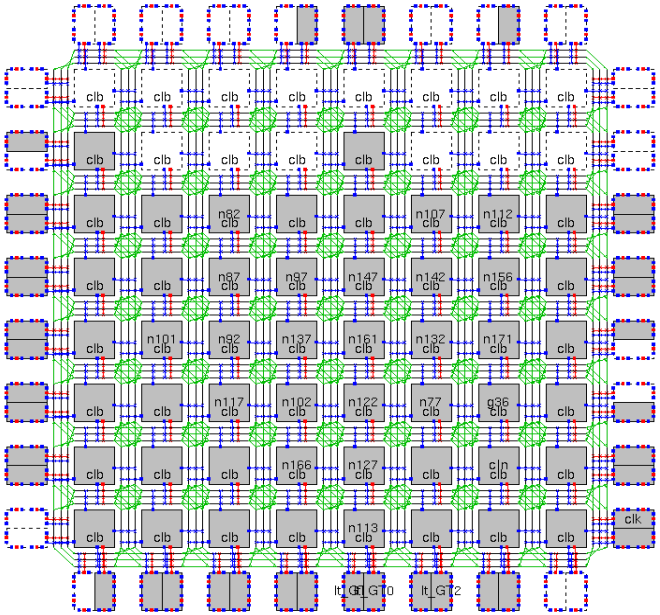




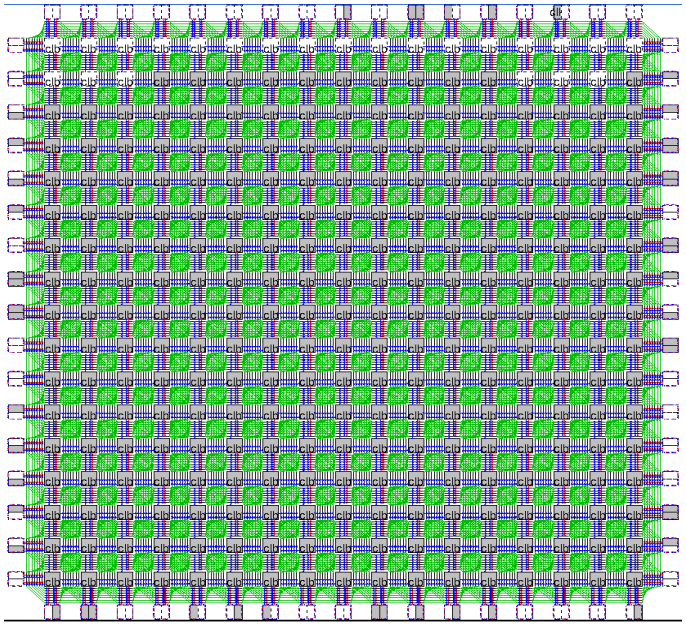
Field Programmable Gate Array (FPGA)



The Program is the Configuration

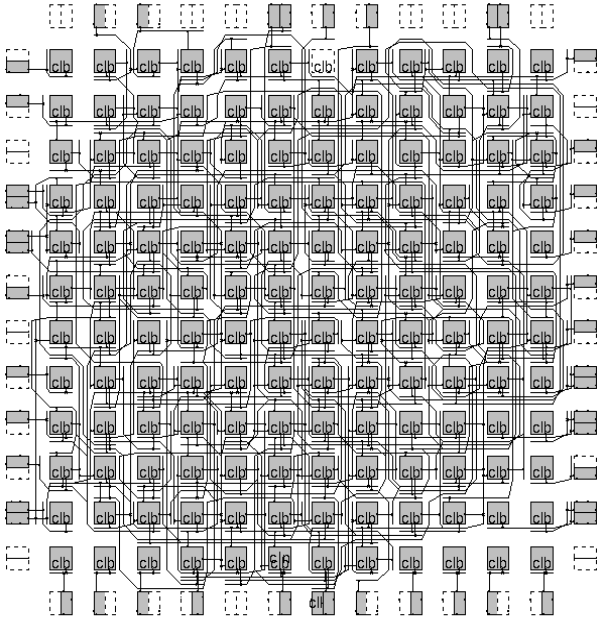


(a) abs

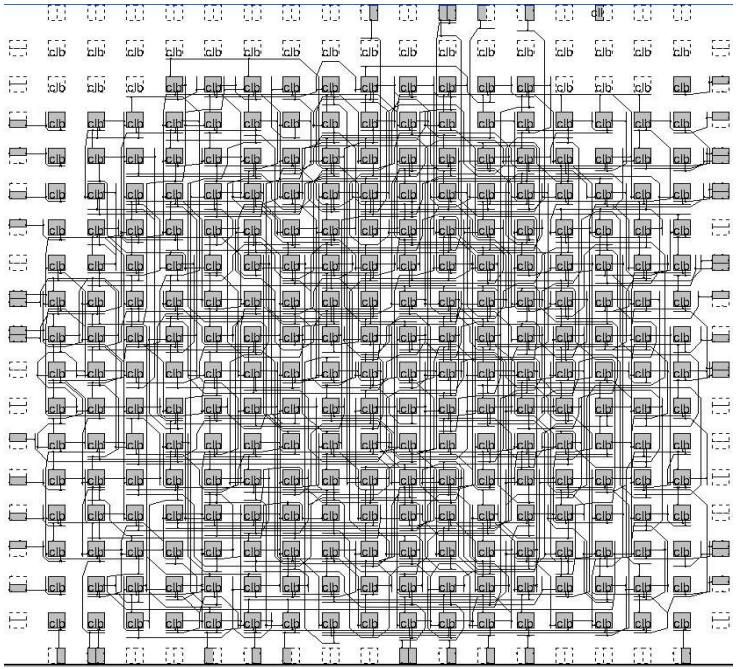


(b) calcNeighbor

The Program is the Configuration



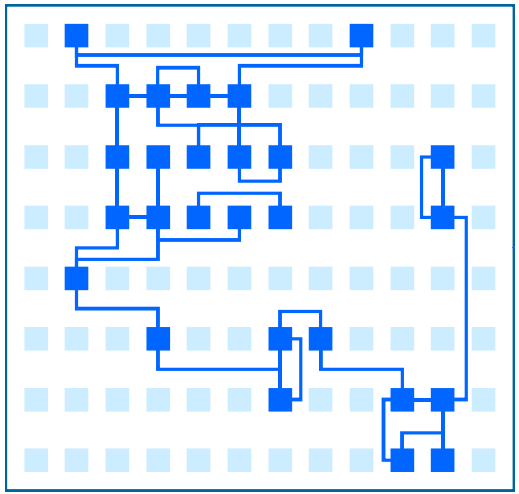
(a) Crc16



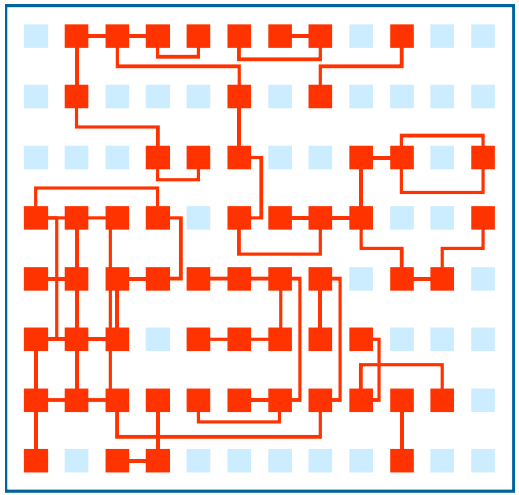
(b) calcNeighbor

Space-Time Computation

```
for(i=1; i<length; i++) {
    if (max < T[i]) {
       max = T[i];
    }
}
```

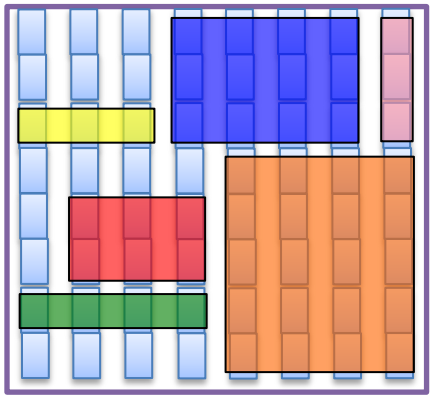


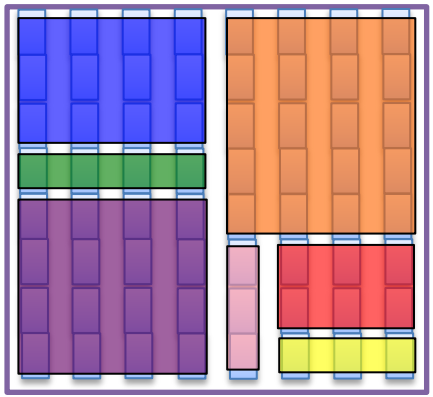
```
for(i=1; i<N; i++) {
   for(j=1; j<M; j++) {
      y[i][j]+=x[i][j]*h[j][i]
   }
}
```

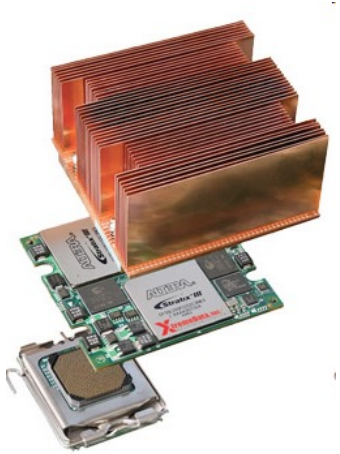


FPGA Acceleration

- FPGAs can run multiple tasks in parallel







- Heterogeneous architectures (CPU+FPGA(+GPU(+NPU)))

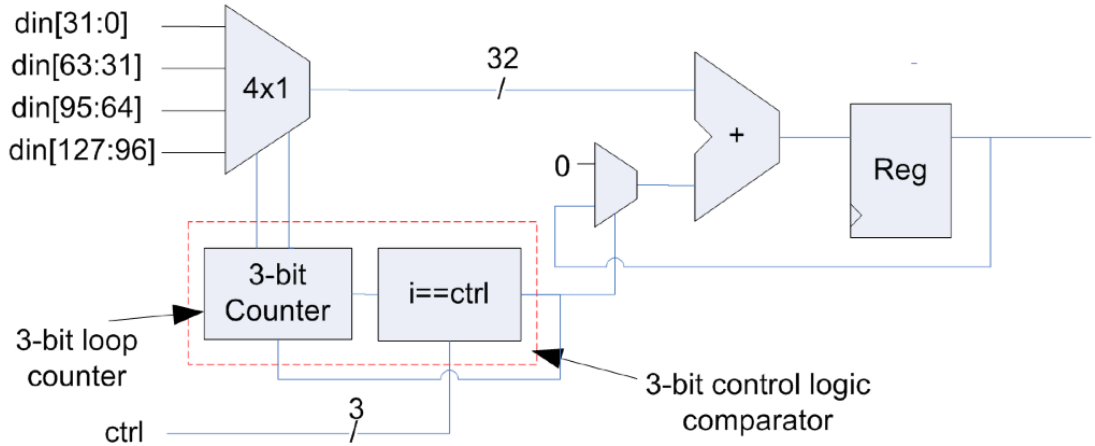
FPGA?

- Facts
  - Intel acquisition of Altera (2016)
  - AMD acquisition of Xilinx (2022)
  - Amazon EC2/F1 instance
    - Up to 8 Xilinx UltraScale+ FPGA devices
  - Microsoft Unveils FPGAs to Accelerate Bing
- Solutions
  - Overlays (e.g., DNN accelerators)
  - High-level synthesis (jointly compile the code and the hardware)
  - New languages for hardware design
    - Chisel, Spinal, Silice, C++, SystemC

High-Level Synthesis of HW Accelerators

- C/C++ to specialized hardware

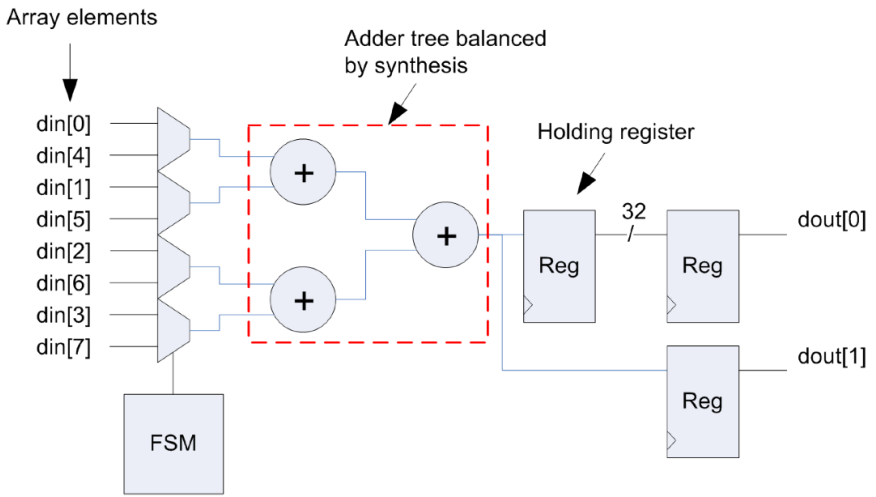
```
void accumulate(int din[2][4], int dout[2]){
  int acc=0;
  ROW:for(int i=0;i<2;i++){
    if (acc>MAX) acc = MAX;
    COL:for(int j=0;j<4;j++){
      acc += din[i][j];
    }
     dout[i] = acc;
  }
}
```



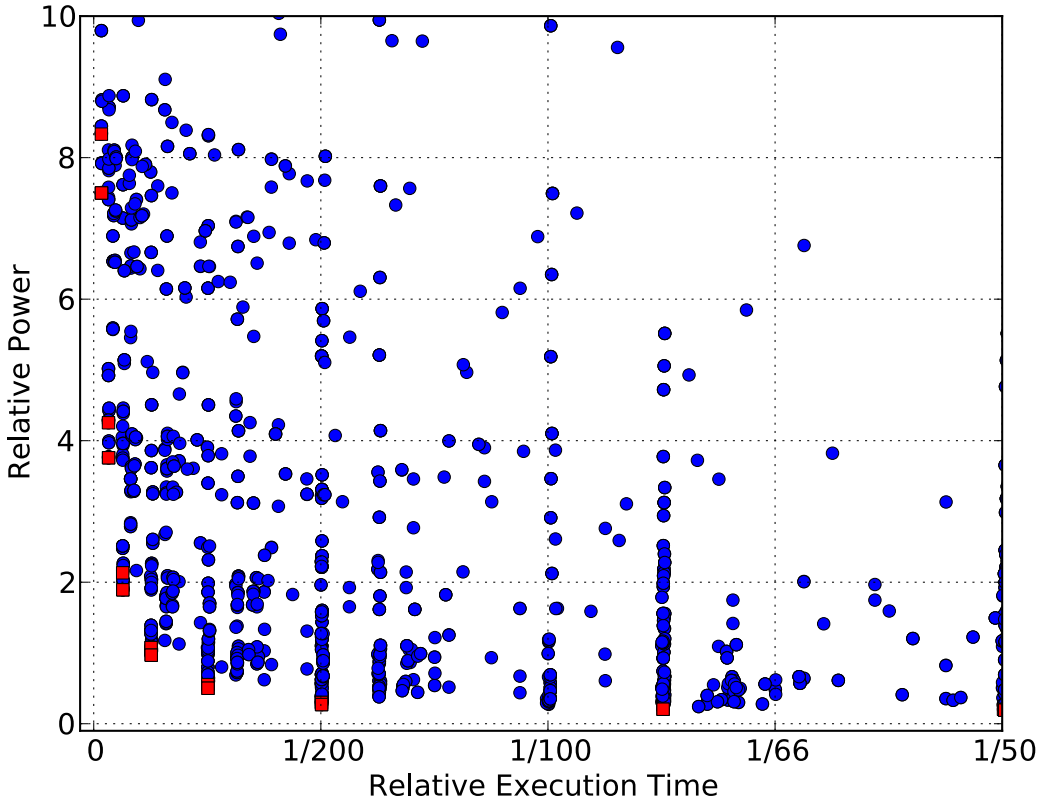
High-Level Synthesis of HW Accelerators

- C/C++ to specialized hardware

```
void accumulate(int din[2][4], int dout[2]){
  int acc=0;
  ROW:for(int i=0;i<2;i++){
    if (acc>MAX) acc = MAX;
    COL:for(int j=0;j<4;j++){
      acc += din[i][j];
    }
     dout[i] = acc;
  }
}
```



Fast Design Space Exploration



Cool Research with RISC-V

and open-source hardware

5 Ways to Modify a RISC-V Based Design

1. Tune the CPU

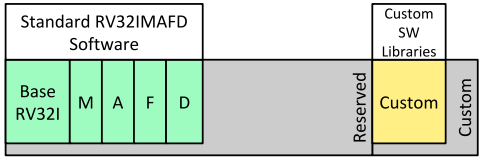
- 32 bits? 64 bits? 128 bits?
- Cache sizes
- Branch Prediction
- Pipeline Depth

2. Tweak the CPU

- Skip multiplications by zero
  - performance? side channel?
- 12-bit reduced-precision floating point hardware

3. Customize your RISC-V ISA

- 12-bit float
- FFT
- GEMM



4. Modify the platform

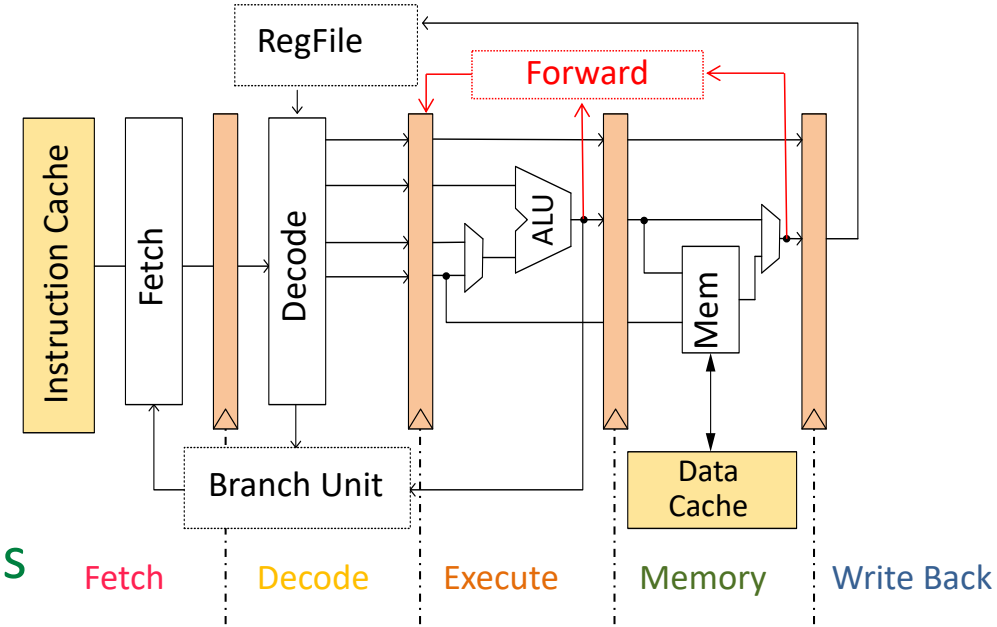
- Add deep learning acceleration

5. Replace your CPU

- Choose an ISA
- Pick or design an implementation

Comet: What You Simulate is What You Synthesize

- Comet is designed from a single C++ specification using High-Level Synthesis (HLS)
  - Hardware == Simulator
- As efficient as RTL
- Very fast simulator
  - ~20 Millions cycles per sec.
- Easy to tune
- Leverages SW dev. techniques

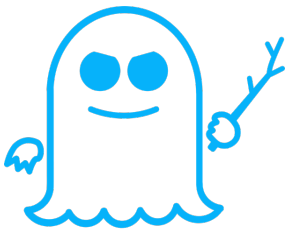


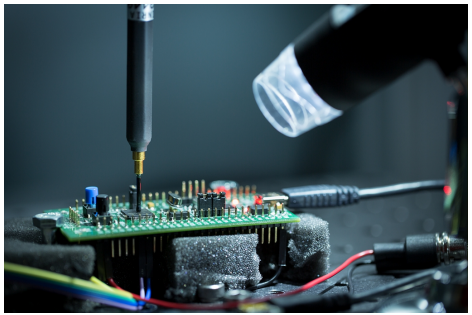
[IEEE/ACM ICCAD'19]

Hardware Security

- Key message here is reproducible research

- Side channel attacks (SCA) and fault attacks (FA)
  - physical, microarchitectural, software, hardware

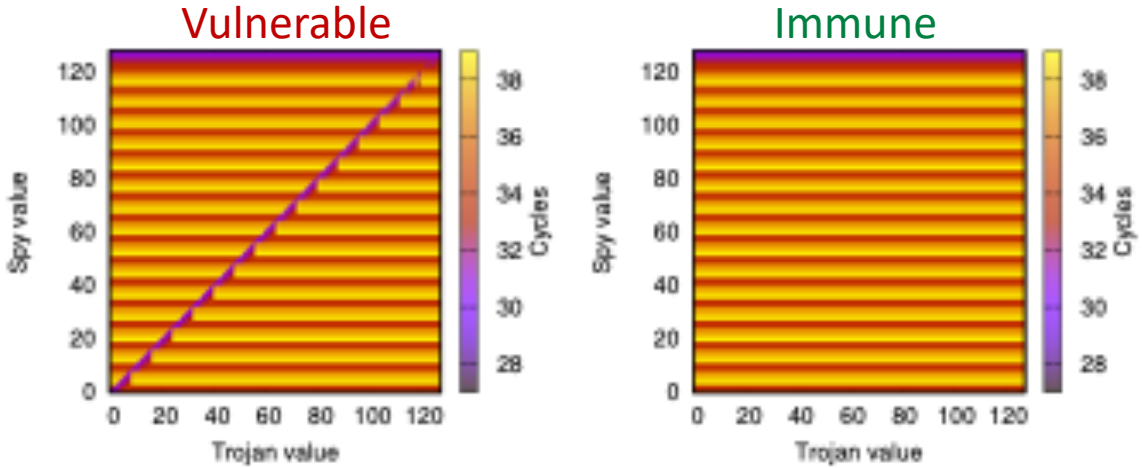




© Inria / Photo C. Morel

Hardware Security

- Build or tune a RISC-V core to demonstrate a technique to improve robustness to SCA or FA
  - e.g., Branch History Table (BHT) as a covert channel
    - M. Escouteloup (PhD), R. Lashermes
    - Results can be reproduced *(build core, build binaries with patched gcc, core simulation)*



Hardware Security: Roadmap

- Secured in-order/out-of-order RISC-V cores

- Security-oriented ISA extensions
  - Several working groups at the RISC-V foundation
  - Inria (R. Lashermes) is co-leading a Working Group at RISC-V International on security extensions
    - Microarchitecture Side Channels Special Interest Group (uSC SIG)
      - *RISC-V strategy to prevent microarchitectural information leakage, with an initial focus on timing side channels*

RISC-V: The ISA of choice for ~~researchers~~ hackers

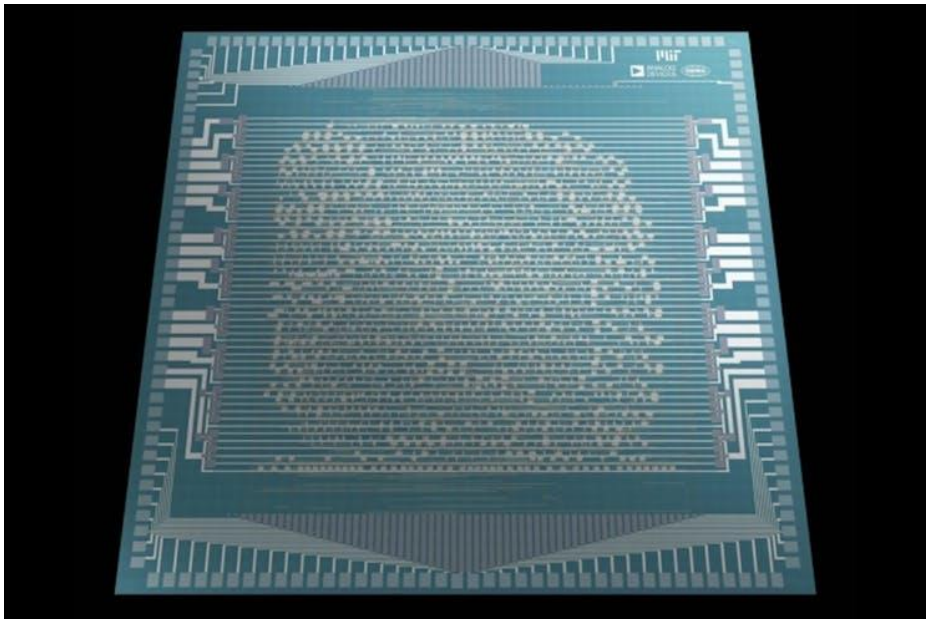
nature

Article | Published: 28 August 2019

Modern microprocessor built from complementary carbon nanotube transistors

RISC-V USES CARBON NANOTUBES

by: Al Williams



RISC-V makes it easy to focus on the cool part of your project

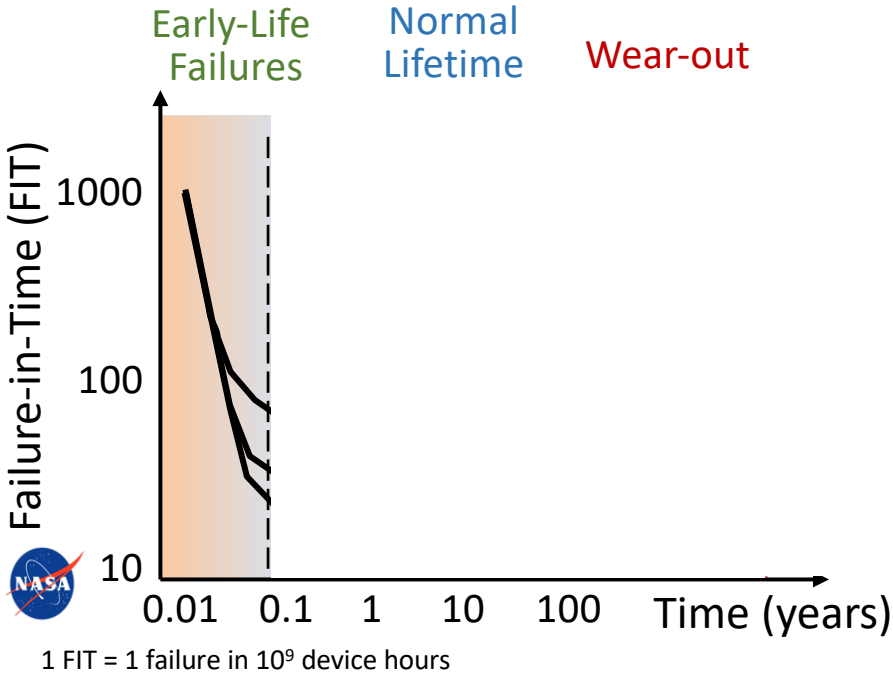
Everlasting Architectures

Can we design a processor that can live for decades?

Chip Reliability Threats

- Defects in the HW
  - Manufacturing defects
- Radiation
  - Particles hit your chip
  - Space, Avionics, Terrestrial
- Wear-out (aging)

- Advanced (or emerging) technologies
  - Increased failure rate
  - Early wear-out



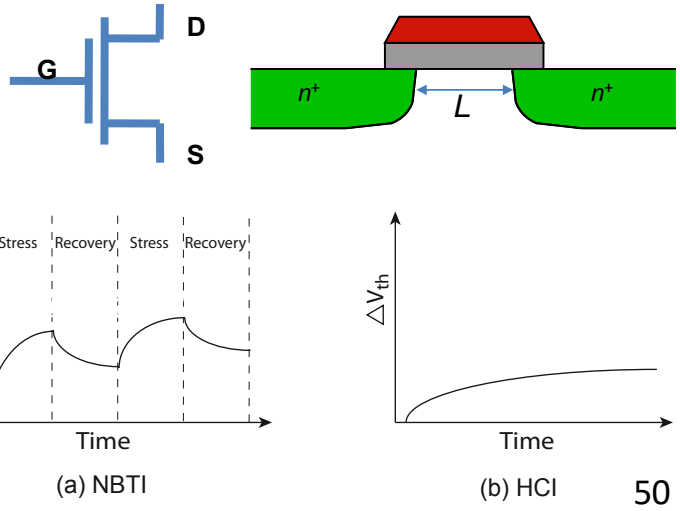
1 FIT = 1 failure in 10^9 device hours

Can we extend chip lifetime?

How Old My Transistors can Live?

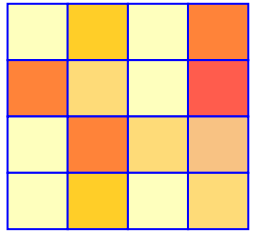
- Chip lifetime requirements
  - e.g., 10-15 years (automotive), 7-10 y (processors), >10 y (data-centers)
  - under temperature and usage (workloads) conditions

- Transistor Aging
  - Gradually causes device to become slower
    - static/dynamic stress
    - temperature
    - …
  - Finally may results in permanent faults



Solutions and Challenges

- Guardbands to absorb any increase in logic path delay
  - Worst-case speed and voltage margins



- Challenge: worst-case guardbands pessimistic and too expensive for current and future chips
- or lifetime is significantly reduced...
- and probability of permanent faults increases with lifetime and technology scaling

Everlasting Architectures?

- Architectures can live longer if they are healed
- Need for

1) performance degradation estimation
   - e.g., timing error analysis, on-line test
2) permanent fault detection
3) self-healing mechanisms
   - Core-level: e.g., reduces #issues in a VLIW
   - System-level: e.g., reduces #cores
   - Hardware reconfiguration, approximations, etc.
   - Aging-driven scheduling
   - JIT compilation/parallelization

Conclusion

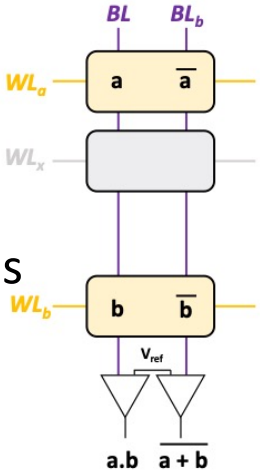
- RISC-V: a standardized, free and open ISA
  - The Linux of hardware
- RISC-V is not a research topic but an enabler

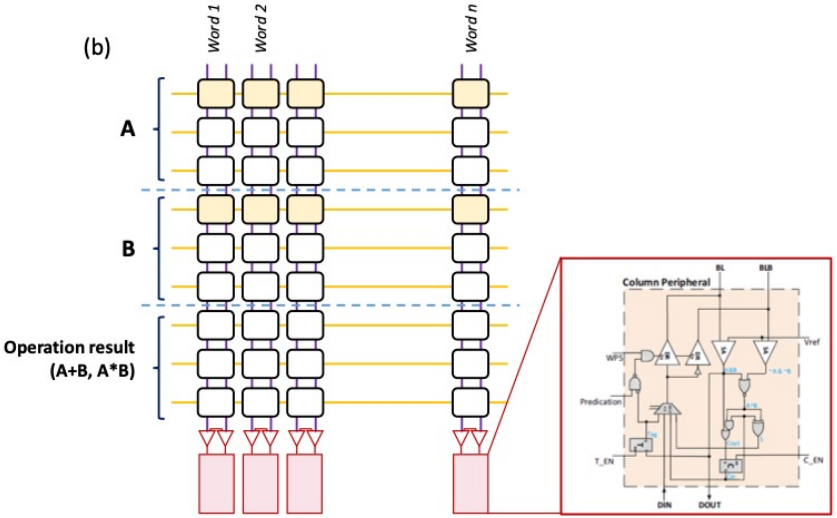
- Open the box!

- (Inria is member of RISC-V International)

What's Hot in Computer Architecture?

- Accelerating Deep Learning of course...
  - Reduced-precision, sparse computations
- Memory still scales
  - Peta-byte of memory
  - But lots of energy to access
- In/Near Memory Computing
  - Remove the Von-Neuman bottleneck





Iron law of processor performance

$$\frac{Time}{Program} = \frac{Instructions}{Program} \times \frac{ClockCycle}{Instruction} \times \frac{Time}{ClockCycle}$$

(1) (2) (3) (4)

(2) depends on code, compiler and ISA

(3) depends on ISA and microarchitecture

(4) depends on microarchitecture and technology

(1) as low as possible

- efficient code, compiler, microarchitecture
- given an ISA and a technology

Energy Efficiency

$$Power = \underbrace{\frac{Operations}{Second}}_{\text{Performance}} \times \underbrace{\frac{Joules}{Operation}}_{\text{Energy Efficiency}}$$

Performance: e.g., Tera op/s (TOPS)

Energy Efficiency: e.g., TOPS/Watt

- Power budget is fixed
- How to increase energy efficiency while maintaining performance?
  - Specialized hardware that computes at the right (lowest) precision

Opportunities for computer architecture research with open-source hardware

The case of RISC-V

Olivier Sentieys

Univ. Rennes, Inria, Irisa, Taran

olivier.sentieys@inria.fr