Optimized Softfloat Routines for RISC-V

- Project Scope
  - produce optimized softfloat routines for RISC-V inspired by code size/performance of existing Arm routines

Arm libgcc object file

```
size _arm_addsubsf3.o
   text    data     bss     dec     hex filename
    592       0       0     592     250 _arm_addsubsf3.o
```

→ **≈220** lines of assembly code

RISC-V libgcc object file

```
 size addsf3.o
   text    data     bss     dec     hex filename
    860       0       0     860     35c addsf3.o
```

Macro expansion of 12 lines of code →

≈590 lines of C code

- Goals
  - potential performance improvements for deeply embedded/IoT applications
  - reduce code size for softfloat applications

Information Classification: General

RISC-V verification techniques for implementations and custom extensions

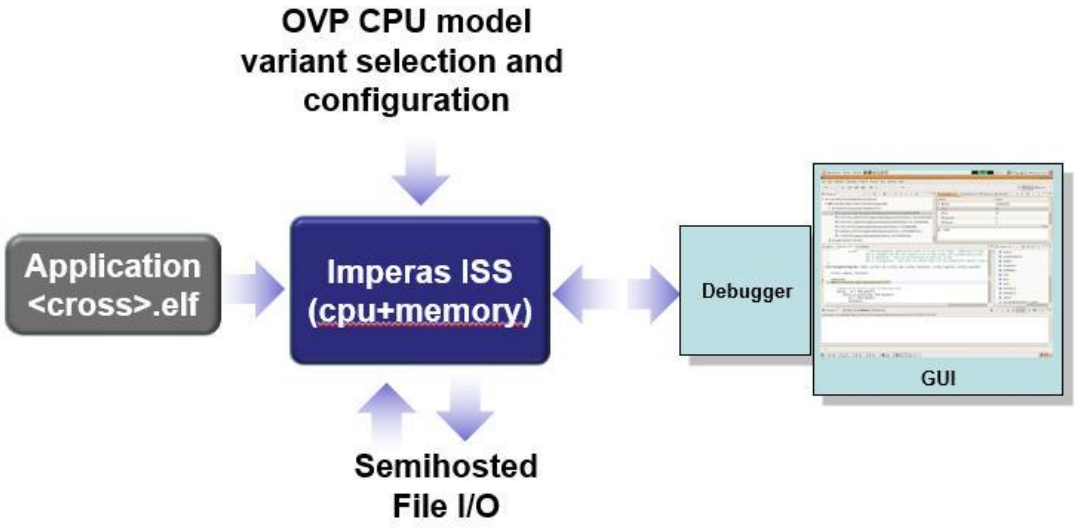
Simon Davidmann, CEO, SimonD@imperas.com

© Imperas Software, Ltd.

RISC-V Workshop Zurich
June 11 - 13, 2019
ETH Zurich, Zurich

Free riscvOVPsim on GitHub

- Simplest usage of Imperas is as Instruction Set Simulator (ISS)
- Full RISC-V model & simulator
  - 2.2, 2.3 User spec
  - 1.10, 1.11 Priv spec
  - Bit Manipulation
  - Vectors
- Free download & usage from GitHub
- riscvOVPsim is used as a reference in the RISCV.org Foundation's Compliance Task Group Group & Bit Manipulation GitHub repositories



- https://github.com/riscv/riscv-compliance
- https://github.com/riscv/riscv-bitmanip

Information Classification: General
© Imperas Software, Ltd.

Flow to add new custom instructions

imperas

Characterize C Application

- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

→

Develop New Custom Instructions

- Design Instructions
- Add to Application
- Add to Model
- Add Timing

→

Characterize New Instructions in Application

- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

↓

Optimize & Document model

- Instruction Coverage
- Line Coverage
- Instruction Performance
- Generate PDF model doc

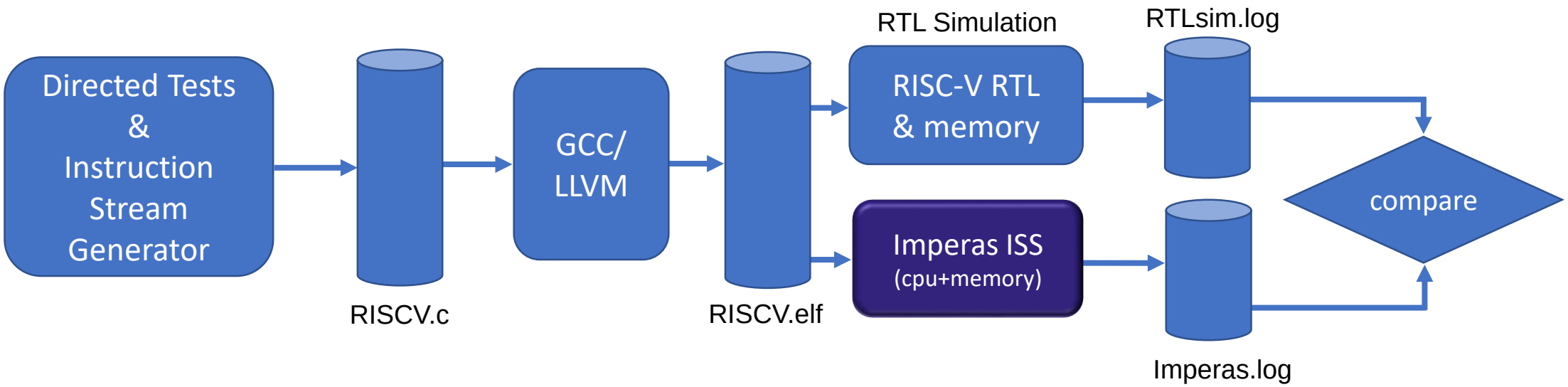
←

Release & Deploy

- Check RISC-V Compliance
- Use as reference for RTL Design Verification
- Use in Imperas/OVP Platforms, EPKs
  - Heterogeneous / Homogeneous
  - Multi-core, Many-core
- Imperas Multi-Processor Debug, VAP tools
- Port OS, RTOS (Linux, FreeRTOS…)
- Use in many simulation envs (inc. SystemC)
- Deliver to end users

Information Classification: General
© Imperas Software, Ltd.

RTL Design Verification
Comparing Model with RTL implementations



- There are many different approaches to use the model with new custom instructions
  - Co-Simulation and log compare
  - Step & compare (tandem verification)
  - …

Information Classification: General
© Imperas Software, Ltd.

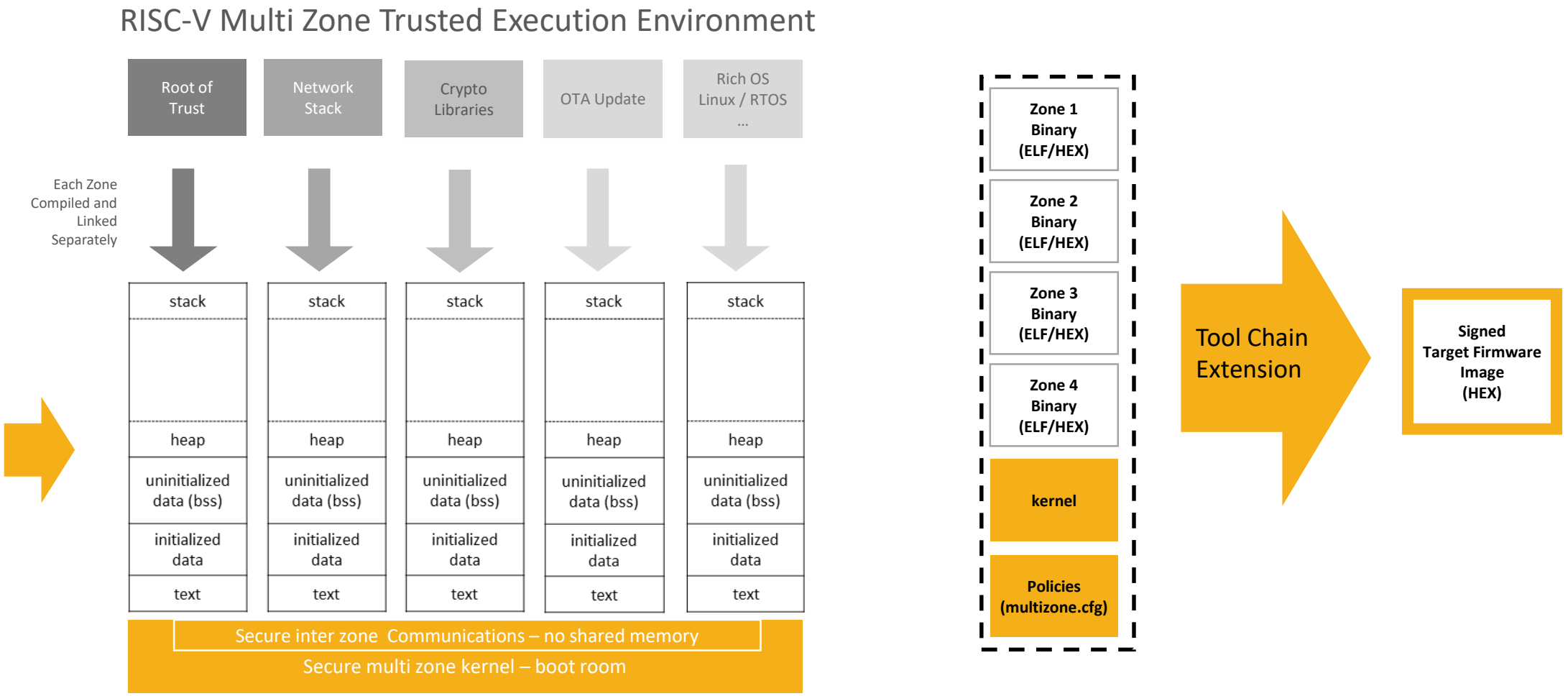
A New Zero-Trust Model Entirely Based on Free and Open Standards

Cesare Garlati, Hex Five Security

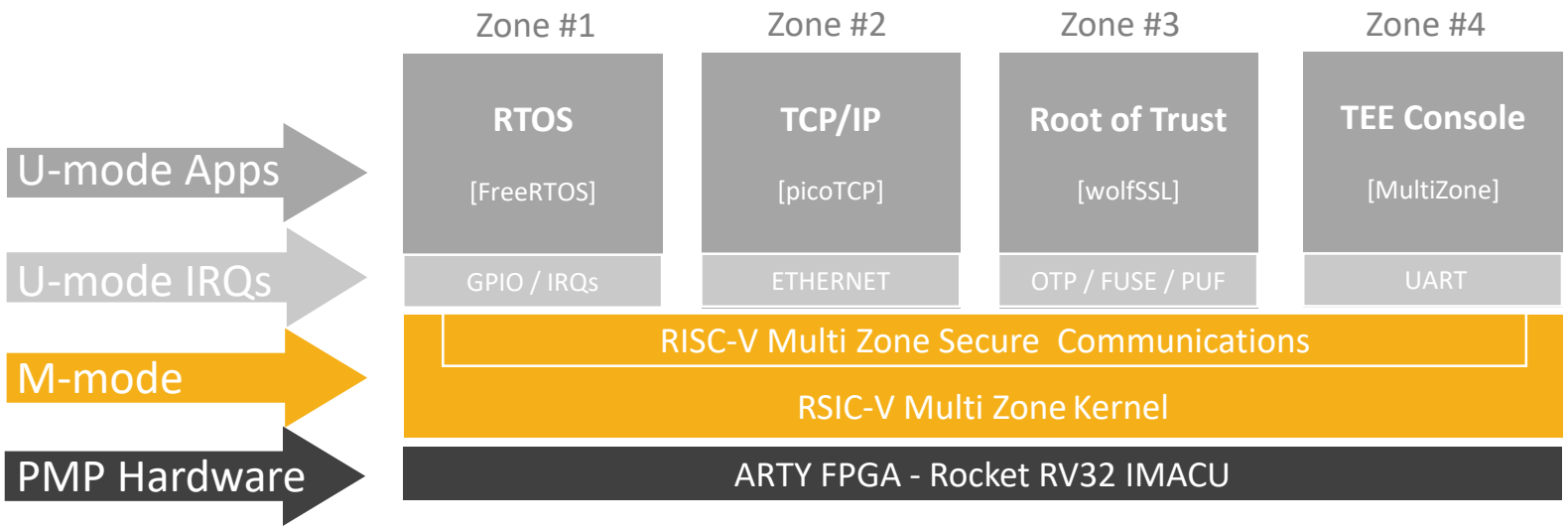
RISC-V Workshop Zürich 2019

Information Classification: General

RISC-V Multi Zone API - Data Model

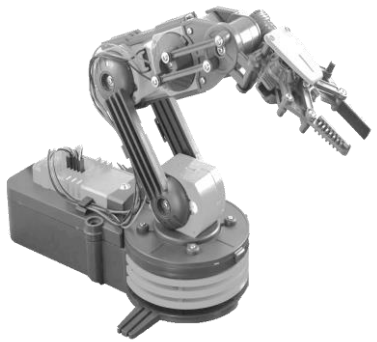


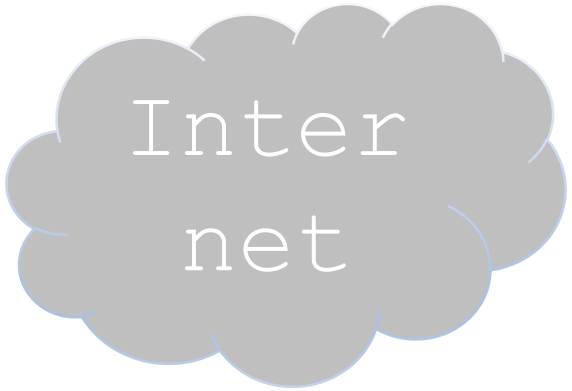
Use Case: RISC-V Secure IoT Stack



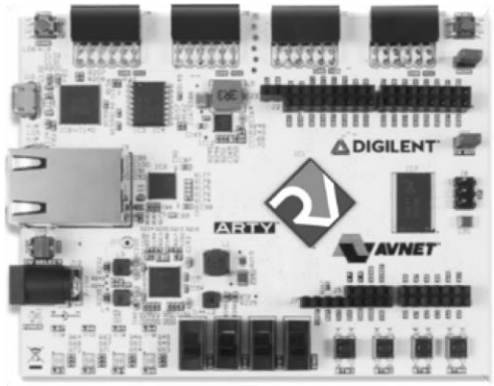
Crypto

TLSv1.3, Cipher TLS_AES_128_GCM_SHA256
Peer signing digest: SHA256
Peer signature type: ECDSA
Server Temp Key: ECDH, P-256, 256 bits
Server public key is 256 bit
Private Key ASN1 OID: prime256v1
Private Key NIST CURVE: P-256





TLS 1.3 / ECC



SPI / USB

UART

Hex Five MultiZone™ Security

Hex Five Security, Inc. is the creator of MultiZone™ Security, the first trusted execution environment for RISC-V. Hex Five open standard technology provides policy-based hardware-enforced separation for an unlimited number of security domains, with full control over data, code and peripherals. Contrary to traditional solutions, Hex Five MultiZone™ Security requires no additional cores, specialized hardware or changes to existing software. Open source libraries, third party binaries and legacy code can be configured in minutes to achieve unprecedented levels of safety and security.

http://hex-five.com

RISC-V based Virtual Prototype

- RV32IMAFC+SUN, RV64GC+SUN
- Implemented in SystemC/C++
  - TLM-2.0 compliant
  - approx. 12k LOC (w/o comments, blanks)
- **Open Source via GitHub**
  - http://www.systemc-verification.org/riscv-vp
  - MIT license
  - FDL 2018 overview paper:
    "Extensible and Configurable RISC-V based Virtual Prototype"
    http://www.informatik.uni-bremen.de/agra/doc/konf/2018FDL_RISCV_VP.pdf

RISC-V based Virtual Prototype

- **SW debug** capabilities with **Eclipse IDE**
- **Linux**, **FreeRTOS** and **Zephyr OS** support
- Simulation of **multi-core** Platforms

---

1. VP Verified with Coverage-guided Fuzzing

DATE 2019 paper: "Verifying Instruction Set Simulators using Coverage-guided Fuzzing"

2. SW Verification using Concolic Testing

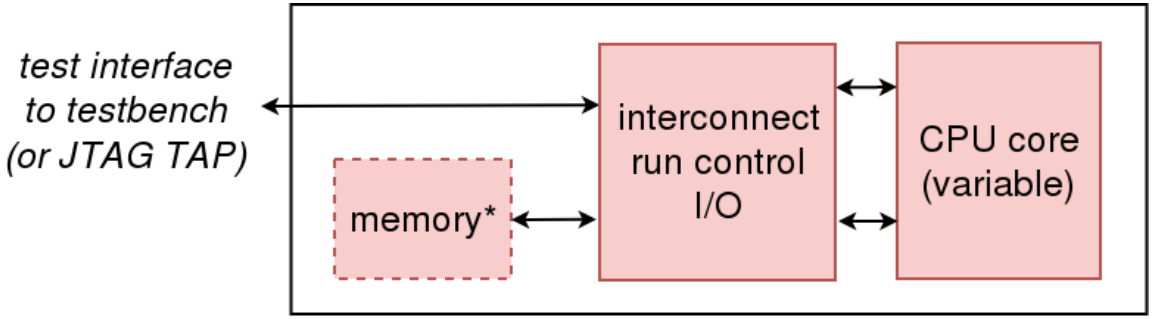
DAC 2019 paper: "Early Concolic Testing of Embedded Binaries with Virtual Prototypes: A RISC-V Case Study"

An Energy Benchmarking Framework for RISC-V Systems-on-Chip

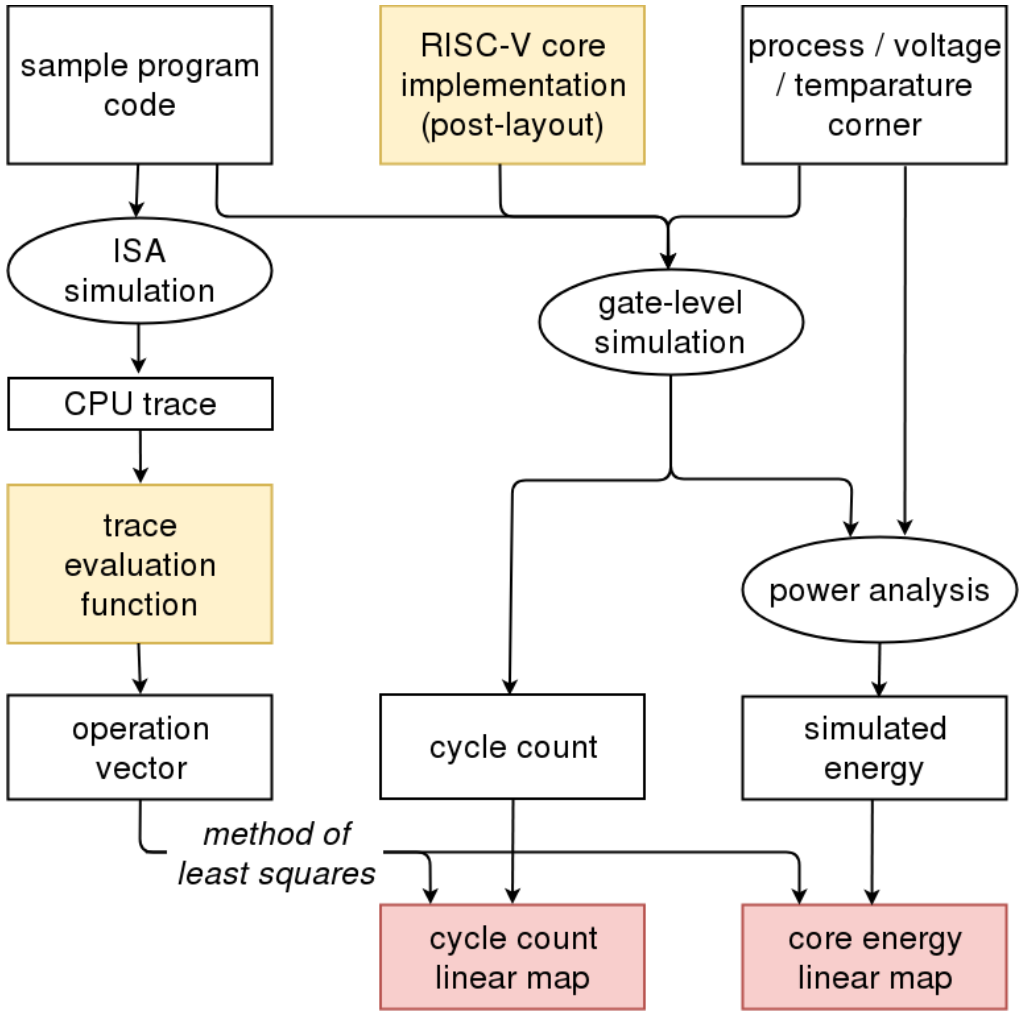
Motivation

- instruction stream properties
  → energy consumption
- enable research in microarchitecture, implementation & circuit level low-power SoCs
- dynamic system-level modeling

System block diagram



Methodology based on ISA simulation



Information Classification: General

An Energy Benchmarking Framework for RISC-V Systems-on-Chip

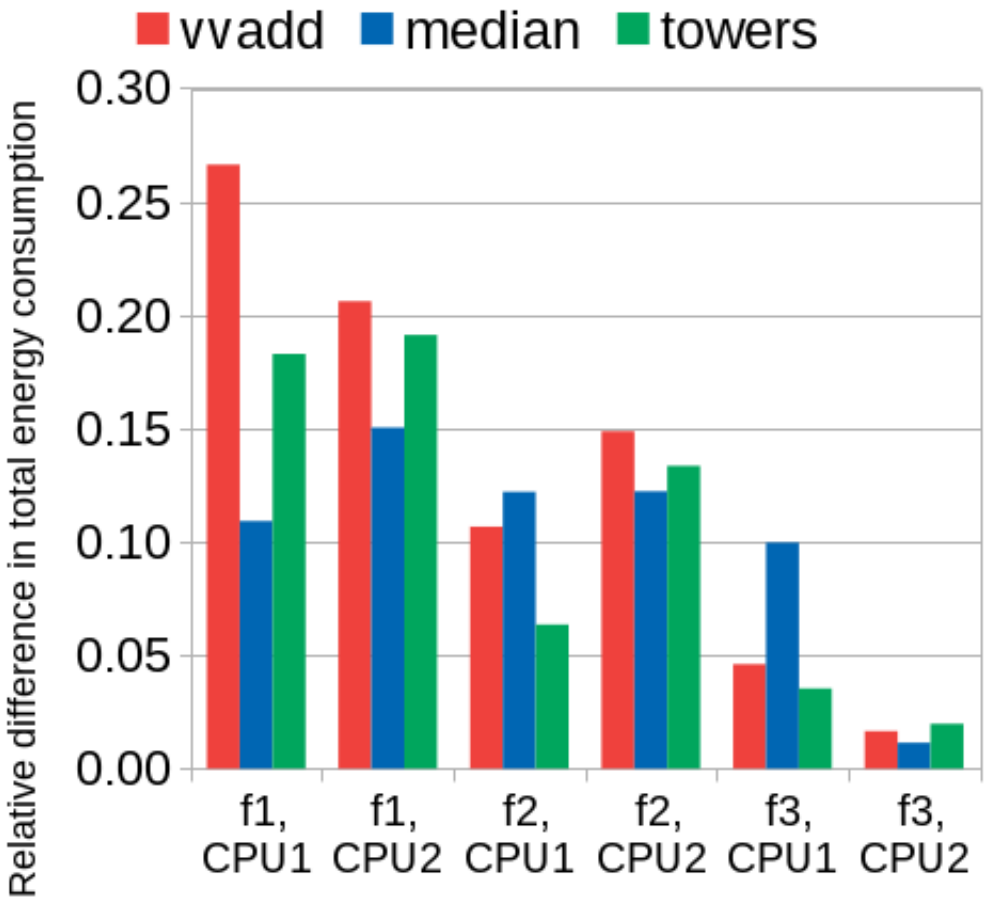
Results

- Linear maps can be used for evaluation in design exploration.

ISA simulation based cycle & energy estimation:

- Depending on processor, trace evaluation function and software, model results differ by 2…10% from gate-level simulation.
- >100x speed-up compared to gate-level sim
- lightweight alternative to TLM
- proposed dynamic system-level simulation

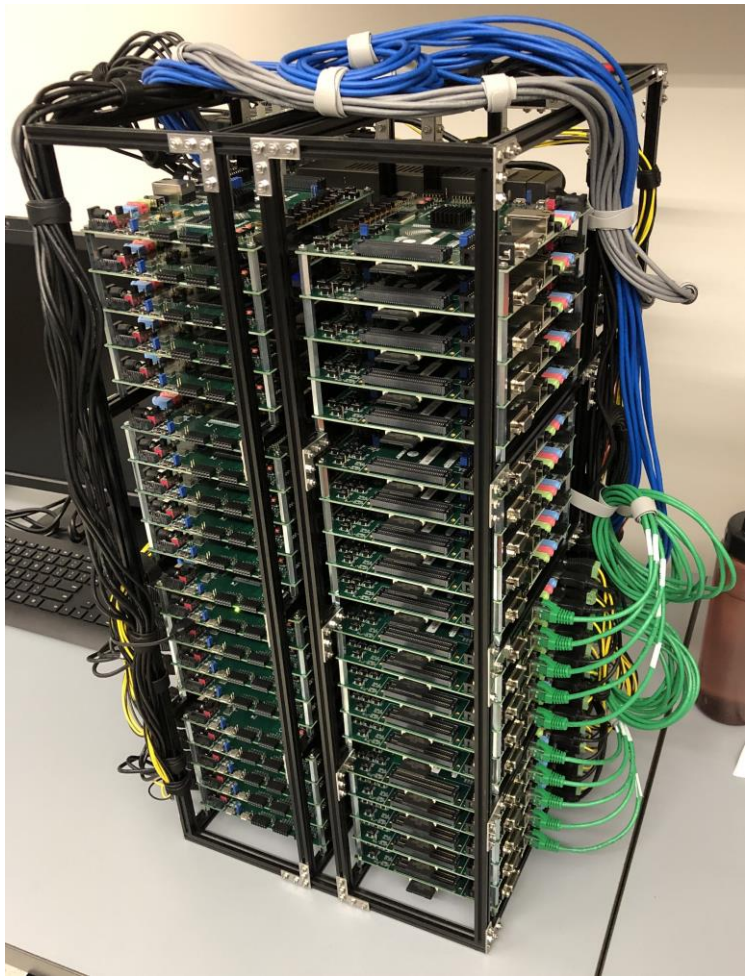
Model accuracy: total energy



Information Classification: General

Project Zeno Benchmark Cluster

- RISC-V testbench
- 50 Zedboards
  - Zynq-7000 SoC
  - 85k Logic cells
  - 100 + RV cores
- 3.4M Total logic cells
- 50 port gigabit ethernet switch
- ascslab.org/zeno.html



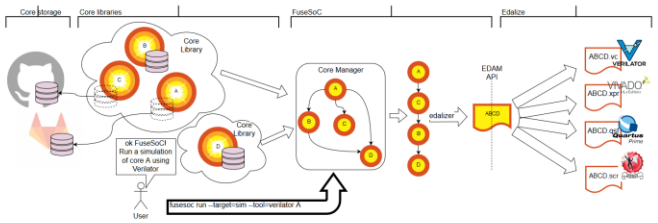
Information Classification: General

FuseSoC

Award-winning

An award-winning package manager and build system for HDL code. FuseSoC helps to increase reuse and decrease maintenance of IP cores and lets you focus on your core business instead of your cores.

Package manager
FuseSoC is your rpm for reconfigurable chips, your apt-get for ASIC. It manages libraries of IP cores, keeps track of dependencies between them and provides a uniform way to query and perform actions on your cores



Build system
FuseSoc is your CMake for CPLD, autotools for ASIC, scons for silicon. FuseSoC provides a uniform interface for all your EDA tools to make it easy to switch between simulators and use the same interface for synthesis and simulation tasks

FuseSoC makes it easier to

Use

Lint or synth? Func or formal? FuseSoC doesn't care what your EDA tool does with your HDL files. This gives you the flexibility to reuse your core descriptions and use the same interface regardless if you're creating an FPGA image, running formal or functional verification or performing code analysis

Habla Chisel? Sprechen Sie Migen? Your EDA tool probably only speak Verilog (or VHDL), but don't worry. FuseSoC has a mechanism called generators that can run your Chisel compiler, RAM generator, IP-XACT generator, C compiler or whatever you need on the fly to produce files that the EDA tools can digest. Generators can be shared between cores for maxium reuse

Reuse

Easy reuse of internal IP. Many companies create sizable libraries of IP cores over time and then spend a lot of time and effort attempting to reuse them in new projects. FuseSoC lets you easily build up a library of cores that can be reused and shared between projects, even if the projects are using different tools.

Easy reuse of third party IP In order to save time when building a product, it's often a good idea to reuse existing third party commercial or FOSSi cores for common building blocks. By using FuseSoC you can easily reuse what others have already built and fill in the gaps with your own cores.

And the converse is true as well. As an IP designer you can make it easier for others to reuse your work by adding a FuseSoC core description file

Olof Kindgren

Maintain

New project, new tools. Starting a new project you might want to reuse cores that have been simulated or synthesized with a different tool than the one you intend to use for the new project. No problem. Instead of manually setting up the configuration files for the new tool, FuseSoC lets you describe the properties of your cores in a tool-agnostic core description file and creates the project files for your EDA tool of choice automatically.

Code is a cost, not an asset. You might tempted to create your own package manager or build system. But before doing that, please take a minute to consider all the thousands of lines of TCL, Makefiles, Perl or Python scripts you will need to maintain, how much time you will need to train new engineers on your particular tooling and how dependent you will be on the single person who actually understands how it all fits together

Wow! This sounds great but I also need

Additional tool support. FuseSoC currently support 15 different simulators, synthesis tools and linters. Is your favorite tool missing? Write it down here

Additional cores There are hundreds of cores available today as FuseSoC packages, but not that's not all. Is there some core you would love to use but is not available as a FuseSoC package? Write it down here

Additional features FuseSoC is already a versatile multi tool, but is it missing *that* feature to make it really useful? Write it down here

qamcom
qamcom
https://www.qamcom.se
OlofKindgren
https://linkedin.com/in/olofkindgren
FossiFoundation
https://fossi-foundation.org

Information Classification: General

MANCHESTER 1824
The University of Manchester

RISC-V on Maxine-VM: An educational use-case

F. Zakkak, F. Blanaru, C. Kotselidis

Maxine-VM:

- Meta-circular research VM for Java
- Multiple JIT-compilers (baseline and optimizing)
- Multiple GC algorithms (MMTk compatibility)
- Multiple ISAs (RISC-V, x86_64, ARMv7, Aarch64)

Maxine-VM RISC-V status:

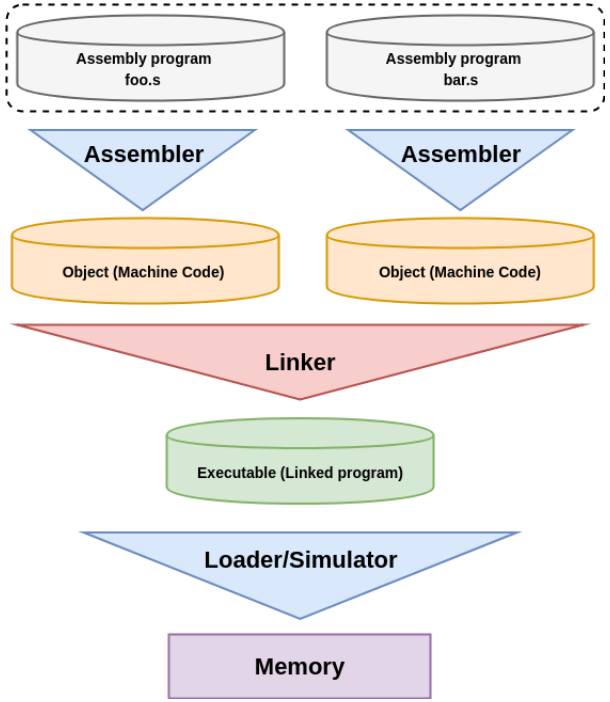
- Currently in booting phase (finalized by Q3)
- Set of tools for visualizing:
  - JIT'ed Java code for RISC-V
  - Full compilation path from Java code to RISC-V assembly
  - Ideal for educational purposes for compilers and RISC-V

Information Classification: General

RISC-V Workshop

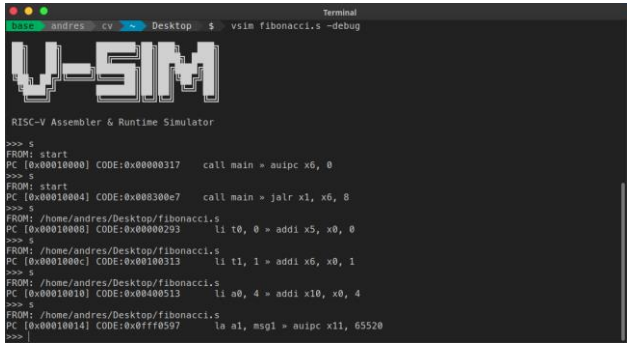
V-Sim: An Open Source and Education-Oriented RISC-V Assembler and Runtime Simulator

Architecture

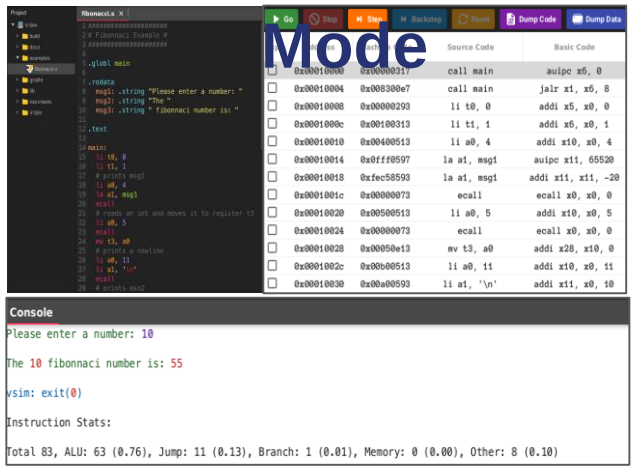


Documentation

CLI



GUI Mode



https://andrescv.github.io/V-Sim

Information Classification: General

XCrypto: A General Purpose Crypto ISE for RISC-V

University of BRISTOL

- An alternative crypto ISE which *does not rely on the vector extension.*
- Designed for *embedded cores* requiring side-channel resiliance:
  - Maximises opportunities to include coutermeasures.
  - Minimises cost of applying said countermeasures wrt. Power/Perf/Area.
- RISC(ish) approach: many general instructions, not one per algorithm:
  - Allows hardware and software side-channel countermeasures to co-exist.
  - Recognises that *crypto standards will change* over the life of an ISA.
- `github.com/scarv/xcrypto.git`
- Part of the SCARV project: Building an open source side-channel resistant RISC-V CPU for secure and efficient embedded cryptography.

Information Classificatior

The first porting of Linux OS support for Approximate Memory management on RISC-V

Giulia Stazi, Mauro Olivieri and Francesco Menichelli

SAPIENZA
UNIVERSITÀ DI ROMA

Dept. of Information Engineering, Electronics and Telecommunications

- **Approximate Memories (AxM):**
  - circuit-level or architecture-level techniques (i.e. voltage scaling, refresh rate reduction) to reduce energy consumption at the expense of rising error rate in bit cells
  - **ETAs**: applications can allocate data in AxM if they can tolerate a certain degradation in output quality

Machine learning

Data mining

Image processing

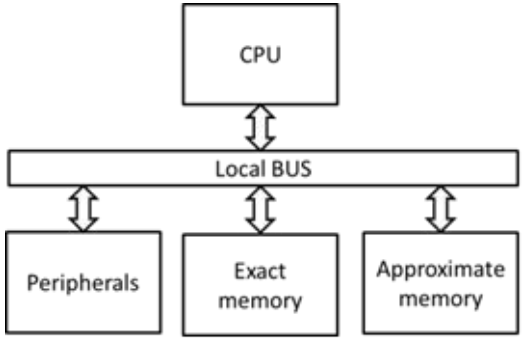
In this work we describe the first porting of Approximate Memory support on Linux OS for RISCV32 and RISCV64 architectures.

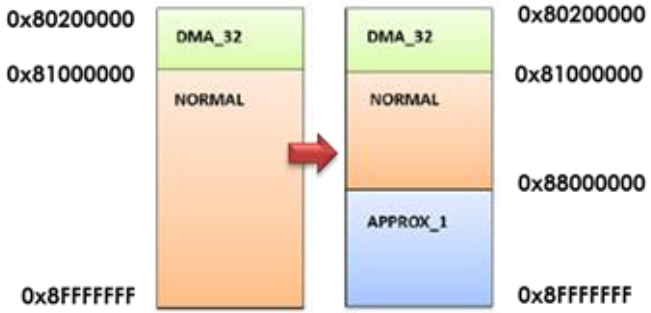
RISC-V

11-12 June 2019

Information Classification: General

General implementation and example of application






```
[    0.000000] Linux version 4.20.0 (giulia@vlsi3) (gcc version 7.4.0 (Bui
9fd5a)) #50 SMP Wed Feb 6 20:44:22 CET 2019
[    0.000000] printk: bootconsole [early0] enabled
[    0.000000] initrd not found or empty - disabling initrd
[    0.000000] Zone ranges:
[    0.000000]   DMA32    [mem 0x0000000080200000-0x0000000080ffffff]
[    0.000000]   Normal   [mem 0x0000000081000000-0x0000000087ffffff]
[    0.000000]   Approximate [mem 0x0000000088000000-0x000000008fffffff]
[    0.000000] Movable zone start for each node
```

The kernel is currently running on **SiFive_u** virtual machine emulated in **QEmu** v3.1.0

Example: H.264 encoder using exact memory for frame buffers (A) and approximate memory (B), ***error rate 1E-3 [errors/(bit x s)]***

Information Classification: General

OpenHW Group

Proven Processor IP

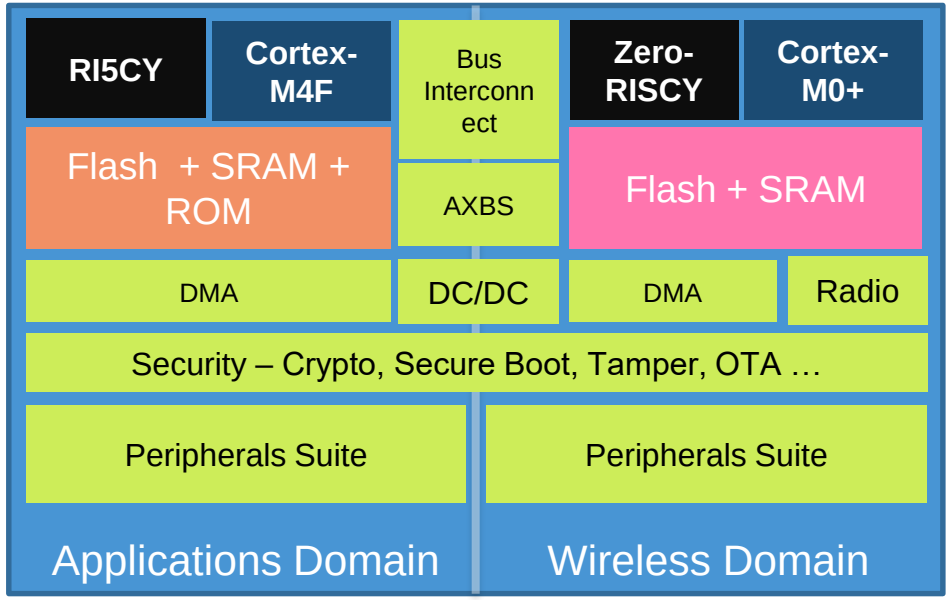
CORE-V Demo Preview

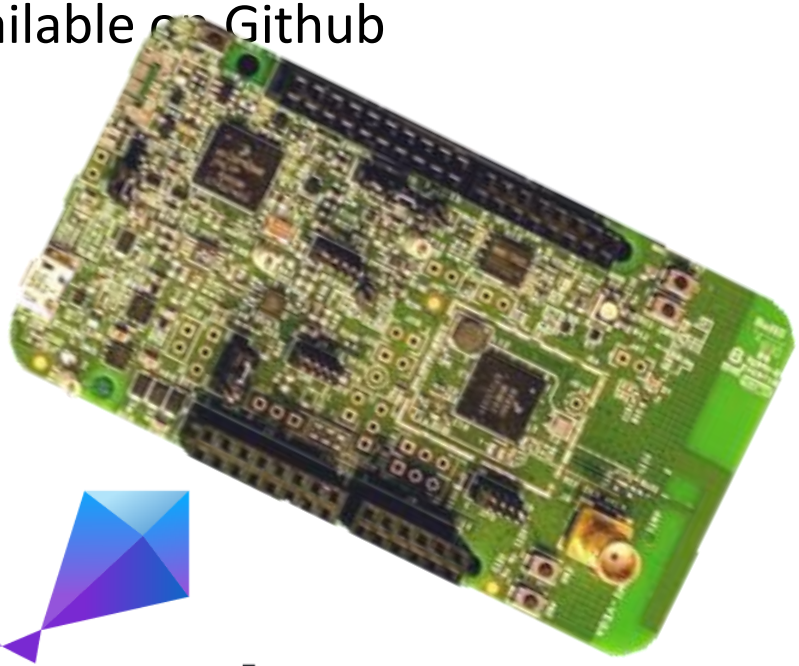
Rick O'Connor rickoco@openhwgroup.or
@rickoco

© OpenHW Group - RISC-V / WOSH ETH Zurich

CORE-V NXP VEGA Demo

- Main application running on CORE-V (RI5CY) core
- BLE controller SW (Link Layer and PHY) running on Ibex (Zero-RISCY) core
- Micropython + drivers, all upstreamed and available on Github





Zephyr™ Project

Information Classification: General
© OpenHW Group - RISC-V / WOSH ETH Zurich

CORE-V Verification Test Bench Demo

(proof of concept)

- Metrics: System Verilog simulator running on Google Cloud
- Imperas: Golden reference model of RISC-V CPU for comparison to DuT

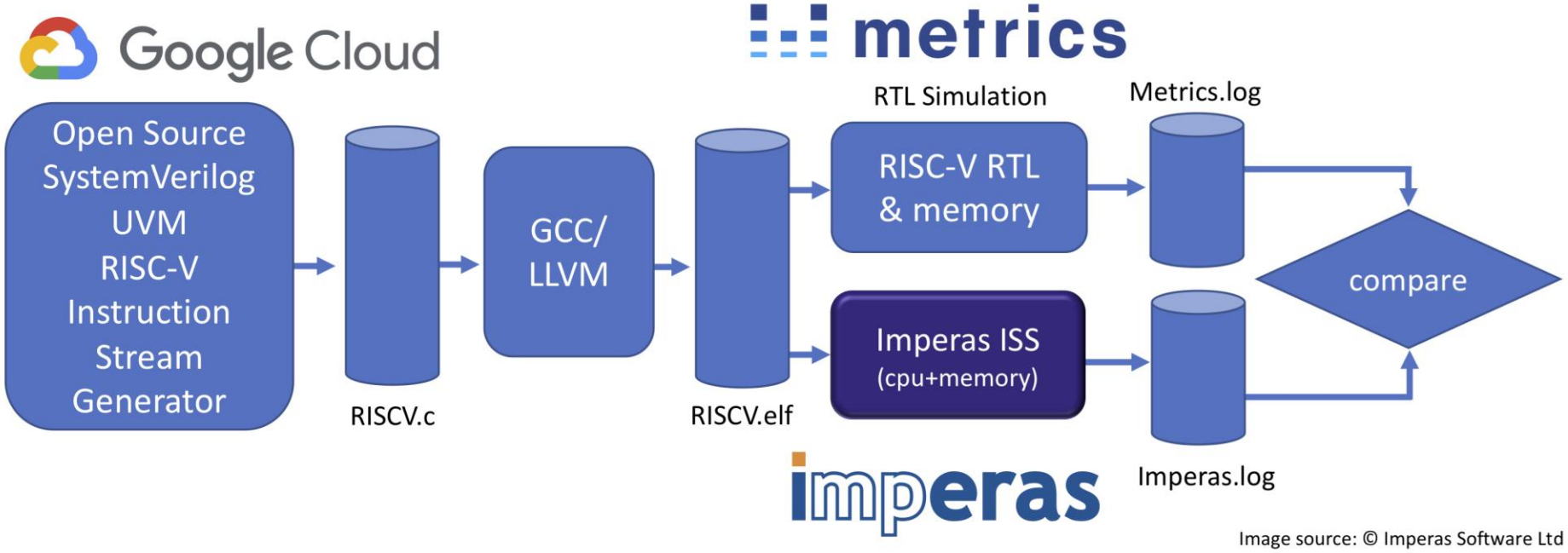


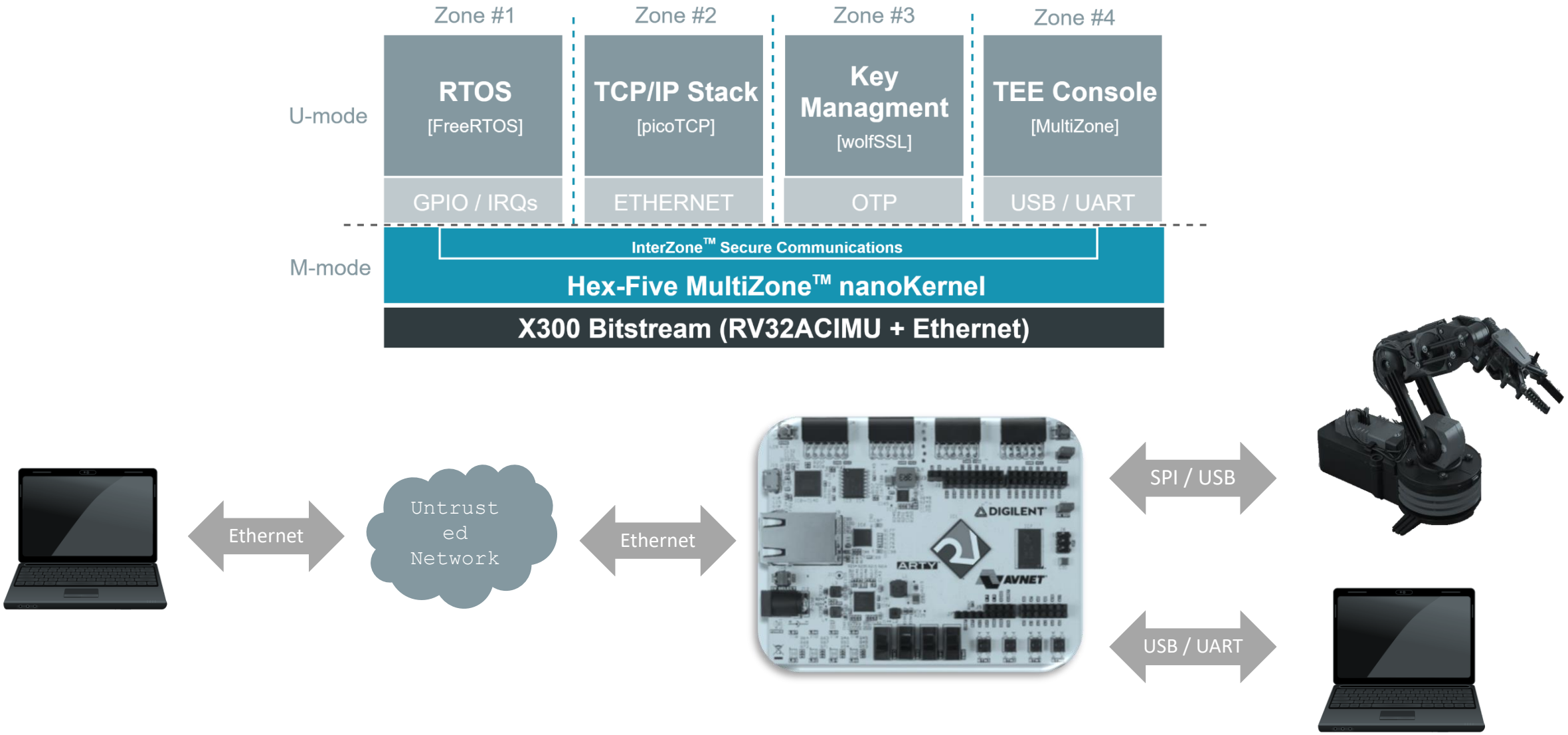
Image source: © Imperas Software Ltd

© OpenHW Group - RISC-V / WOSH ETH Zurich

HOW TO BUILD SECURE RISC-V IOT DEVICES?

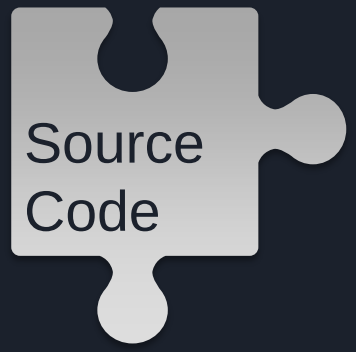
- **FreeRTOS** – widespread RTOS for embedded systems.
- **wolfSSL** – de-facto standard crypto library for embedded.
- **picoTCP** – self-contained TCP/IP stack.
- **MultiZone** – Trusted Execution Enviornment and secure boot.
- **Xilinx XEmacLite** – AXI Ethernet Lite MAC IP.
- **Mod. E300 SoC** – modified Rocket for Xilinx Arty7.
- **GNU Toolchain** – programming tools for developing software.

SECURE IOT STACK FOR RISC-V



Analyzing RISCV Binaries

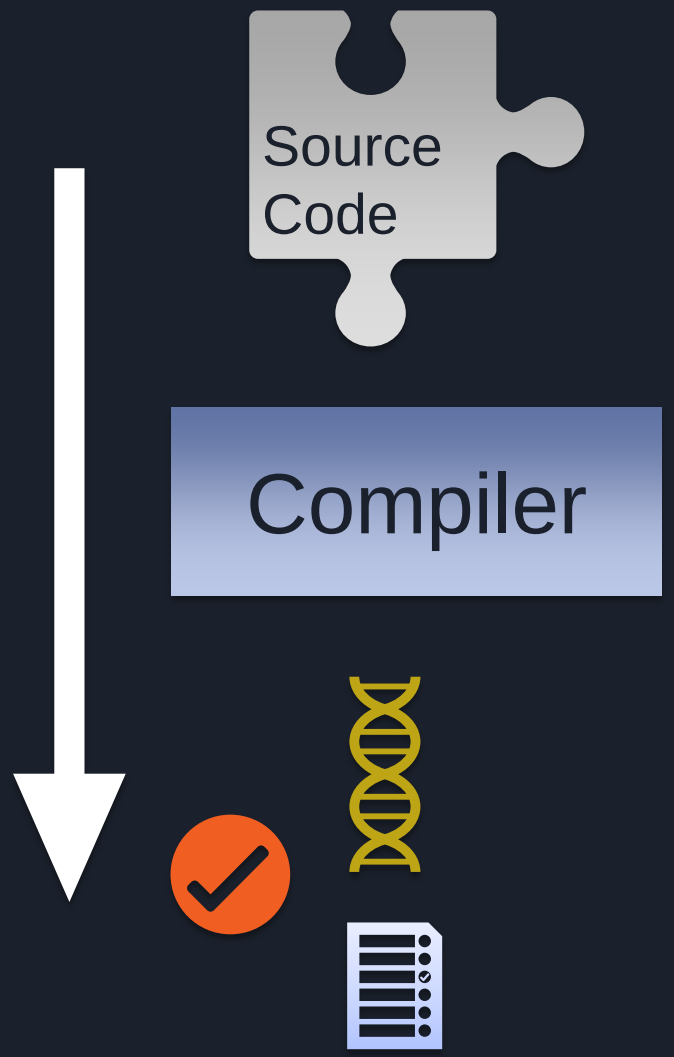
David Demicco and Aravind Prakash, Binghamton University



Information Classification: General

Analyzing RISCV Binaries

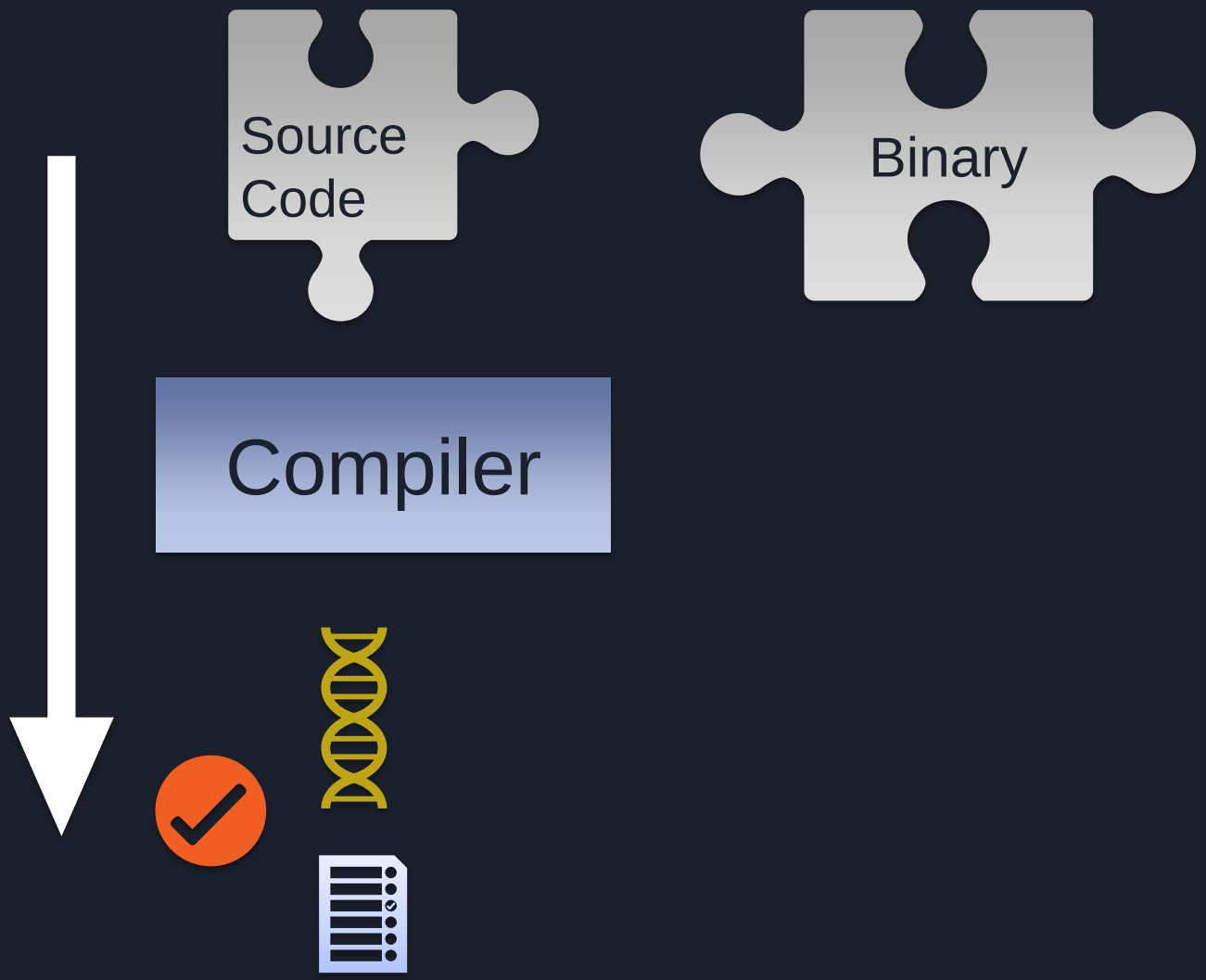
David Demicco and Aravind Prakash, Binghamton University



Information Classification: General

Analyzing RISCV Binaries

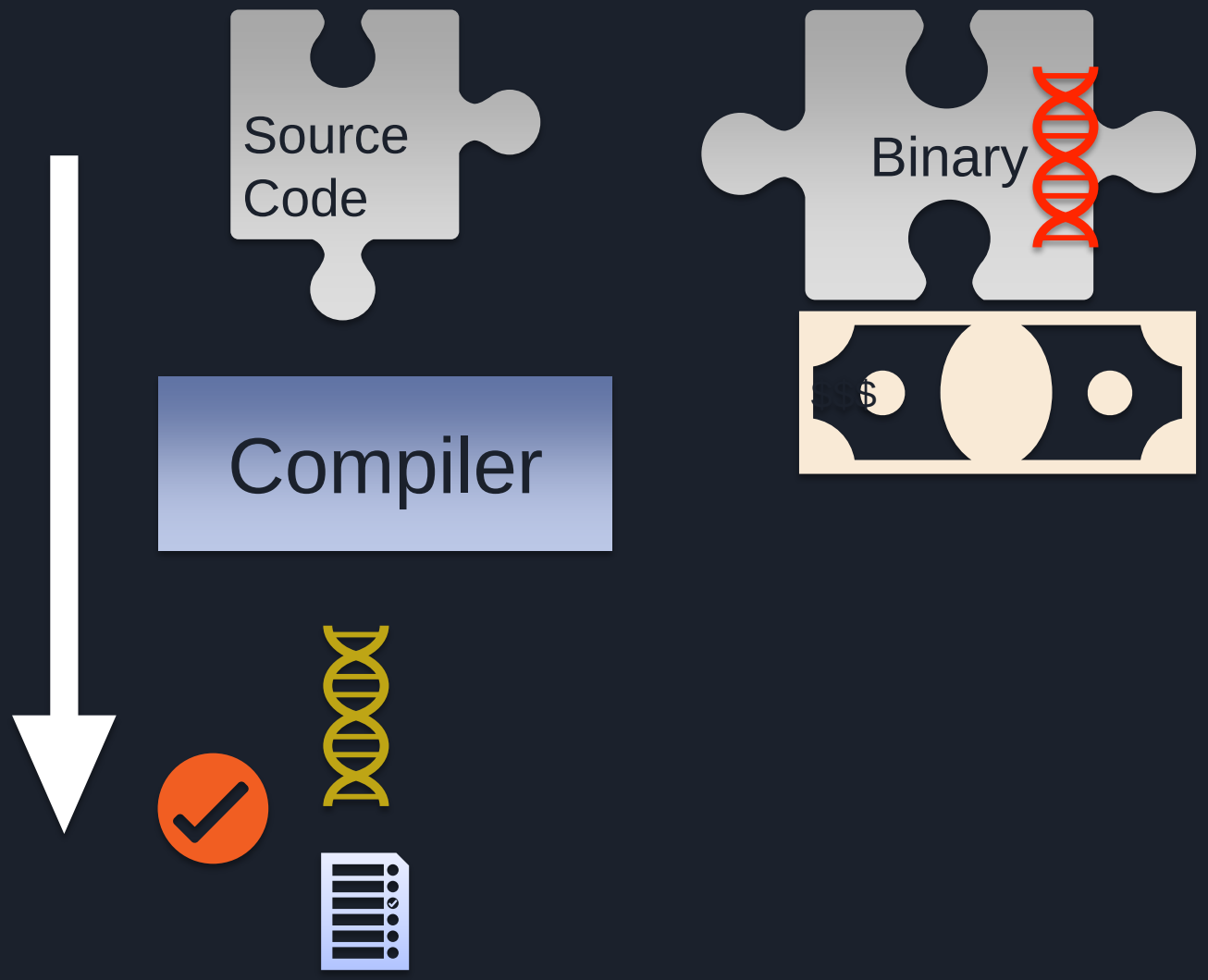
David Demicco and Aravind Prakash, Binghamton University



Information Classification: General

Analyzing RISCV Binaries

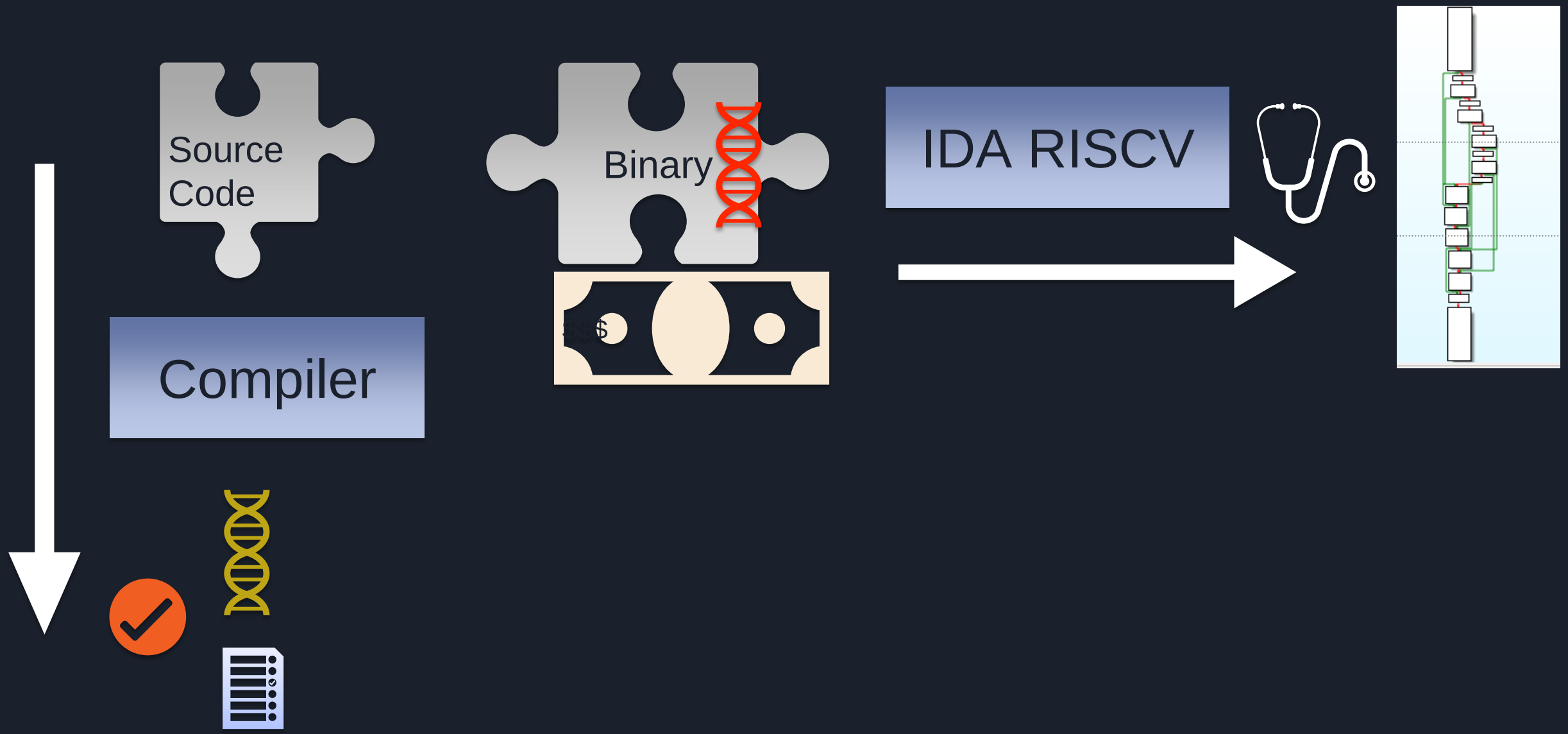
David Demicco and Aravind Prakash, Binghamton University



Information Classification: General

Analyzing RISCV Binaries

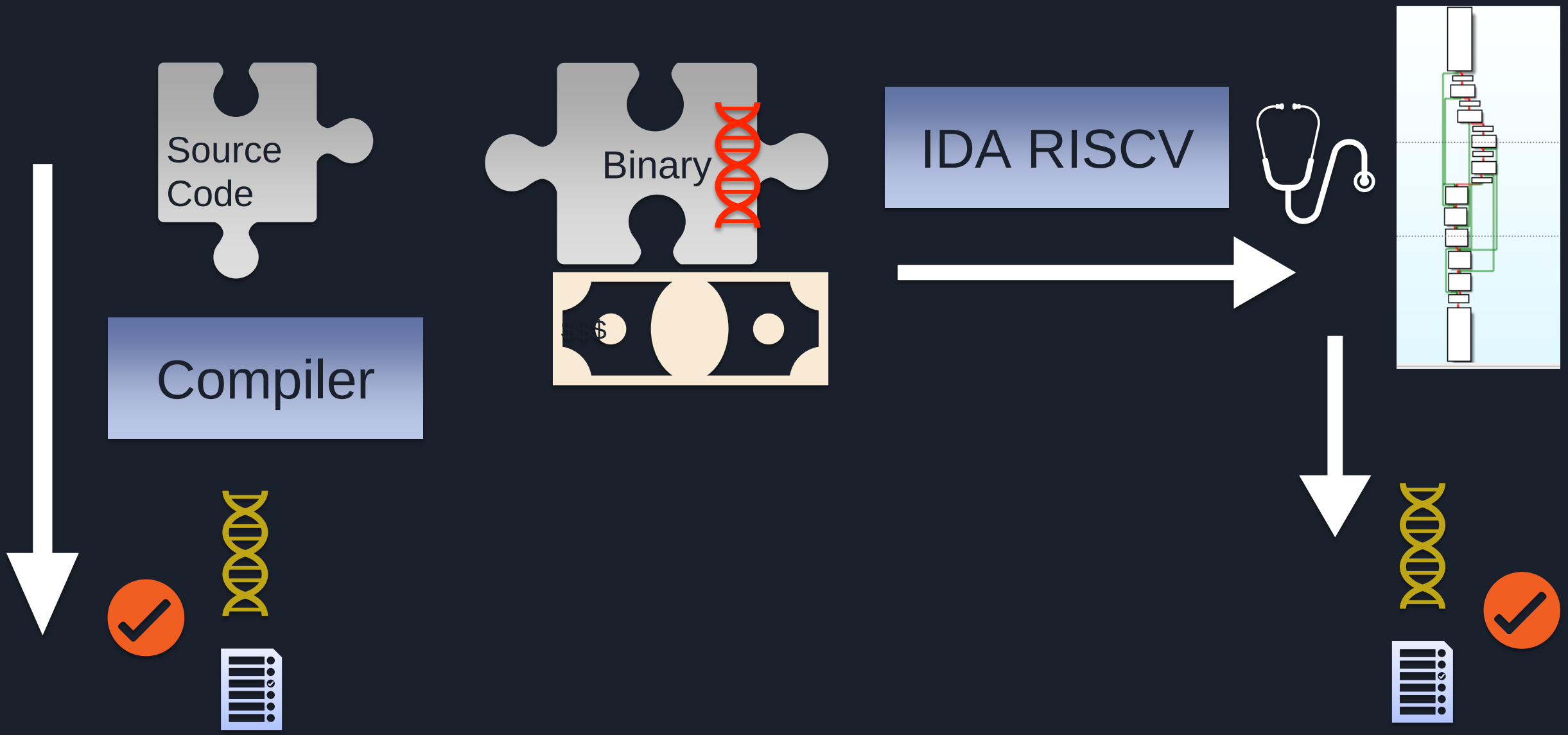
David Demicco and Aravind Prakash, Binghamton University



Information Classification: General

Analyzing RISCV Binaries

David Demicco and Aravind Prakash, Binghamton University

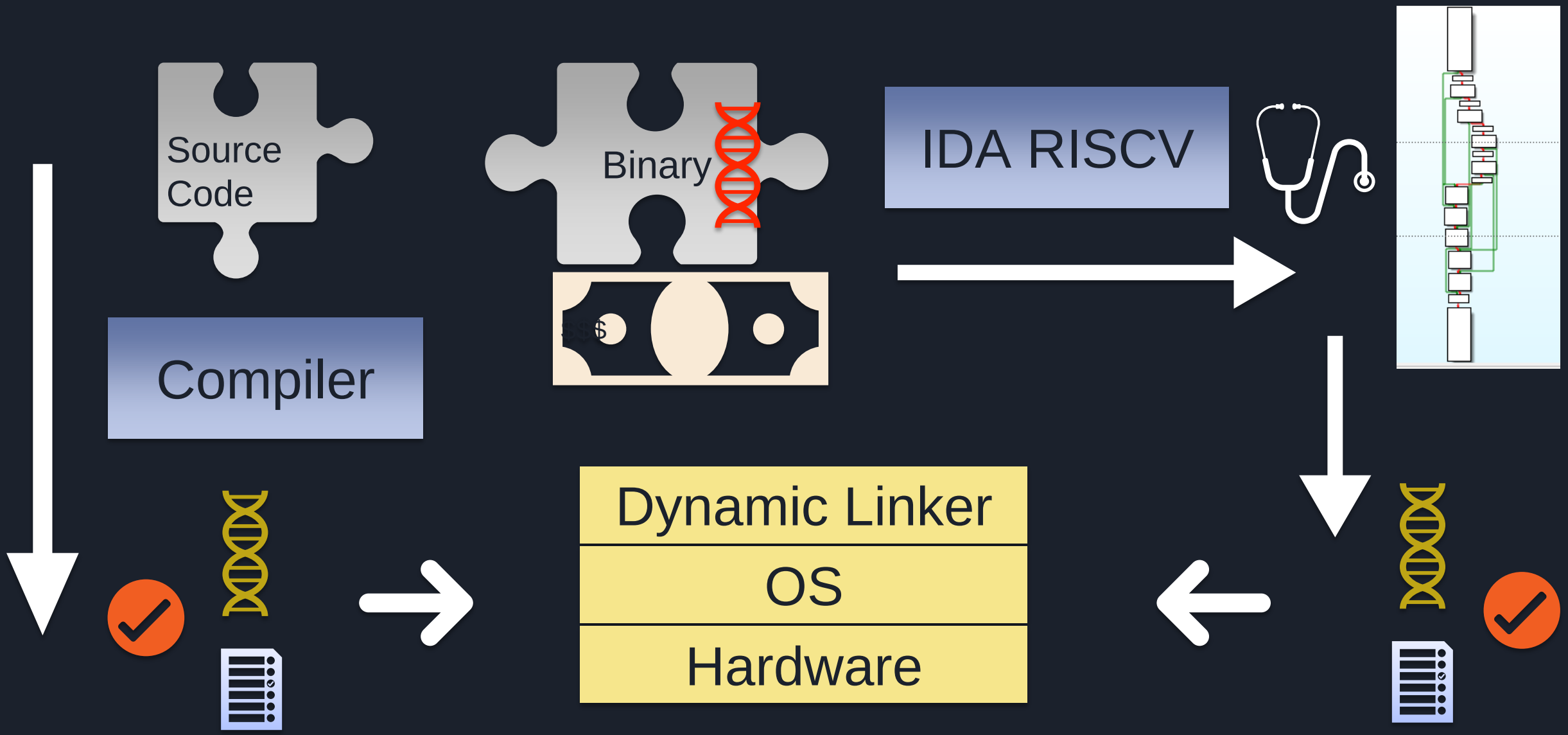


Information Classification: General

Analyzing RISCV Binaries

David Demicco and Aravind Prakash, Binghamton University

BINGHAMTON UNIVERSITY
State University of New York

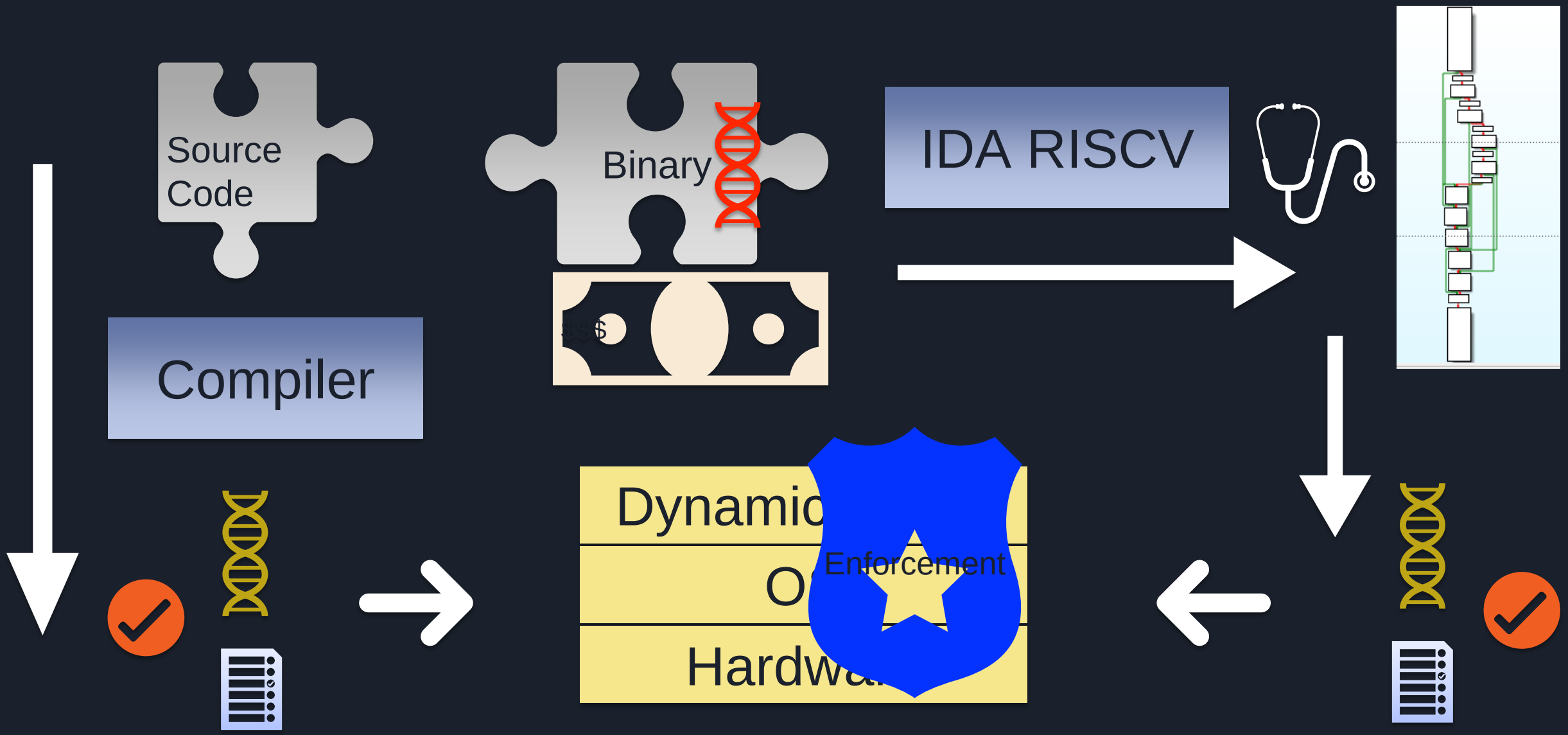


Information Classification: General

Analyzing RISCV Binaries

David Demicco and Aravind Prakash, Binghamton University

BINGHAMTON UNIVERSITY
State University of New York

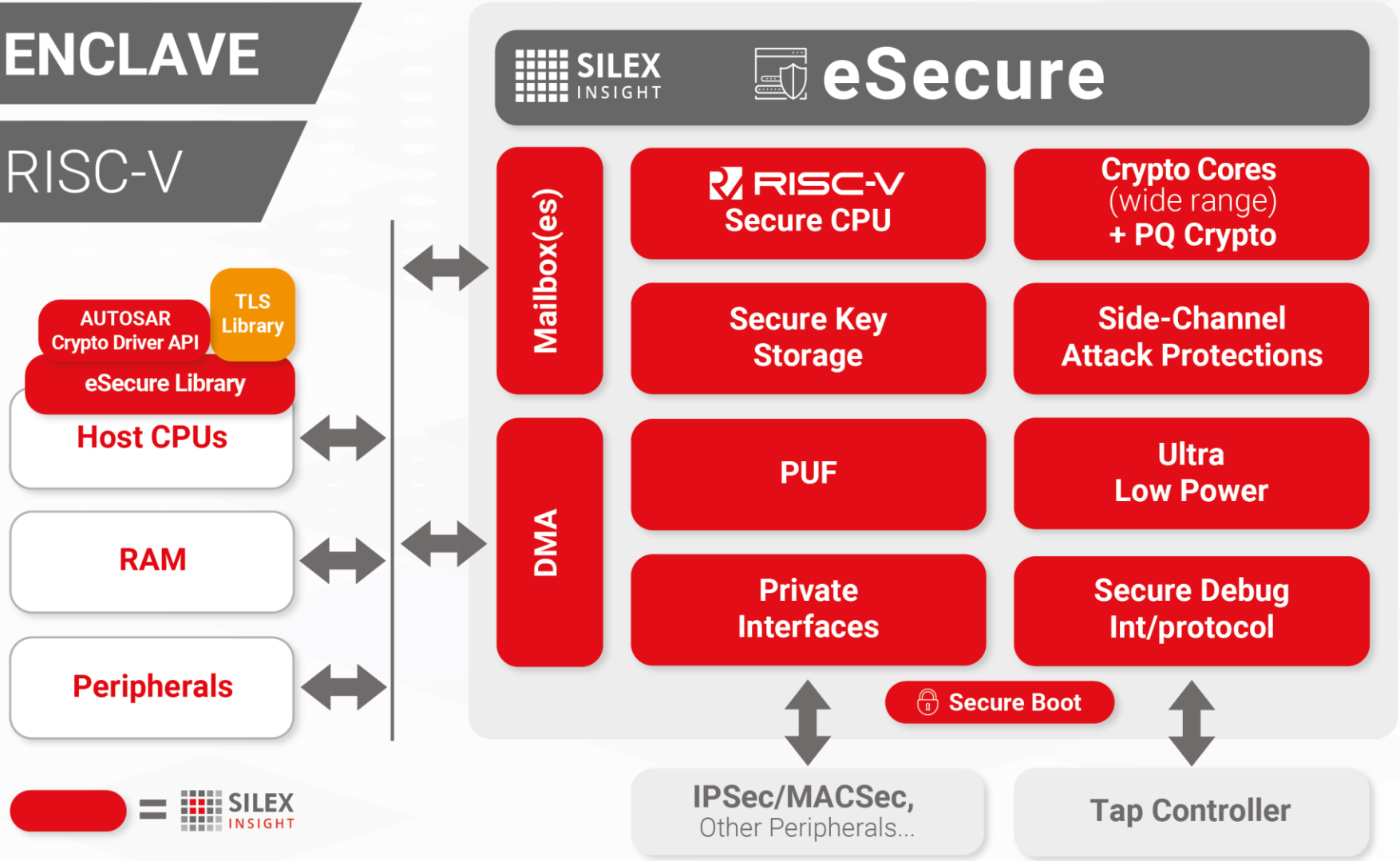


Information Classification: General

SILEX INSIGHT

SECURITY ENCLAVE

BASED ON RISC-V



Information Classification: General

Fault Injection Framework to Study Soft Error Resilience of RISC-V Cores on FPGA

Jhalak Sharma, Subir K. Roy, Nanditha Rao - IIIT- Bangalore, Karnataka, India

GOAL:

To design a controller-based fault injection framework for RISC-V core, program it on FPGA and to study the probability of multiple bit-flips.

INTRODUCTION:

- Soft errors are random, non-recurring change of state due to energetic particles interacting with the silicon.
- High energy particles strike a semiconductor substrate, create charge → glitch or a transient current → Single Event Transient (SET).
- If transient current crosses the critical charge (Qc) of the memory cell, the stored data may flip and cause a soft

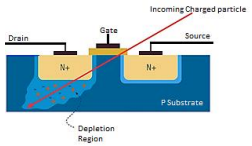


Fig 1. Generation of SET

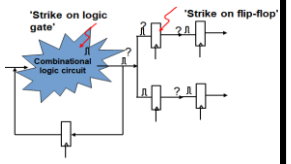


Fig 2. SET propagation

- Significant impact on Circuit reliability.
- Can propagate and get captured in register/flip-flops

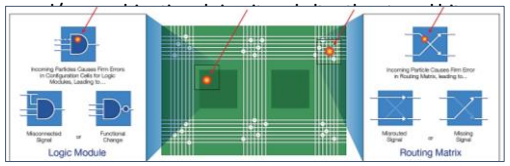
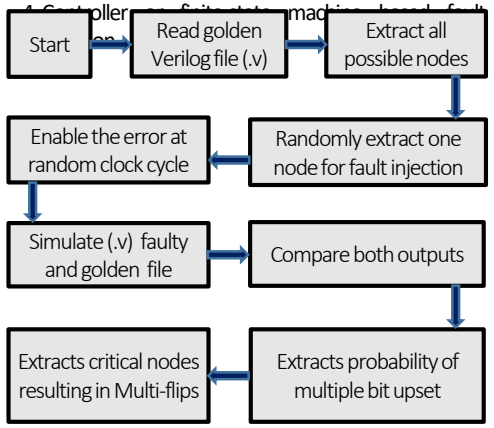


Fig 3. Soft Error in Logic and Routing module [4]

- The study will evaluate the soft error vulnerability of cores such as rocket and BOOM.
- Reduced voltage and shrinking node capacitance makes the devices more susceptible to soft errors.

METHODOLOGY:

1. Fault injection in the Verilog netlist
2. Golden and fault injected Monte-Carlo simulations
3. Statistics to evaluate MBUs and extract critical nodes.
4. Controller based finite state machine based fault injection



Repeat till the probability hits 95% confidence level

EXPERIMENTAL SETUP:

1. Golden RTL simulated for 5000 clock cycle.
2. Fault injected at random node at random clock cycle, simulated for 5000 cycles.

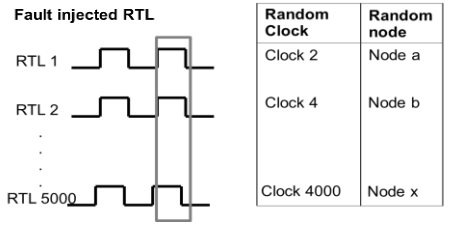


Fig 3. Setup for fault injection

RESULTS:

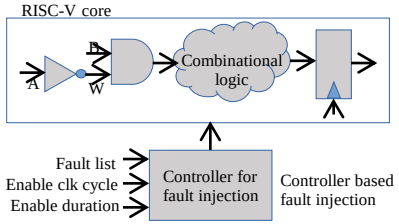


Fig 4. Fault injection Framework

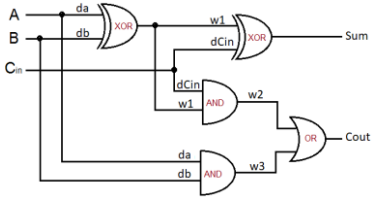
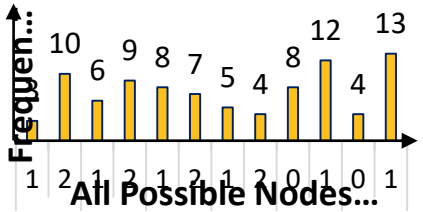
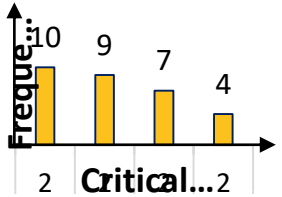


Fig 5. Design Under Test- Full Adder

Observations





1. Total simulation clock cycle = 100 cycles
2. Probability of Single bit upset = 47%
3. Probability of Multi bit upset = 30%
4. Probability of Fault Masked = 12%

FUTURE WORK:

- Entire experiment is expected to run for different digital designs benchmarks such as: ISCAS 85, ISCAS 89, ITC 99 and test these by programming it on FPGAs.
- Based on the error probabilities, propagation characteristics and vulnerable nodes, we will need to use suitable error correction schemes.

REFERENCES:

1. ABDUL RAFAY KHATRI, M. M. (n.d.). Instrumentation Technique for FPGA based Fault Injection Tool. *Recent Advances in Financial Planning and Product Development*, (p. 7).
2. Ghazanfar Asadi, M. B. (2005). Soft Error Rate Estimation and Mitigation for SRAM Based FPGAs. *FPGA'05, February 20–22, 2005, Monterey, California, USA.*, (p. 12).
3. Haissam Ziade1, R. A. (2011). A New Fault Injection Approach to Study the Impact of Bitflips in the Configuration of SRAM-Based FPGAs. *The International Arab Journal of Information Technology.*
4. https://www.microsemi.com/product-directory/reliability/4883-see

CONTACT INFO

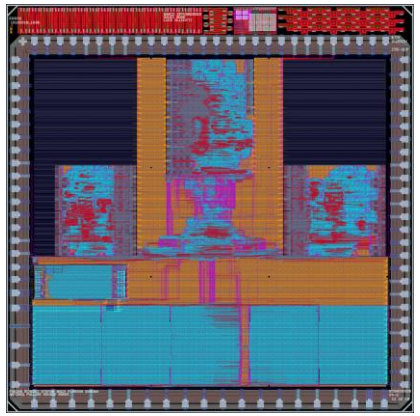
1. jhalak.sharma@iiitb.org 2. subir@iiitb.ac.in 3. nanditha.rao@iiitb.ac.in (*CEEMS Lab, International Institute of Information Technology Bangalore. Bengaluru – 560100, India*)

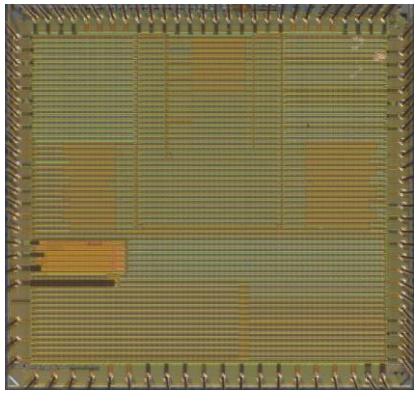
Information Classification: General

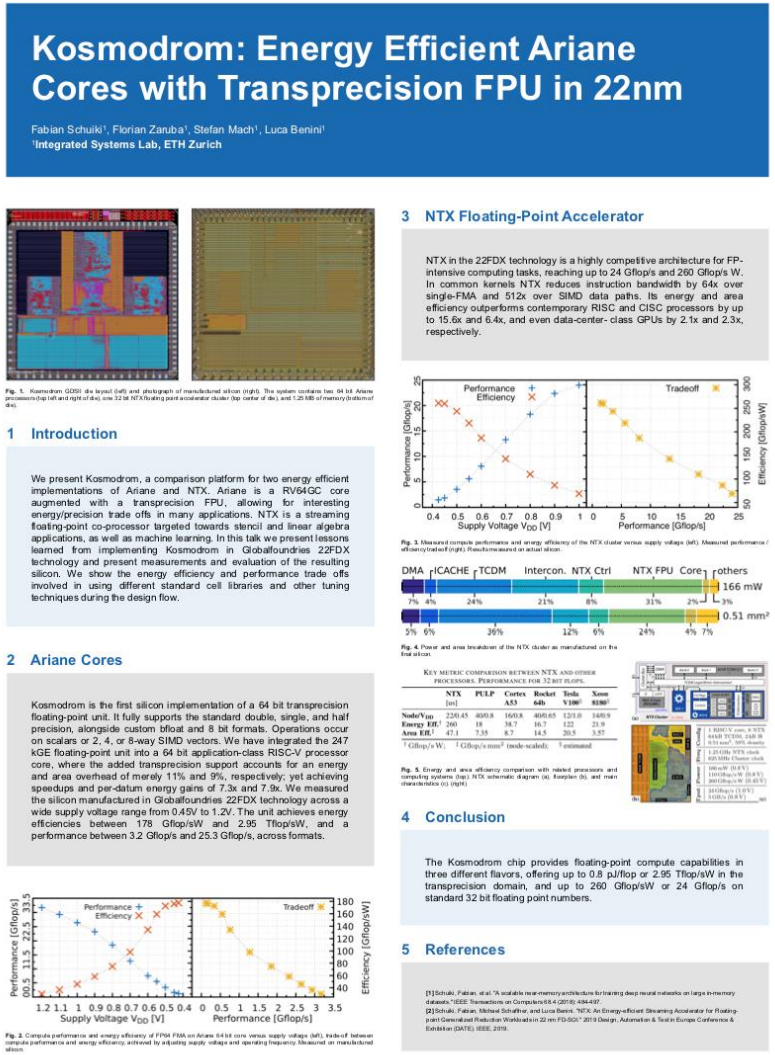
Kosmodrom: Energy Efficient Ariane Cores with Transprecision FPU in 22nm

Fabian Schuiki, Florian Zaruba, Stefan Mach, Luca Benini

- First silicon implementation of a transprecision floating-point unit
  - double, single, half, bfloat, custom 8 bit format
- Two Ariane cores (different cell flavors)
  - Efficiency: 178 Gflop/sW to 2.95 Tflop/sW
  - Performance: 3.2 Gflop/s to 25.3 Gflop/s
- One NTX cluster (32 bit float acceleration)
  - Efficiency up to 260 Gflop/sW
  - Performance up to 24 Gflop/s
- Come talk to us!







ETH zürich

Kosmodrom: Energy Efficient Ariane Cores with Transprecision FPU in 22nm

Fabian Schuiki[1], Florian Zaruba[1], Stefan Mach[1], Luca Benini[1]
[1]Integrated Systems Lab, ETH Zurich

1 Introduction

We present Kosmodrom, a comparison platform for two energy efficient implementations of Ariane and NTX. Ariane is a RV64GC core augmented with a transprecision FPU, allowing for interesting energy/precision trade offs in many applications. NTX is a streaming floating-point co-processor targeted towards stencil and linear algebra applications, as well as machine learning. In this talk we present lessons learned from implementing Kosmodrom in Globalfoundries 22FDX technology and present measurements and evaluation of the resulting silicon. We show the energy efficiency and performance trade offs involved in using different standard cell libraries and other tuning techniques during the design flow.

2 Ariane Cores

Kosmodrom is the first silicon implementation of a 64 bit transprecision floating-point unit. It fully supports the standard double, single, and half precision, alongside custom bfloat and 8 bit formats. Operations occur on scalars or 2, 4, or 8-way SIMD vectors. We have integrated the 247 kGE floating-point unit into a 64 bit application-class RISC-V processor core, where the added transprecision support accounts for an energy and area overhead of merely 11% and 9%, respectively; yet achieving speedups and per-datum energy gains of 7.3x and 7.9x. We measured the silicon manufactured in Globalfoundries 22FDX technology across a wide supply voltage range from 0.45V to 1.2V. The unit achieves energy efficiencies between 178 Gflop/sW and 2.95 Tflop/sW, and a performance between 3.2 Gflop/s and 25.3 Gflop/s, across formats.

3 NTX Floating-Point Accelerator

NTX in the 22FDX technology is a highly competitive architecture for FP-intensive computing tasks, reaching up to 24 Gflop/s and 260 Gflop/s W. In common kernels NTX reduces instruction bandwidth by 64x over single-FMA and 512x over SIMD data paths. Its energy and area efficiency outperforms contemporary RISC and CISC processors by up to 15.6x and 6.4x, and even data-center- class GPUs by 2.1x and 2.3x, respectively.

4 Conclusion

The Kosmodrom chip provides floating-point compute capabilities in three different flavors, offering up to 0.8 pJ/flop or 2.95 Tflop/sW in the transprecision domain, and up to 260 Gflop/sW or 24 Gflop/s on standard 32 bit floating point numbers.

5 References

Information Classification: General

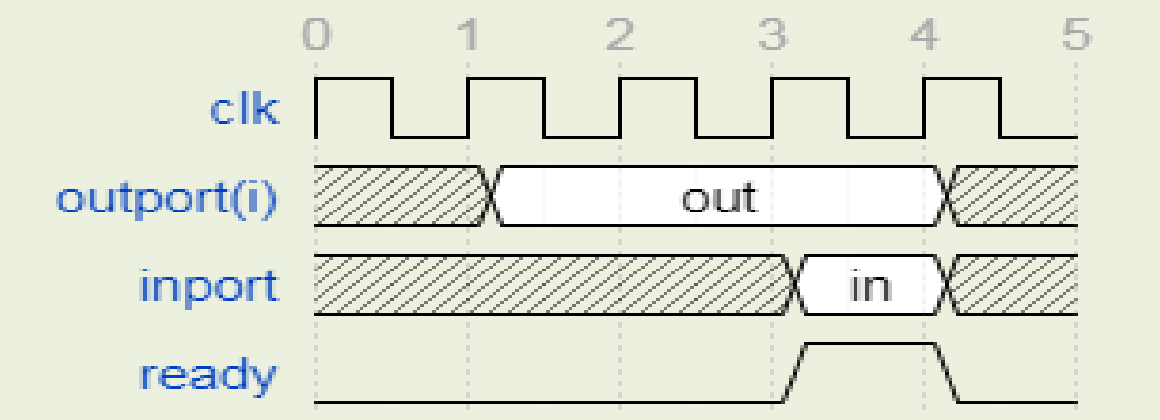
Andes Custom Extension™ (ACE)

Custom Port for Direct RTL Connection

```
port outport {          //4 32-bit output ports
    num= 4;
    width= 32;          //bit[31]: valid bit
    io_type= out;
}
port inport {           //a 32-bit input port
   width= 32;
   io_type= in;
}
port ready {            //a ready signal
   io_type= in;
}
```

```
insn acp_hdshk {
  op= {out gpr rslt, outport OP, in gpr odata};
  implied_op= {inport, ready};
  csim= %{
    acp_write_OP(odata | (1<<31));
    while( ! acp_read_ready())
        ace_advance_cycle(1);
    rslt = acp_read_inport() + odata;
    acp_write_OP(0x7fffffff);        //clear valid bit
  %};
};
```

acp_hdshk.ace



acp_hdshk.v

```
//ACE_BEGIN: acp_hdshk
   assign OP = {1,odata[30:0]};  //all 0's if invalid
   assign rslt = inport + odata;
   assign ace_dovalid = ready;   //instr completed
//ACE_END
```

RISC-V FreeStart *with RISK-Free RISC*

- **FreeStart:**
  - **AndesCore™ N22 processor**:
    - Full-featured, configurable, ecosystem-supported
    - Up to 3.95 Coremark/MHz
    - With no upfront fee, reasonable production royalty
  - **AndeSight™**: Feature-rich IDE available for free download
  - **Optional Support Package**, including an AHB platform IP
- **Website: freestart.andestech.com**

Unbounded Formal Verification of RISC-V CSRs with Interval Property Checking

RISC-V Workshop

11-12 June 2019, Zurich

Nicolae Tusinschi | nicolae.tusinschi@onespin.com

Product Specialist Design Verification | **OneSpin Solutions**

Functional Verification of RISC-V Cores

Does the RTL precisely implement the RISC-V ISA spec?

RISC-V processor cores are hard to verify

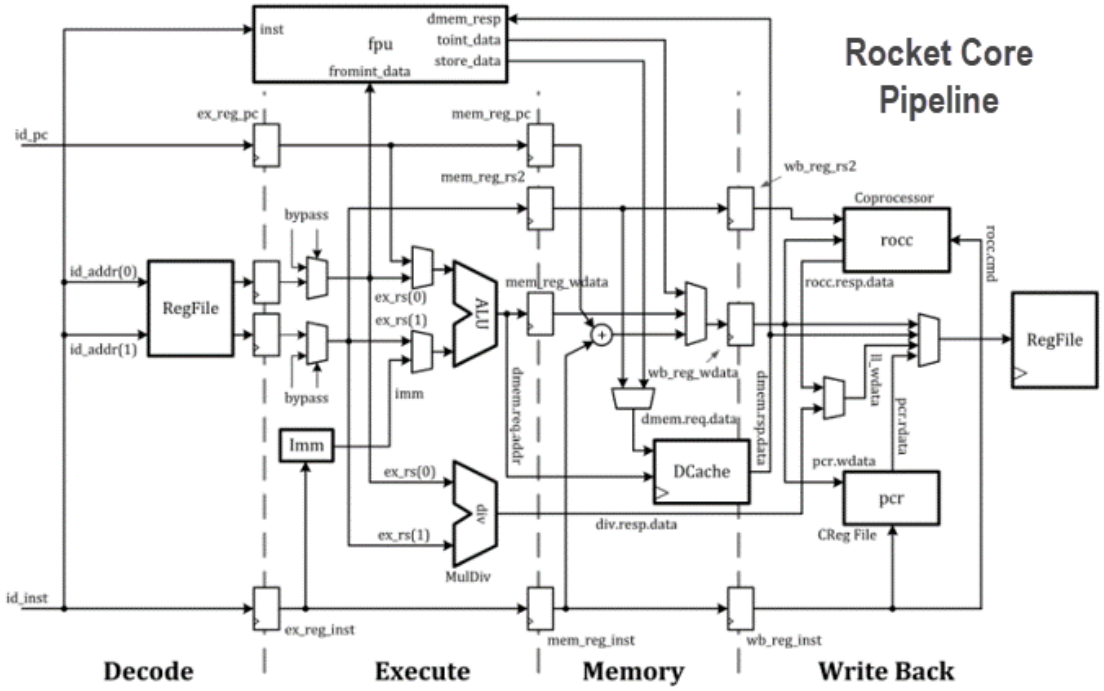
- Complex micro architectures to achieve PPA targets
- Brach prediction – Forwarding – Out-of-order execution …

Formal verification

- Exhaustive verification finds corner-case bugs
- The only technology that can prove bug absence

Challenges

- Complexity issues lead to bounded proofs
- Hard to write good quality, reusable assertions



OneSpin RISC-V Integrity Verification Solution

Industry's first formal RISC-V compliance solution

RISC-V ISA formalized as SystemVerilog Assertions (SVA)

- Decoupled from microarchitectural details

Enables 100% unbounded formal proof for implementation of all instructions against ISA specification

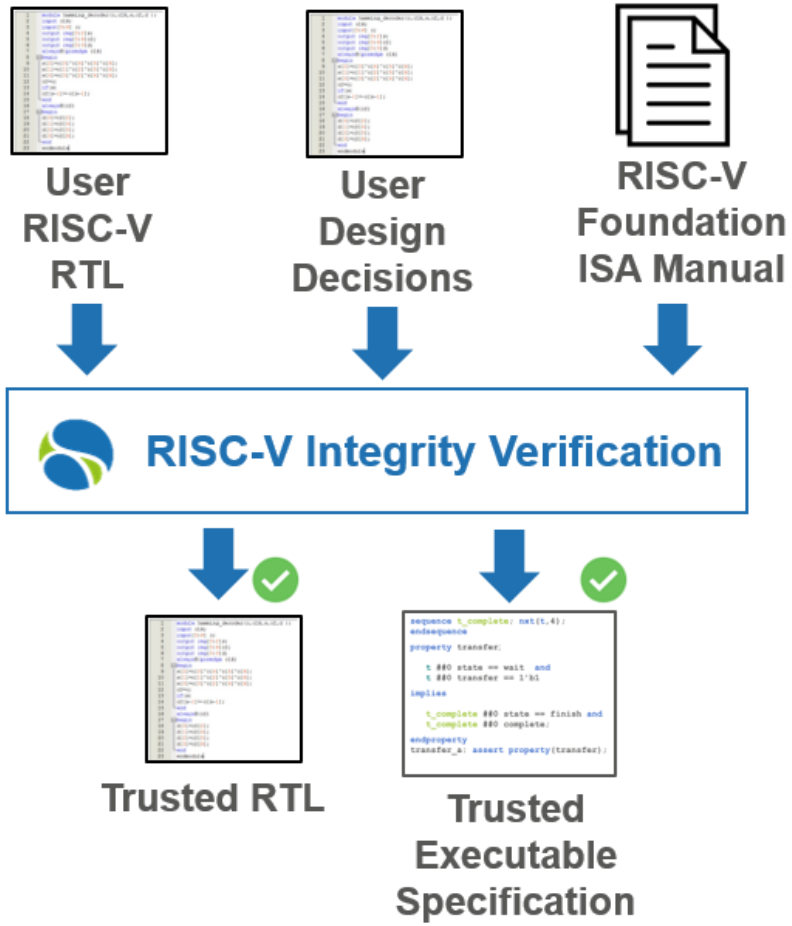
- Guarantees that core is fully compliant

Enables verification of entire core implementation

- Leverages user-provided microarchitectural details

Detects any extra functionality beyond the ISA specification

- Formally verifies trust

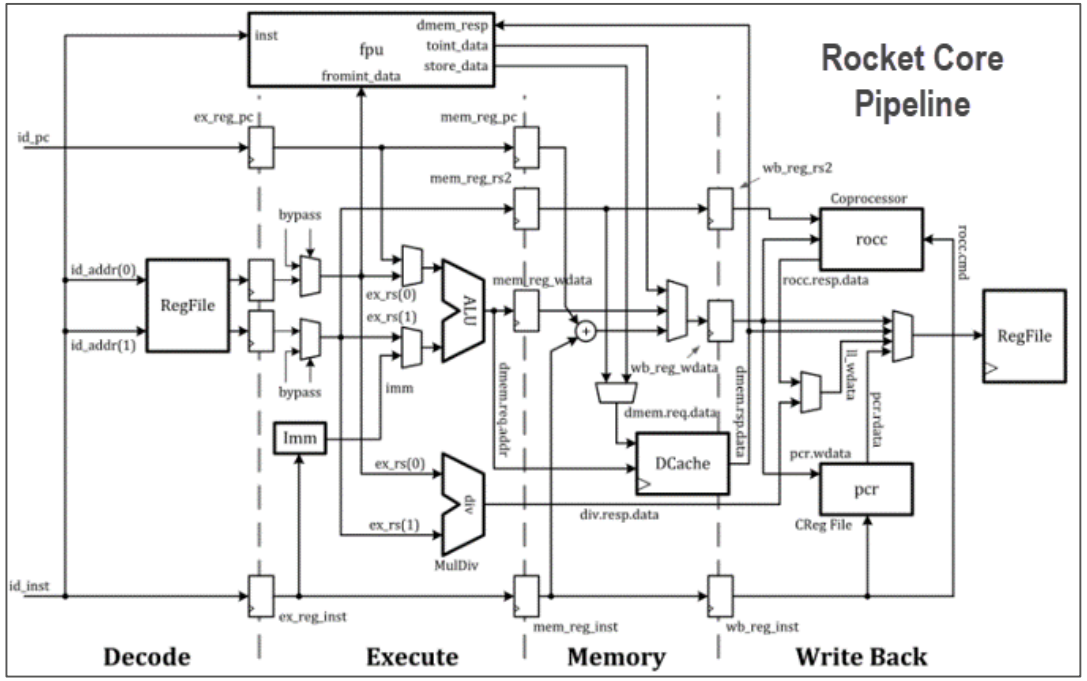


Application Example 64-Bit Rocket Core

Results with OneSpin RISC-V Formal App (Prototype)

Design issues reported*):

- **Issue 1757**: Jump instructions store different return – instruction fetch unit responsible to prevent this issue
- **Issue 1752**: **DIV result not written** in register file – confirmed and fixed in RTL
- **Issue 1861**: Replay of illegal opcodes / generating memory accesses - Illegal opcodes access cache
- **Issue 1868**: **Undocumented non-standard instruction** - opcode 32'h30500073 / CEASE instruction
- **Issue 1949: Undocumented CSR** that reads back 0



Performance

- Less than 5 days to setup verification
- Less than 10 minutes run time per property
- Less than 2 hours total runtime
- Each property proven with unbounded result and reachable witness

*) https://github.com/freechipsproject/rocket-chip/issues/

Thank you!

For questions or more information on OneSpin's RISC-V Integrity Verification Solution, get in touch:

nicolae.tusinschi@onespin.com

© 2019 OneSpin Solutions