2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

UVM-based verification of a RISC-V Processor Core Using a Golden predictor model and a Configuration Layer

Speaker: Ritesh Goel – *Mentor, A Siemens Business*

Authors: Marcela Zachariasova, Lubos Moravec – *Codasip Ltd.*

John Stickley, Hans van der Schoot, Shakeel Jeeawoody – *Mentor, A Siemens Business*

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

CONTENT

- *What is RISC-V*
- *Codasip Automation Flow*
- *RISC-V Verification Strategies*
  - *Configuration Layer*
  - *Golden Predictor Model*
  - *Emulation for Running Tests*
- *What have we achieved*
- *Summary*

What is RISC-V

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

RISC-V Introduction

- Free to use, modern and open instruction set architecture (ISA)
- Originally designed to support research and education, now standard for industry implementations under **RISC-V Foundation**
- No patents

Flexibility OF RISC-V ISA

- 1 RISC-V ISA standard ⇨ many RISC-V HW architecture variants:
  - Base is only integer instruction set ("**I** */***E**")
  - Can be enhanced by standard extensions: integer multiplication and division ("**M**"), atomic instructions for handling real-time concurrency ("**A**"), IEEE floating point ("**F**") with double-precision ("**D**") and quad-precision ("**Q**")
  - Can have compressed instructions ("**C**")
  - Can have different number of the registers (16 or 32) of different sizes (32 or 64 bits).

- We do not verify only 1 processor but many of its variants with enabled/disabled extensions.

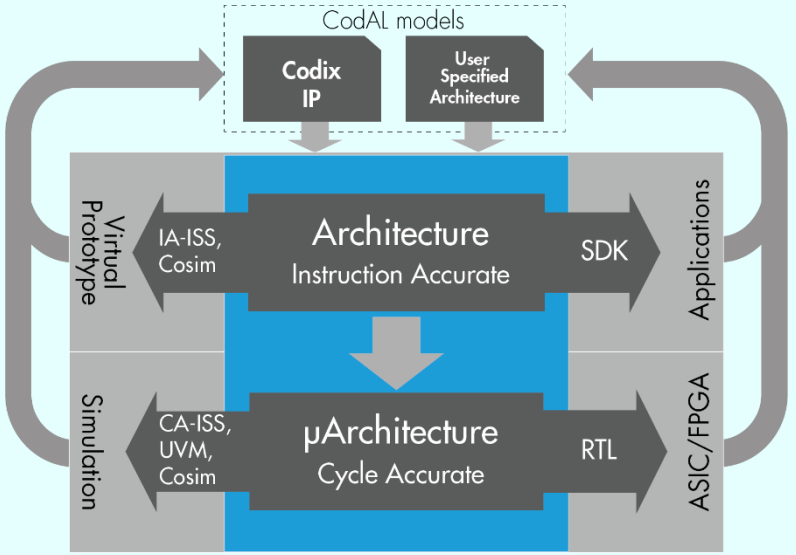
Codasip Automation Flow

CUSTOMIZED RISC-V CORES

Berkelium Processor IP

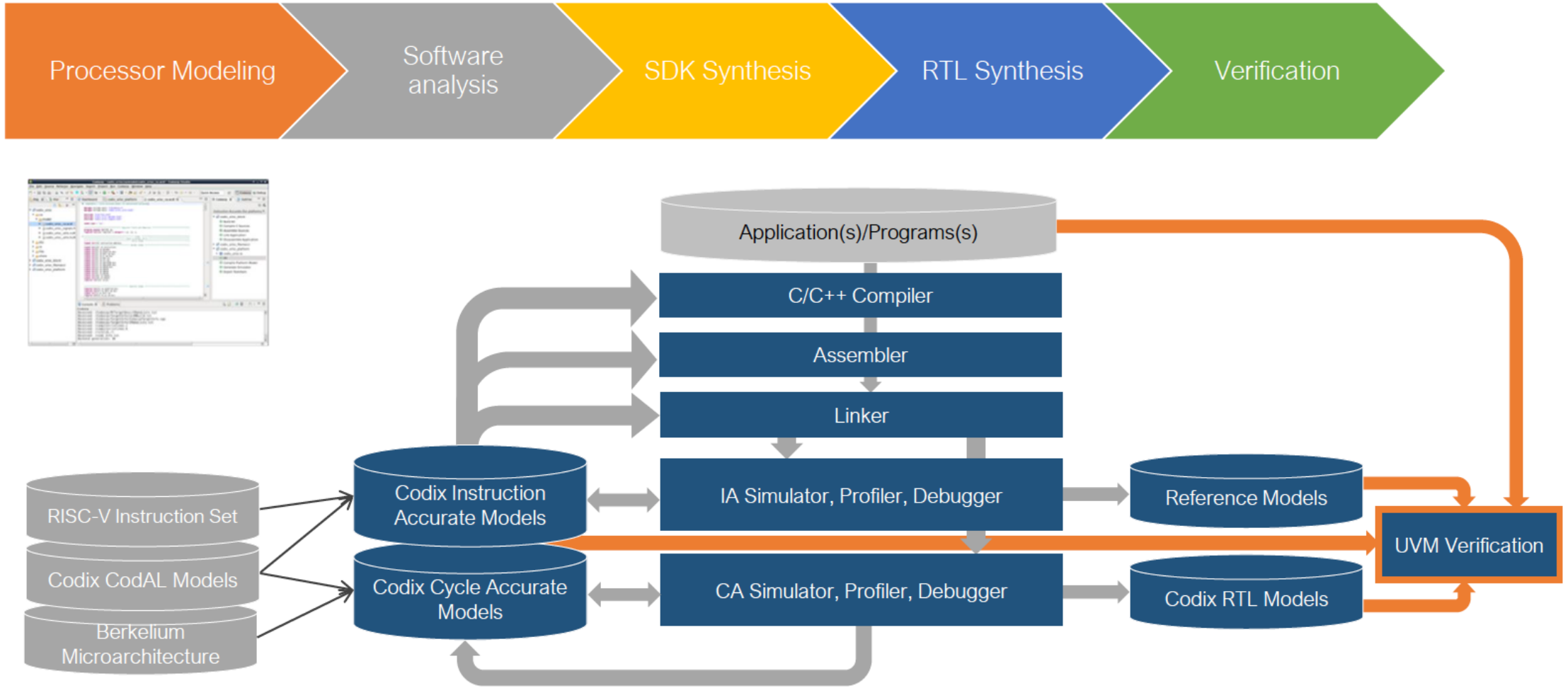
- RISC-V ISA compliant
- Bk-1: no pipeline, 32 bit
- Bk-3: 3-stage, 32 bit
- Bk-5: 5-stage, 32-bit or 64 bit
- Support for standard and **custom extensions**
- Full JTAG and debug support
- Sleep modes and low power support

Codasip Studio processor design and customization environment



- Profile target RISC-V applications
- Explore optimization opportunities, and implement extensions
- Automatically regenerate RTL and software tools
- Comprehensive verification and simulation

UNIQUE AUTOMATION TECHNOLOGY



2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

RISC-V Verification Strategies

Verification Strategies

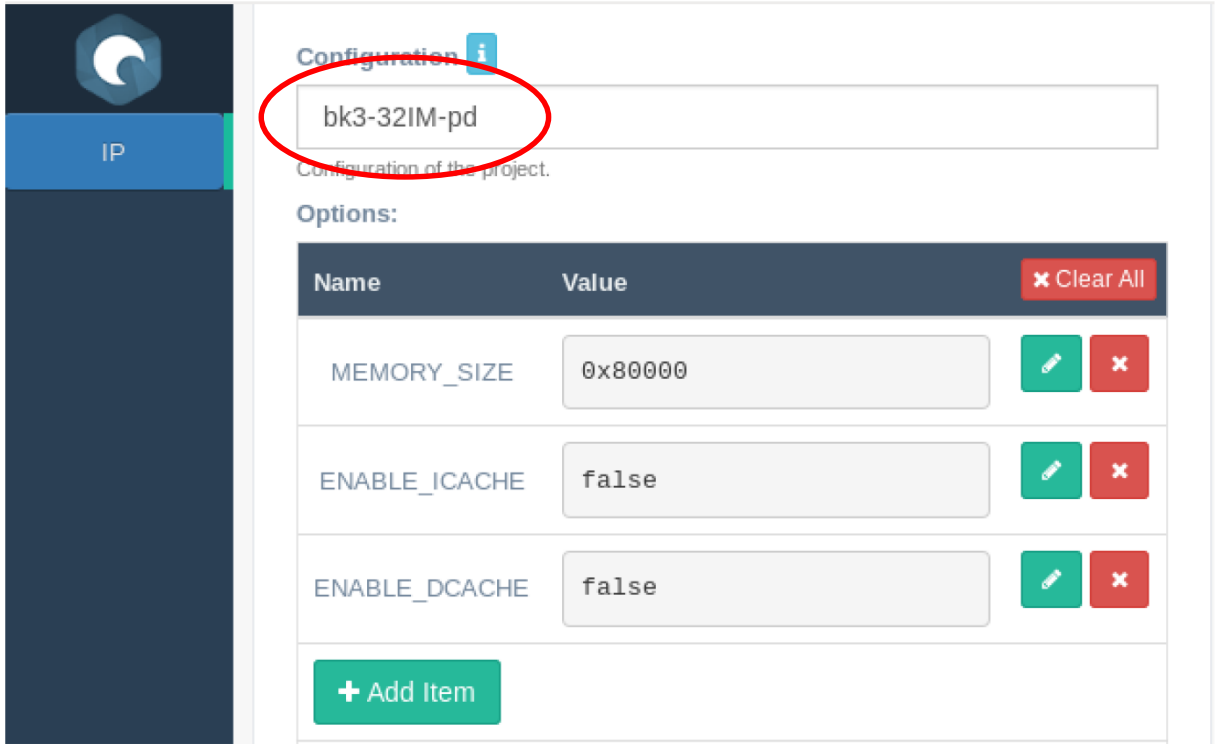
1. Defining configuration layer for RISC-V design and verification
2. Defining a golden predictor model based on ISA simulator
3. Utilizing emulation to effectively perform tests

1. Configuration Layer

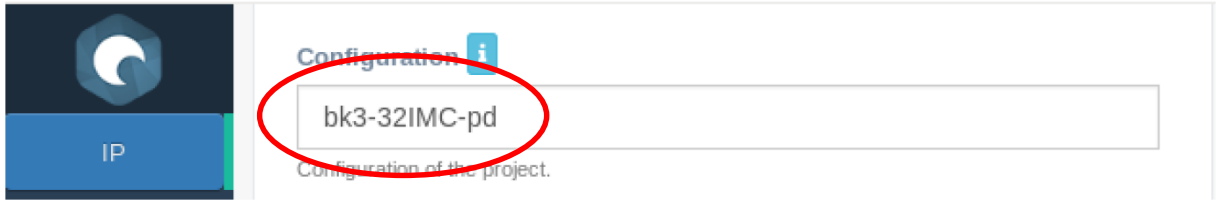
- Many RISC-V core variants = many RTL source codes and UVM test-benches
- Optimization: configurability at a suitable layer
- Options:
  1. Configuration layer ⇨ automation tool ⇨ generated RTL + generated UVM
  2. RTL and UVM configurable via static compile time constructs and scripts
  3. UVM configurable via dynamic runtime configuration approaches

Option 1: Codasip Studio

Bk3-32IM-pd



Bk3-32IM**C**-pd

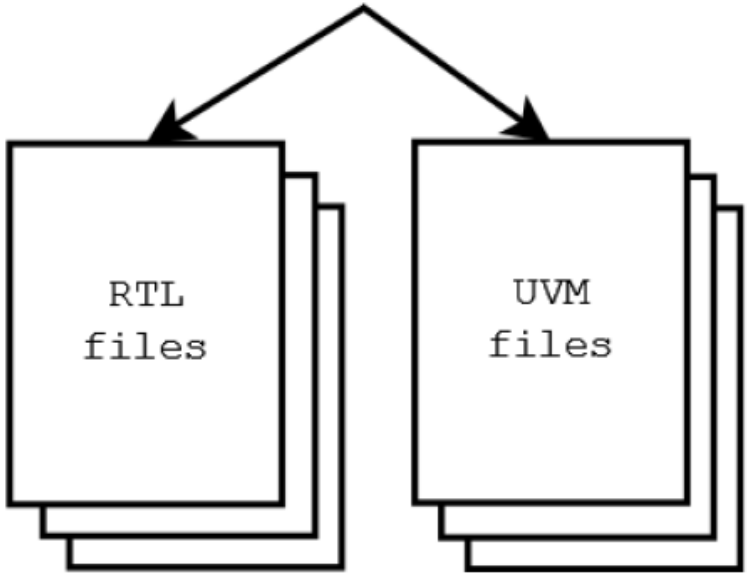


Standard ISA:
I = integer ISA, 32 GPRs
E = integer ISA, 16 GPRs
M = multiplication extension
C = compressed instructions
F = floating-point ISA

Codasip hardware extensions:
p – parallel multiplier
d – JTAG debug interface

Option 2: static compile time Constructs

```
Compile of configurations ´bk3-32IMC-pd´:
+define+EXTENSION_M+EXTENSION_C+...
```




```
/* Decoder description
 * file : codex_berkelium_ca_core_dec_t_decoder.svh
*/
// enumeration code for every decoded instruction
typedef enum {
  add_,
  and_,
  .....
} m_instruction;

// "M" instruction
`ifdef EXTENSION_M
  typedef enum {
      mul_,
      mulh_,
      .....
  } m_instructuon_ext_m;
`endif

`ifdef EXTENSION_M
   bind HDL_DUT_U.codex_berkelium.core.decompressor_16b32b
icodix_berkelium_ca_core_decompressor_16b32b_t_probe probe
(……);
`endif
```

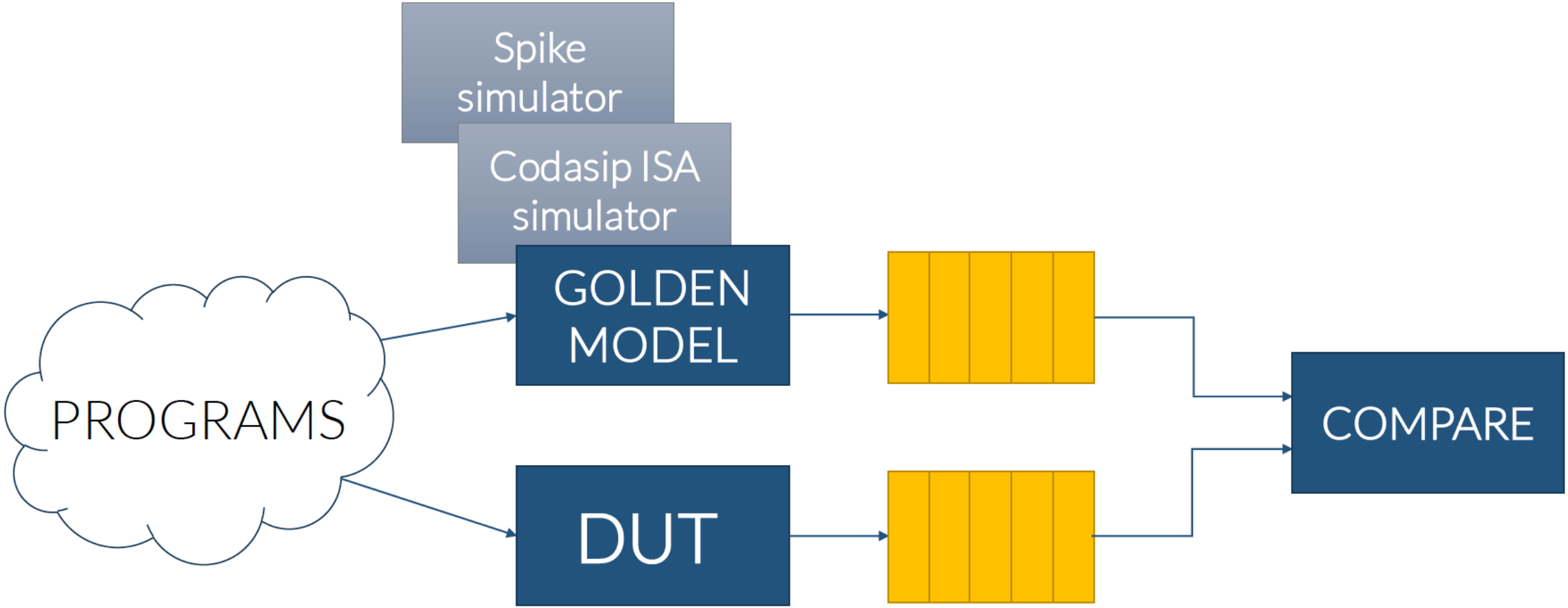
Option 3: Dynamic Runtime Configuration

```
/* Decoder agent configuration
 * file : codex_berkelium_ca_core_dec_t_agent_config.svh
*/
// uvm_config_db configuration and set
function void set_decoder_config_params ();
   // set configuration info
    decoder = new();
    ……..
    decoder.extension_M = 1;
    ……..
     ……
    uvm_config_db #(decoder_config)::set(this, “*”, “decoder_config”, decoder);
endfunction




……….
……..
cvp_instructions : coverpoint m_instruction(….);
if(m_config.extention_M)
begin
     cvp_instructions_ext_m : coverpoint m_transaction_h.m_instruction_ext_m { ……. }
end
```

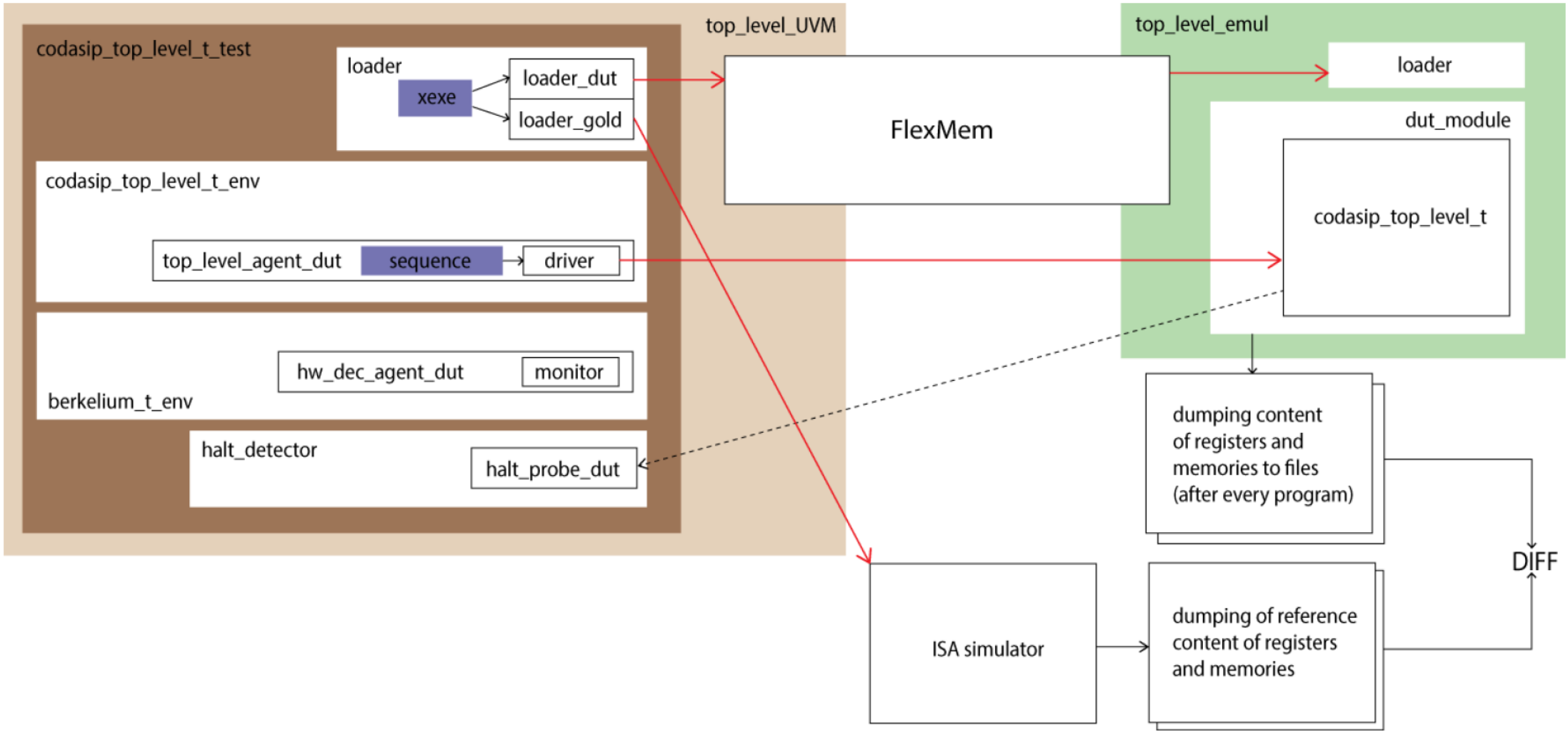
2. ISA simulator as Golden Model



3. Emulation For Fast Verification

- 48 RISC-V variants x 10,000 programs x 500 instructions = unbearable simulation runtime!
- Porting UVM to Veloce emulator:
  1. Comparison of pure simulation and emulation environment runtimes for 1 program.
  2. Creating 2 top-levels – simulation and emulation. For the emulation, the processor is moved into emulator and agents are divided between emulator and simulator for the best efficiency.
  3. Connecting UVM objects while analyzing and minimizing/removing bottlenecks.

Codasip UVM ported to Veloce



2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

What we have achieved

COMPARISON OF RISC-V VERIFICATION CONFIGURATION APPROACHES

ADVANTAGES AND DISADVANTAGES OF THE CONFIGURATION METHODS

	Pros	Cons
Configuration set at higher abstraction level and RTL + UVM are generated	Easy setting of desired configuration. Better readability of generated source files. Very fast.	The generated RTL + UVM are dedicated just to one processor configuration. Price - generator presented in this paper is a paid tool.
Ifdefs for manually written RTL and UVM files	Support of multiple processor configurations.	Worse readability of code in source files.
uvm_config_db for manual-ly written UVM files	Support of multiple processor configurations in UVM source files.	Worse readability of code in source files. Limited only to UVM.

COMPARISON OF RISC-V EMULATION APPROACHES

	Average runtime for 1 program in seconds (~100 000 instructions)	Acceleration achieved
Pure simulation-based verification vs. pure emulation-based verification	**128 (simulation)** **1.28 (emulation)**	**100x**
Emulation-based verification with simple test-bench and pipes	**32**	**4x**
Emulation-based verification with UVM, FlexMem and external ISA simulator	**1.7**	**75x**

Summary

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Summary

- RISC-V flexibility introduces verification challenge in the number of variants that must be verified
- Verification strategy that targets variability:
  - Configurability (we introduced 3 approaches)
  - Utilizing existing ISA simulators as golden predictor model (Codasip, Spike simulator)
  - Employing emulation for running huge amount of tests (we provided recommendations how to proceed by porting UVM to Veloce)

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Q & A