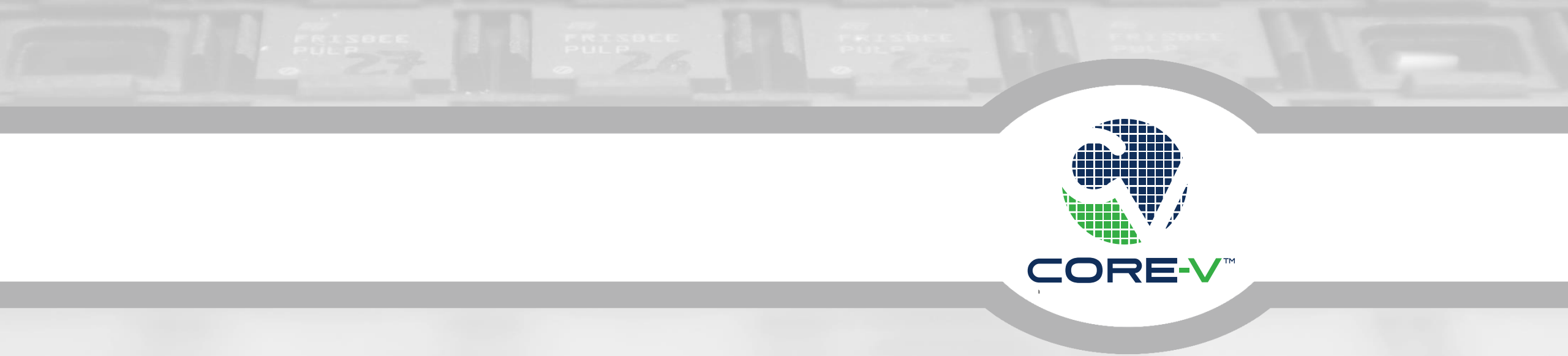


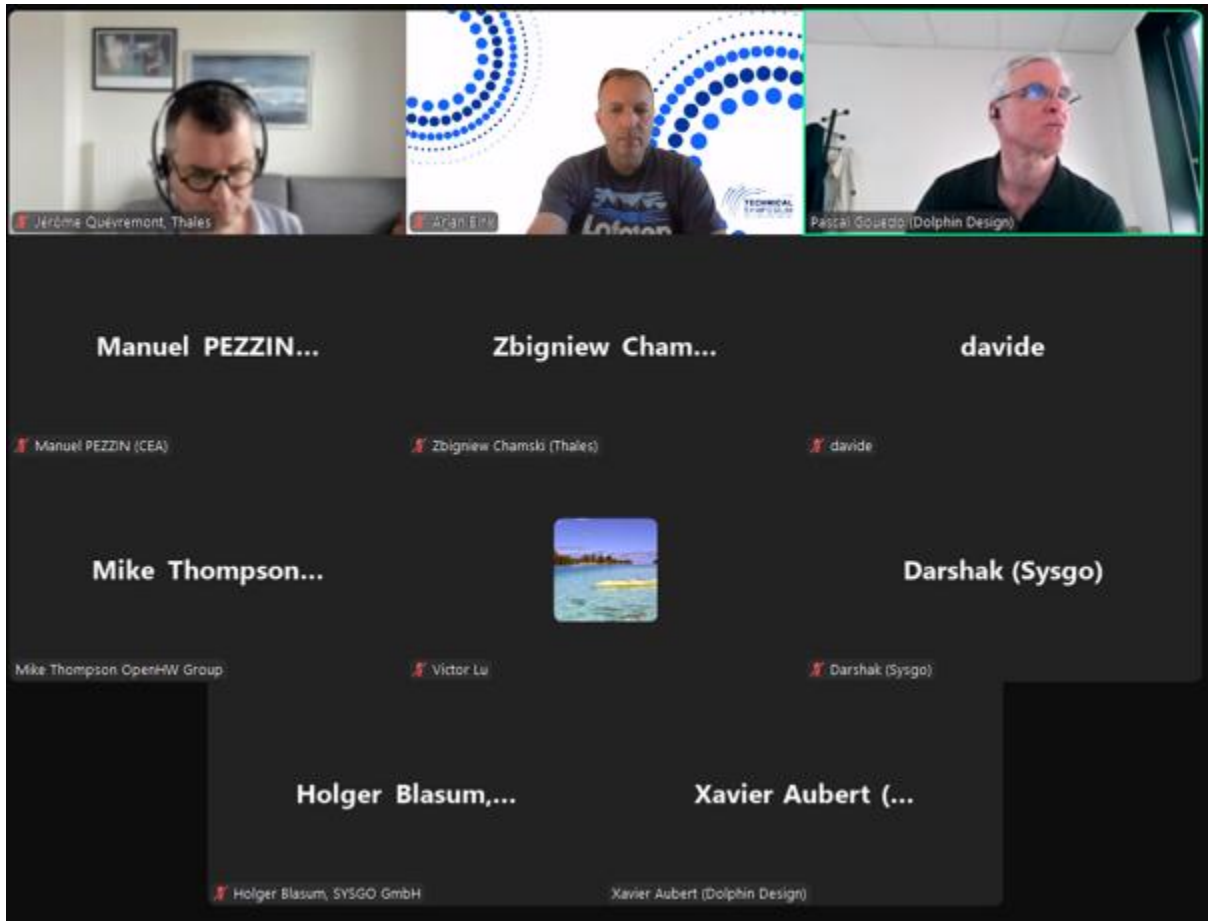
CORES TG – June 3 2024

Arjan Bink

Jérôme Quevremont

Davide Schiavone

Attendance

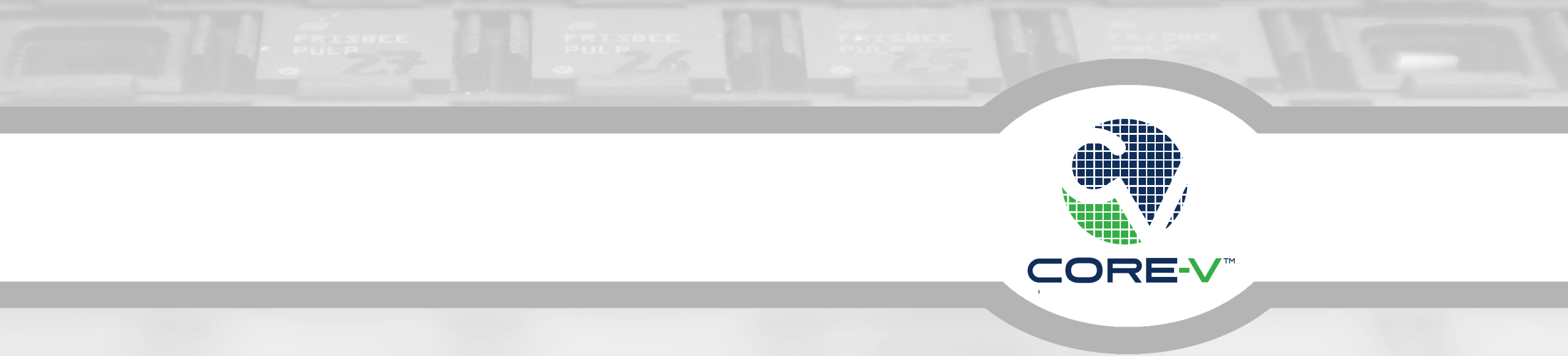

Jérôme Quévremont, Thales
Arjan Bink
Pascal Gouedo (Dolphin Design)
Manuel PEZZIN...
Zbigniew Cham...
davide
Manuel PEZZIN (CEA)
Zbigniew Chamski (Thales)
davide
Mike Thompson...
Darshak (Sysgo)
Mike Thompson OpenHW Group
Victor Lu
Darshak (Sysgo)
Holger Blasum,...
Xavier Aubert (...
Holger Blasum, SYSGO GmbH
Xavier Aubert (Dolphin Design)

Agenda

- CV32E40Pv2 status (Pascal Gouédo)
- CVA6 – Configuration management (Zbigniew Chamski)

Request from Duncan Bees

- “can we start putting individual project reports/slide updates into their own subdirectory?”
  - programs/TGs/cores-task-group/cve20
  - programs/TGs/cores-task-group/cva6
  - programs/TGs/cores-task-group/cv32e40p
  - Etc.



CV32E40Pv2 status

Pascal Gouédo

Yoann Pruvost

Bee Nee Lim

May 6, 2024

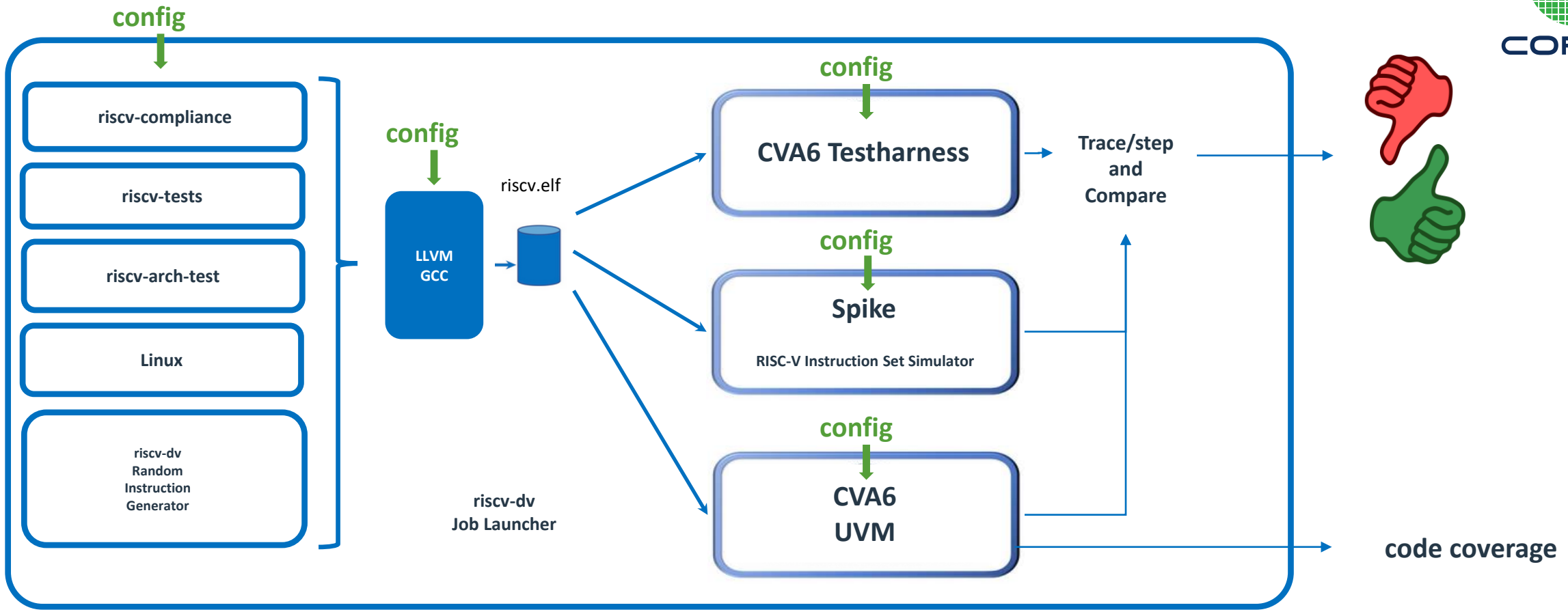
THALES
Building a future we can all trust

Configuring CVA6: from SV to Yaml

Zbigniew CHAMSKI
Jerôme LE NOIR
Gilles MALFREYT
Abdessamii OUKALRAZQOU

www.thalesgroup.com
2024-06-03

Context



- 32-bit: CV32A6*X, CV32A6_embedded, CV32A6_im[a][f][c]_{sv0,sv32}, CV32A6_ima_sv32_fpga
- 64-bit: CV64A6_imafdc[h][v]_sv39_{hpdcache,openpiton,polara,wb}

Challenges

- Variety of configurations (16+ in total): XLEN, supported privilege levels, extensions
- Maintainability/readability of configurations
  - Hidden vs. explicit information (hardcoded choices, implicit values)
- Consistency of each configuration
  - Presence and semantics of CSRs qnd CSR fields
- “Single” entry point for processable configuration info
  - Three components: ISA, CSRs, platform properties
- Aim: Common source for configuration files
  - RTL
  - Spike
  - CSR design documentation and tests
  - Test suites
- Out of scope (for now): Annotated ISA documentation

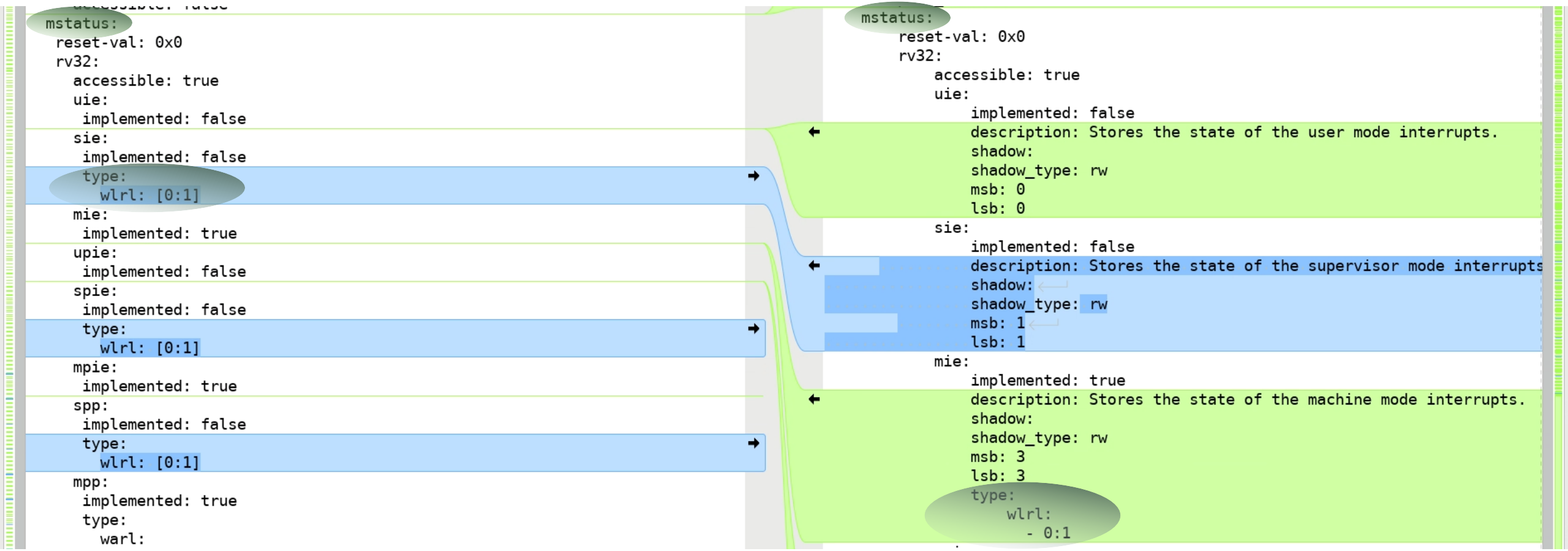
Leverage community work: riscv-config

- Endorsed by RISC-V Int'l. (https://github.com/riscv-software-src/riscv-config)
  - ***Stated purposes:***
    - *describe implementation configurations*
    - *select and configure tests of RISCOF*
  - ***Input (yaml) :*** *ISA + list of matching std CSRs, custom CSR descriptions, limited platform information*
  - ***Actions:***
    - *Check **consistency** of ISA and CSR information and **compatibility** with RISC-V ISA specs (unpriv and priv)*
    - *Fill **implied** information about CSRs*
  - ***Output (yaml)****: ISA + CSR descriptions extended with implied info and guaranteed to match RISC-V ISA spec*
- Avantages and shortcomings (at first glance)
  - (+) Outputs a fairly complete description of CSRs, including properties implied by the RISC-V ISA specs
  - (+) Endorsed by RISC-V Int'l.
  - (-) Missing information at platform/system level: memory layout and properties, peripherals (presence, mem mapping)…
  - (-) Documentation issues (ReadTheDocs out of date, doc not easy to regenerate locally)

Example 1: Added CSR info

```
hart_ids: [0]                                          hart_ids: [0]
hart0: &hart0                                          hart0:
  ISA: RV32IMCZicsr_Zicntr_Zifencei                      ISA: RV32IMCZicsr_Zicntr_Zifencei
  User_Spec_Version: '2.3'                               User_Spec_Version: '2.3'
  supported_xlen: [32]                                   supported_xlen:
                                                           - 32
  physical_addr_sz: 32                                   physical_addr_sz: 32
  pmp_granularity: 5                                     pmp_granularity: 5
  misa:                                                  misa:
   reset-val: 0x40001104                                     reset-val: 0x40001104
   rv32:                                                     rv32:
     accessible: true                                          accessible: true
     mxl:                                                      mxl:
       implemented: true                                         implemented: true
       type:                                                     type:
           warl:                                                     warl:
              dependency_fields: []                                     dependency_fields: []
              legal:                                                    legal:
                - mxl[1:0] in [0x1]                                       - mxl[1:0] in [0x1]
              wr_illegal:                                               wr_illegal:
                - unchanged                                               - unchanged
     extensions:                                                               description: Encodes the native base integer ISA width.
       implemented: true                                                       shadow:
       type:                                                                   shadow_type: rw
           warl:                                                               msb: 31
              dependency_fields: []                                            lsb: 30
              legal:                                                  extensions:
                - extensions[25:0] in [0x0000000:0x3FFFFFF]               implemented: true
              wr_illegal:
```

Example 2: Overspecification (SIE) vs. implicit property (MIE)



File layout (after CVA6 PR #2160 merge)

- vendor/riscv/riscv-config: Vendorized riscv-config tree (v3.18.1)

> Project-side additions

```
config/riscv-config/
├── Makefile
├── README.md
└── cv32a65x
    ├── generated
    │   ├── custom_gen.yaml
    │   ├── isa_gen.yaml
    │   └── platform_gen.yaml
    └── spec
        ├── custom_spec.yaml
        ├── isa_spec.yaml
        └── platform_spec.yaml
```

Generated by `riscv-config`

HAND-WRITTEN (based on examples)

```
config/gen_from_riscv_config/
├── README.md
├── cv32a65x
│   ├── csr
│   │   ├── csr.md
│   │   └── csr.rst
│   ├── isa
│   │   ├── isa.md
│   │   └── isa.rst
│   └── spike
│       └── spike.yaml
├── requirements.txt
├── scripts
│   ├── libs
│   │   ├── csr_factorizer.py
│   │   ├── csr_updater.py
│   │   ├── isa_updater.py
│   │   └── utils.py
│   └── riscv_config_gen.py
├── templates
│   └── isa_template.yaml
└── updaters
    └── cv32a65x
        ├── csr_updater.yaml
        └── isa_updater.yaml
```

Tasks ahead

- riscv-config
  - *Review and fix* the input spec files, use CV32A65X as carrier example
  - Extend *platform information*
    - Memory layout and properties
    - Platform components: CLIC/PLIC, mappings of peripherals… MAYBE correlate with/extract from Device Tree information??
  - Add *user-specified constraints*
    - Example: No D extension if XLEN == 32
    - Hooks available in **riscv-config**
- gen_from_riscv_config
  - Improvements to *ISA and CSR documentation* (may require enhancements to **riscv-config**)
  - Generation of *Spike Yaml parameter file*
  - Config-driven *test generation and control* (e.g., 'no **sbreak** tests if **S** mode not present')
  - *RTL parameters*…

THALES
Building a future we can all trust

Thank you

www.thalesgroup.com

Thank you!