2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

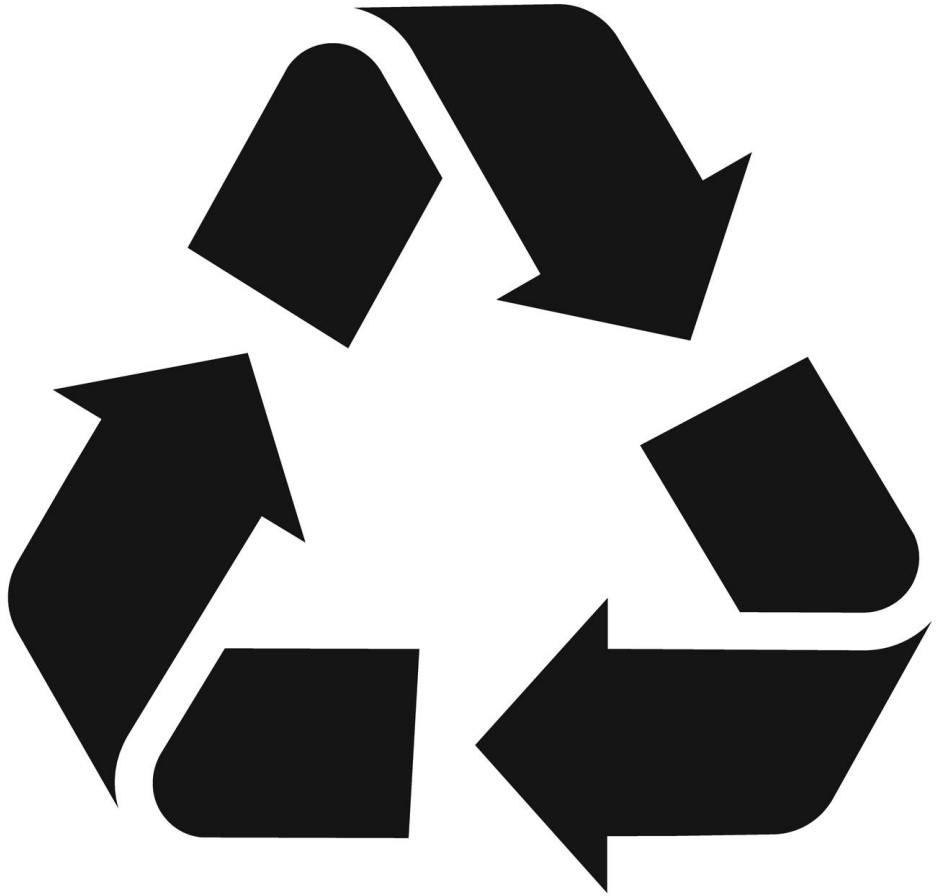
SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

Automate the Pain Away: HW/SW Interface Design Methodology

Alvin Santos, Insaf Meliane, Tim Schneider

ARTERIS IP

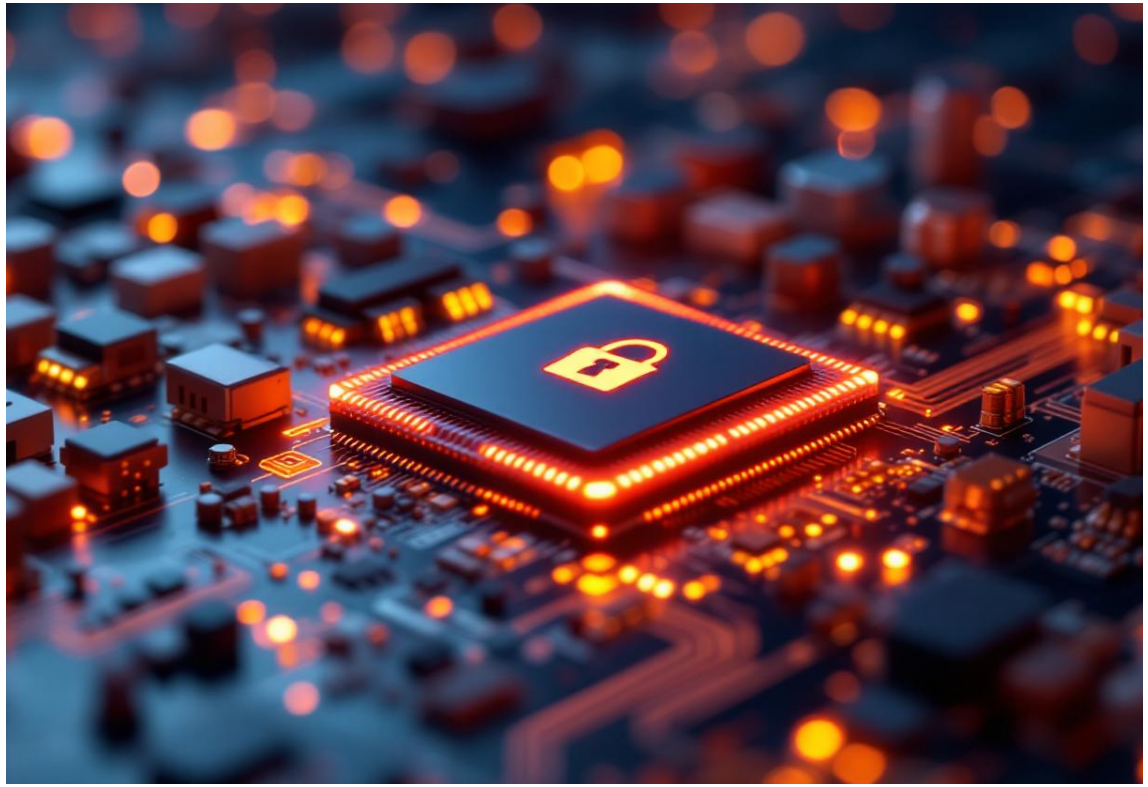
Complexity of Modern SoCs



Reuse and 3rd party IP are a significant portion of designs

Writing and maintaining home-grown solutions

Manual effort to keep up with changing standards, enhancements & bug fixes



Increased SoC complexity is expanding the hardware attack surface

Case Study: Introduction

- PULPino RISC-V SoC as demonstration vehicle
- Magillem platform for integration and verification automation

SoC Integration Automation with the Magillem Platform

Magillem Packaging

Magillem Connectivity

IP-XACT
1685-2009
1685-2014
1685-2022

Protocols | IP-XACT | RTL | IP

Import and Packaging

Port, parameter, bus interface, source files (fileset) capture, editing, visualization and checks

SoC Assembly

(Instantiate, Configure, Connect, Hierarchy, Restructuring, Partition, Incremental design)

IP IP

API libraries
(TGI, RTL, etc.)

Elaborated Data Model

CLI
(Python, Tcl)

Flow Automation

Checkers (Std + Customs) + Reports

Generators (Std + Customs)

UPF | AMD | XILINX | SYNOPSYS | cadence | SIEMENS


HDL top level netlist

Magillem Registers

IP-XACT | CSRSpec | SystemRDL

Import and Packaging

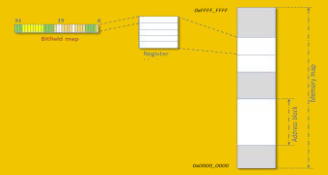
Memory map, bridge capture, editing, visualization and checks

IP IP

IP-XACT

HW/SW Interface

(Address map elaboration, System Map calculation)



Checkers + Reports

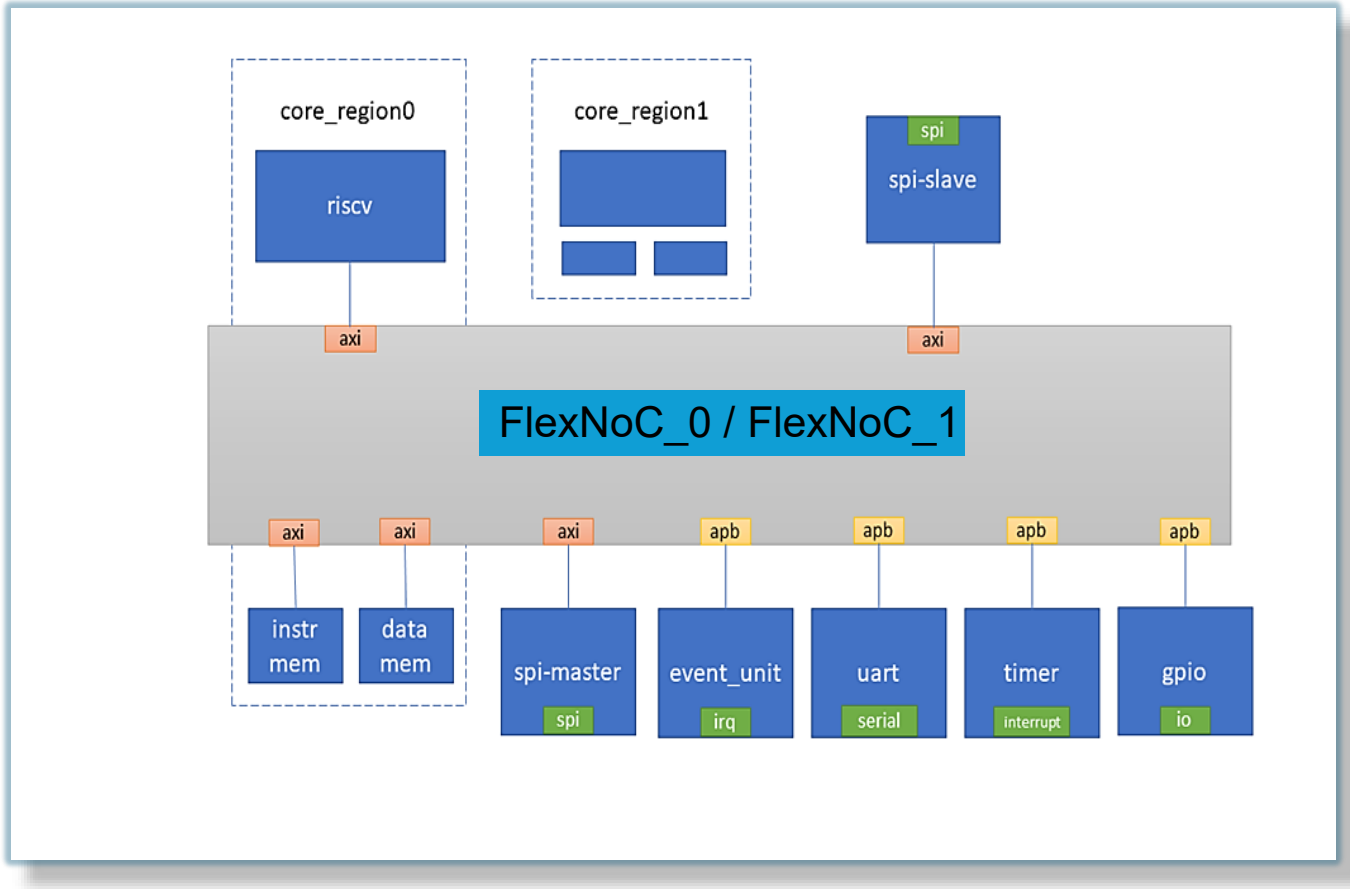
Generators (+spec cross-compile)

RTL | UVM | hal | CSRSpec | SystemRDL | IP-XACT

- Easy Configuration
- Continuous Integration
- Automated Generation
- Faster Cycle Time to Completion
- Better Quality
- IP Reuse
- Generate Derivative Designs

PULPino SoC Integration

- Pulpino (RiscV) Example platform
- AXI4: Risc Core, Inst/data mem, SPI
- APB: Peripherals
- Dual FlexNoC IP - Arteris SIP group

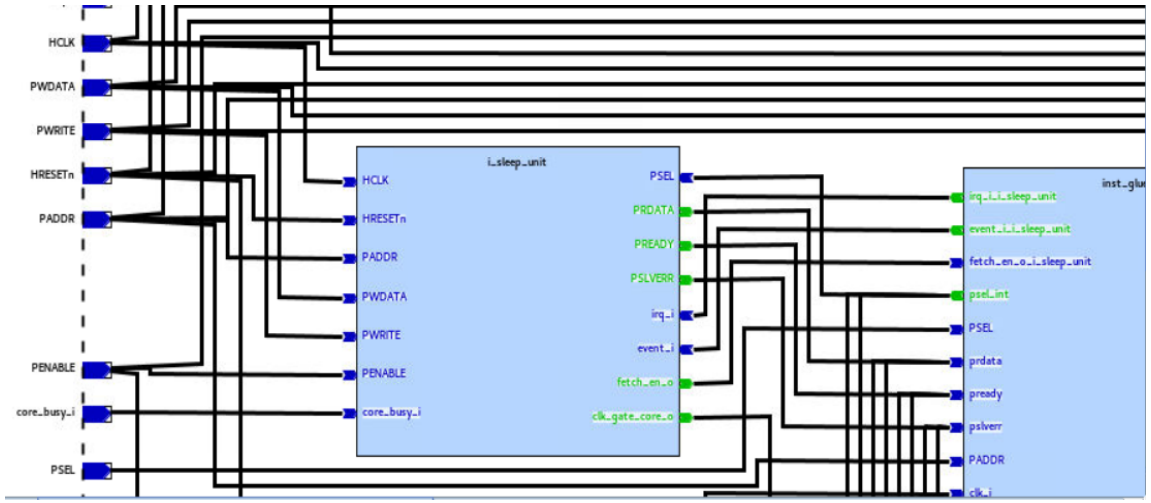


What are we going to do?

- Import & Package IPs
- Load AMBA bus definitions
- Import RTL IP
- Map Physical IP ports to AMBA logical ports
- Merge Memory Map into Component IP
- Build Top-Level Design
- Instantiate and Configure Components
- Connect bus Interfaces and AdHocs (wires)
- Validation
- Run Checkers and Generate Connectivity Report
- Check System Map / Generate System map report
- Hierarchy Manipulation / Repartitioning
- Outputs Generation
- RTL Netlists
- UVM RAL
- HTML
- C Header

Import RTL and Generate IP-XACT

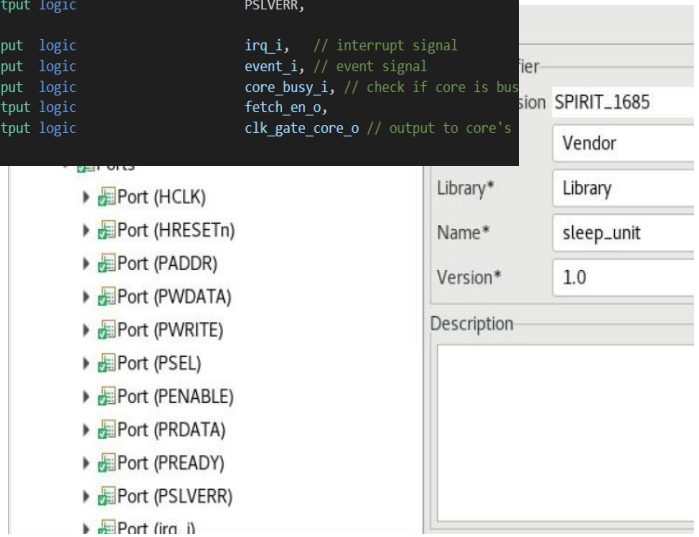
```
##########################################
# Import RTL and create IP-XACT files
##########################################
Import.hdlHierImport(
    '-fileList', ['apb_event_unit.sv', 'generic_service_unit.sv', 'sleep_unit.sv'],
    '-includeLocation', ['include'],
    '-language', 'systemverilog',
    '-xmlLocation', ipxact_dir
)
##########################################################
```



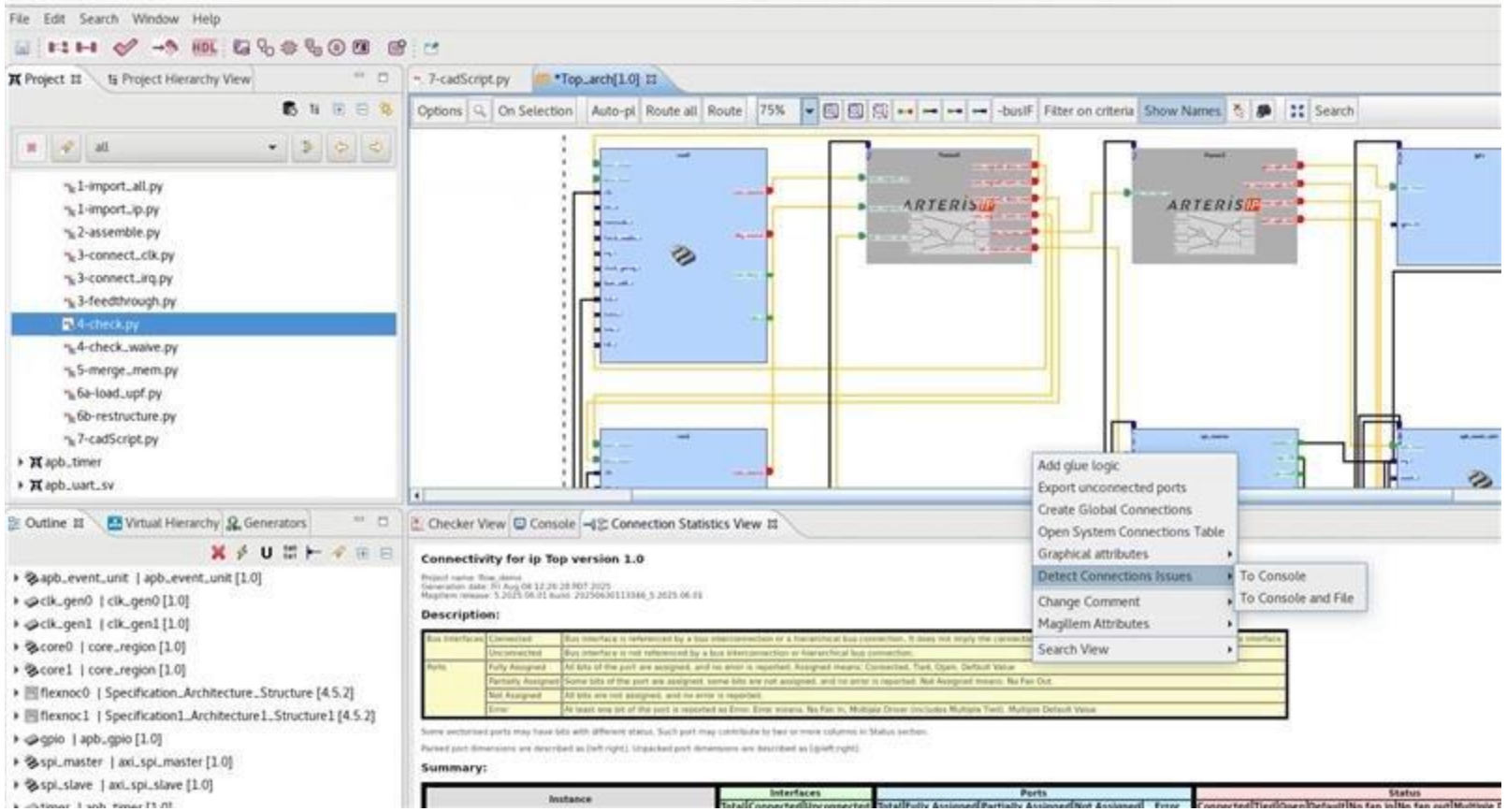
1-import_ip.py 9+ sleep_unit.sv

home > cwang > Projects > FlowDemo > pulpino-master > ips > apb > apb_event_unit > sleep_unit.sv

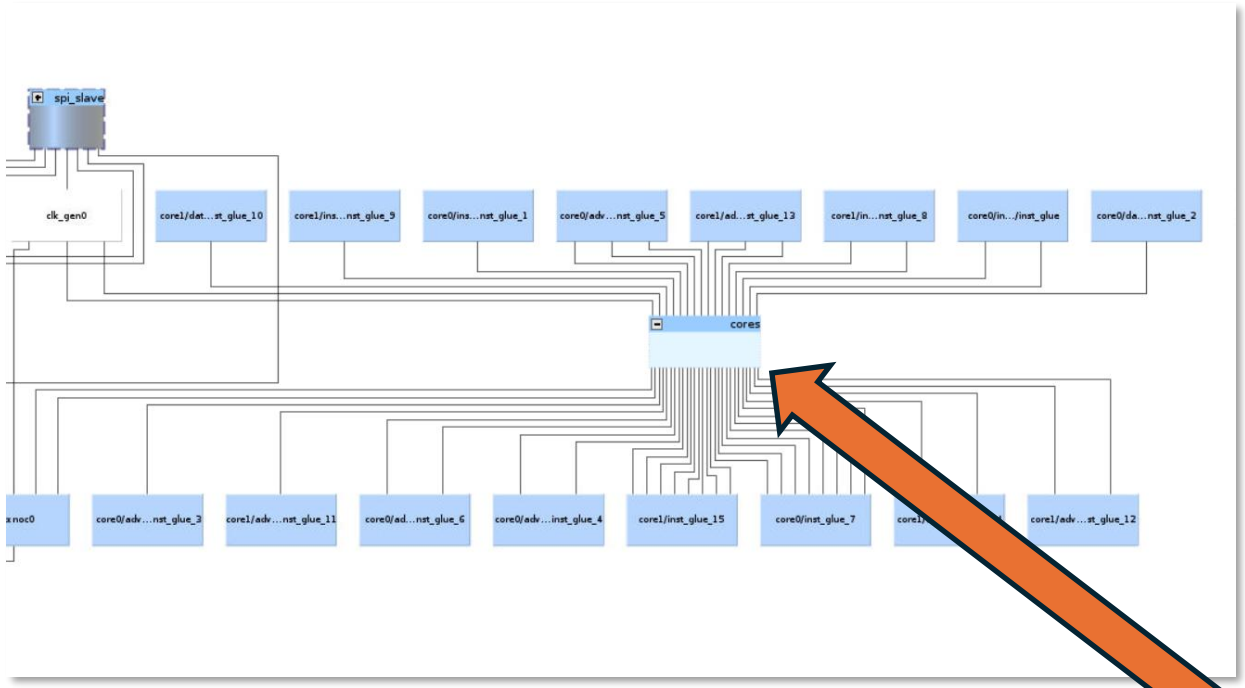
```
12
13  module sleep_unit
14  #(
15      parameter APB_ADDR_WIDTH = 12  //APB slaves are 4KB by default
16  )
17  (
18      input  logic                     HCLK,
19      input  logic                     HRESETn,
20      input  logic [APB_ADDR_WIDTH-1:0] PADDR,
21      input  logic              [31:0] PWDATA,
22      input  logic                     PWRITE,
23      input  logic                     PSEL,
24      input  logic                     PENABLE,
25      output logic              [31:0] PRDATA,
26      output logic                     PREADY,
27      output logic                     PSLVERR,
28
29      input  logic                     irq_i,   // interrupt signal
30      input  logic                     event_i, // event signal
31      input  logic                     core_busy_i, // check if core is bus
32      output logic                     fetch_en_o,
33      output logic                     clk_gate_core_o // output to core's
34  );
```



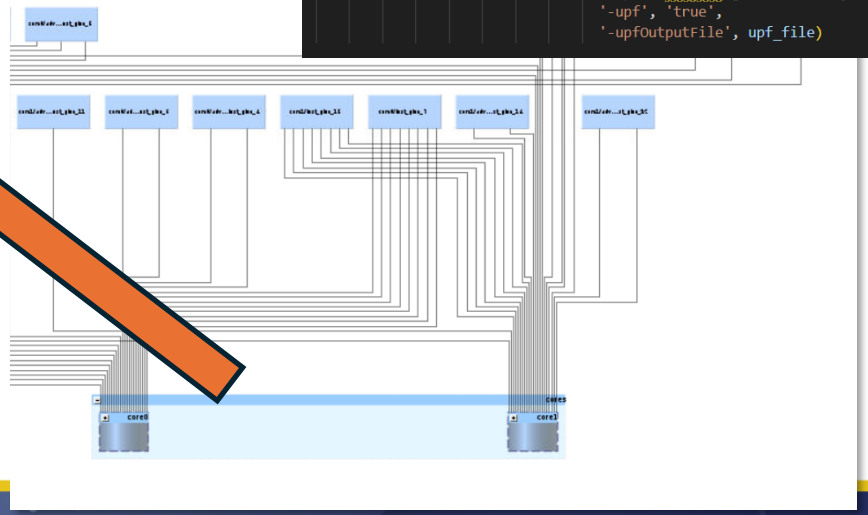
Automap Bus Interfaces and Connect Components



Restructured RTL

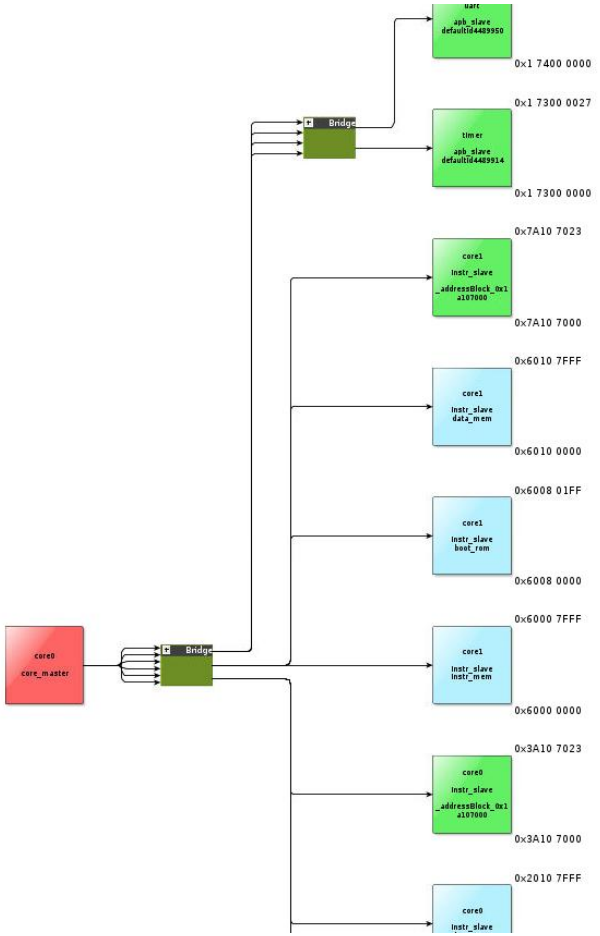



```
project.setCurrentProject('flow_demo')
setPreference('vh_upf_mode', 'true')
top = ['Vendor', 'Library', 'Top', '1.0']

proj_dir = os.path.join(project.getWorkSpaceLocation(), project.getCurrentProject())
vh_file = os.path.join(proj_dir, '_'.join(top) + '.hierarchy')
root_dir = os.path.expandvars('$ARTERIS_IPD_FLOW_DEMO')
upf_file = root_dir + "/output.upf"

try:
    vh.deleteVirtualHierarchy('-project', project.getCurrentProject(),
                              '-vlnv', ['Vendor', 'Library', 'top_vh', '1.0'], '-physical')
except:
    pass

vh.initSession('-vlnv', ['Vendor', 'Library', 'top_vh', '1.0'],
               '-project', project.getCurrentProject(),
               '-component', top)
vh.save(vh_file)

vh.add('-name', 'cores', '-path', '/', '-vlnv', ['Vendor', 'Library', 'core_subsystem', '1.0'])
vh.move(['/core0', '/core1'], '/cores')

vh.add('-name', 'peripherals', '-path', '/', '-vlnv', ['Vendor', 'Library', 'periph_subsystem', '1.0'])
vh.move(['/uart', '/timer', '/gpio', '/apb_event_unit', '/spi_master'], '/peripherals')

vh.changePowerDomain('-powerDomain', 'pd_peripherals', '-instance', '/peripherals')
vh.changePowerDomain('-powerDomain', 'pd_cores', '-instance', '/cores')

vh.closeSession('-save')

vh.generatePhysicalHierarchyFiles('-vh', vh_file,
                                  '-directory', proj_dir,
                                  '-np', 'true',
                                  '-npn', project.getCurrentProject() + '_pd',
                                  '-upf', 'true',
                                  '-upfOutputFile', upf_file)
```

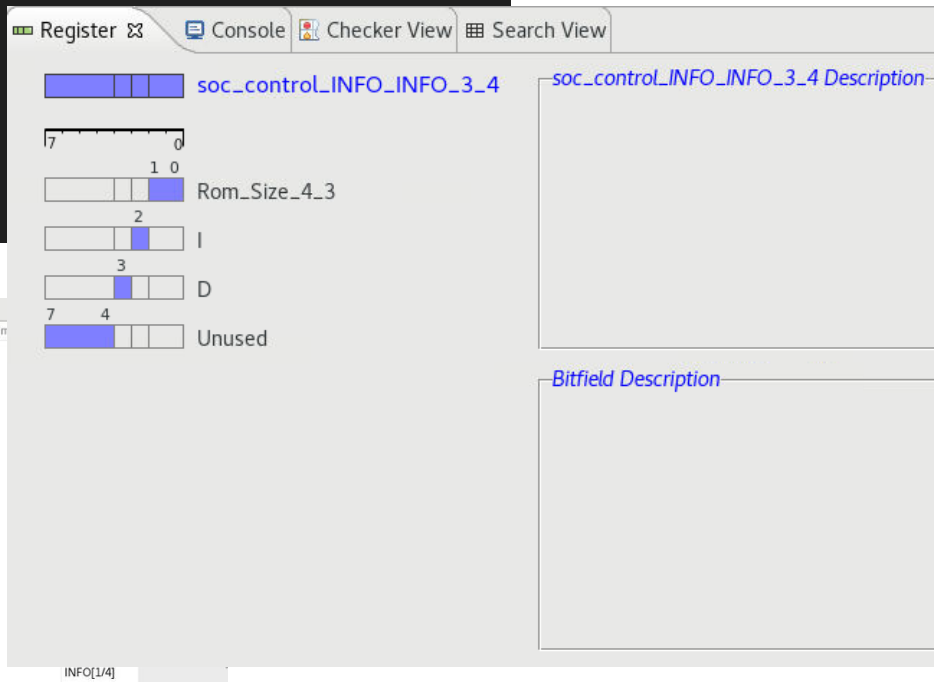


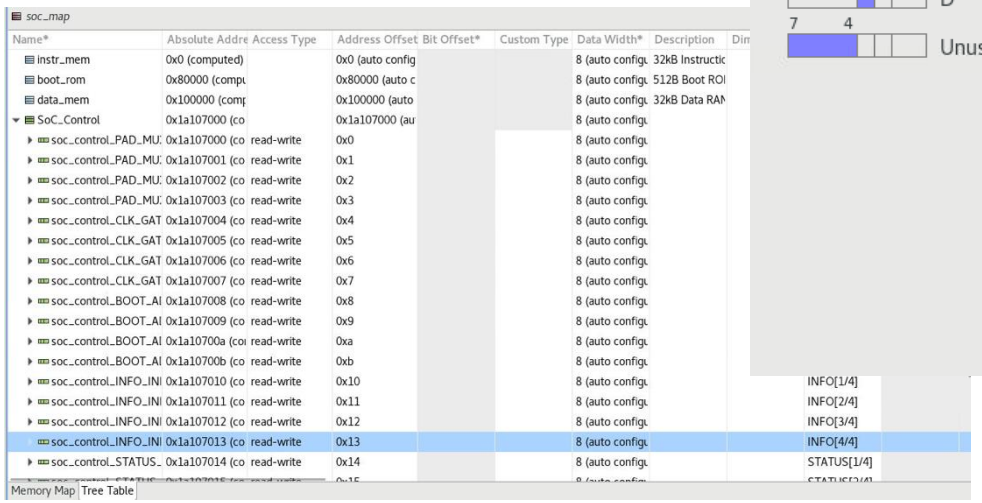
Move, Merge, or Flatten physical and virtual hierarchy: Fast & safe response to meet power & floor-planning constraints

Generate a Memory Map and Register Definitions

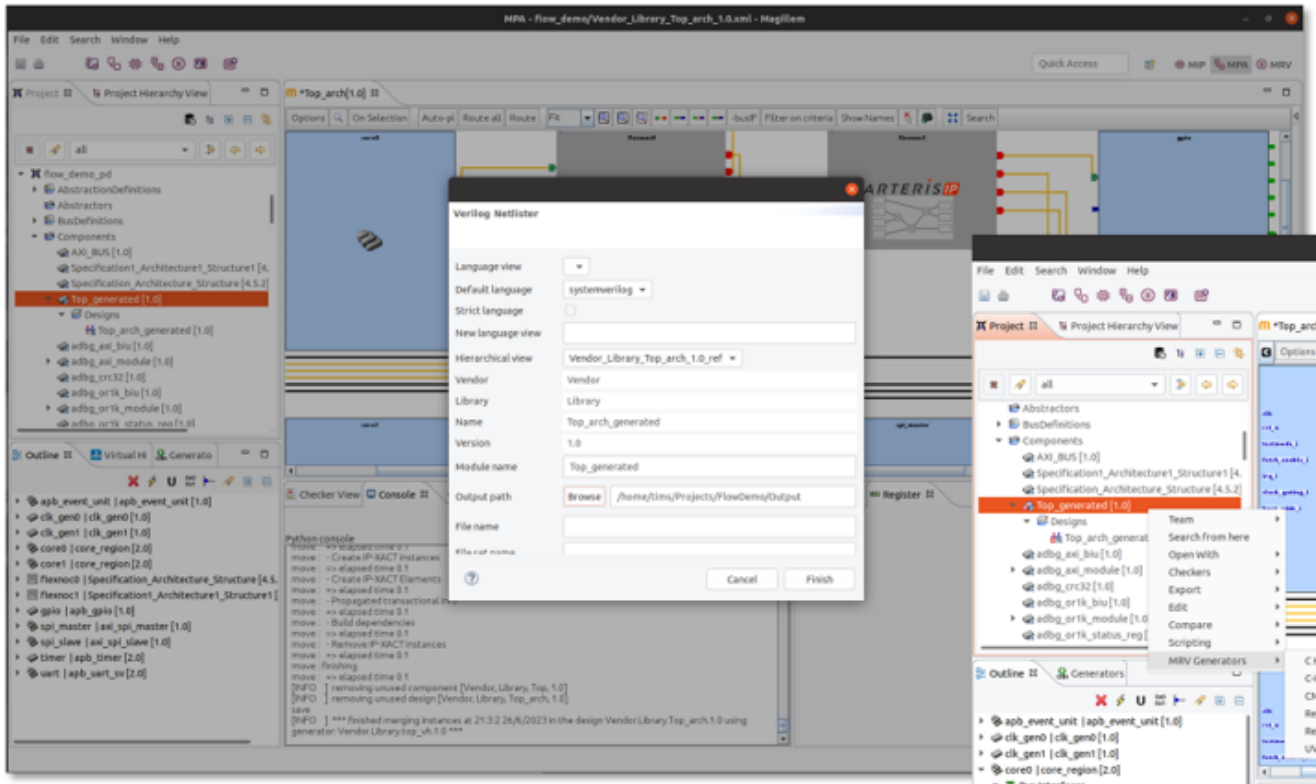



```
// INFO at offset 0x10 (Absolute: 0x1A10_7010)
// Note: Explicit offset needed due to address gap (0x700C is unused)
register {
  property register_reset_value = 0x0000_8082;
  field {  } [31:28] Unused;
  field {  } [27] D;
  field {  } [26] I;
  field {  } [25:21] Rom_Size;
  field {  } [20:13] Inst_Ram_Size;
  field {  } [12:5] Data_Ram_Size;
  field {  } [4:0] Version;
} [0x10] INFO;
```

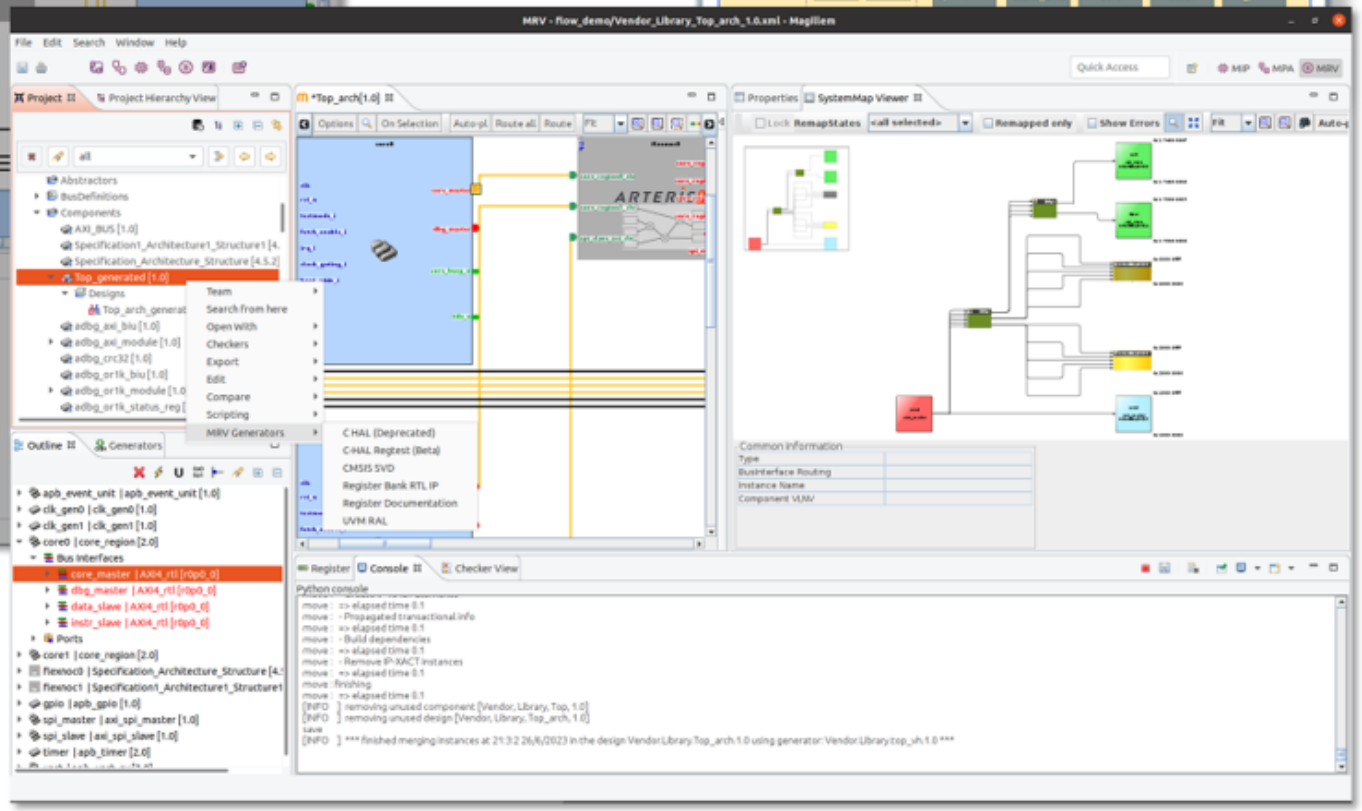




Output Generation

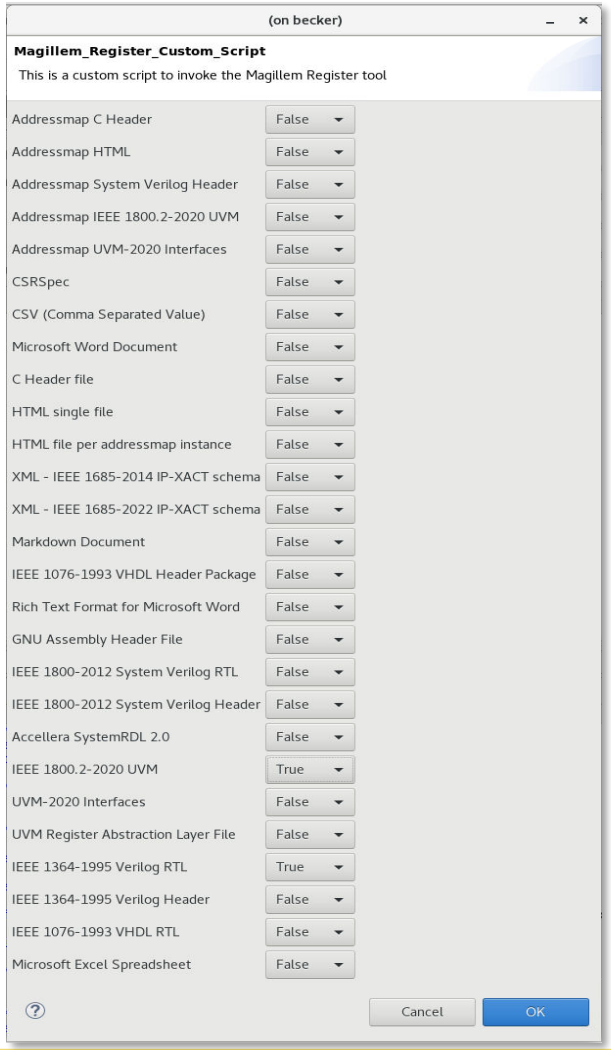


HDL Netlist Generation



Register Bank RTL, UVM RAL and Documentation

Output Generation



- IPXACT 2014/2022
- UVM RAL
- SystemVerilog RTL
- HTML
- Excel Spreadsheet

Case Study: UVM-Based Verification

- Magillem Registers exports UVM RAL based on SystemRDL/CSRSpec/IP-XACT
- UVM testbench performs access tests, reset checks, mirror/predict
- Ensures alignment between RTL and software-visible registers

Case Study: UVM Register Model

```
15  //
16  // Generated on: Wed Aug  6 11:45:10 2025
17  //           by: cwang
18  //
19
20  `ifndef \CSR_VENDOR_LIBRARY_CORE_REGION_MM_2.0
21  `define \CSR_VENDOR_LIBRARY_CORE_REGION_MM_2.0
22
23
24  package \csr_pkg_Vendor_Library_core_region_mm_2.0 ;
25  import uvm_pkg::*;
26  `include "uvm_macros.svh"
27
28  // Memory: core_region_mm::instr_slave::instr_mem
29  // Source filename: /home/cwang/Projects/FlowDemo/FlowDemo_new/workspace/flow_demo/Vendor_Library_core_region_mm_2.0.xml, line: 1457
30  class csr_mem_core_region_mm_instr_slave_instr_mem extends uvm_mem;
31
32    function new (string name = "csr_mem_core_region_mm_instr_slave_instr_mem");
33      super.new(name, 64'h8000, 8, "RW", UVM_NO_COVERAGE);
34    endfunction: new
35
36    `uvm_object_utils(csr_mem_core_region_mm_instr_slave_instr_mem)
37
38  endclass : csr_mem_core_region_mm_instr_slave_instr_mem
39
40  // Memory: core_region_mm::instr_slave::boot_rom
41  // Source filename: /home/cwang/Projects/FlowDemo/FlowDemo_new/workspace/flow_demo/Vendor_Library_core_region_mm_2.0.xml, line: 1514
42  class csr_mem_core_region_mm_instr_slave_boot_rom extends uvm_mem;
```

Case Study: HTML Documentation

Addressmap Information for 'Vendor_Library_core_region_mm_2.0'

Input File Information ☐ Header File Information ☐ Enum Information ☐

component core_region_mm expand all collapse all

Identifier	core_region_mm
Attributes	clk=""

addressmap instr_slave address map

Identifier	instr_slave
Access	R/W

memory instr_mem

Identifier	instr_mem
Description	32kB Instruction RAM
Offset	0x0
Word Count	0x8000
Access	R/W
Attributes	_address_="0x0" _aub_="8" _blockName_="instr_mem" _busInterfaceName_="instr_slave" _componentLibrary_="Library" _componentName_="core_region" _componentVendor_="Vendor" _componentVersion_="2.0" _memoryMapName_="soc_map" _nbInstance_="1"

Attributes	_address_="0x1a107000" _aub_="8" _blockName_="_addressBlock_0x1a107000" _busInterfaceName_="instr_slave" _componentLibrary_="Library" _componentName_="core_region" _componentVendor_="Vendor" _componentVersion_="2.0" _memoryMapName_="soc_map" _nbInstance_="1" _reserved_="false"

register soc_control_PAD_MUX_PAD_MUX_0_4

Identifier	soc_control_PAD_MUX_PAD_MUX_0_4
Title	PAD_MUX[1/4]
Offset	0x0
Access	R/W
Reset Value	0x00
Reset Mask	0xFF
Attributes	_address_="0x1a107000" _blockName_="_addressBlock_0x1a107000" _busInterfaceName_="instr_slave" _componentLibrary_="Library" _componentName_="core_region" _componentVendor_="Vendor" _componentVersion_="2.0" _customType_="RW" _isTestable_="true" _memoryMapName_="soc_map" _nbInstance_="1" _readType_="read" _registerName_="soc_control_PAD_MUX_PAD_MUX_0_4" _reserved_="false" _writeType_="write"

Identifier	Title	Bit	Access	Reset	Attributes	Description
PADMUX_7_0	PADMUX[7:0]	[7:0]	R/W	0x00	_blockName_="_addressBlock_0x1a107000" _busInterfaceName_="instr_slave" _componentLibrary_="Library" _componentName_="core_region" _componentVendor_="Vendor" _componentVersion_="2.0" _customType_="RW" _memoryMapName_="soc_map" _readType_="read" _registerName_="soc_control_PAD_MUX_PAD_MUX_0_4" _reserved_="false"	

Case Study: Results and Benefits

- 3× speed, 5× capacity in register management
- 35% time savings in HW/SW interface development
- Single source of truth, automated validation
- Reduced risk of design failure

Guidelines for Robust HSI Design

Early Involvement of Software Teams

- Software can do some things more easily than RTL
- Keep the software team in lockstep with RTL teams
- Use automation on a golden source input

The product doesn't ship
unless the device driver works!

Hierarchical vs Leaf Address Maps

- Hierarchical address maps contain only other address maps
- Leaf address maps contain other memory objects, i.e. registers and memories
- Putting them together will work, but it may cause issues for specific outputs

Simplifies UVM backdoor paths!

Decide Top-Level Word Size Early

- Multiple IPs mean potentially various access widths
- Set all developed maps the same word size
- Make the top the largest access size

Simplifies verification and the need for shadow registers!

Cautiously Use Byte Enables

- Decide early!
- Can mix and match IPs, but system map needs to be one or other
- Adds complications to HW verification and software writing

Simplifies verification!

Fields as the Atom

- Each output has different requirements and focus
  - UVM is register - aligned
  - C Header is typically register - aligned
  - RTL is flop-aligned
- Using fields gives a wide range of support

Simplifies various outputs!

Avoid Mixed Access in Registers

- Don't mix functionality and purpose
- Don't cram fields
- There is enough room for specific registers doing specific things

Simplifies software development!

Don’t Smash Your System Map

- Give your maps some room between each other
- Group like things together
- It simplifies the decoders

Future proofs your designs.

Design for Security

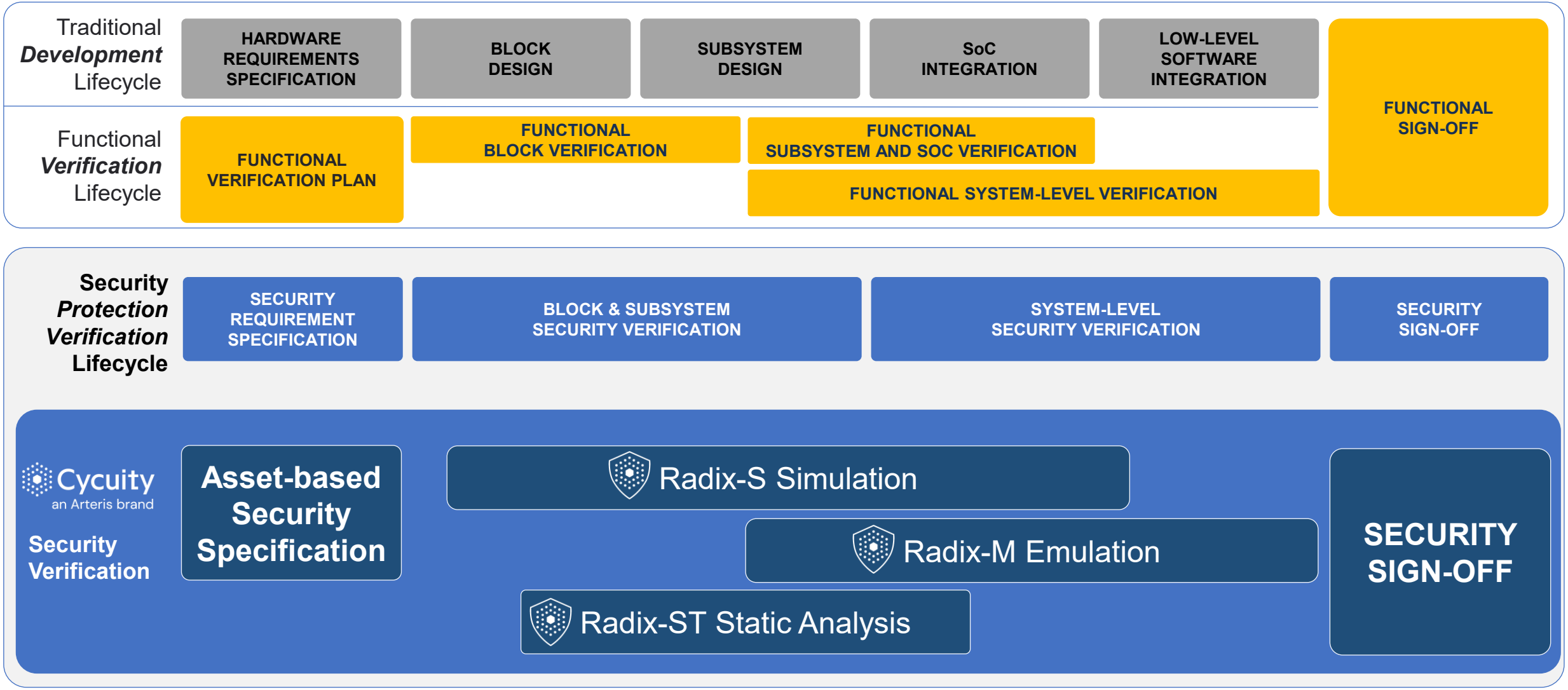
- Identify security-sensitive assets
- Develop verifiable security requirements
- Apply systematic security verification at each stage of design
- Collect security verification evidence

Enable traceable security assurance.

Arteris Is Investing in Tools to Take the Pain away from HW Security Verification

- **Radix-ST**: Static RTL analysis to identify Common Security Weaknesses (CWE) early in the HW development cycle
- **Radix-S**: Subsystem-level security verification capable of identifying 80% of common hardware weaknesses in the Common Weakness Enumeration (CWE) database maintained by MITRE
- **Radix-M**: System-level security verification with system FW/SW

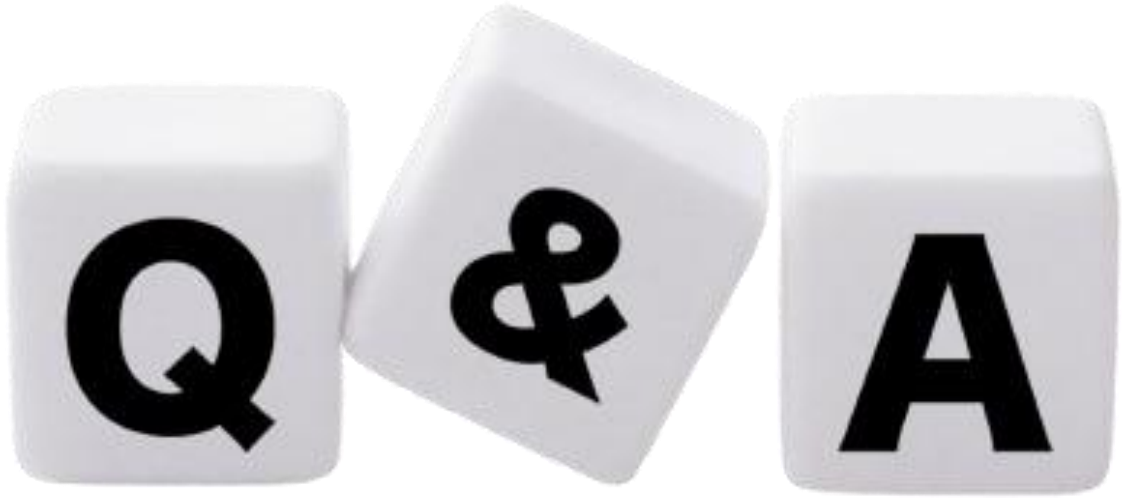
Security Verification in Lockstep with Chip Development



Conclusion

- Blending standards and automating SoC integration
- Adoption of IPXACT and SystemRDL/CSRSpec
- Future directions: AI-assisted integration, formal verification

Thanks for your attention!



Learn more at arteris.com