



SoC HUB

# Introduction to IP-XACT and Kactus2

26 January 2022



## HDL level

- RTL code has three aspects mixed
  - Behavioral description
  - Structural description
  - Control for configuration
- Implicit references that get evaluated late in the design flow
- Works fine for small design, but vulnerable to errors in large projects
  - 100k files, multiple vendors, ...
  - Wrong path/files, conflicts in naming, custom scripts dependent on file version, ...
- Coding style agreements does not seem to help

## SoC level

- The scale is so large that nobody can comprehend the whole system in detail
- Abstractions above RTL must be used for design space exploration
- Multitude of tools, languages and specification styles
- Design for deadline often compromises design for reuse
- Integration of IPs from different vendors is difficult without any agreed rules for interoperability

# IP-XACT ™

## IEEE-1685 Standard

Standard Structure for Packaging, Integrating, and  
Reusing IP within Tool Flows



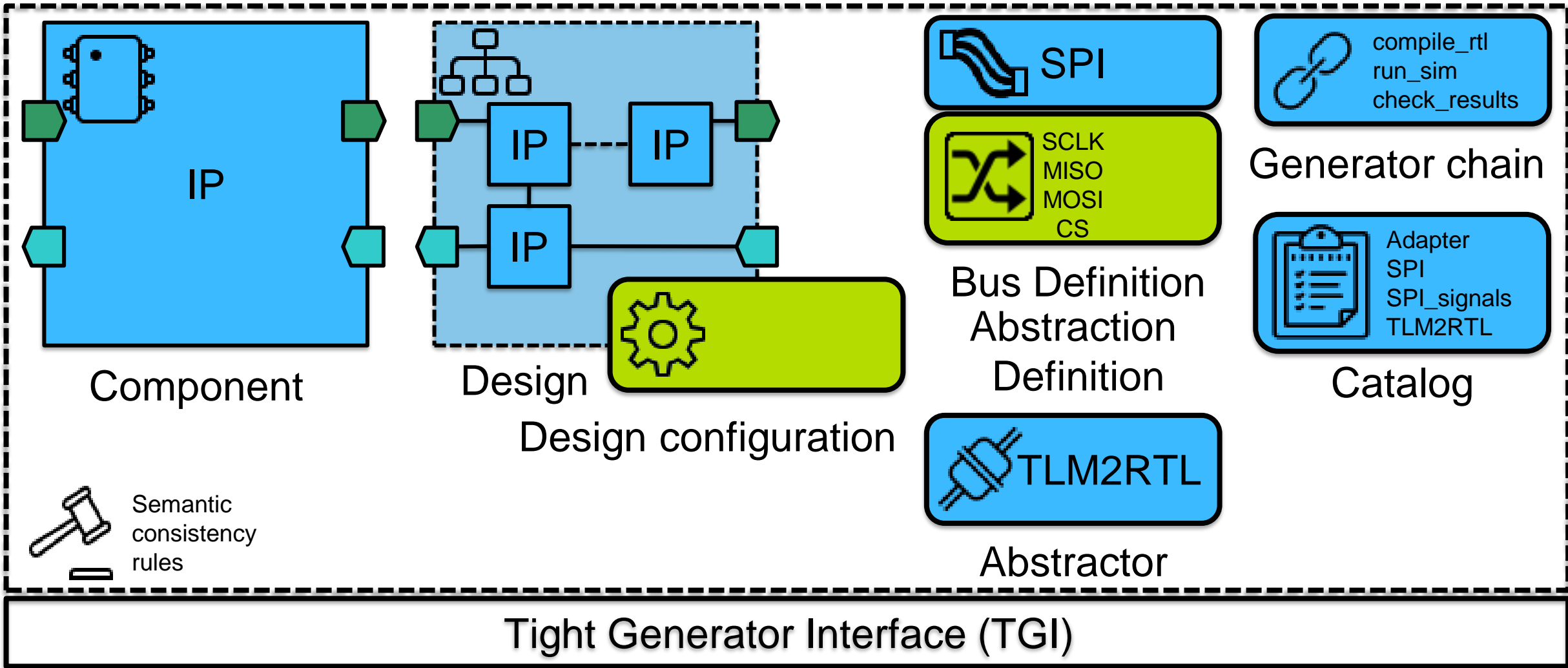
# SoC HUB IP-XACT

- Electrical data sheet for Intellectual Property (IP)
- Industry standard for design data exchange
  - XML format
  - 790 unique elements, 241 attributes
  - Vendor extensions
  - Generator interface for tool flow
- First maintained by SPIRIT consortium
- Now IEEE standard 1685, maintained by Accellera Initiative
  - Members: AMD, Arm, Cadence, NXP, Nvidia, ...

```
<ipxact:vendor>tuni.fi</ipxact:vendor>
<ipxact:library>cpu.structure</ipxact:library>
<ipxact:name>cpu_example</ipxact:name>
<ipxact:version>1.0</ipxact:version>
<ipxact:busInterfaces>
  <ipxact:busInterface>
    <ipxact:name>spi_master</ipxact:name>
    <ipxact:busType vendor="tuni.fi" library="interface" name="spi" version="1.0"/>
  </ipxact:busInterface>
</ipxact:busInterfaces>
```



# SoC HUB Overview






# SoC HUB Document identification

- All top-level documents have a unique identifier, VLNV
- Example: `tuni.fi:peripheral.components:uart:1.0`

Vendor	Library	Name	Version
--------	---------	------	---------
- Cross-references by VLNV
  - No broken file paths



## SoC HUB **Vendor extensions**

- Standard means for customization
- Any content
- Most, but not all, elements extendable
- Not compliant with other tools 
  - Potential danger for vendor lock-in
  - e.g. Xilinx adds ~200 elements

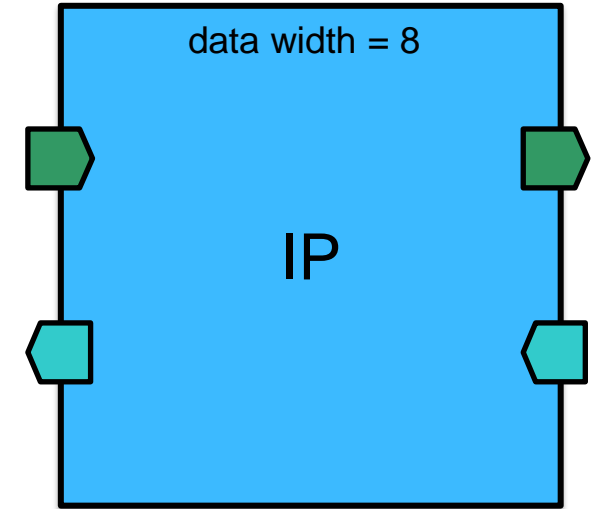
# IP-XACT core elements





# SoC HUB **Component**

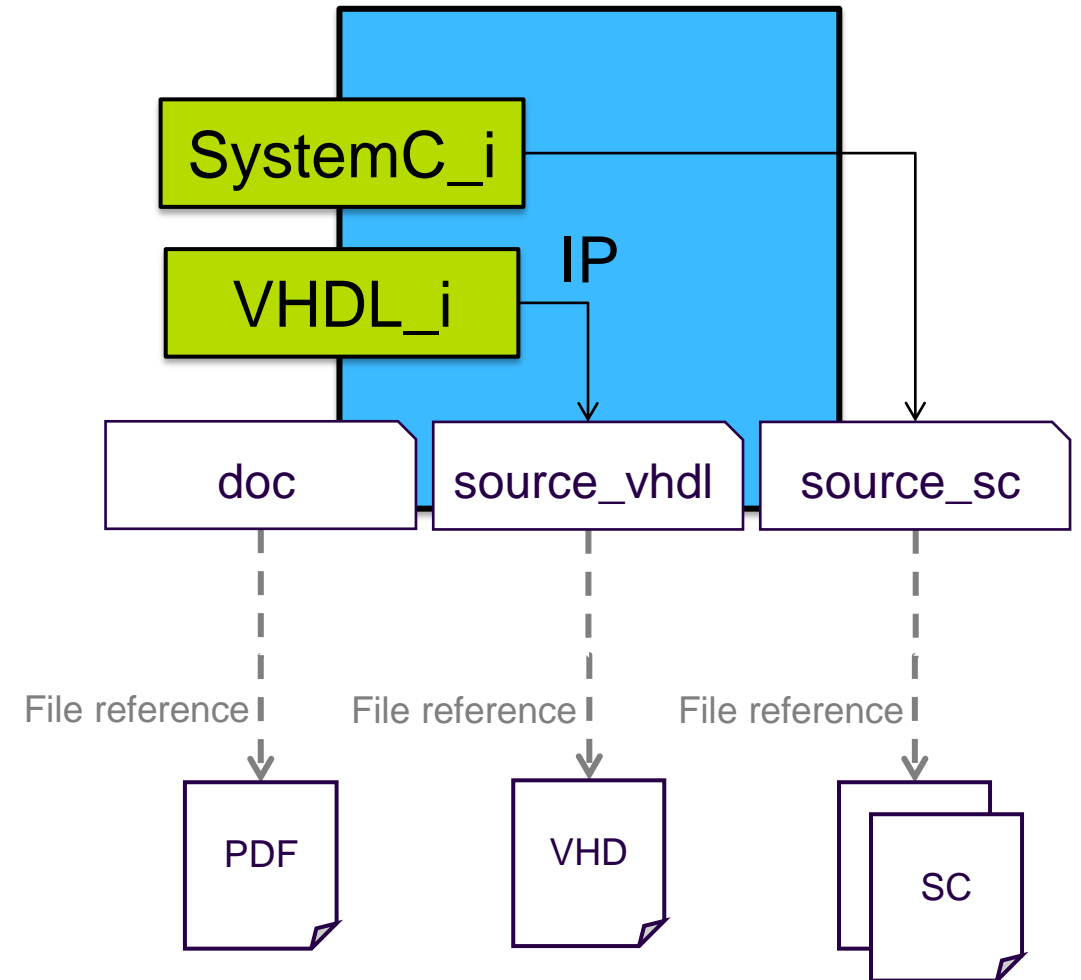
- Static structural model of a single IP
  - processor, uart, crossbar...
- Equivalent to VHDL entity or Verilog module
- Reusable: configurable by **parameters**
- May include multiple implementations
  - Same external interface





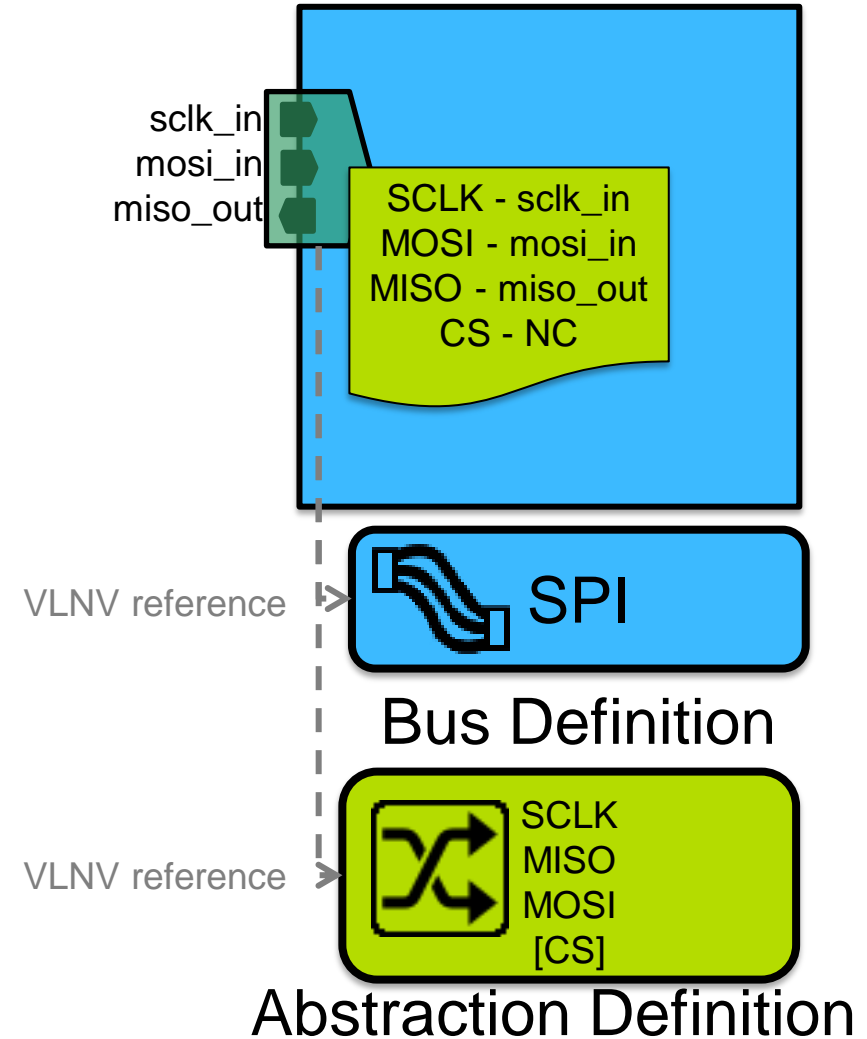
# SoC HUB Component implementations

- Relevant **files** are listed in **FileSets**
- **Component instantiations** define implementation specific details
  - Filesets
  - Language
  - Module name
  - Module parameters (language specific)
  - Library/package



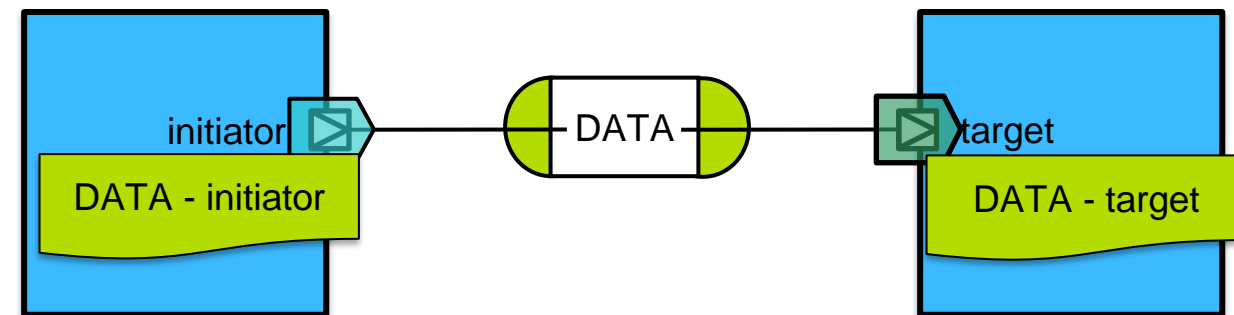
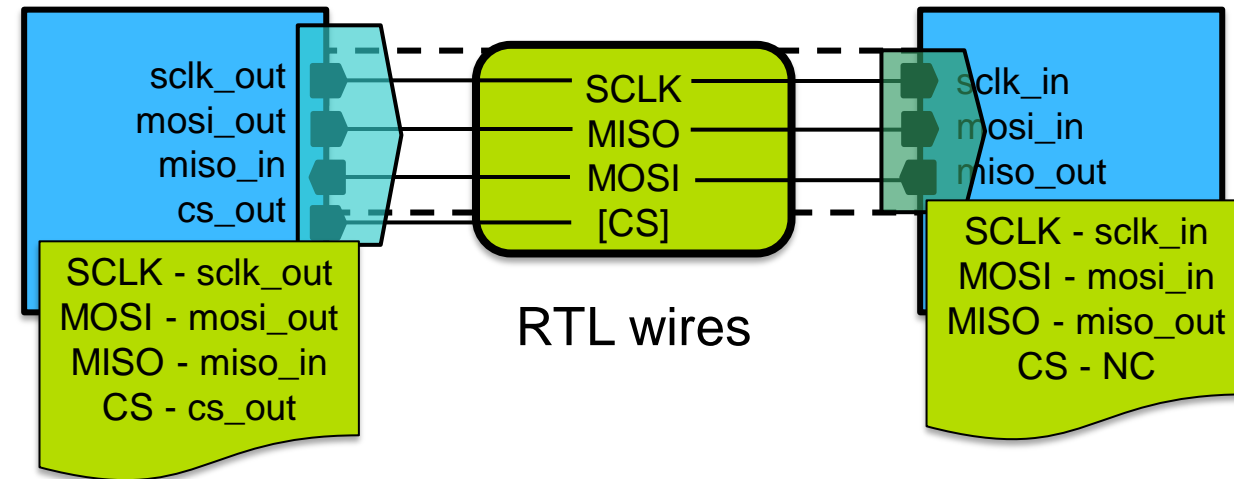
# SoC HUB Component interface

- Physical **ports** equivalent to HDL ports
- **Bus interface** groups ports
  - **Bus definition** specifies the protocol
  - **Abstraction definition** defines the logical **signals** on the bus
  - **Port maps** link physical ports to logical signals
  - 7 modes per component role:
    - Master, Slave, System
    - MirroredMaster, MirroredSlave, MirroredSystem
    - Monitor



# SoC HUB Interconnections

- **Interconnection** are made between component bus interfaces in a **Design**
  - Connects all ports mapped in the end bus interfaces
  - Avoids ad-hoc connections between individual ports
- Similar for both RTL and TLM models
  - RTL signal called **wire**
  - TLM signal called **transactional**
  - RTL-TLM cross-over with **abstractors**

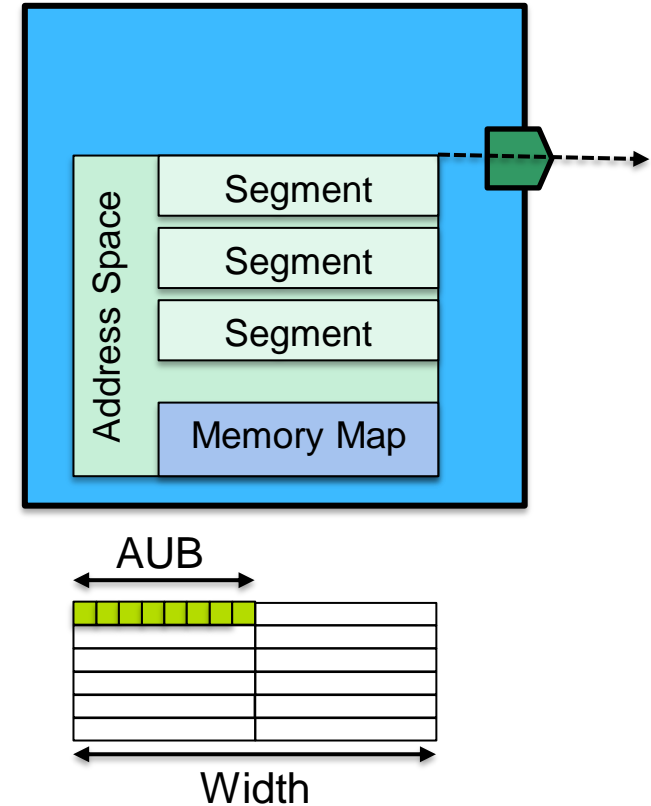


TLM transactionals



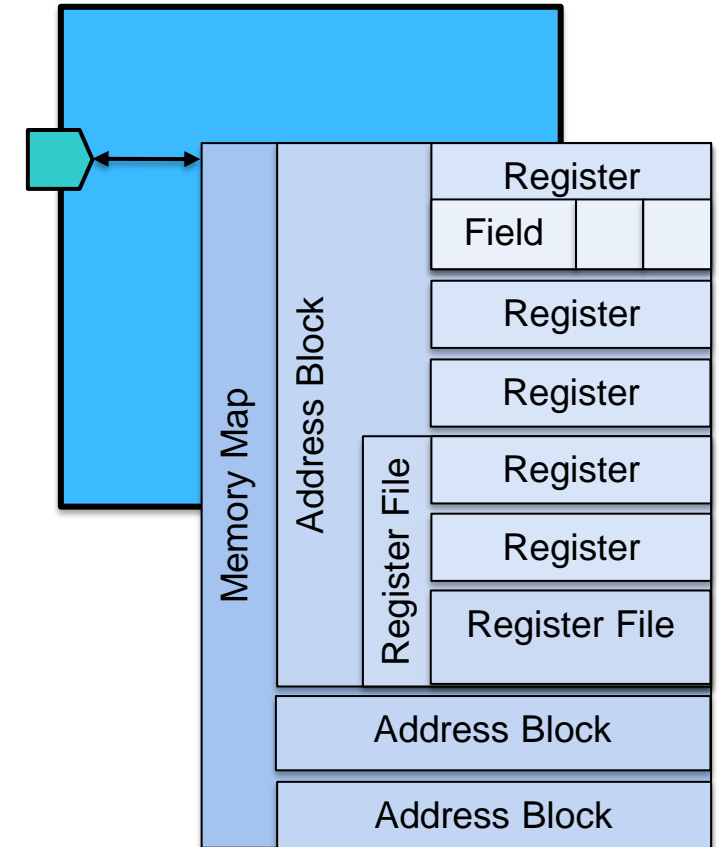
# SoC HUB Component address spaces

- **Address space** defines the address range that is available out of a master bus interface
  - **Address unit bits (AUB)** defines the number of data bits in each address increment (8 bits by default)
  - **Range** specifies the memory space as the **number of AUBs**
  - **Width** specifies the maximum bit size of a single transfer
- **Segments** describe sections of the address space
- **Local memory map** describes memory visible only in the containing address space



# SoC HUB Component registers

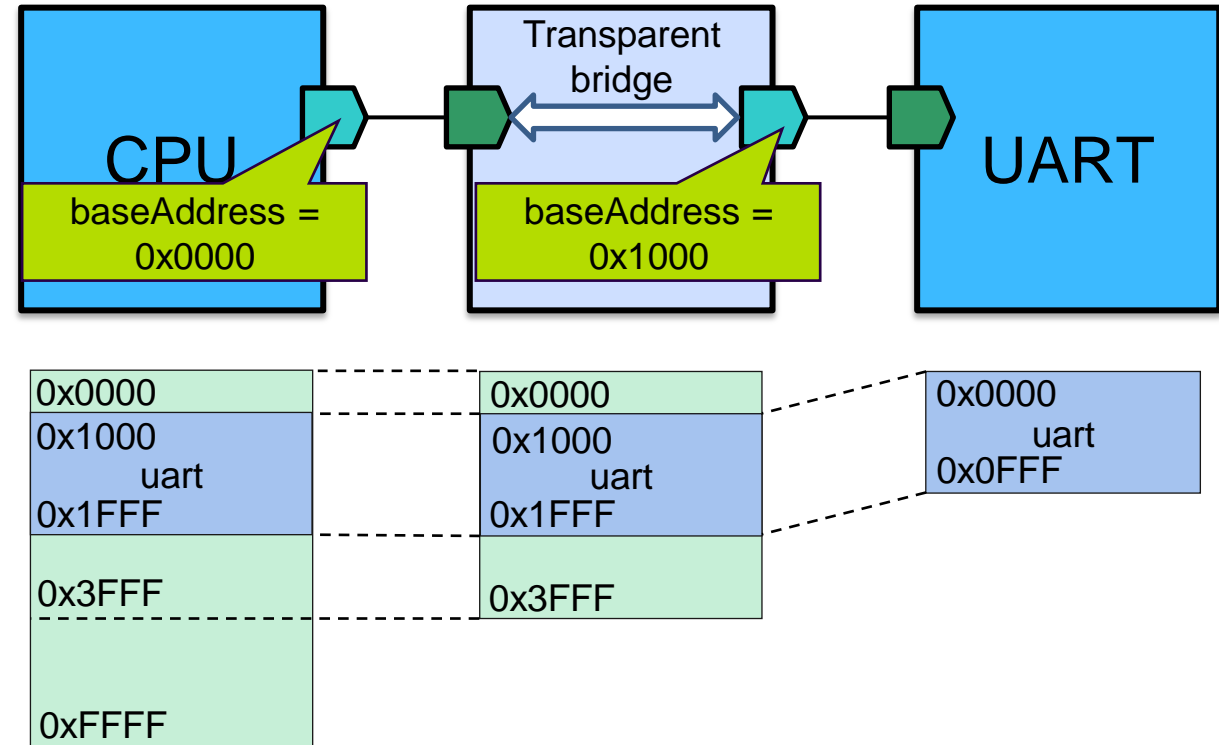
- IP-XACT memory model is hierarchical:
  - **Memory map** is accessible through a slave bus interface
  - **Address blocks** are continuous blocks of memory, reserved areas or registers
  - **Registers** define the software interface
  - **Fields** describe register bits
  - **Enumerated values** describe field bit pattern intent
- **Register files** group registers and other register files
- Address locations are defined as local offsets





# SoC HUB Bridges

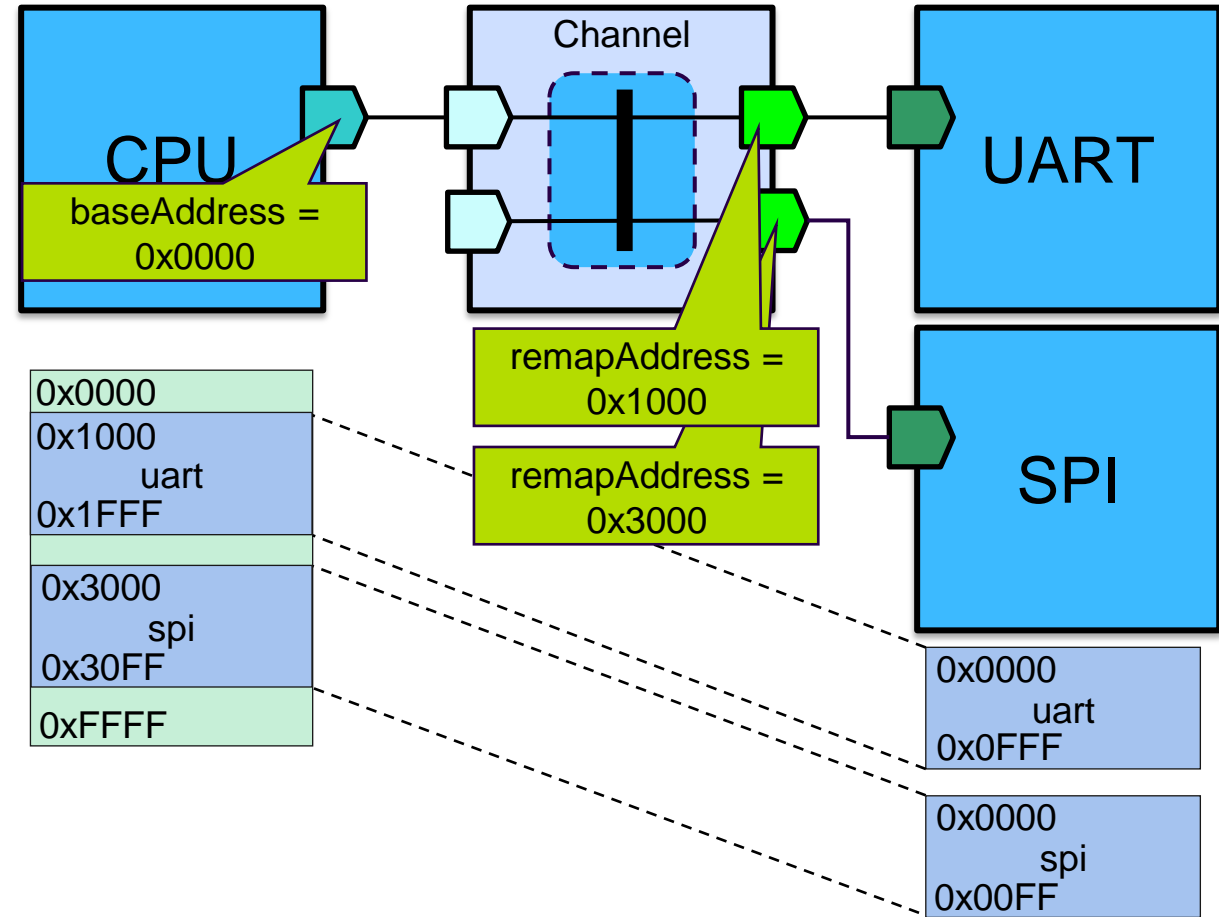
- **Transparent bridge** provides direct access through a component
  - Connects a slave interface to master(s)
  - Component may introduce protocol changes
  - Addressing remains unchanged
- **Opaque bridge** hides the connected IP memory map





# SoC HUB Channels

- **Channel** connects multiple mirrored bus interfaces into a single bus
  - Slave addresses are consistent for all masters
  - Only one master may initiate a transfer simultaneously

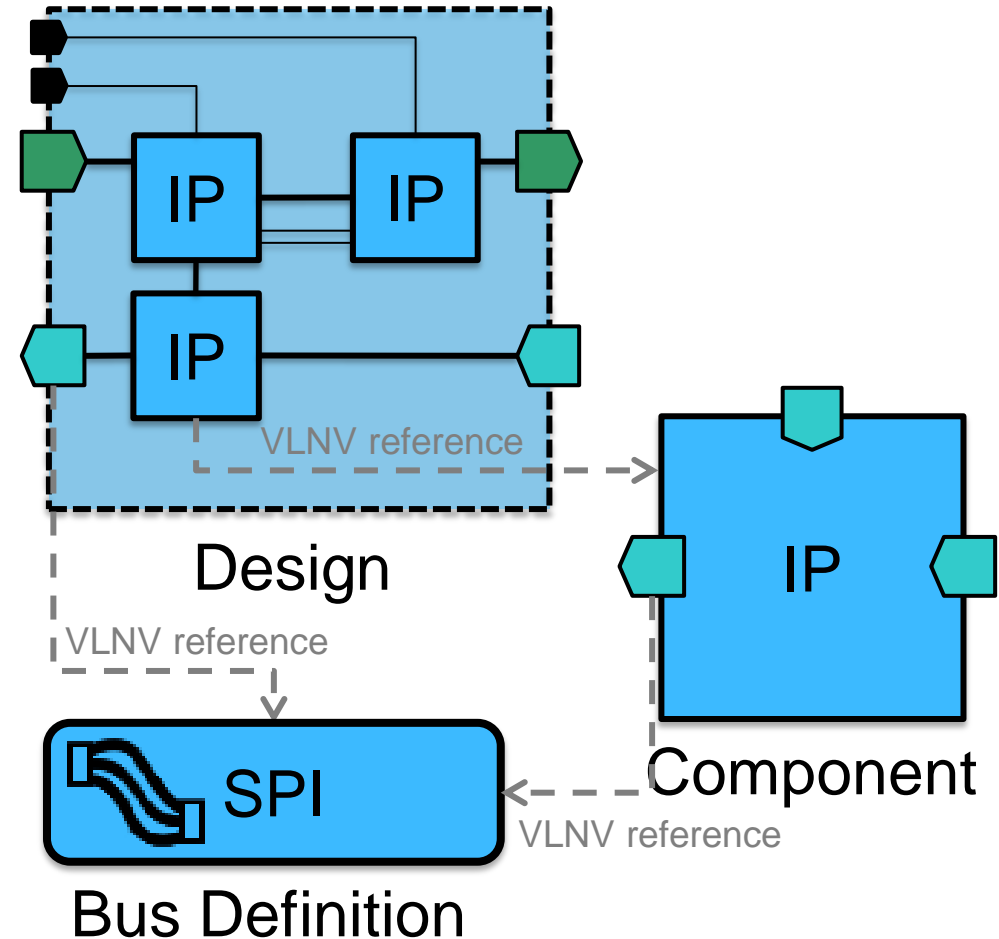






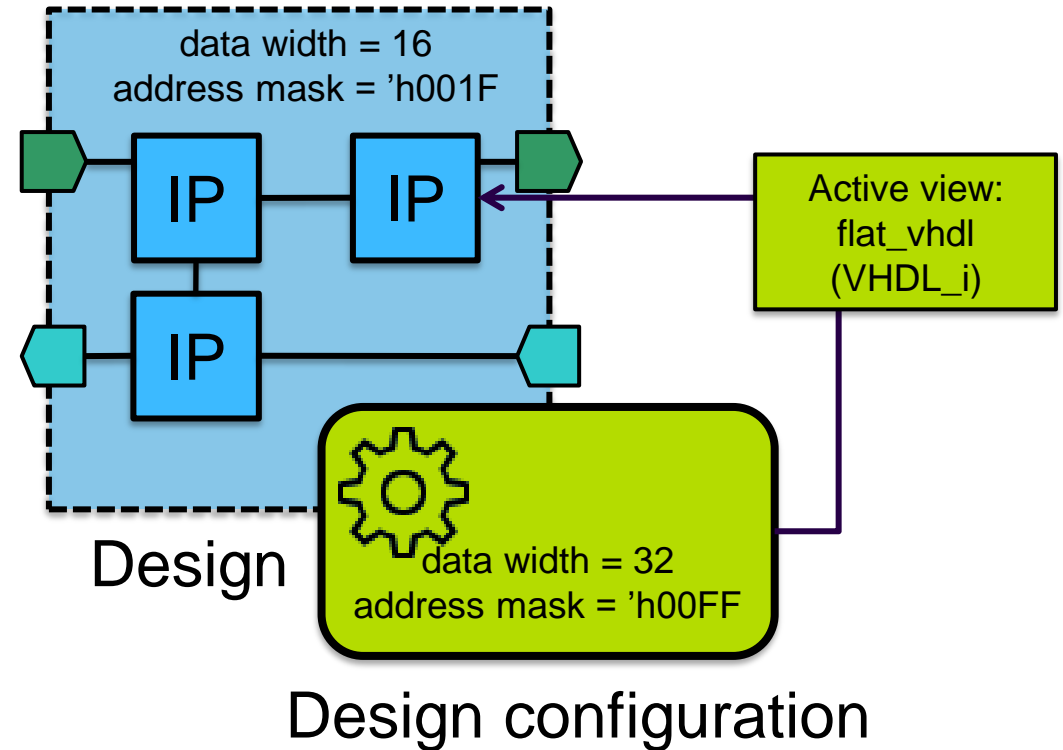
# SoC HUB Design

- A model of the system structure
- **Component instances** define sub-components
- **Interconnections** for communication
  - Formalized by Bus definitions
- **Ad-hoc connections** between ports



# SoC HUB Design configurations

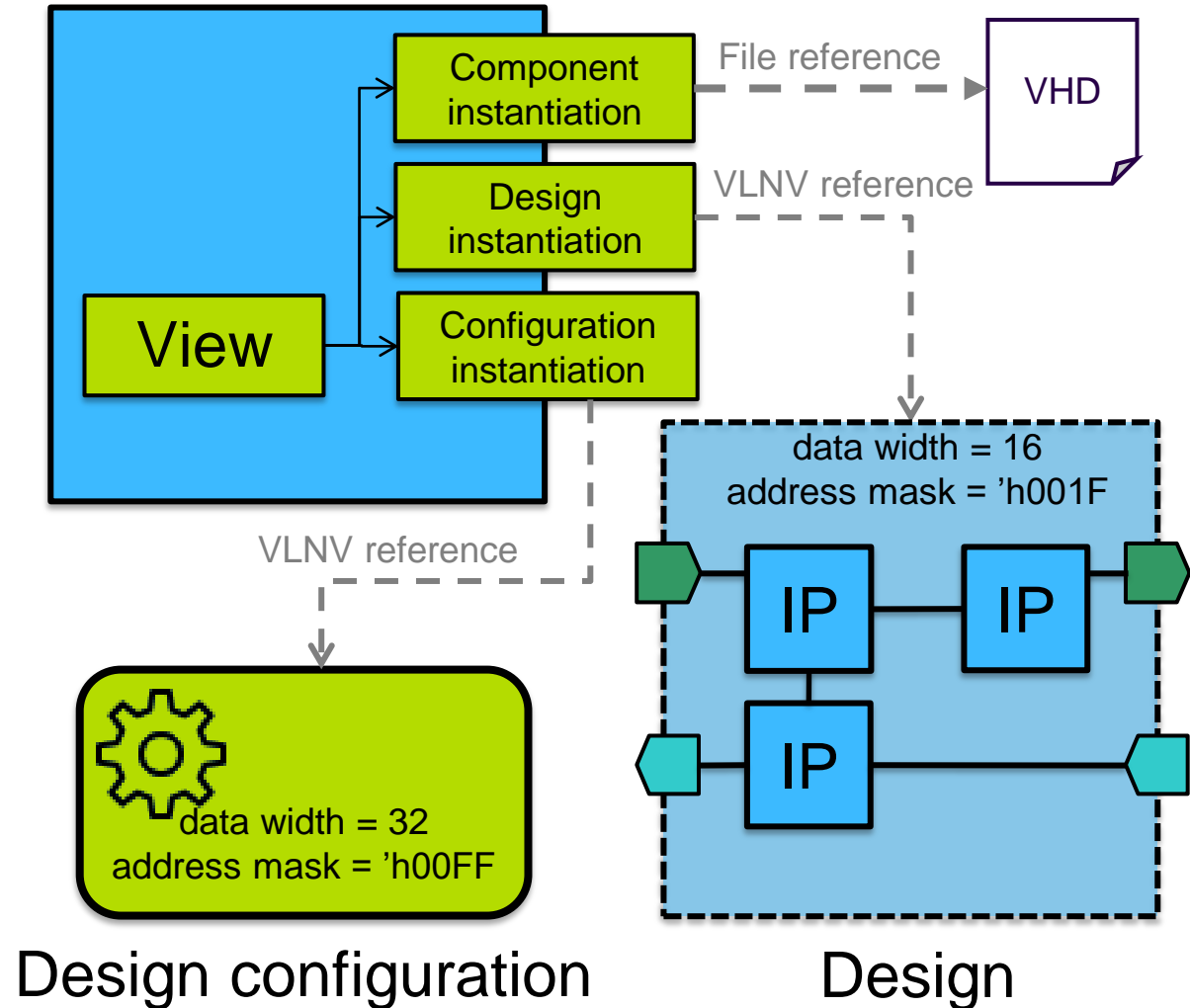
- A design is configured by
  - Parameter values
  - Component instance **active views**
- Active view selects the instance implementation
- Cannot add/remove instances or connections





# SoC HUB Component views

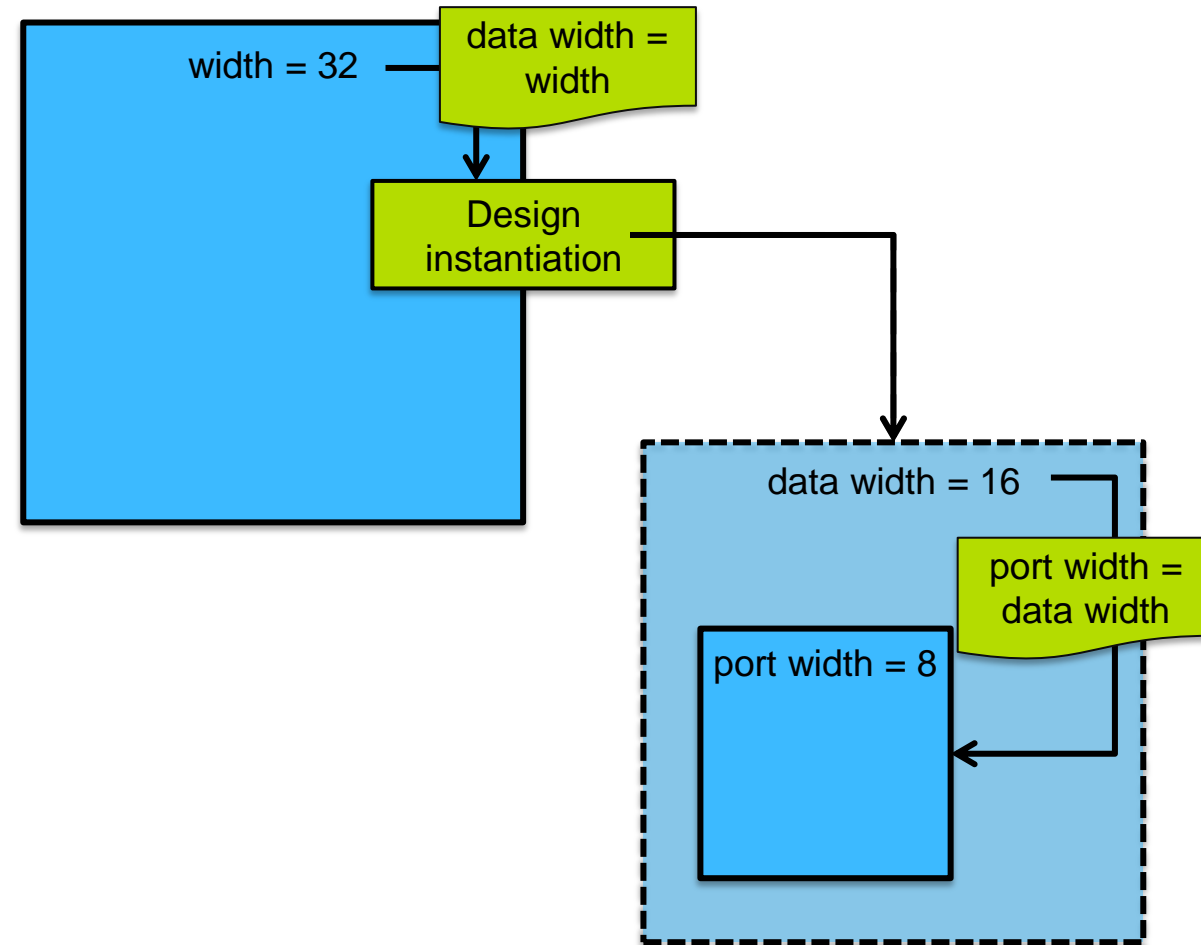
- A **view** defines a representation of the component
  - RTL, TLM, simulation, synthesis...
- Refer any combination of instantiations:
  - **Component instantiation** for selecting the implementation
  - **Design instantiation** for identifying the hierarchical design
  - **Design configuration instantiation** for selecting the configuration for the design





# SoC HUB Parameter propagation

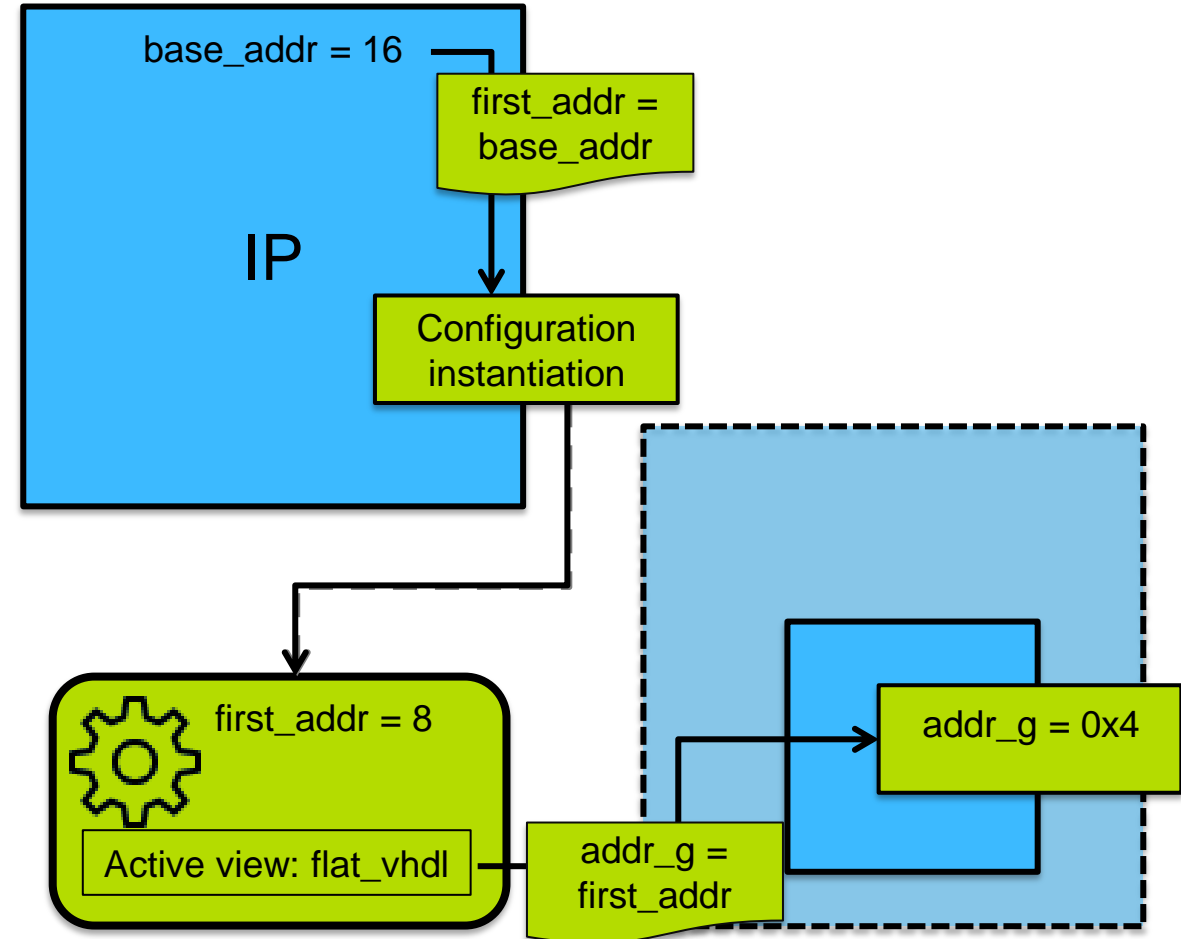
- Design may override parameter values in component instances
  - **Configurable element values**
  - Unless set to resolve **immediate** value in the component (default)
- Design instantiation may override design parameters
  - May reference containing component parameters





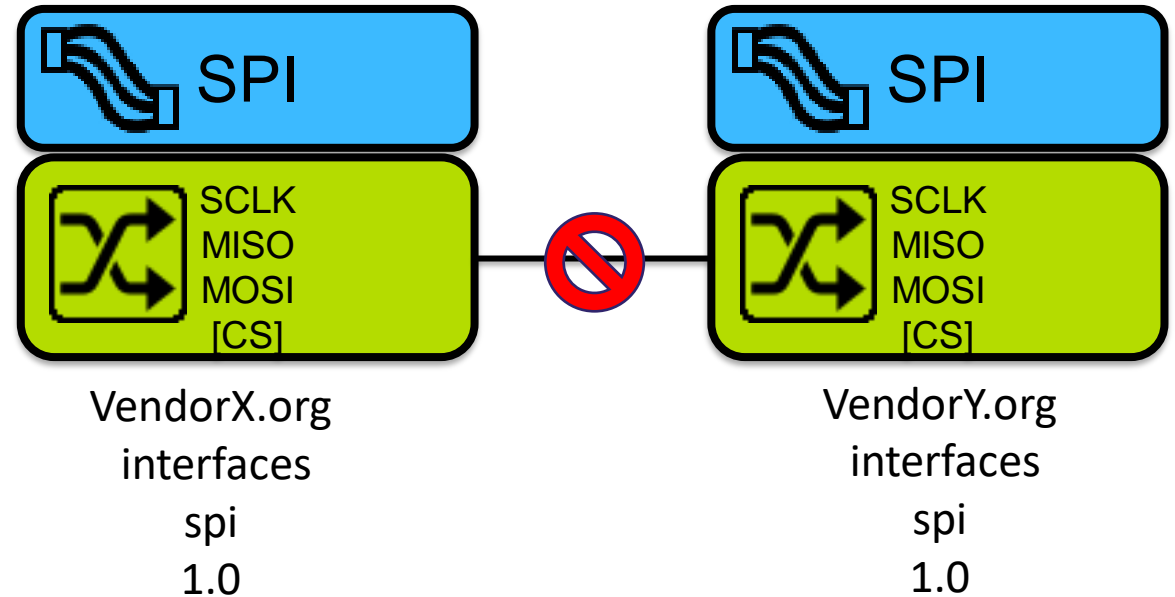
# SoC HUB Module parameter propagation

- Design configuration may override **module parameters** in component instantiation
- Design configuration instantiation may override design configuration parameters
  - May reference component parameters
- Propagation rules:
  - Value propagates down one level in hierarchy
  - May only reference parameters in the same document



- Companies seem to be most interested in:
  - File management
    - Help organize IP related files
  - Connectivity
    - Help plug-and-play IP-blocks
    - Benefits the HW team
  - Address definitions
    - Help manage tables of registers and memory maps
    - Benefits the SW team
- The standard is versatile in some respects, limited in others
  - Complementing standards: SystemRDL and Unified Power Format (UPF)

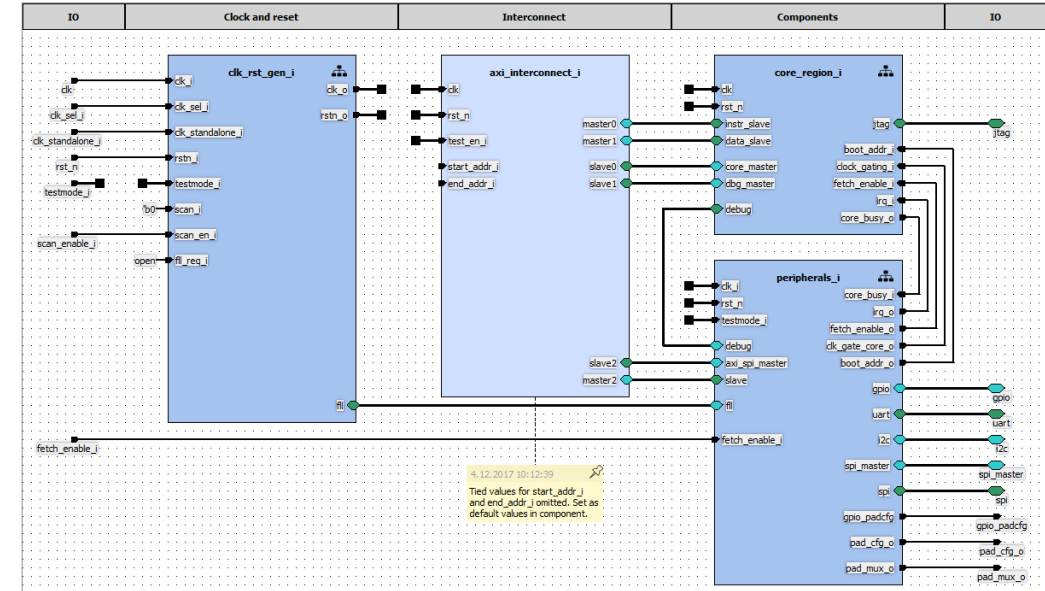
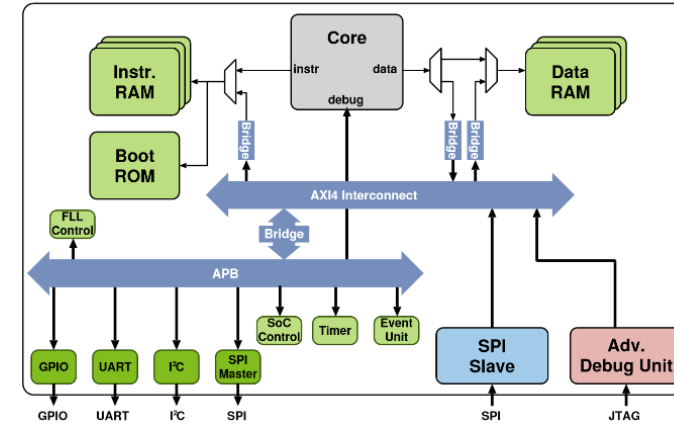
- No compatibility of bus/abstraction definitions from different vendors
  - Very few are publicly available
- Trying to force IP-XACT to conform to
  - Overly generic HDL
  - HDL with structure, behavior and configuration mixed
  - (File)name dependencies, virtual libraries
- Vendor extensions complicate data exchange





# SoC HUB Lessons learned from packaging RISC-V

- Open-source RISC-V microprocessor project, PULPino [1], approximately 250 SystemVerilog files across 21 repositories
- The project was packaged with Kactus2 resulting in 95k lines in 169 files [2]
- Enforce good HDL practices early
  - Decouple structure and behavior
  - Design module interfaces carefully
- Reference bus definitions are mandatory for true compatibility of IPs



[1] A. Traber and M. Gautschi, "Pulpino: Datasheet," ETH Zurich, Tech. Rep., 2016. [Online]. Available: <https://github.com/pulp-platform/pulpino/blob/master/doc/datasheet/datasheet.pdf>

[2] E. Pekkarinen and T. D. Hämmäläinen, "Modeling RISC-V Processor in IP-XACT," 2018 21st Euromicro Conference on Digital System Design (DSD), 2018, pp. 140-147.





# SoC HUB HDL challenge: structure and behavior

- Glue logic for simple data manipulation added as part of structural description
  - Typically multiplexing
- All behavior must be contained within components
  - Move glue logic to new components
  - Impractical for basic ANDs etc.

```
logic is_boot, is_boot_q;
...
boot_rom_wrap
  #(
    .DATA_WIDTH ( DATA_WIDTH )
  )
  boot_rom_wrap_i
  (
    .clk      ( clk                ),
    .rst_n    ( rst_n              ),
    .en_i     ( en_i & is_boot     ),
    .addr_i   ( addr_i[ ROM_ADDR_WIDTH-1:0] ),
    .rdata_o  ( rdata_boot         )
  );

assign rdata_o = (is_boot_q == 1'b1) ? rdata_boot : rdata_ram;

always_ff @(posedge clk, negedge rst_n)
  begin
    if (rst_n == 1'b0)
      is_boot_q <= 1'b0;
    else
      is_boot_q <= is_boot;
  end
```

Snippet from instr\_ram\_wrap.sv



# SoC HUB HDL challenge: conditional structure

- Configuration value enables/disables part of structure
  - Module instantiation
  - Wires
  - Ports
- Very flexible in (System)Verilog
- Inherently IP-XACT structure is static
  - **isPresent** attribute is sufficient where applicable
  - With instances consider creating an alternate design

```
`ifndef VERILATOR
  apb_uart apb_uart_i (
    ...
  );
`else
  apb_uart_sv
  #(
    .APB_ADDR_WIDTH( 3 )
  )
  apb_uart_i
  (
    ...
  );
`endif
```

Snippet from peripherals.sv.



# SoC HUB HDL challenge: generate loops

- Create regular structure
  - Module instantiations
  - Wire connections
- Replace with static instances

```
generate
  genvar i;
  for (i = 0; i < APB_NUM_SLAVES; i = i + 1) begin
    cluster_clock_gating core_clock_gate
    (
      .clk_o      ( clk_int[i]
                    ),
      .en_i       ( peripheral_clock_gate_ctrl[i] ),
      .test_en_i  ( testmode_i
                    ),
      .clk_i      ( clk_i
                    )
    );
  end
endgenerate
```

Snippet from peripherals.sv.



# SoC HUB HDL challenge: configurable interfaces

- Reusable modules with configurable interface
  - N master and M slave interfaces
- Bus interfaces are statically defined.  
Options:
  - Define maximum set, leave any unused unconnected or use `isPresent`
  - Automatically generate a component with correct number of bus interfaces and ports from a template

```
`include "axi_bus.sv"


module axi_node_intf_wrap
#(
    parameter NB_MASTER      = 4,
    parameter NB_SLAVE      = 4,
    parameter AXI_ADDR_WIDTH = 32,
    ...
)
(
    // Clock and Reset
    input logic clk,
    input logic rst_n,
    input logic test_en_i,

    AXI_BUS.Slave slave[NB_SLAVE-1:0],

    AXI_BUS.Master master[NB_MASTER-1:0],

    // Memory map
    input logic [NB_MASTER-1:0][AXI_ADDR_WIDTH-1:0] start_addr_i,
    input logic [NB_MASTER-1:0][AXI_ADDR_WIDTH-1:0] end_addr_i
);
```

Snippet from `axi_node_intf_wrap.sv`



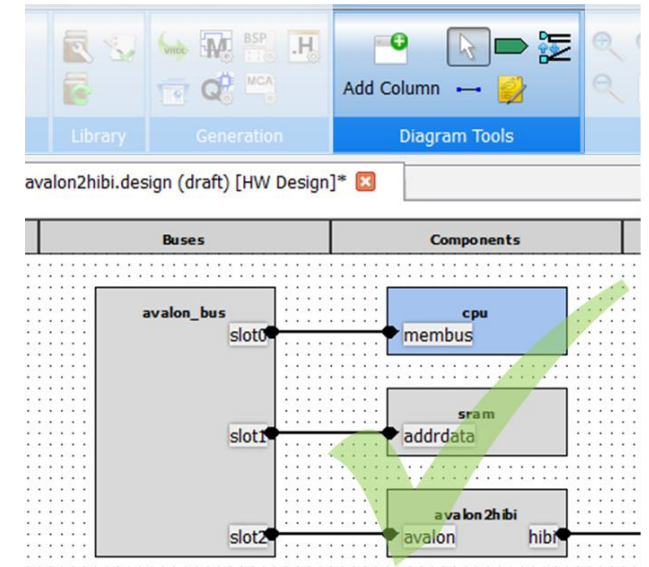
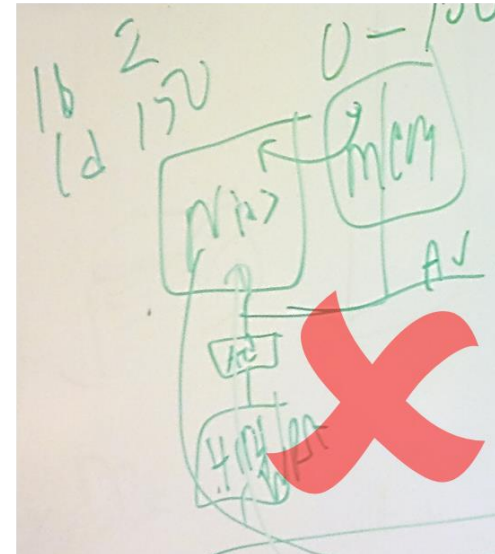
# Kactus<sup>2</sup>

## IP-XACT Design Environment



# SoC HUB Project motivation

- IP-XACT the most promising standard for common data exchange format
  - IP integrator companies
  - IP providers, SMEs, subcontractors
  - University teaching and research
- Disjoint effort in tool development
- Commercial IP-XACT tools are:
  - Expensive
  - Difficult to use
  - Disregarding the standard





# SoC HUB **Kactus2 project**

- Project started in TUT in 2009
  - Collaboration with 9 embedded system companies
  - Since then collaboration/support requested by 20+ companies
- Kactus2 released open-source in 2011
  - Initially a graphical editor for IP-XACT XML
  - Now over 500k lines of C++/Qt in total
- 19,896 downloads since release

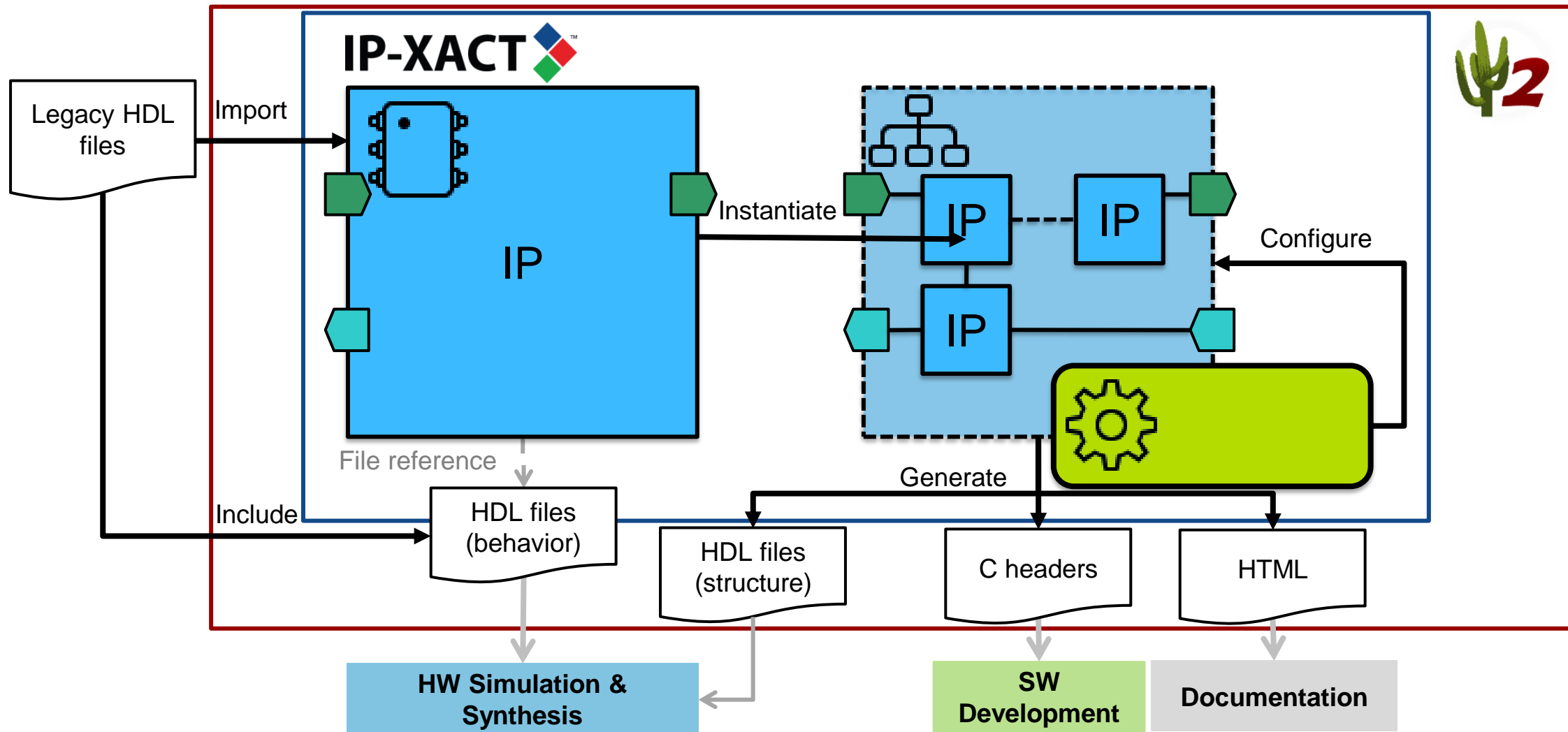


**GitHub**

<https://github.com/kactus2/kactus2dev>



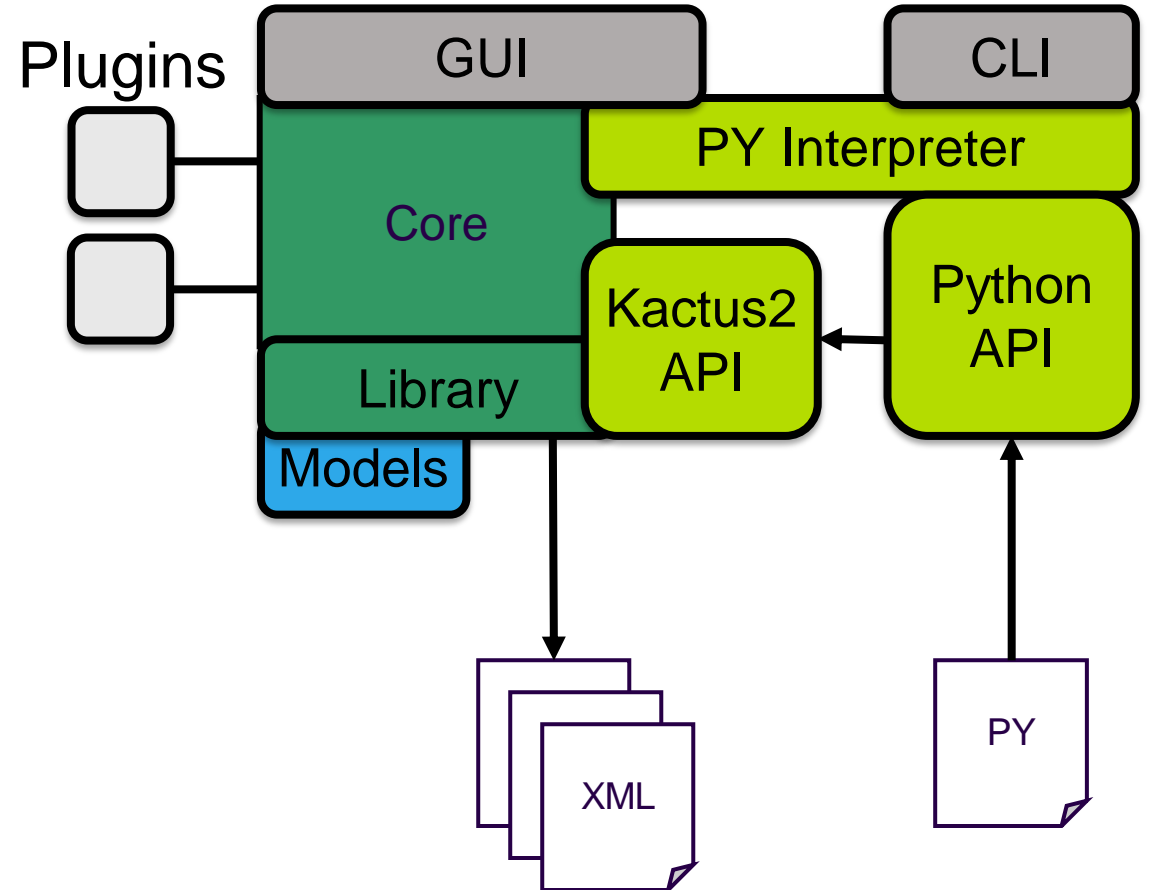
# SoC HUB IP-XACT-based design flow





# SoC HUB Kactus2 structure

- All access to IP-XACT files is controlled by the library interface
  - Data is parsed into model objects
- User input with graphical user interface (GUI) or command-line interface (CLI)
  - Both have access to Python interpreter since version 3.9.0
- A set of available functions are defined in PythonAPI
  - Scripting for batch jobs
  - Access to core functions and library through Kactus2 API
- Extendable with plugins
  - Importers for fast RTL to IP-XACT conversion
  - Generators for code/documentation
  - Source analyzers for file dependency information





SoC HUB

# Kactus2 graphical user interface

Toolbar

The screenshot displays the Kactus2 GUI with several key components:

- Toolbar:** Located at the top, it contains icons for File, Library, Protection, Edit, Generation, View, Configuration Tools, Workspace, and System.
- IP-XACT Library:** A sidebar on the left showing a hierarchy of components and libraries.
- Editor space:** The central area showing a table of ports for a component named 'core\_example.design (1.0)'.
 

Wire ports (14)		Transactional ports (0)		Ports									
Name	#	Name	Direction	Left (higher) bound, f(x)	Right (lower) bound, f(x)	Width	Type	Default value, f(x)	Array left, f(x)	Array right, f(x)	Port tags	Ad-hoc	Description
instruction_feed	1	instruction_feed	in	INSTRUCTION_WIDTH-1	0	28							
mem_address_o	2	mem_address_o	out	ADDR_WIDTH-1	0	9							
mem_data_o	3	mem_data_o	out	DATA_WIDTH-1	0	32							
mem_data_i	4	mem_data_i	in	DATA_WIDTH-1	0	32							
mem_we_o	5	mem_we_o	out	0	0	1							
clk_i	6	clk_i	in	0	0	1						✓	The mandatory clock, as this is synchronous logic.
rst_i	7	rst_i	in	0	0	1						✓	The mandatory reset, as this is synchronous logic.
mem_slave_rdy	8	mem_slave_rdy	in	0	0	1							
mem_master_rdy	9	mem_master_rdy	out	0	0	1							
addr_o	10	addr_o	out	INSTRUCTION_ADDRESS_WIDTH-1	0	8							
local_address_o	11	local_address_o	out	ADDR_WIDTH-1	0	9							
local_write_data	12	local_write_data	out	DATA_WIDTH-1	0	32							
local_write_o	13	local_write_o	out	0	0	1							
local_read_data	14	local_read_data	in	DATA_WIDTH-1	0	32							
- Context-sensitive help:** A panel on the right providing detailed information about wire ports, including definitions for direction, bounds, type, and ad-hoc ports.
- Notification and error messages:** A panel at the bottom left showing system messages and error counts.
- Python console:** A panel at the bottom right displaying the output of a Python script.
- Selection-sensitive editors:** A panel on the far right showing a component instance diagram and its associated metadata table.

IP-XACT document library

Editor space

Notification and error messages

Python console

Context-sensitive help

Selection-sensitive editors



# SoC HUB Bus definitions editor

- Kactus2 groups together bus and abstraction definition
- Abstraction definition includes
  - Qualifiers (address, data, clock, reset, any)
  - Presence (required, optional, illegal)
- Signal conditions can be created
  - e.g. some signal is not allowed in master interface

General (Bus Definition)

Bus definition  
Vendor: tut.fi  
Library: interface  
Name: spi  
Version: 1.0

Extended bus definition  
Vendor:   
Library:   
Name:   
Version:

Description  
Serial Peripheral Interface bus for multislave full duplex communication.

Constraints  
 Allow non-mirrored connections  
 Support broadcast  
 Addressable bus

Max masters on bus:   
Max slaves on bus:

System group names

Signals (Abstraction Definition)

Abstraction definition  
Vendor: tut.fi  
Library: interface  
Name: spi.absDef  
Version: 1.0

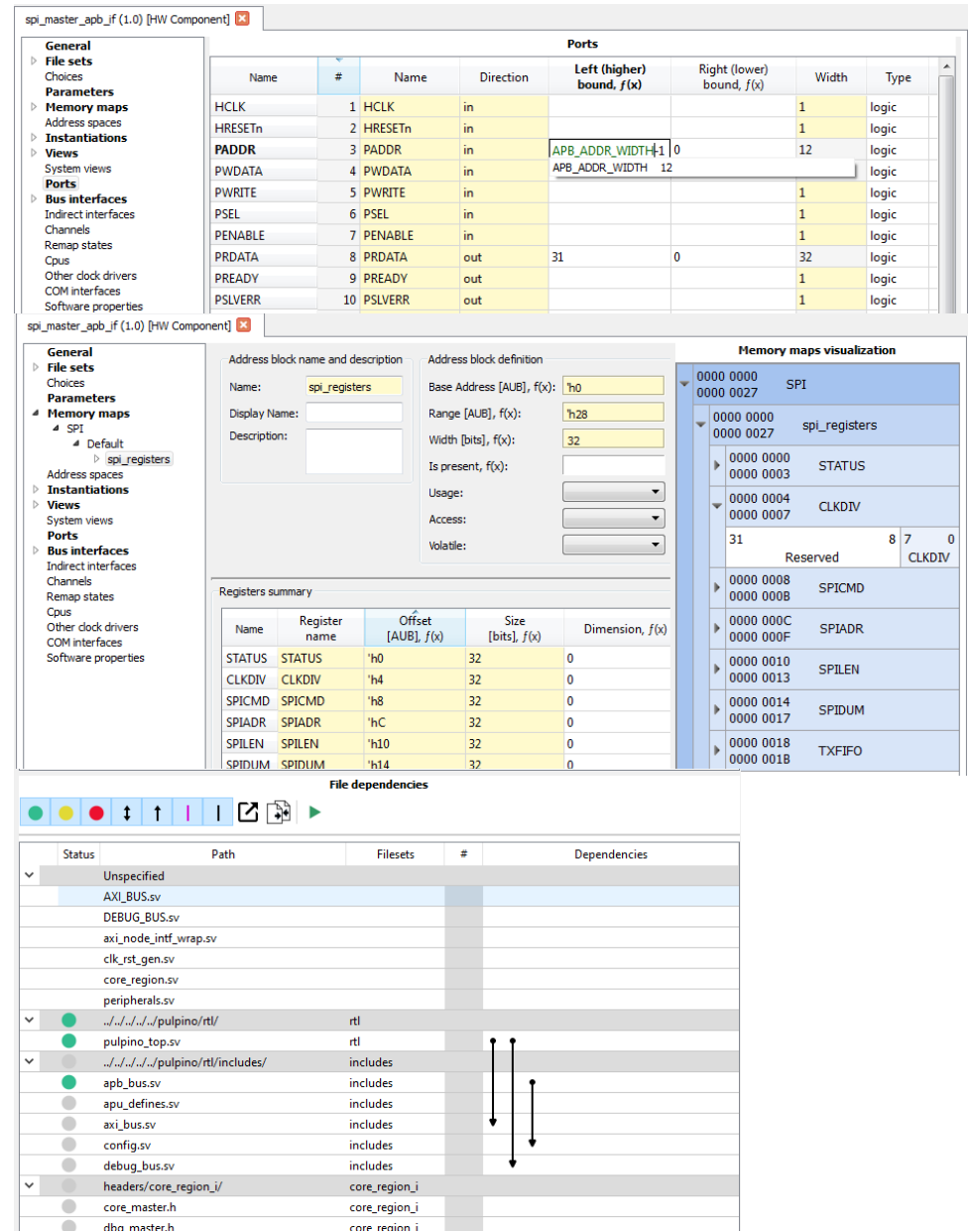
Extended abstraction definition  
Vendor:   
Library:   
Name:   
Version:

Description

Wire ports    Transactional ports

Name	Mode	Presence	Direction	Width	Default value	Driver	Qualifier	System group	Description
MISO	master		in	1					Master input, slave output.
MISO	slave		out	1					Master input, slave output.
MOSI	master		out	1					Master output, slave input.
MOSI	slave		in	1					Master output, slave input.
SCLK	master		out	1					Clock from master to slave.
SCLK	slave		in	1					Clock from master to slave.
SS	master		out						Slave select, may have variable...
SS	slave		in						Slave select, may have variable...

- Define IP details e.g.
  - Ports
  - Parameters
  - Registers
  - Related files, file sets
- File set dependency analysis
- Memory map visualization
- Automatic validity checks



The screenshot displays the SoC HUB Component editor interface for a component named 'spi\_master\_apb\_if (1.0)'. The interface is divided into several panels:

- Ports Table:** A table listing 10 ports with their names, directions, and widths.
 

Name	#	Name	Direction	Left (higher) bound, f(x)	Right (lower) bound, f(x)	Width	Type
HCLK	1	HCLK	in			1	logic
HRESETn	2	HRESETn	in			1	logic
PADDR	3	PADDR	in	APB_ADDR_WIDTH-1	0	12	logic
PWDATA	4	PWDATA	in	APB_ADDR_WIDTH	12		logic
PWRITE	5	PWRITE	in			1	logic
PSEL	6	PSEL	in			1	logic
PENABLE	7	PENABLE	in			1	logic
PRDATA	8	PRDATA	out	31	0	32	logic
PREADY	9	PREADY	out			1	logic
PSLVERR	10	PSLVERR	out			1	logic
- Registers Summary Table:** A table listing registers with their names, offsets, sizes, and dimensions.
 

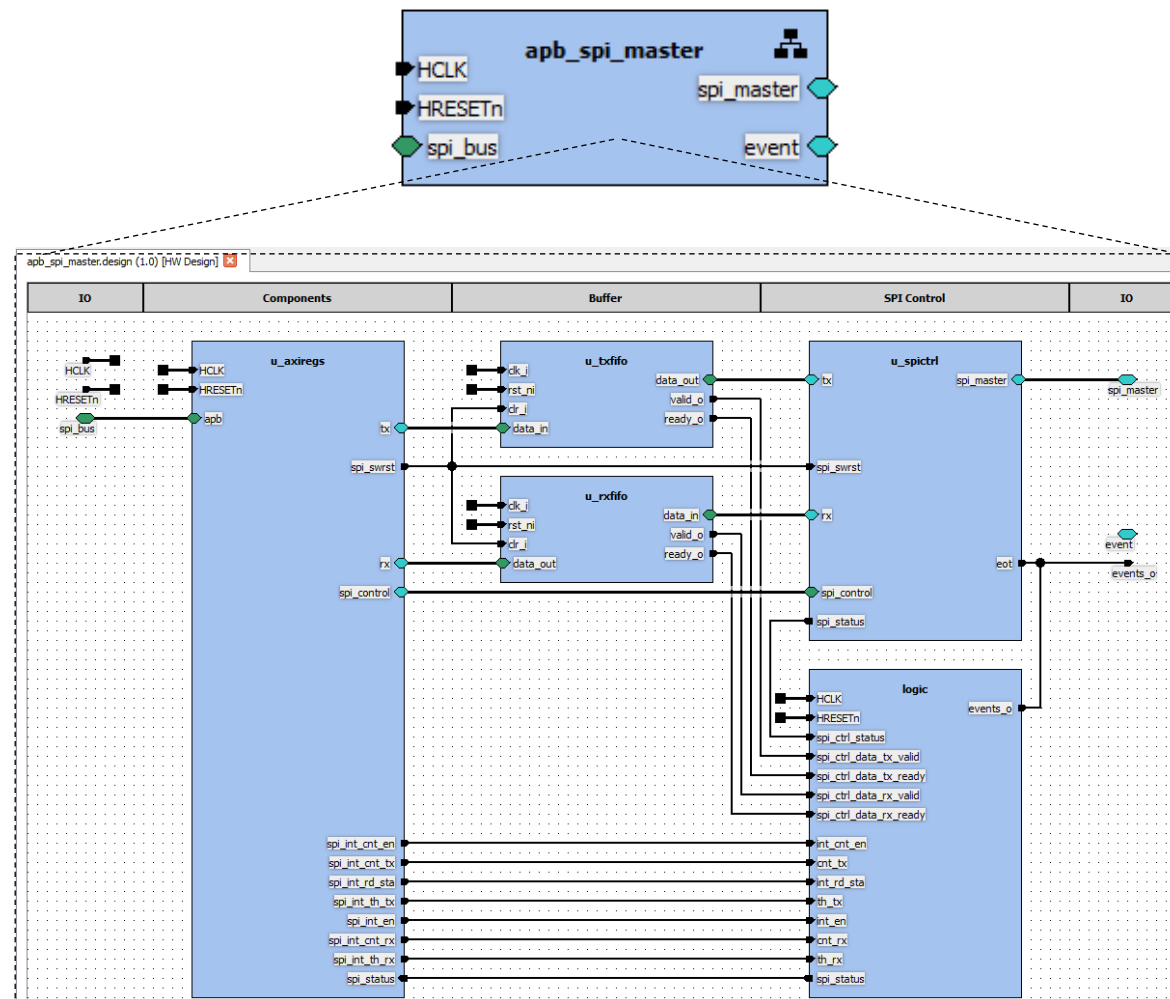
Name	Register name	Offset [AUB], f(x)	Size [bits], f(x)	Dimension, f(x)
STATUS	STATUS	'h0	32	0
CLKDIV	CLKDIV	'h4	32	0
SPICMD	SPICMD	'h8	32	0
SPIADR	SPIADR	'hC	32	0
SPILEN	SPILEN	'h10	32	0
SPIIDUM	SPIIDUM	'h14	32	0
- Memory maps visualization:** A tree view showing memory addresses and their associated components, such as 'spi\_registers', 'STATUS', 'CLKDIV', 'Reserved', 'SPICMD', 'SPIADR', 'SPILEN', 'SPIDUM', and 'TXFIFO'.
- File dependencies:** A table showing the dependencies between file sets.
 

Status	Path	Filesets	#	Dependencies
Unspecified	AXI_BUS.sv			
	DEBUG_BUS.sv			
	axi_node_intf_wrap.sv			
	clk_rst_gen.sv			
	core_region.sv			
	peripherals.sv			
●	./../../../../pulpino/rtl/	rtl		
●	pulpino_top.sv	rtl		
▼	./../../../../pulpino/rtl/includes/	includes		
●	apb_bus.sv	includes		
●	apu_defines.sv	includes		
●	axi_bus.sv	includes		
●	config.sv	includes		
●	debug_bus.sv	includes		
▼	headers/core_region_i/	core_region_i		
●	core_master.h	core_region_i		
●	rtba_master.h	core region i		

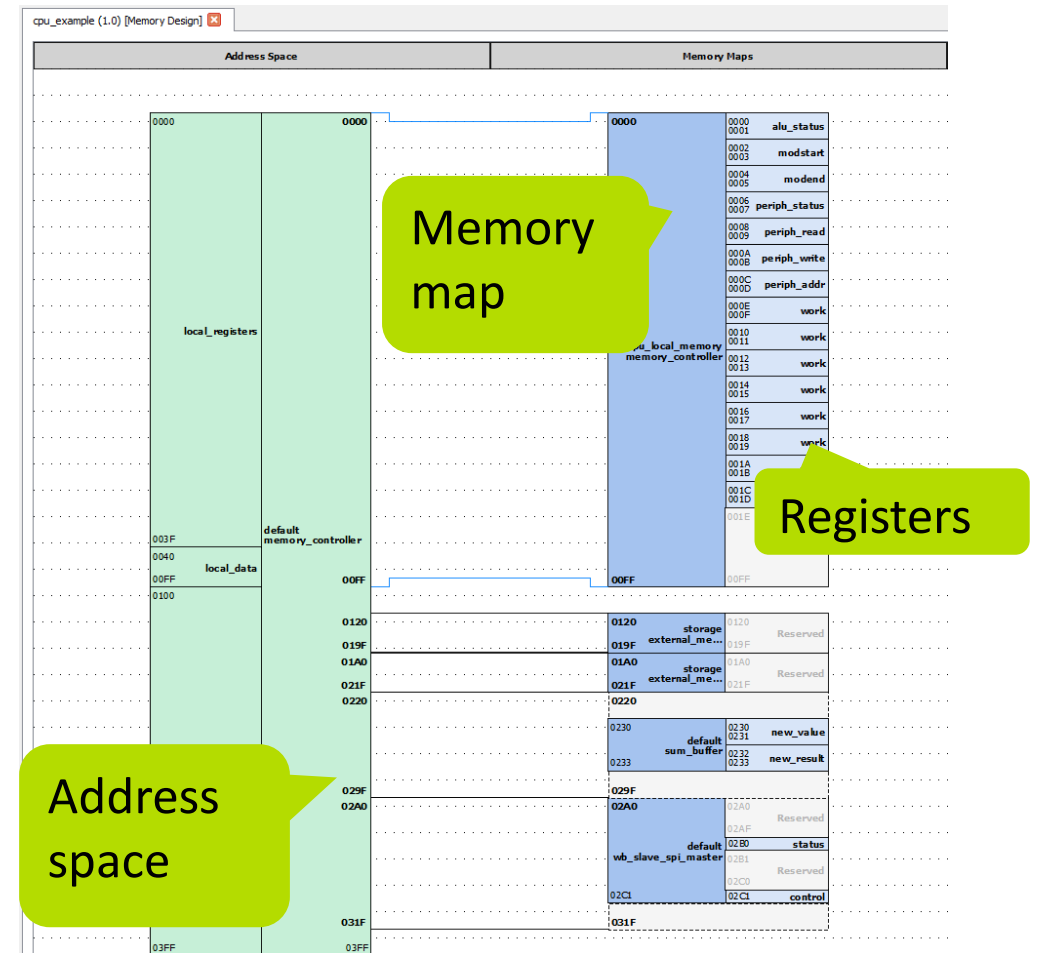


# SoC HUB Design editor

- Create hierarchical structure using existing components
- Configure instances
- Speculate with draft instances
- Automatic checks for valid connections



- Align address spaces with memory maps
  - Multiple components
  - Multiple hierarchy levels
- Resolve register addressing based on connectivity
- HW designer can check address definitions in all components



[1] M. Teuho, E. Pekkarinen and T. Hämäläinen, "Visualization of Memory Map Information in Embedded System Design," 2018 21st Euromicro Conference on Digital System Design (DSD), 2018, pp. 163-166 .

# Summary



## SoC HUB **Summary**

- IP-XACT targets to ease reuse and data exchanged between IP vendors
- Lack of commonly shared bus definitions and examples slow down adoption
- Components and designs capture the SoC structure
- Kactus2 is the open-source IP-XACT tool by Tampere University





# SoC HUB Resources

- SoC Hub: [www.sochub.fi](http://www.sochub.fi)
- Kactus2 home: <https://research.tuni.fi/system-on-chip/tools/>
- Kactus2 source code and issue tracking: <https://github.com/kactus2/kactus2dev>
- Kactus2 installer download: <https://sourceforge.net/projects/kactus2/>
- Kactus2 video tutorials: <https://www.youtube.com/user/Kactus2Tutorial/>

