Tampereen yliopisto
Tampere University

Modelling RISC-V architecture in Kactus2

System-on-Chip Research Group, Tampere University

CSW Tampere

14th October 2020

Biography: Esko Pekkarinen

- Doctoral Researcher at TAU
  - Design tools and IP-XACT-based design
  - Teaching on System Design and Real-time Systems
- Kactus2 IP-XACT design tool developer since 2013
- Previous research topics
  - Simulating Wireless Senson Networks (MSc work)
  - Modeling applications on MP-SoCs (BSc work)

Motivation

- RISC-V is a standardized ISA which
  - has optional and custom extensions, and
  - numerous implementations.
- High abstraction level is mandatory in modern embedded system design.
  - IP reuse
  - Multiple configurations
  - Automation
- Yet modelling is seen as tedious compared to traditional HDL-driven design by the designers
  - Legacy code bases difficult to adapt
  - Benefits clear only on the long-term

Samsung to Use SiFive RISC-V Cores for SoCs, Automotive, 5G Applications

by Anton Shilov on December 12, 2019 11:00 AM EST

https://www.anandtech.com/show/15228/samsung-to-use-riscv-cores

Western Digital Rolls-Out Two New SweRV RISC-V Cores For Microcontrollers

by Anton Shilov on December 13, 2019 5:00 PM EST

Western Digital has added two new processor cores — the SweRV Core EH2 and the SweRV Core EL2 — into its SweRV portfolio of microcontroller CPUs. And, keeping in line with past parts, and the company has made their register-transfer level (RTL) design abstraction available to the industry for free. In addition, the company has also introduced the first hardware reference design for OmniXtend cache coherent memory over Ethernet protocol, and transferred management and support of the architecture to Chips Alliance.

https://www.anandtech.com/show/15231/western-digital-rollsout-two-new-swerv-riscv-cores

Design challenges

HDL level

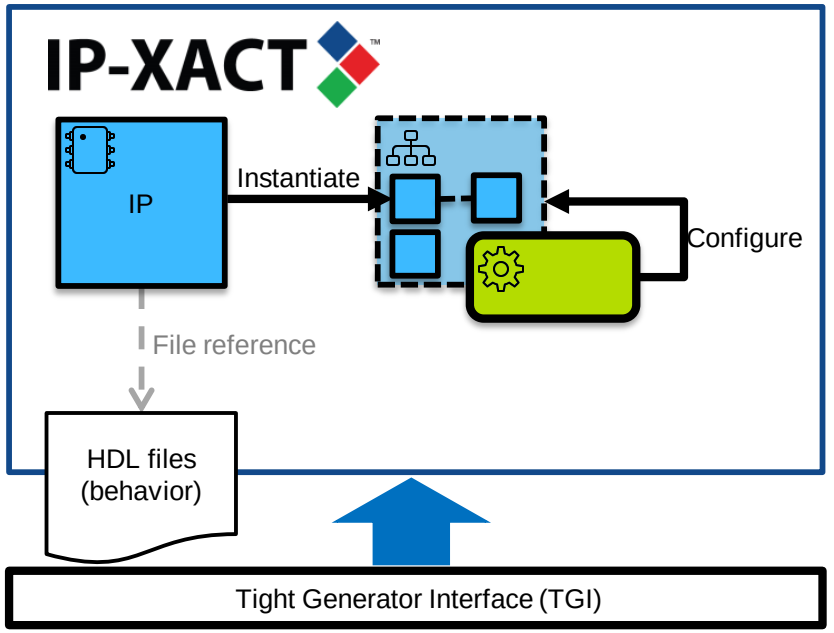
- HDLs have three aspects mixed
  - Behavioral description
  - Structural description
  - Control for configuration
- Implicit references are evaluated late in the design flow
- Works fine for small design, but vulnerable to errors in large projects.
  - 100k+ files, multiple vendors, …
  - Wrong path/files,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

SoC level

- Abstractions above RTL must be used for design space exploration
- Multitude of tools, languages and specification styles
  - Different formats (syntax)
  - Meaning (semantics)
  - Intention (how language or tool is applied)
- Design for deadline often compromises design for reuse
- Integration of IPs from different vendors is difficult without any agreed rules for interoperability
  - The scale is so large that nobody can comprehend the whole system in detail

IEEE 1685-2014 IP-XACT

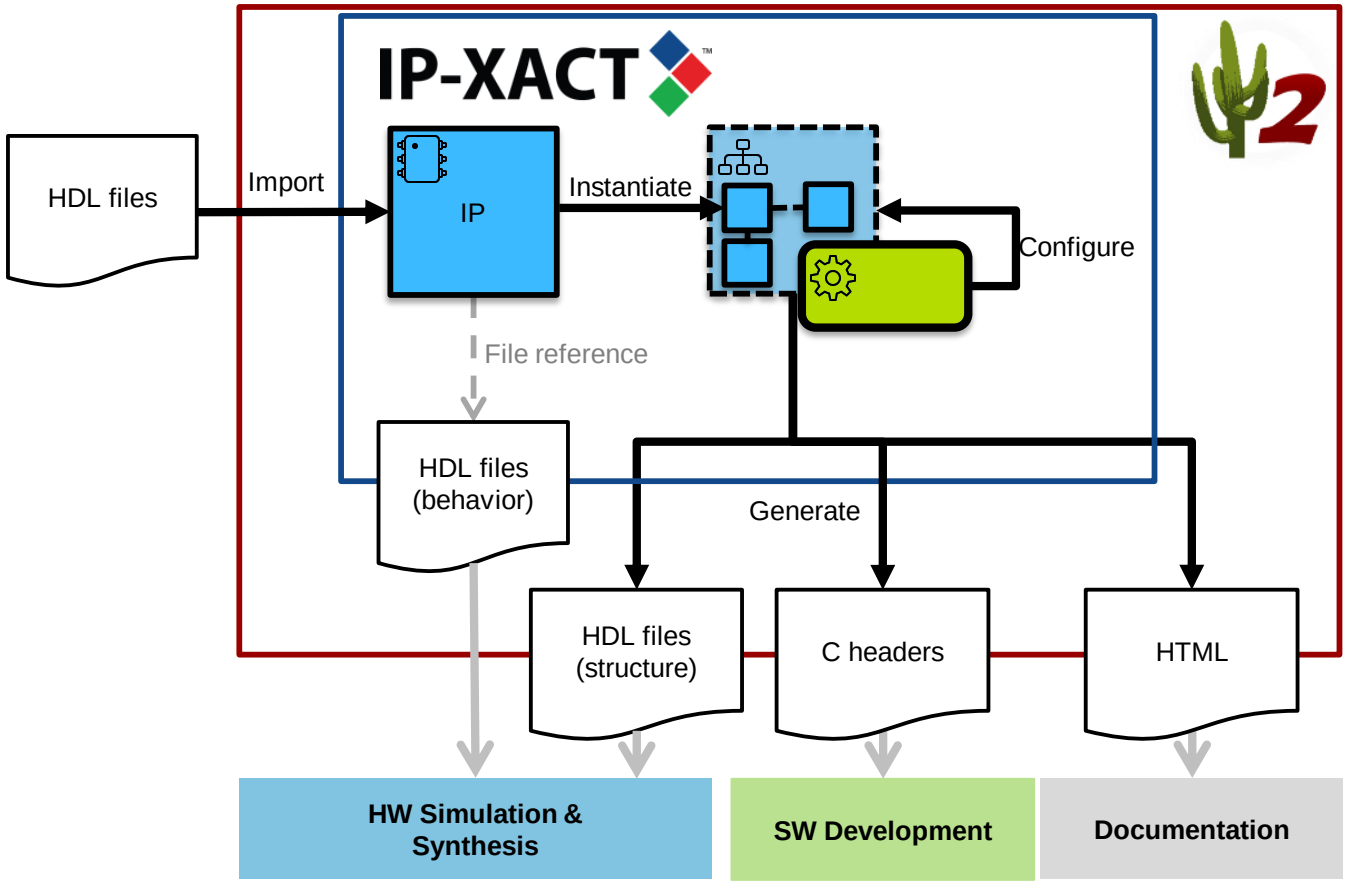
- Standardized XML format for
  - IP-block model, *component*
  - SoC *design* model
  - Integration and configuration flow
  - Tool interfaces
- De-facto standard in the industry
- Vendor, implementation language and tool independent
- Easy reuse of IPs leads to better productivity



Kactus2

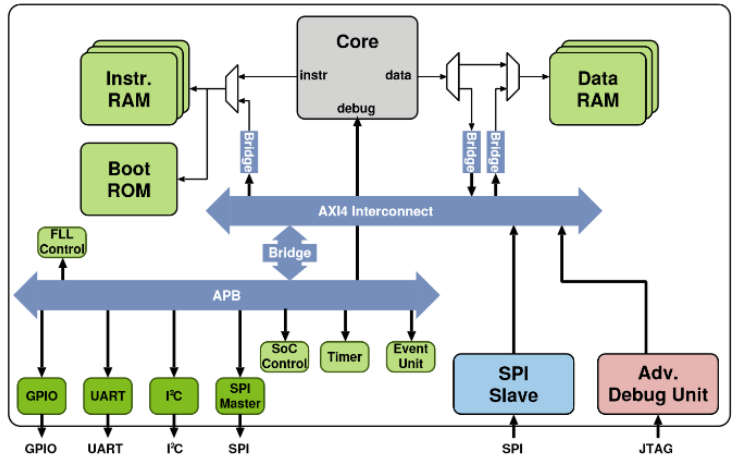
- Open-source IP-XACT based toolset for embedded system design
  - VHDL and Verilog import
  - Generator plugins
- First released in 2011, and has stayed relevant due to
  - standard compatibility,
  - graphical user interface, and
  - usability.

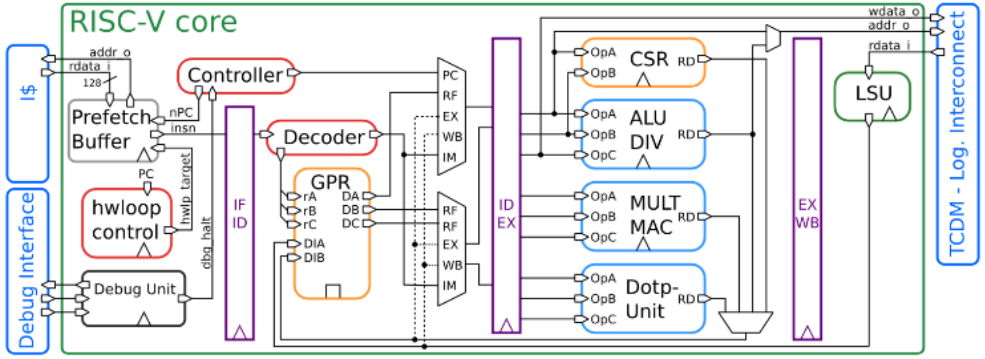
*"For our latest project I did the whole toplevel wiring with K2.
Once the database is in place, it's really impressive how fast thousands of connections can be done with this tool.
I like it!!!"*



PULPino RISC-V microcontroller

- Published open-source
  - 68k lines in 250 HDL files
  - 21 repositories
- Four core configurations:
  - RI5CY: RV32ICM[F] + PULP extensions
  - zero-riscy: RV32ICM
  - micro-riscy: RV32EC

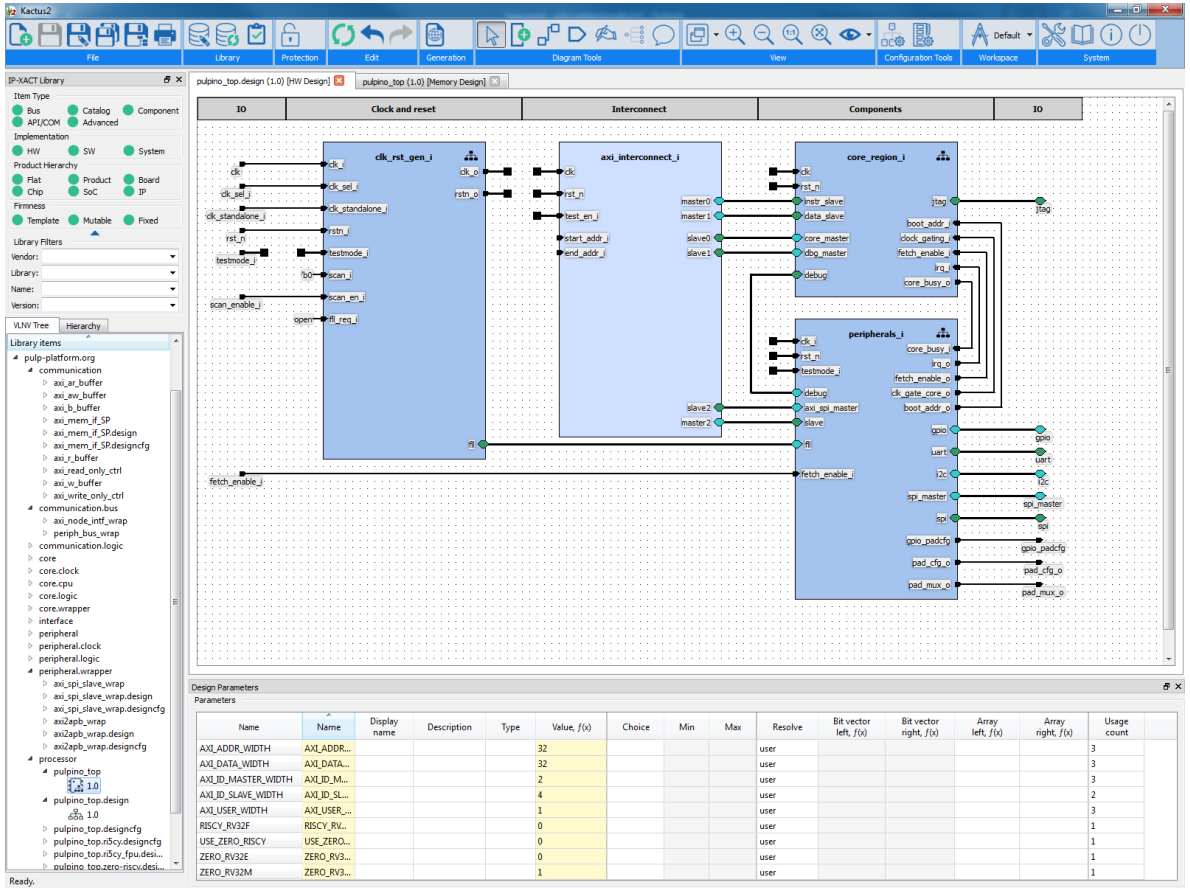




PULPino Datasheet. https://github.com/pulp-platform/pulpino/blob/master/doc/datasheet/datasheet.pdf

PULPino IP-XACT modelling

- Component interface imported from source files
  - Ports and parameters
  - Behavior kept in original sources
- Sub-module instantiations and connectivity created manually
  - Signal and port grouping to buses and *bus interfaces*
- Configuration is done through parameters and component views.
- The IP-XACT model is available at https://github.com/kactus2/pulpinoexperiment
  - 169 IP-XACT files, 95k lines of XML



IP-XACT modelling challenges

Conditional structure

- Configuration value enables/disables part of structure
  - Module instantiation
  - Wires
  - Ports
- Very flexible in HDL
- Inherently IP-XACT structure is (mostly) 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.

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.

Coupling structure and behavior

- Glue logic added for simple manipulation of data 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

Configurable interfaces

- Reusable modules with configurable interface
  - N master and M slave intefaces
- IP-XACT *bus interfaces* are statically defined. Options:
  1. Define maximum set, leave any unused unconnected or use *isPresent* for limits
  2. Automatically generate a *component* with correct number of *bus interfaces* and ports from 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

Current and future work

- Exploring IP-XACT modeling capability within system-level project
  - Targeting for code generation on SoC level
- IP-XACT modeling of PULPissimo microcontroller
  - uDMA for better CPU utilization compared to PULPino
  - SDIO and camera interface peripherals

Conclusion

- Modelling effort benefits from good practices in code and project organization
  - Clear and well-defined interfaces
  - Decoupling of structure and behavior
  - One module per source file
- IP-XACT cannot capture all HDL features
- Modelling work targets for improved
  - understandability,
  - reusability, and
  - productivity.