PULP PLATFORM
Open Source Hardware, the way it should be!

An Open-Source Platform for High-Performance Non-Coherent On-Chip Communication

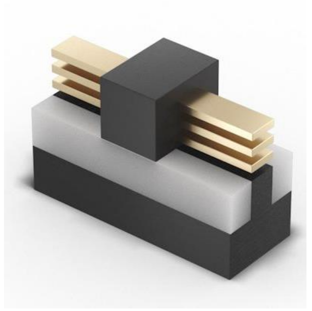
Thomas Benz **<tbenz@iis.ee.ethz.ch>**

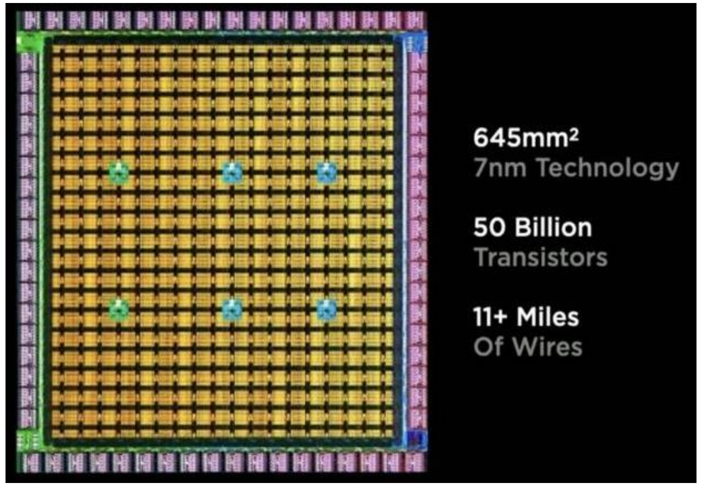
ETH zürich

http://pulp-platform.org @pulp_platform https://www.youtube.com/pulp_platform

Motivation

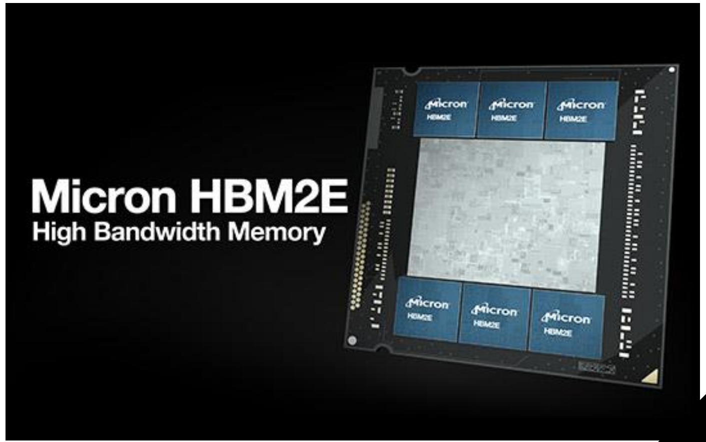
- **Trend towards more complex ICs**
  - larger die sizes
  - feature scaling (Intel 20A)
  - Increasing heterogeneity (ML accelerators)





Tesla D1: 450MiB on-chip SRAM

- **Huge amount of high BW memory**
  - on-chip: SRAM
  - off-chip: HBM2E



➢ **Need for high-BW point-to-point data transfers**

Major on-chip protocols

- Intel: Ultra Path Interconnect
- AMD: Scalable Data Fabric
- IBM: Power9 on-chip interconnect

Not available for third parties
(or only under royalties)

ARM® AMBA®
Interconnect Standards

- ARM: Advanced eXtensible Interface (AXI) (and others)

open standard that can be used without royalties

AXI Implementations

- Synopsys: DesignWare IP Solutions for AMBA Interconnect
- Cadence: VIP only
- ARM: AMBA Products (CoreLink NIC-400, CCI-500, ...)

proprietary, expensive

- Xilinx: LogiCORE IP Products (Interconnect, Data Width Converter, ...)

licensed with Xilinx products, but FPGA only
generated & adapted with IP Integrator

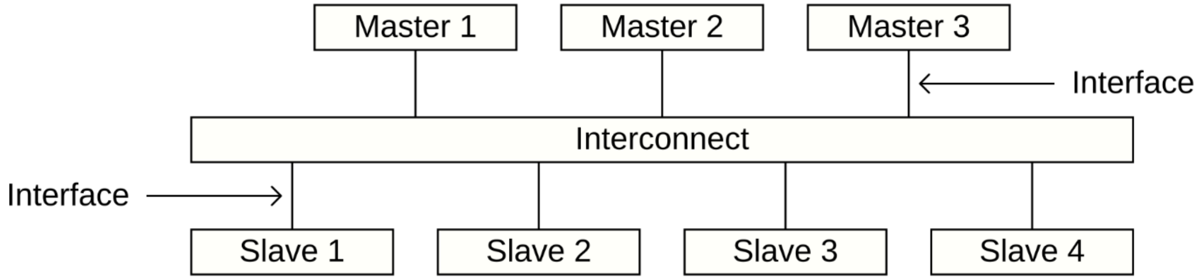
- **FOSS, technology-independent implementation?**

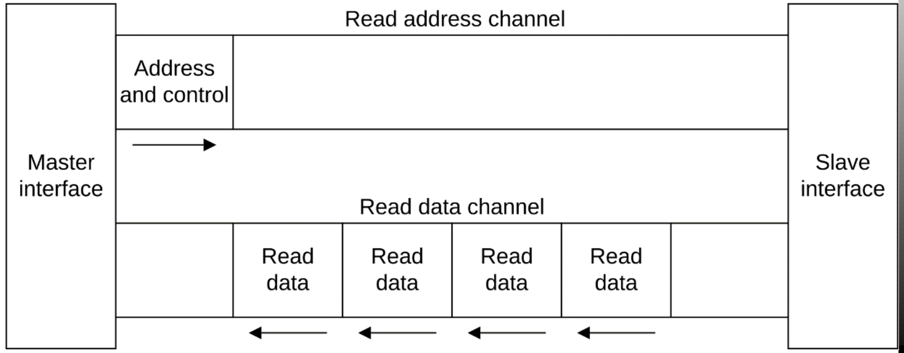
ETH Zurich PULP platform AXI

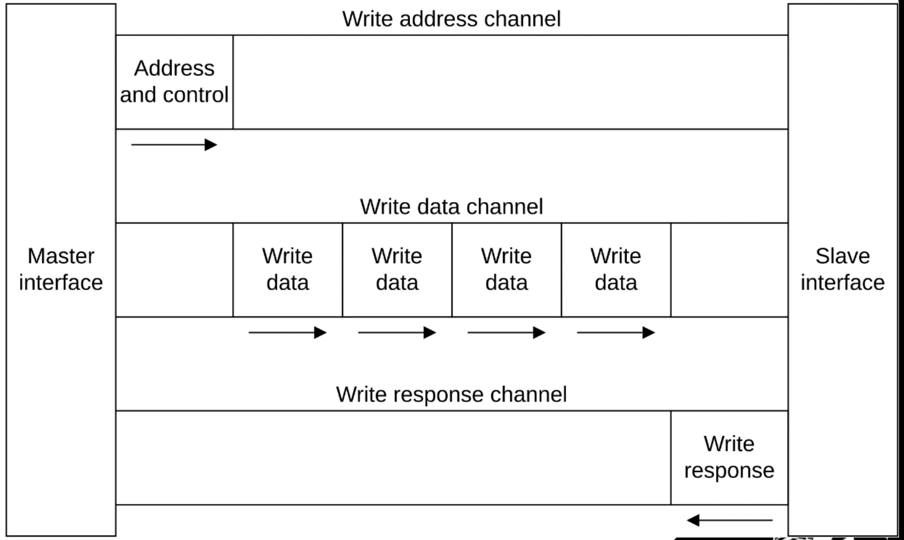
- FOS, technology-independent
- AXI4 and AXI4-Lite synthesizable IPs in SystemVerilog
  - Written and optimized by hand
- Extensive verification infrastructure
  - UVM-compatible
- Full architectural description and extensive documentation
- Fully customizable and extensible
  - User signals are routed
  - Achieve best performance by customizing to the application

AXI Architecture / Terminology

- 5 independent transaction channels
  - Valid, ready, last handshake
- Components
  - Master, slave, interconnect
- Master initiates AXI operation to slave
  - Set of required op. -> transaction
  - Burst of individual data beats

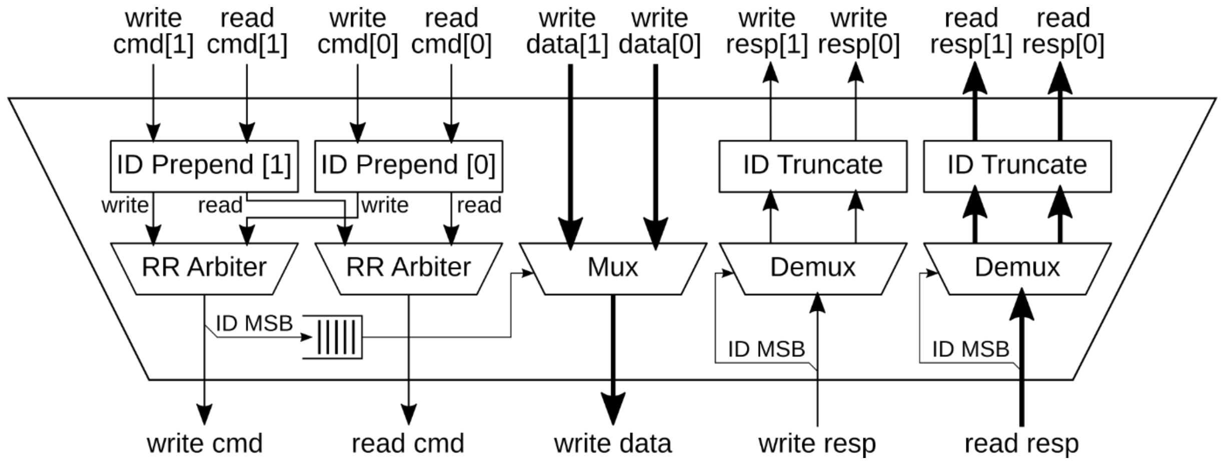






AXI Multiplexer

- Connect multiple slave ports to one master port

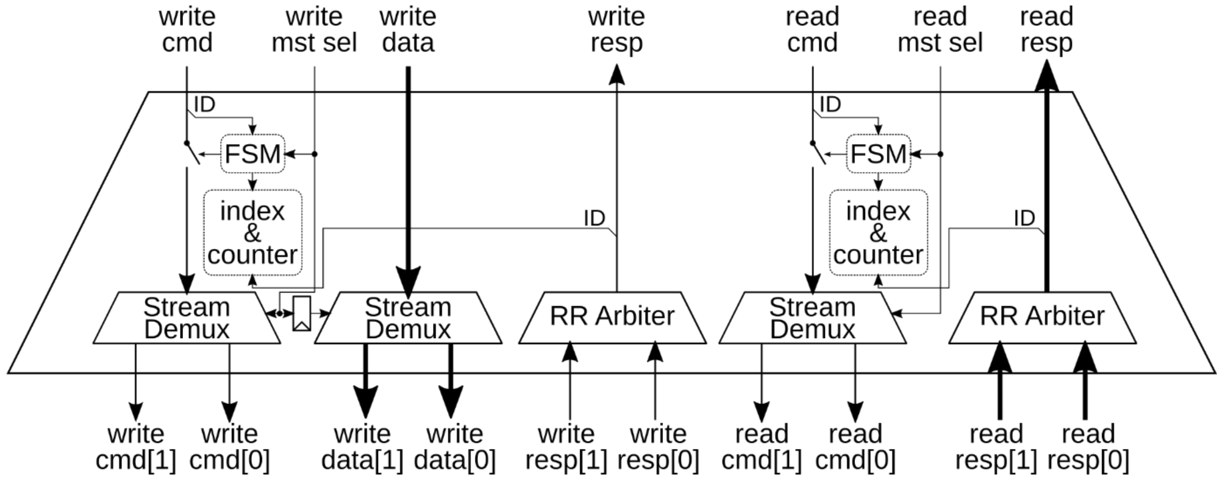


- Operation:
  - Multiplexing forward channel
  - Fair round-robin arbitration
  - Demultiplexing backward channel

- Complexity: backward channel
  - Critical path: $O(log\ S)$ (arbitration)
  - Area: $O(S)$ (arbitration)

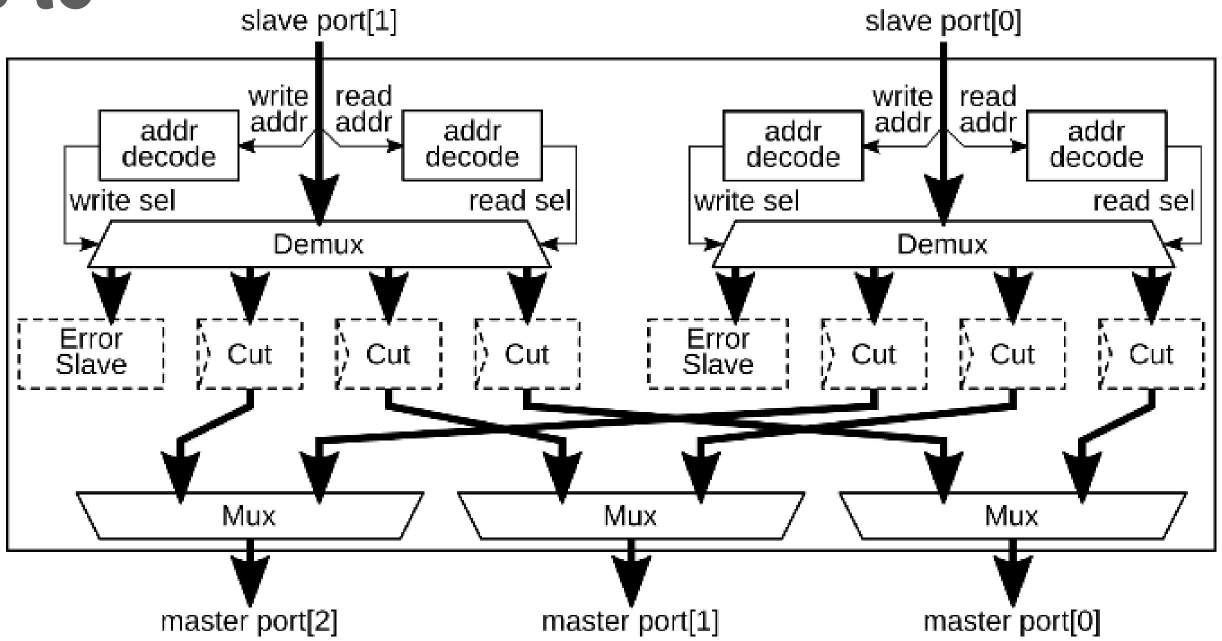
AXI Demultiplexer

- Connects one slave port to multiple master ports
- Operation:
  - Externally select master port
  - Store id information to route reordered responses
- Complexity: keep ordering
  - Critical path: $O(M), O(I)$
  - Area: $O(M), O(2^I)$



AXI Crossbar (X-bar)

- Connects N master ports to M slave ports
- Operation:
  - Address decoding (slave ports)
  - Master selection (demultiplexer)
  - Multiplexing
  - Optionally: add cuts, error slave
- Complexity:
  - Critical path: $O(M + I)$ (demux)
  - Area: $O(MS + 2^I S)$ (S demux, M mux)



Additional Design IPs

- ID remapper and serializer
- Data Upsizer and downsizer
- Simplex and duplex on-chip SRAM controller
- AXI-attached last level cache (LLC)
- Multi-channel AXI DMA engine
- And many more ☺

We have a full set of AXI design IPs to build industry-grade interconnects

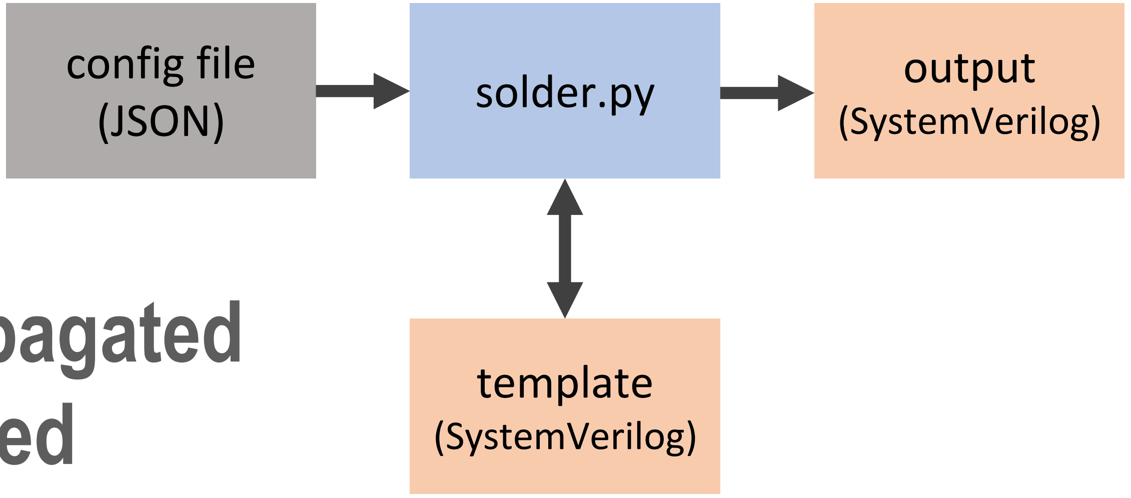
ETH zürich

Building large Systems from our IPs

- Our IPs are written and optimized by hand in SystemVerilog
- Naturally: use SystemVerilog to create the AXI system
  - AXI has many signals -> Tedious and error-prone process
  - We provide a macro-based solution to create AXI types and connect buses
  - ➤Even with SV generate constructs: limit fast exploration
- One solution: Use HLS to describe topology (and generate V code)
  - We would have to throw away our optimized IPs ☹
- Solution: use template-based HLS strategy

Solder: Template & IP-based HLS

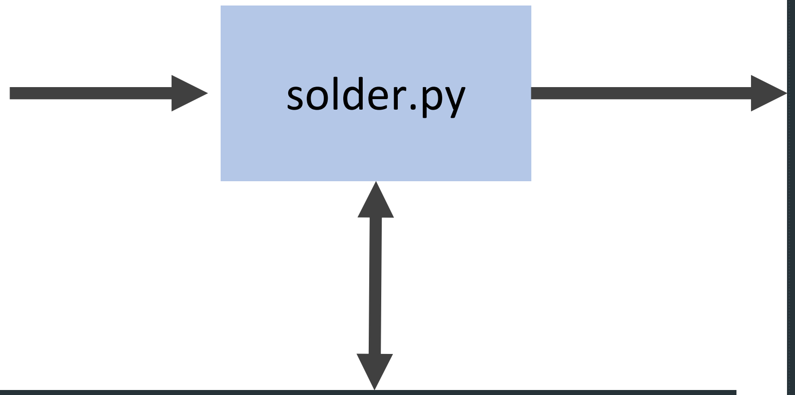
- Python-based application
  - Configuration file for:
    - Parameters, Addresses
    - Other user-defined constants
  - Mako templateing for SV base
- Address maps, routes propagated and sanity checks performed
- Interconnect, SoC, testbench, documentation, linker script, … generation
- Generates understandable (modifiable) SystemVerilog



Example: AXI Adaptation

```
JSON config:
wide data width: 512
narrow data width: 64
…
```

Configuration of fixed values



```
/////////////////////////////
// Wide to Narrow Crossbar //
/////////////////////////////
<%
  soc_wide_xbar.out_soc_narrow \
    .change_iw(context, soc_narrow_xbar.in_soc_wide.iw, "soc_wide_narrow_iwc") \
    .change_dw(context, soc_narrow_xbar.in_soc_wide.dw, "soc_wide_narrow_dw", to=soc_narrow_xbar.in_soc_wide)
%>
```

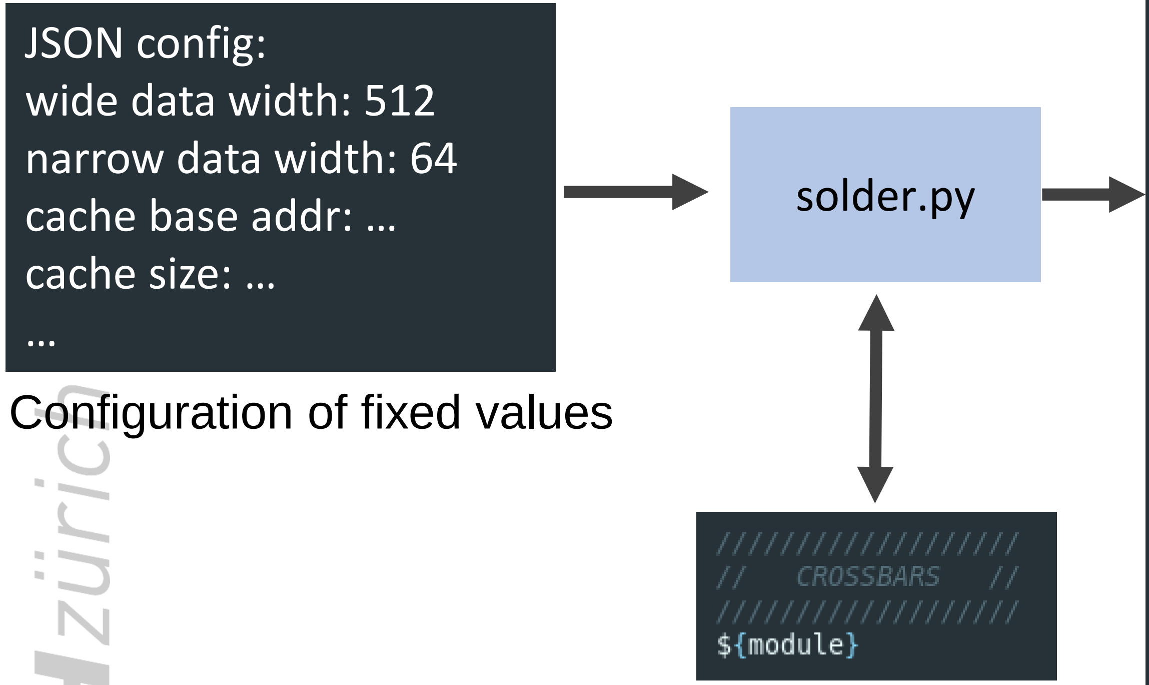
Template: describing connectivity

```
/////////////////////////////
// Wide to Narrow Crossbar //
/////////////////////////////
axi_a48_d512_i4_u0_req_t  soc_wide_narrow_iwc_req;
axi_a48_d512_i4_u0_resp_t soc_wide_narrow_iwc_rsp;

axi_id_remap #(
    .AxiSlvPortIdWidth(9),
    .AxiSlvPortMaxUniqIds(16),
    .AxiMaxTxnsPerId(4),
    .AxiMstPortIdWidth(4),
    .slv_req_t(axi_a48_d512_i9_u0_req_t),
    .slv_resp_t(axi_a48_d512_i9_u0_resp_t),
    .mst_req_t(axi_a48_d512_i4_u0_req_t),
    .mst_resp_t(axi_a48_d512_i4_u0_resp_t)
) i_soc_wide_narrow_iwc (
    .clk_i(clk_i),
    .rst_ni(rst_ni),
    .slv_req_i(soc_wide_xbar_out_req[SOC_WIDE_XBAR_OUT_SOC_NARROW]),
    .slv_resp_o(soc_wide_xbar_out_rsp[SOC_WIDE_XBAR_OUT_SOC_NARROW]),
    .mst_req_o(soc_wide_narrow_iwc_req),
    .mst_resp_i(soc_wide_narrow_iwc_rsp)
);
axi_dw_converter #(
    .AxiSlvPortDataWidth(512),
    .AxiMstPortDataWidth(64),
    .AxiAddrWidth(48),
    .AxiIdWidth(4),
    .aw_chan_t(axi_a48_d64_i4_u0_aw_chan_t),
    .mst_w_chan_t(axi_a48_d64_i4_u0_w_chan_t),
    .slv_w_chan_t(axi_a48_d512_i4_u0_w_chan_t),
    .b_chan_t(axi_a48_d64_i4_u0_b_chan_t),
    .ar_chan_t(axi_a48_d64_i4_u0_ar_chan_t),
    .mst_r_chan_t(axi_a48_d64_i4_u0_r_chan_t),
    .slv_r_chan_t(axi_a48_d512_i4_u0_r_chan_t),
    .axi_mst_req_t(axi_a48_d64_i4_u0_req_t),
    .axi_mst_resp_t(axi_a48_d64_i4_u0_resp_t),
    .axi_slv_req_t(axi_a48_d512_i4_u0_req_t),
    .axi_slv_resp_t(axi_a48_d512_i4_u0_resp_t)
) i_soc_wide_narrow_dw (
    .clk_i(clk_i),
    .rst_ni(rst_ni),
    .slv_req_i(soc_wide_narrow_iwc_req),
    .slv_resp_o(soc_wide_narrow_iwc_rsp),
    .mst_req_o(soc_narrow_xbar_in_req[SOC_NARROW_XBAR_IN_SOC_WIDE]),
    .mst_resp_i(soc_narrow_xbar_in_rsp[SOC_NARROW_XBAR_IN_SOC_WIDE])
);
```

Generated SystemVerilog Code

Example: AXI X-Bar



```
JSON config:
wide data width: 512
narrow data width: 64
cache base addr: …
cache size: …
…
```

Configuration of fixed values

solder.py

```
///////////////////
//   CROSSBARS   //
///////////////////
${module}
```

Template

```
///////////////////
//   CROSSBARS   //
///////////////////

/// Address map of the `wide_xbar_quadrant_s1` crossbar.
xbar_rule_48_t [4:0] WideXbarQuadrantS1Addrmap;
assign WideXbarQuadrantS1Addrmap = '{
'{ idx: 1, start_addr: 48'h10000000000, end_addr: 48'h20000000000 },
'{ idx: 2, start_addr: cluster_base_addr[0], end_addr: cluster_base_addr[0] + ClusterAddressSpace },
'{ idx: 3, start_addr: cluster_base_addr[1], end_addr: cluster_base_addr[1] + ClusterAddressSpace },
'{ idx: 4, start_addr: cluster_base_addr[2], end_addr: cluster_base_addr[2] + ClusterAddressSpace },
'{ idx: 5, start_addr: cluster_base_addr[3], end_addr: cluster_base_addr[3] + ClusterAddressSpace }
};

wide_xbar_quadrant_s1_in_req_t   [4:0] wide_xbar_quadrant_s1_in_req;
wide_xbar_quadrant_s1_in_resp_t  [4:0] wide_xbar_quadrant_s1_in_rsp;
wide_xbar_quadrant_s1_out_req_t  [5:0] wide_xbar_quadrant_s1_out_req;
wide_xbar_quadrant_s1_out_resp_t [5:0] wide_xbar_quadrant_s1_out_rsp;

axi_xbar #(
    .Cfg          (WideXbarQuadrantS1Cfg),
    .Connectivity (30'b011111101111110111111011111110),
    .AtopSupport  (0),
    .slv_aw_chan_t(axi_a48_d512_i4_u0_aw_chan_t),
    .mst_aw_chan_t(axi_a48_d512_i7_u0_aw_chan_t),
    .w_chan_t     (axi_a48_d512_i4_u0_w_chan_t),
    .slv_b_chan_t (axi_a48_d512_i4_u0_b_chan_t),
    .mst_b_chan_t (axi_a48_d512_i7_u0_b_chan_t),
    .slv_ar_chan_t(axi_a48_d512_i4_u0_ar_chan_t),
    .mst_ar_chan_t(axi_a48_d512_i7_u0_ar_chan_t),
    .slv_r_chan_t (axi_a48_d512_i4_u0_r_chan_t),
    .mst_r_chan_t (axi_a48_d512_i7_u0_r_chan_t),
    .slv_req_t    (axi_a48_d512_i4_u0_req_t),
    .slv_resp_t   (axi_a48_d512_i4_u0_resp_t),
    .mst_req_t    (axi_a48_d512_i7_u0_req_t),
    .mst_resp_t   (axi_a48_d512_i7_u0_resp_t),
    .rule_t       (xbar_rule_48_t)
) i_wide_xbar_quadrant_s1 (
    .clk_i                  (clk_i),
    .rst_ni                 (rst_ni),
    .test_i                 (test_mode_i),
    .slv_ports_req_i        (wide_xbar_quadrant_s1_in_req),
    .slv_ports_resp_o       (wide_xbar_quadrant_s1_in_rsp),
    .mst_ports_req_o        (wide_xbar_quadrant_s1_out_req),
    .mst_ports_resp_i       (wide_xbar_quadrant_s1_out_rsp),
    .addr_map_i             (WideXbarQuadrantS1Addrmap),
    .en_default_mst_port_i('1),
    .default_mst_port_i     ('0)
);
```

Generated SystemVerilog Code

Future Development: Fast Exploration

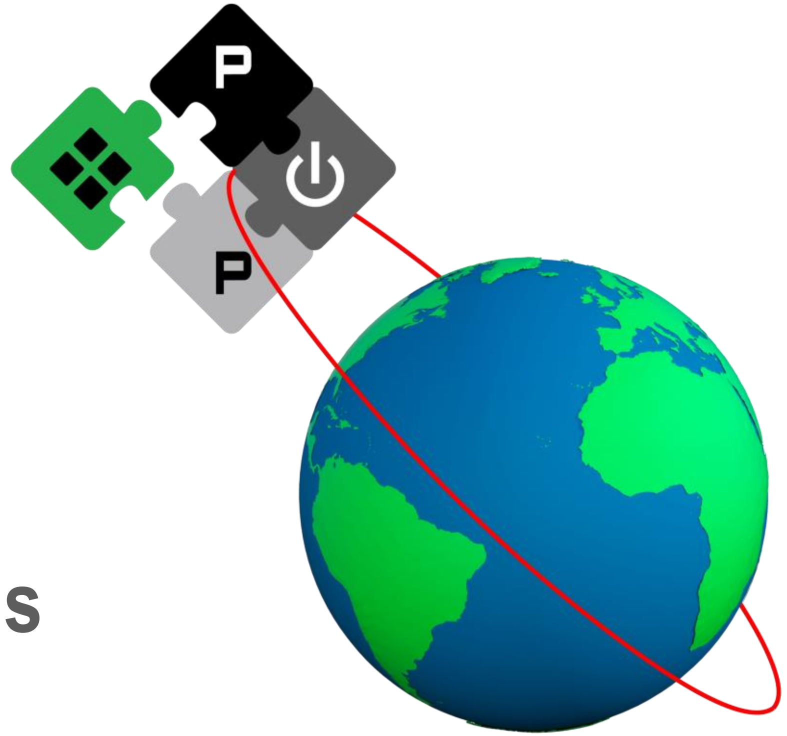
- Solder generates the Interconnect from
  - Config file and SystemVerilog template

➢It has the full overview of the instantiated AXI IPs and their configuration

- From fitted models it is possible:
  - Estimate the critical path of each IP
  - Estimate the size of each IP
- Estimate timing and area of full AXI system
  - ➢Do ultra fast (automated) exploration

Future Development: AXI Extensions

- **Error correction**
  - Space-grade applications
  - Use redundant data (e.g. parity)
  - User Signals
  - Create / check integrity at source / sink
- **Memory stream-based extensions**
  - Custom burst types
  - Encode more complex memory streams
    - N-D transfers with regular strides
    - Scatter / gather operations with arbitrary memory streams



PULP in Space

Conclusion

- We created a synthesizable FOS AXI4 implementation, that can compete with industry-grade solutions
  - Check it out at https://github.com/pulp-platform/axi
- Our AXI4 implementation is written in SystemVerilog, optimized by hand and fully characterized
- We have a template-based HLS approach to create SoCs and their AXI4 interconnects
  - Check out our reference RISCV system: https://github.com/pulp-platform/snitch