



SYNOPSYS® 新思

Optimize Your Own RISC-V Architecture, Using Application-Specific Processor Design Tools: Synopsys ASIP Designer

毛海雪, 新思科技解决方案事业部, 资深应用工程师

August 25, 2023

RISC-V Extensibility

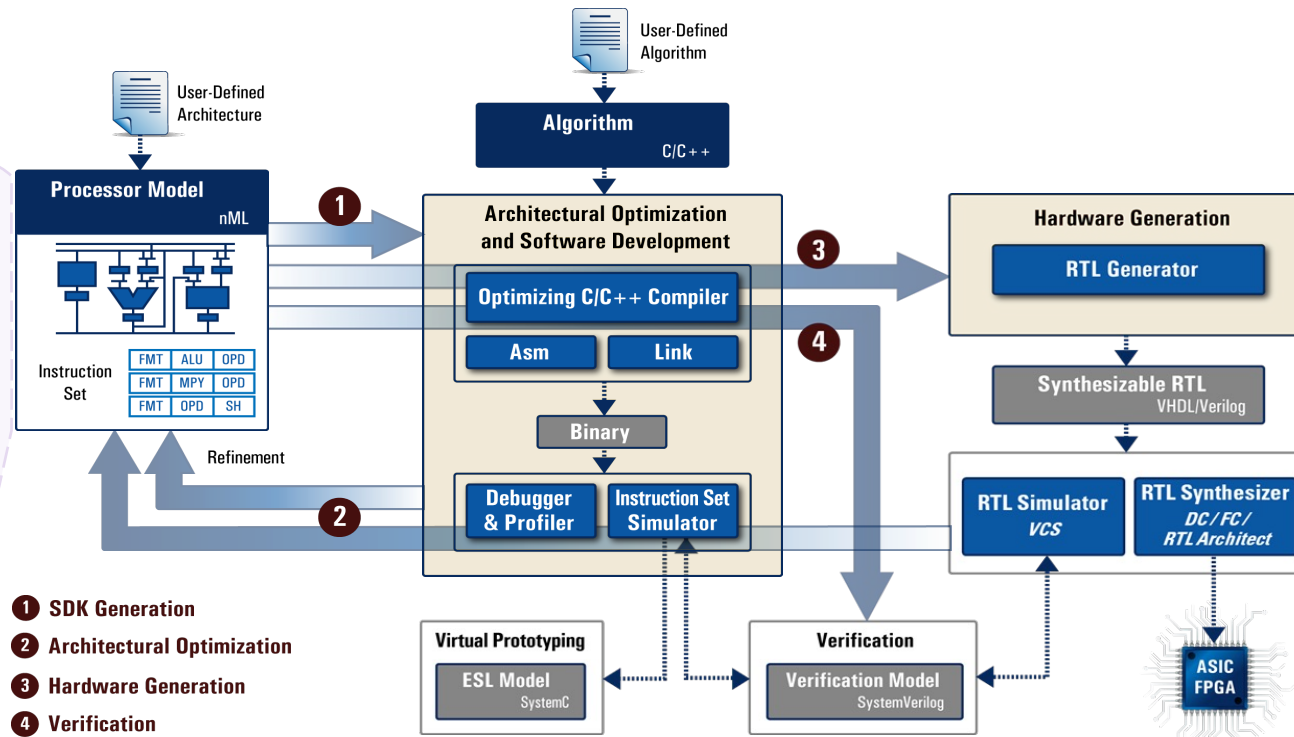
- ISA customization and extensibility are drivers for the growing adoption of RISC-V
 - This results in ASIPs with a RISC-V baseline ISA
 - Preserve RISC-V compatibility
 - Execute SW code and libraries
 - Reuse HW peripherals
- } Designed for general-purpose RISC-V
- Challenges
 - Which extensions are best for the target application domain?
 - How to obtain a high-quality SW Development Kit (SDK), including an optimizing compiler?
 - How to obtain a reliable RTL implementation with excellent PPA?
 - How to verify the design?

ASIP Designer™

```

start trv32p5x;
opn trv32p5x(bit32_ifmt | bit16_ifmt);
opn bit32_ifmt(majOP | majOP_IMM | majLOAD | ... | majCUSTOM3);
opn majOP(alu_rrr_ar_instr | mpy_rrr_instr | div_instr);
opn alu_rrr_ar_instr(op: majOP_fn10, rd: eX, rs1: eX, rs2: eX){
  action{
    stage ID:
      pidX1 = r1 = X[rs1];
      pidX2 = r2 = X[rs2];
    stage EX:
      aluA = pidX1;
      aluB = pidX2;
      switch(op){
        case add: aluR = add (aluA,aluB) @alu;
        case sub: aluR = sub (aluA,aluB) @alu;
        case slt: aluR = slt (aluA,aluB) @alu;
        case sltu: aluR = sltu(aluA,aluB) @alu;
        case xor: aluR = bxor(aluA,aluB) @alu;
        ...
        case sra: aluR = sra (aluA,aluB) @alu;
      }
    stage EX:
      pexX1 = texX1 = aluR;
    stage ME:
      pmexX1 = tmeX1 = pexX1;
    stage WB:
      if (rd: x0) w1_dead = w1 = pmexX1;
      else X[rd] = w1 = pmexX1;
  }
  syntax : "neg " rd "," rs2 op<<sub>> rs1<<x0>>
    | "snez " rd "," rs2 op<<sltu>> rs1<<x0>>
    | "sltz " rd "," rs1 op<<slt>> rs2<<x0>>
    | "sgtz " rd "," rs2 op<<slt>> rs1<<x0>>
    | op " " rd "," rs1 "," PADOP2 rs2;
  image : op[9..3]::rs2::rs1::op[2..0]::rd, class(alu_rrr);
}
...

```



- Industry-leading tool to design your own Application-Specific Instruction-set Processor (ASIP)
- Language-based description of ISA and microarchitecture: nML
- Single processor model ensures that SDK and RTL are in sync
- Architectural exploration with Compiler-in-the-Loop™ & Synthesis-in-the-Loop™
- Licensed as a tool (not IP): product differentiation, no royalties
- Full interoperability with other Synopsys EDA tools

Trv (RISC-V ISA) Models

Shipped with ASIP Designer

Trv**32****p3****xf**

└─ + single-precision floating-point ^{1) 2)}
└─ + HW loop + post-mod + 2-way ILP ¹⁾
└─ pipeline depth: p3 or p5
└─ word length: 32 or 64

¹⁾ optional
²⁾ Trv32 only

Integer models: Trv<mm>p<n>

	32-bit datapath	64-bit datapath
3-stage pipeline	Trv32p3 Trv32p3x	Trv64p3 Trv64p3x
5-stage pipeline	Trv32p5 Trv32p5x	Trv64p5 Trv64p5x

- ISA: RV64IM, RV32IM
 - Integer and multiply instructions
- Micro architecture
 - Protected pipeline, 3 or 5 stages
 - Hardware multiplier
 - Iterative divider
- Optional extensions: Trv<mm>p<n>**x**
 - Two-way static ILP
 - Zero overhead hardware loops
 - Load/stores with post-modify addressing

Floating-point models: Trv32p<n>**f**

	32-bit datapath
3-stage pipeline	Trv32p3f Trv32p3fx
5-stage pipeline	Trv32p5f Trv32p5fx

- ISA: RV32IMZfinx
 - Integer and multiply instructions
 - Single precision float instructions
- Micro architecture
 - Protected pipeline, 3 or 5 stages
 - FPU models based on HardFloat [Hauser]
 - Iterative divider and square-root units
- Optional extensions: Trv32p<n>**fx**
 - Two-way static ILP
 - Zero overhead hardware loops
 - Load/stores with post-modify addressing

Trv Models With Simple Datapath eXtensions (SDX)

Concept

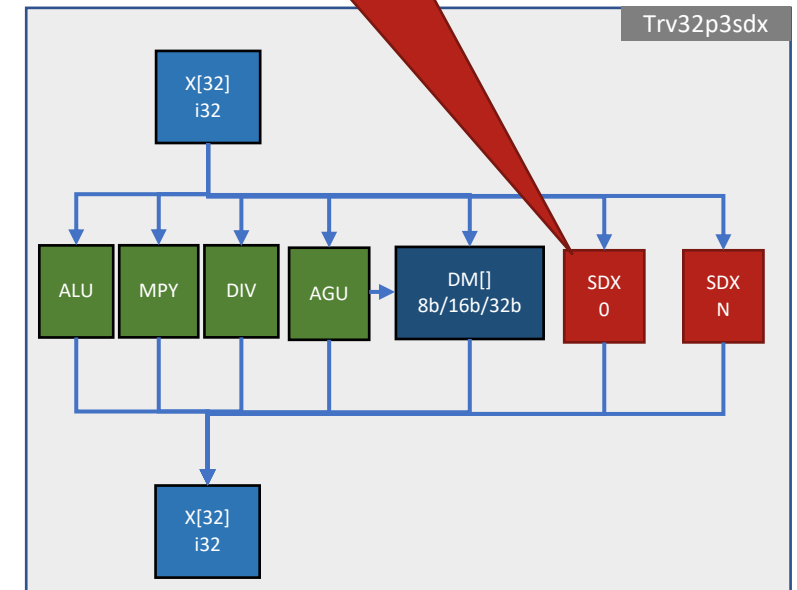
- SDX is a mechanism to add simple extension instructions to Trv (RISC-V)
 - nML model of Trv32p3 with predefined **stubs** for extension instructions
 - User codes the **behavior** of the stubs in PDG (bit-accurate C code)
 - Compiler intrinsics that target the extension instructions are provided (and can be modified)
- Benefits
 - No (deep) nML knowledge required: extensions can be created by SW engineers
 - Fast exploration of extensions with compiler-in-the-loop & synthesis-in-the-loop
- Extensions encoded in RISC-V custom-2 space
- Operand options
 - 3-register (32 and 64-bit)
 - Accumulate
 - Additional single register inputs and outputs

Behavioral model (PDG)

```
w32 sdx0(w32 a, w32 b)
{
  w32 r;
  r[15:0] = a[15:0] + b[15:0];
  r[31:16] = a[31:16] + b[31:16];
  return r;
}
```

Compiler intrinsics


```
int sdx0(int,int);
int add2(int,int);
```

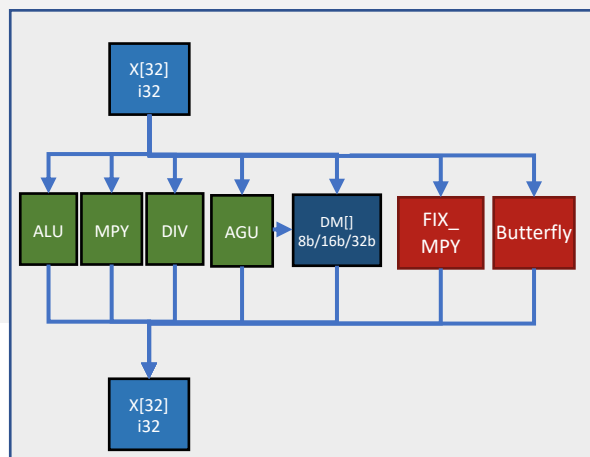


Trv Models With Simple Datapath eXtensions (SDX)


SDX Examples Provided With ASIP Designer

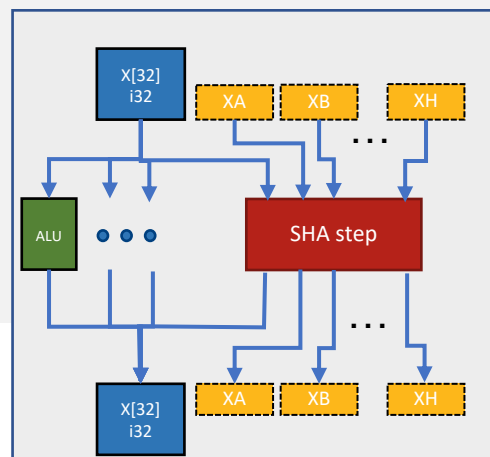
Example: FFT

- SDX instructions accelerating
 - Complex fixed-point multiplication & scaling
`sdx1 rd,rs1,rs2`
 - ABS(x) function: `sdx2 rd, rs1, rs2(x0)`
 - FFT Butterfly: `sdx5 rd,rs1,rs1`
- Specialization:
 - Fractional data types
 - Complex numbers (16bit/16bit -> 32bit register)
- Speedup: 280% Area increase: 31% 

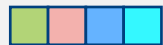



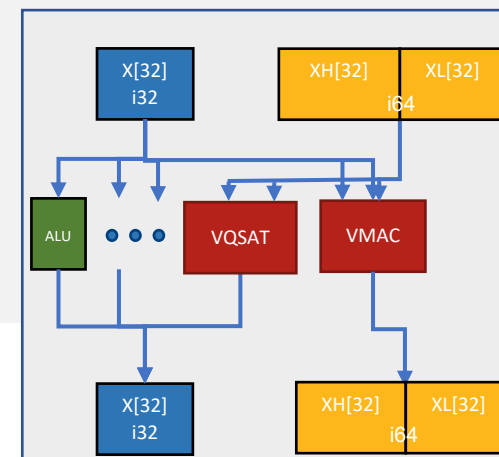
Example: SHA256

- Computes a hash of message W using bitwise AND, OR, XOR operations, shift operations and additions
- Custom data path is ideal to implement the complex hash function in one instruction
- Additional state of the hash functions (8 state variables) require an SDX variant that supports **8 additional register reads and writes**
`sdx7 rd, rs1, rs2, x24,x25,..., x32`
- Speedup: 270% Area increase: 16% 



Example: Keyword Spotting

- Based on small sized Neural Network (3.3M MACs)
- SDX architecture feature: **packed SIMD**
 - 32-bit register contains vector of 4x 8-bit values 
 - Use of register pairs, enabling 64-bit access
`sdx4a_dr dd,rs1,rs2,mode // vmac`
`sdx0_dd rd, ds1,ds2 // vqsat`
- Speedup: 1160% Area increase: 16% 



Large-Scale Trv Extensions

Example: “Tmoby” ASIP for Acceleration of MobileNet v3

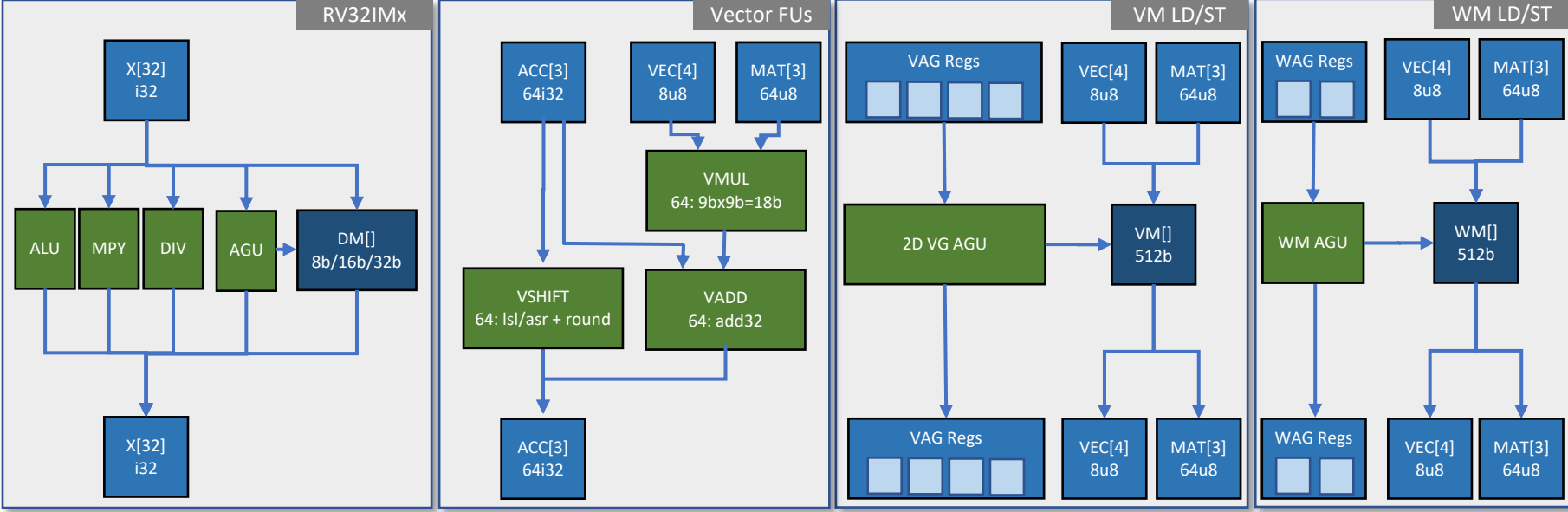
- Acceleration through instruction-level parallelism, SIMD, and/or specialization
- By direct editing of any Trv model
- Typically results in VLIW architecture with a RISC-V scalar issue slot

Application: MobileNet v3

- Accelerated kernels:
 - Conv_2D (pointwise)
 - Depthwise_Conv_2D
 - Add
 - Average_Pool_2D
 - Softmax

tmoby			
sc: bit32_ifmt	vc: slot_vc	vm: slot_vm	wm: slot_wm
majOP	0110011	pnop_instr	pnop_instr
majOP_IMM	0010011	x x xxxlxxxll0xxxll0	xxxxxxxxxxxxxxxxxxxxxxxxl x x x xxxlxxxll
majLOAD	0000011	vec_move	01xxxll0
majSTORE	0100011	vec_ext_upd	00 agv_standalone xxxxxxxxxxxx01agw_standalonexxx01
majBRANCH	1100011	vec_bool	00xxxll0
majJAL	1101111	vec_bc	xxx0xll0xxxll0
majJALR	1100111	vec_mac	01
majLUI	0110111	mux_trans_add_instr	xxx010
majAUIPC	0010111	vec_sh	x010xxxll0
majCUSTOM0	0001011	vec_misc	xxx0xxxll0xxxll0
majCUSTOM1	0101011		
majCUSTOM3	1111011		

- VLIW extension of Trv32p3
 - 90-bit instruction word
 - 4-way ILP
 - Scalar slot: Trv32p3
 - Vector arithmetic slot: SIMD64 MAC 8x8→32
 - 2 vector load/store slots:
 - VM: features
 - WM: weights
 - Vector addressing
- 360x cycle count decrease



ASIP Designer Tight Integration with Synopsys' Flows

Automatic Generation of Interfaces Reduces Overall Time-to-Results

DesignWare Libraries

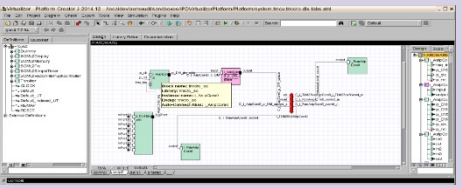
Datapath: Floating Point (Overview)	
DW_fp_add	Floating-Point Adder
DW_fp_addsub	Floating-Point Adder/Subtractor
DW_fp_addsub_DG	Floating-Point Adder/subtractor with Datapath Gating
DW_fp_add_DG	Floating-Point Adder with Datapath Gating
DW_fp_cmp	Floating-Point Comparator

Library components can be incorporated in ASIP processor model

Zebu

Playback in ASIP Designer-generated ISS, using Zebu-generated tracing info

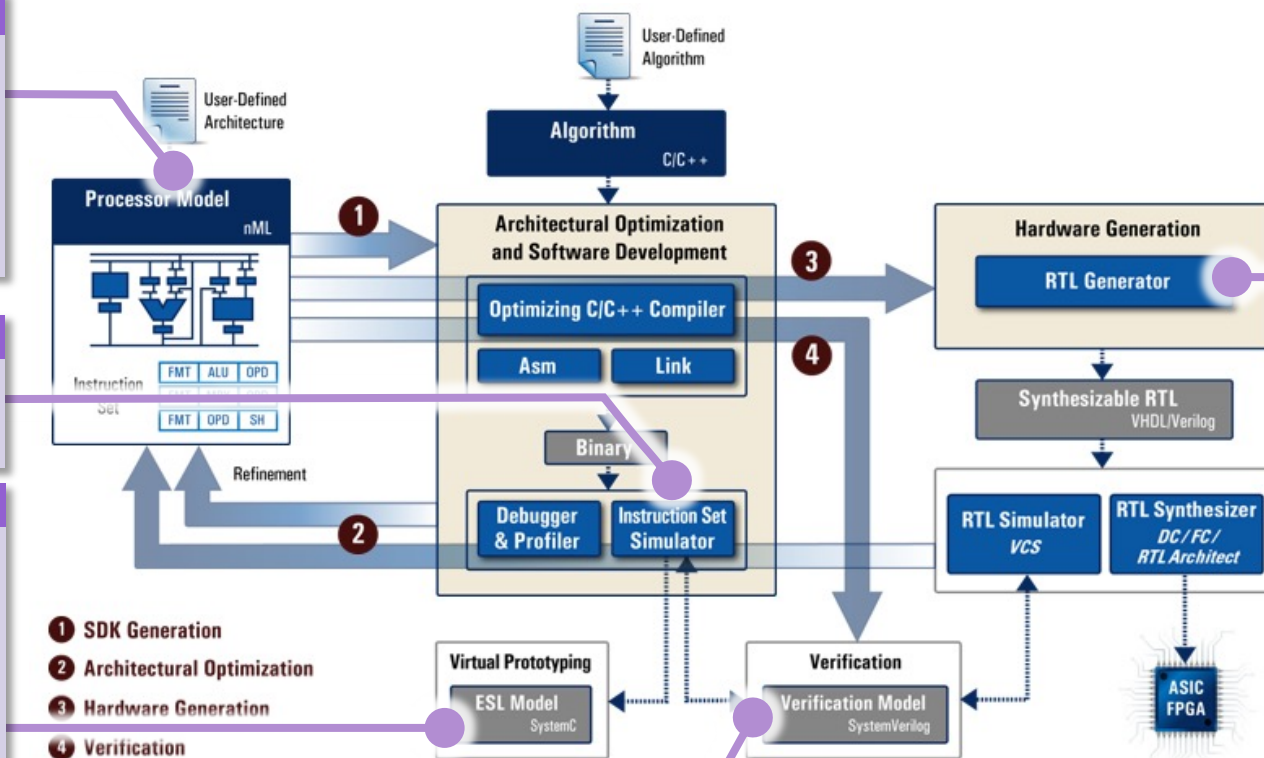
Virtualizer & Platform Architect



SoC architecture design and pre-silicon SW development, using SystemC wrapped ISS of ASIP

VC Formal

Verification of SystemVerilog properties generated by ASIP Designer



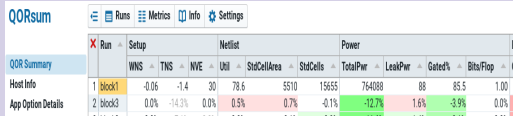
Used by 7 of the Top 10 Semiconductor Developers

Coverity
Static check of ASIP Designer source code base ensures tool quality

Design Compiler & Fusion Compiler

Implementation of ASIP Designer-generated RTL, with best PPA

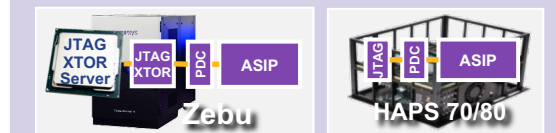
RTL Architect



QORsum		WNS	TNS	NVE	Unit	StatCellArea	StatCells	TotalPwr	LeakPwr	Gated%	BitsFlop	Route
Host Info	1 block	-0.06	-1.4	30	78.6	5510	15455	764088	88	85.5	1.00	
App Option Details	2 block	0.0%	-14.3%	0.0%	0.5%	0.7%	-0.1%	-12.2%	1.6%	-3.9%	0.0%	
Timeline (1/100)	3 block	0.0%	7.1%	3.3%	-0.3%	-0.1%	-0.9%	-11.6%	-1.4%	-2.1%	0.0%	

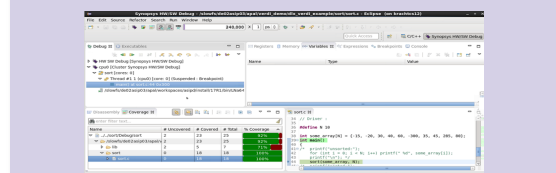
Fast and accurate PPA estimation of ASIP Designer-generated RTL

Zebu & HAPS



High-speed verification and FPGA prototyping with SW debug support, of ASIP Designer-generated RTL

Verdi HW/SW Debug



Debugger connected to RTL simulator with playback support

Spyglass

Linting of ASIP Designer-generated RTL

SYNOPSYS® · 新思

Thank You