High-Level Digital System Design with COSIDE® using MatchLib and Catapult® HLS

Petri Solanti, Senior Field Application Engineer
Siemens EDA, a part of Siemens Digital Industries Software

SIEMENS

Agenda

Challenges in Designing AI/ML Hardware

AI Accelerator Development

Analyzing and Optimizing System Performance

Implementing Accelerator HW with High-Level Synthesis

Conclusions

SIEMENS

AI/ML Application Challenges

Algorithmic Complexity

- Growing faster than the ability of RTL designers to code and verify

Memory Architecture Complexity

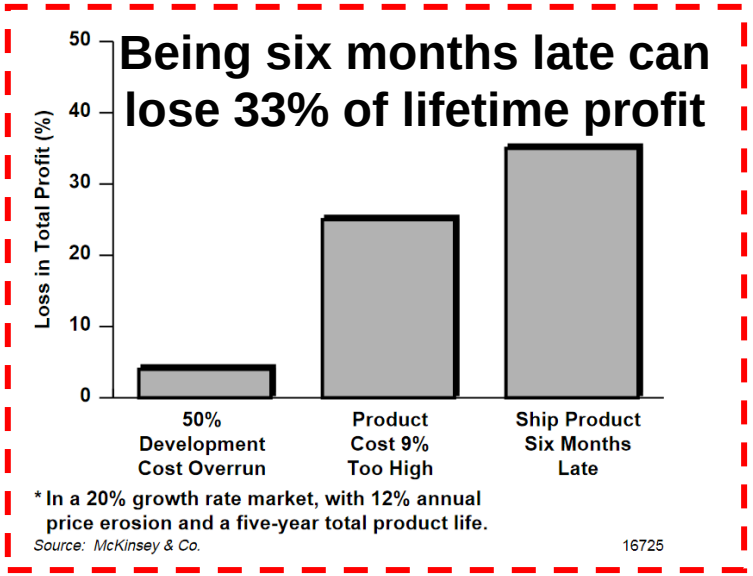
- Efficient data movement is key for power, performance and area

RTL Verification Costs Increasing

- Increased design complexity increases bugs introduced during hand-coding of RTL
- RTL regressions involve server farms, electricity cost, licenses and time

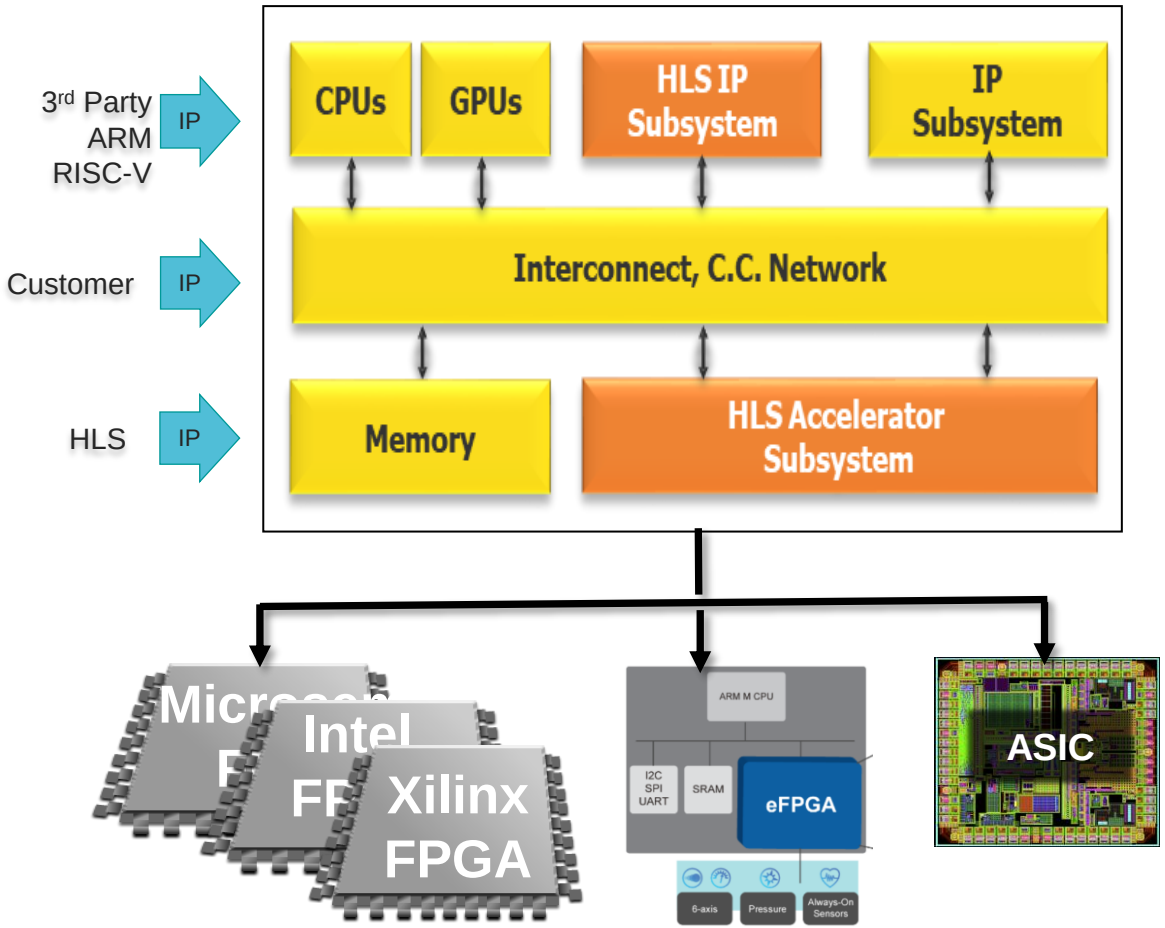
Slips in Design Schedule Kills Total Profit

- Finding bugs during system integration is too late



SIEMENS

Designing HW Accelerator MUCH Faster @ C-Level with HLS



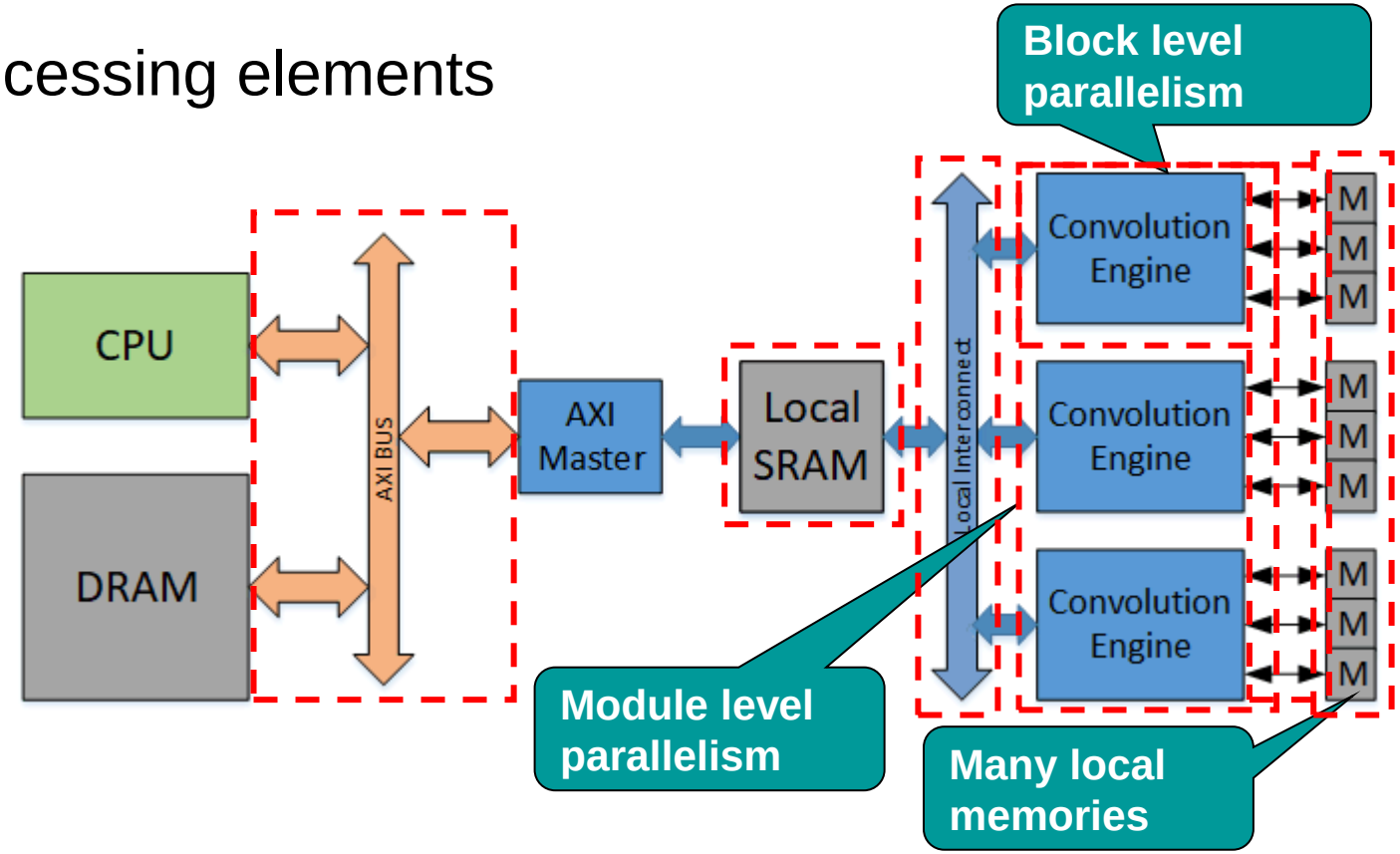
Building a HW Accelerator

- Design it faster and easier from Algorithm
- Get the architecture right with exploration
- Validate system-level performance early
- Plug into system-level design flow – HW/SW/Analog
- Faster verification at the C-level
- Predictable and quality closure for RTL

SIEMENS

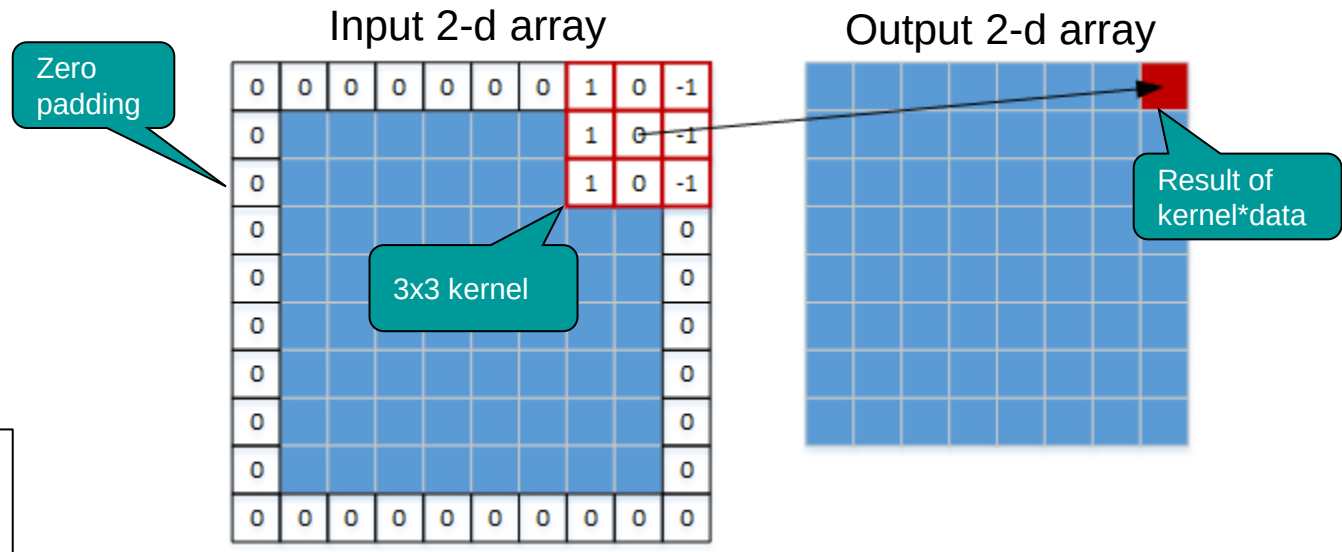
CNN Architectural Challenges

- Memory architectures need to leverage data reuse and parallelism
- May have multiple engines or processing elements
  - Block level parallelism
  - Module level parallelism
- Many local memories
- Complex interconnect
  - AXI4, local interconnect, etc.



Example: Implementing 2-d 3x3 Image** Convolution Algorithm

- 2-d kernel
- 2-d input array
- Padding left, right, and top and bottom
- Stride = number of elements kernel jumps by

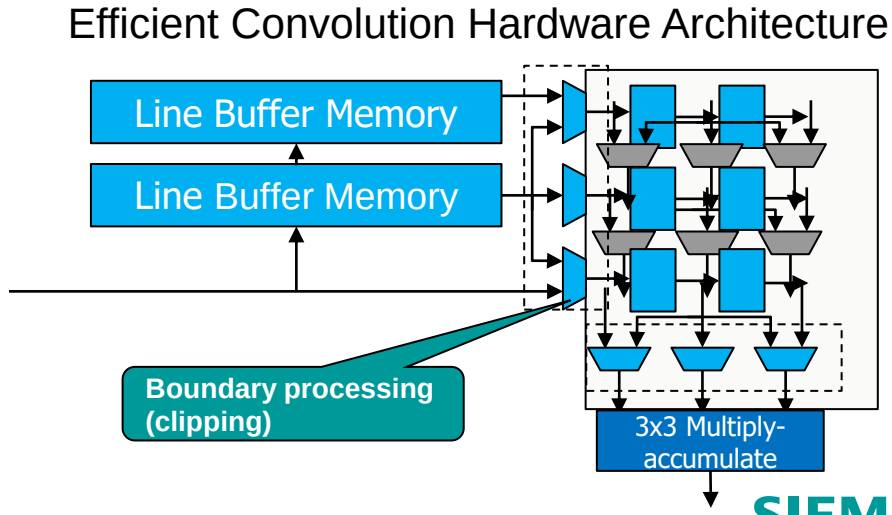


2-d Convolution Algorithm – stride 1, zero pad

```
HEIGHT:for(int r=0;r<IN_HEIGHT;r++){
  WIDTH:for(int c=0;c<IN_WIDTH;c++){
    KERNEL_R:for(int i=0;i<3;i++){
      KERNEL_C:for(int j=0;j<3;j++){
        if(r+i-1 < 0 || r+i-1 >= IN_HEIGHT)
          data = 0;//zero pad
        else if(c+j-1 < 0 || c+j-1 >= IN_WIDTH)
          data = 0;//zero pad
        else
          data = data_in[r+i-1][c+j-1];
        acc += data * kernel[i][j];
  }
   result[c] = acc;
}
```

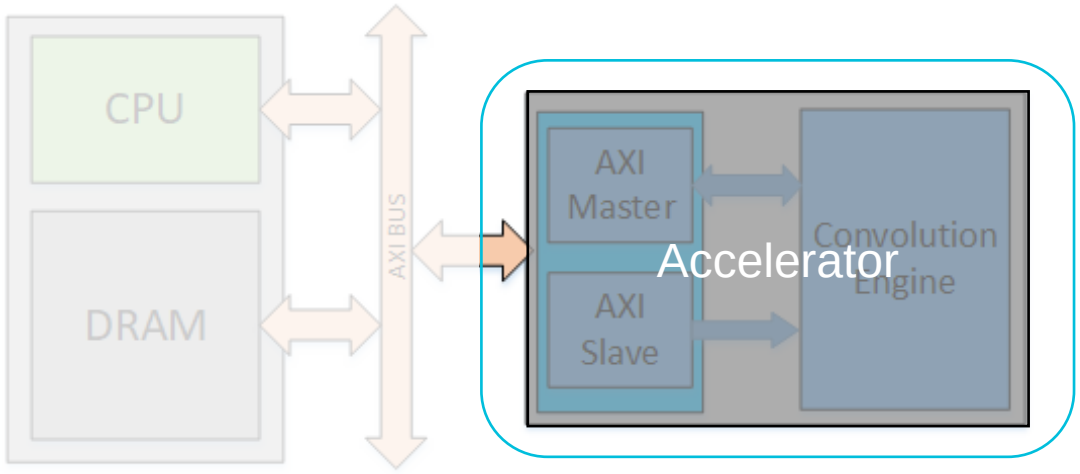
** ML convolution usually has 2 loops more

!=



AI Accelerator Development

- Implementing AI accelerator for software driven AI system
- Requires AXI master and slave interfaces and the accelerator algorithm
- Balancing between HW model abstraction level and SystemC complexity
- Need a tool flow that enables easy modeling, detailed analysis and seamless RTL implementation path



ML accelerator block diagram

Complex Bus Protocols are Easily Modelled with MatchLib

- AXI4 Master and Slaves are part of the MatchLib library
  - AXI4 segmenter transparently manages burst size, 4k boundaries, etc
- AXI4 Read Master I/F with built-in segmentation
  - r_master<>

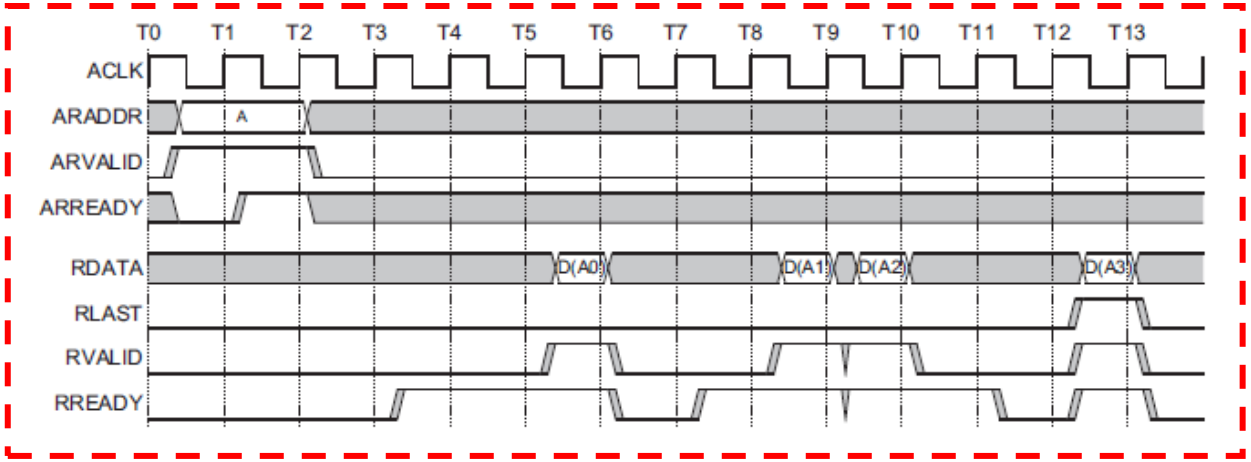
AXI4 Write Master I/F with built-in segmentation

  - w_master<>

AXI4 Slave I/F with built-in segmentation

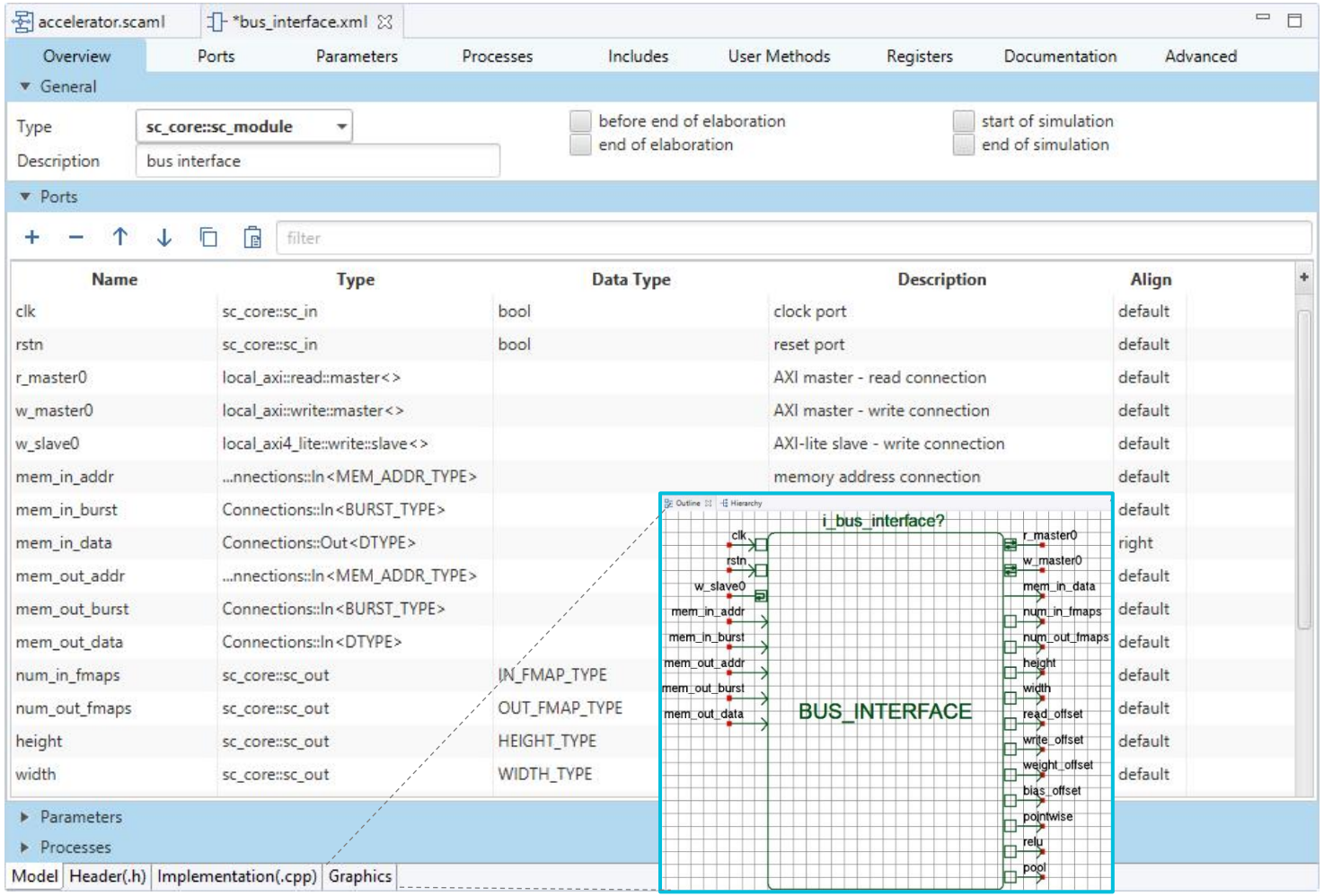
  - slave<>
- Interface reads/writes use connections Push/Pop methods
  - Generates AXI4 transactions

```
class bus_if: public sc_module
…
 r_master<>  r_master0;
…
while (1) {
   …
   r_payload r = r_master0.r.Pop();
   …
 }
…
```



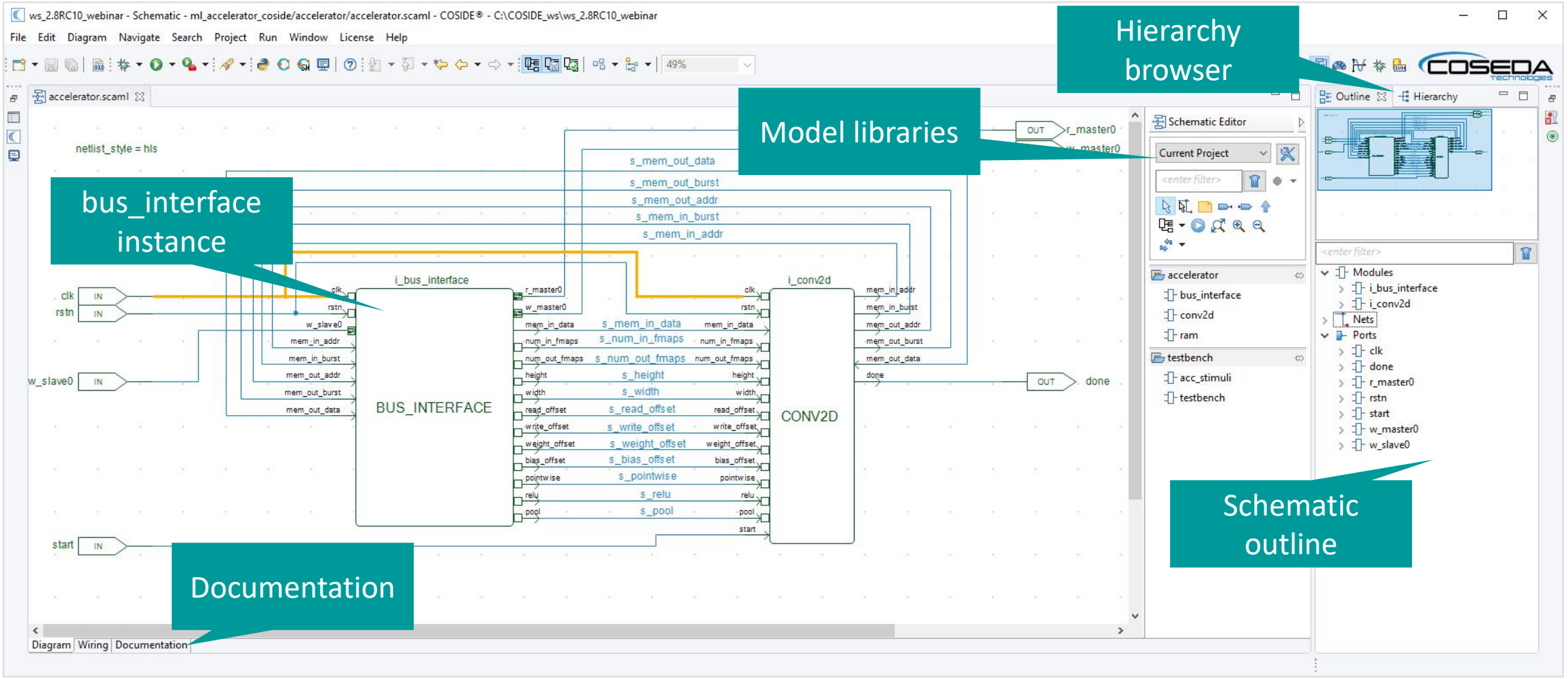
Creating Accelerator Modules with COSIDE Module Editor

- Port definition
  - SystemC
  - MatchLib Connections
- Parameter definition
- Process definition
- Enable HLS compliance
- Code generation
- Description for documentation purpose



SIEMENS

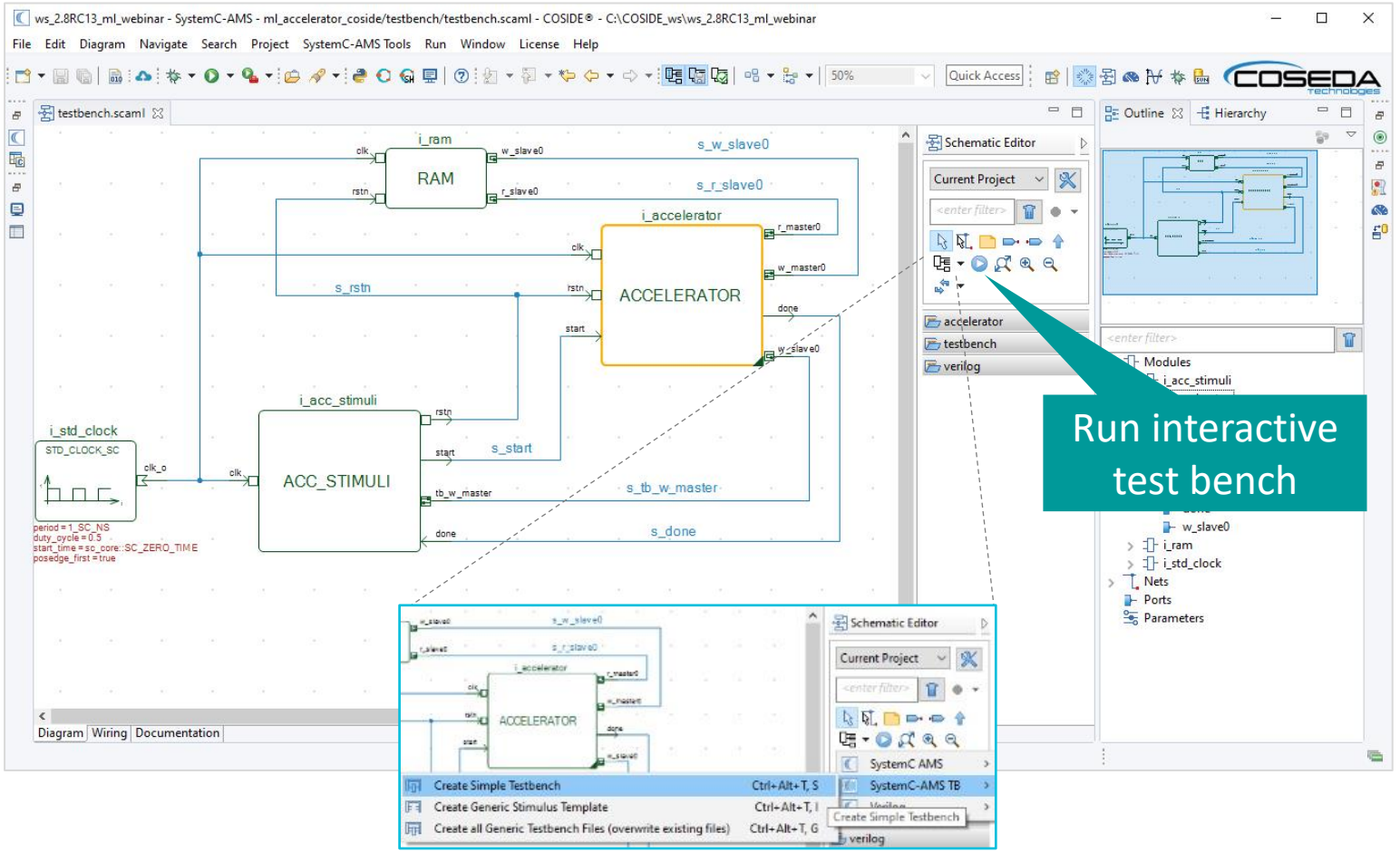
Assembling AI Accelerator with COSIDE Schematic Editor



SIEMENS

Creating Testbench with COSIDE Schematic Editor

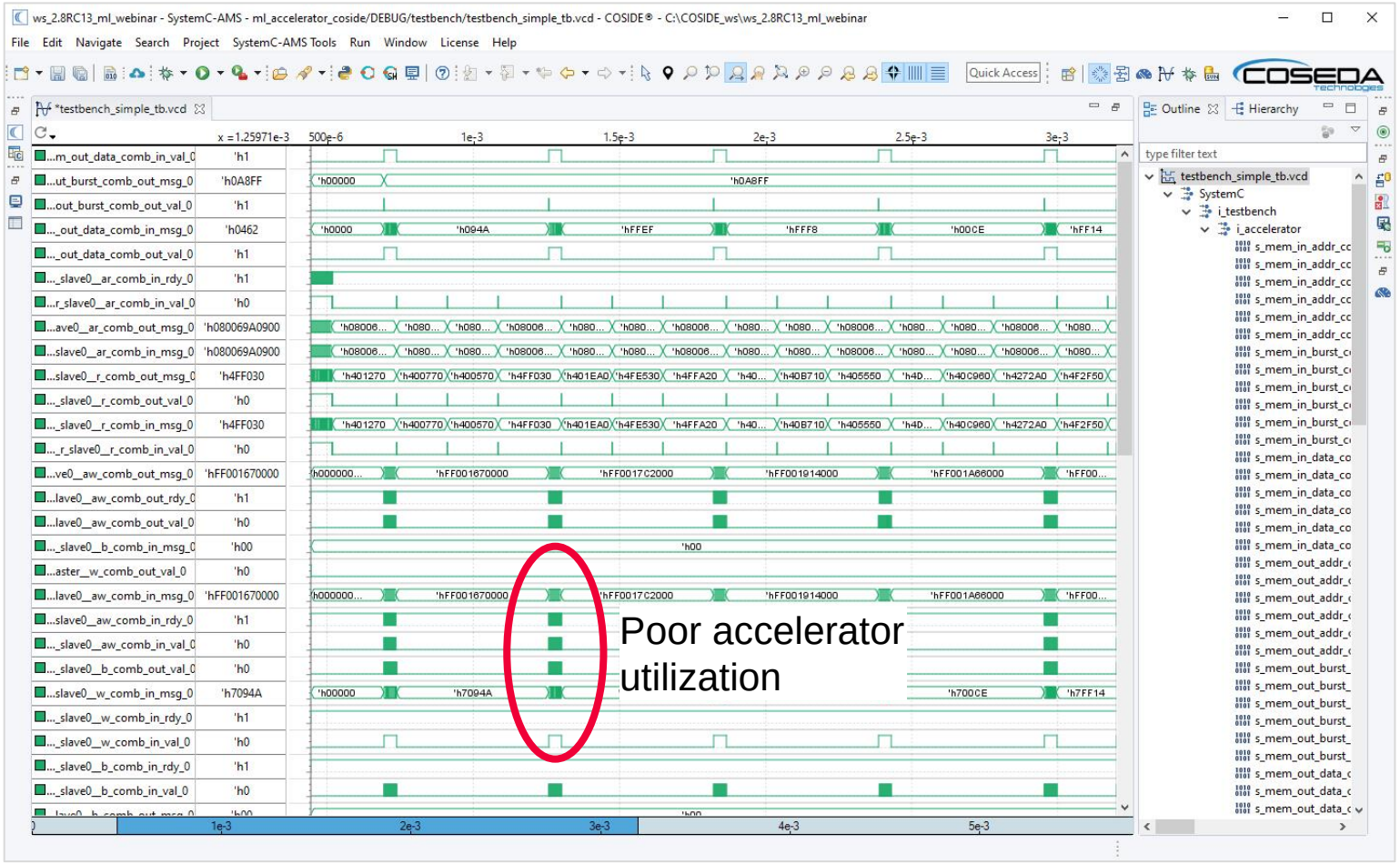
- Test bench code generation
- Test bench concepts:
  - Interactive/simple test bench
  - Stimuli sequence based
  - UVM-SystemC
- Code compilation done in background



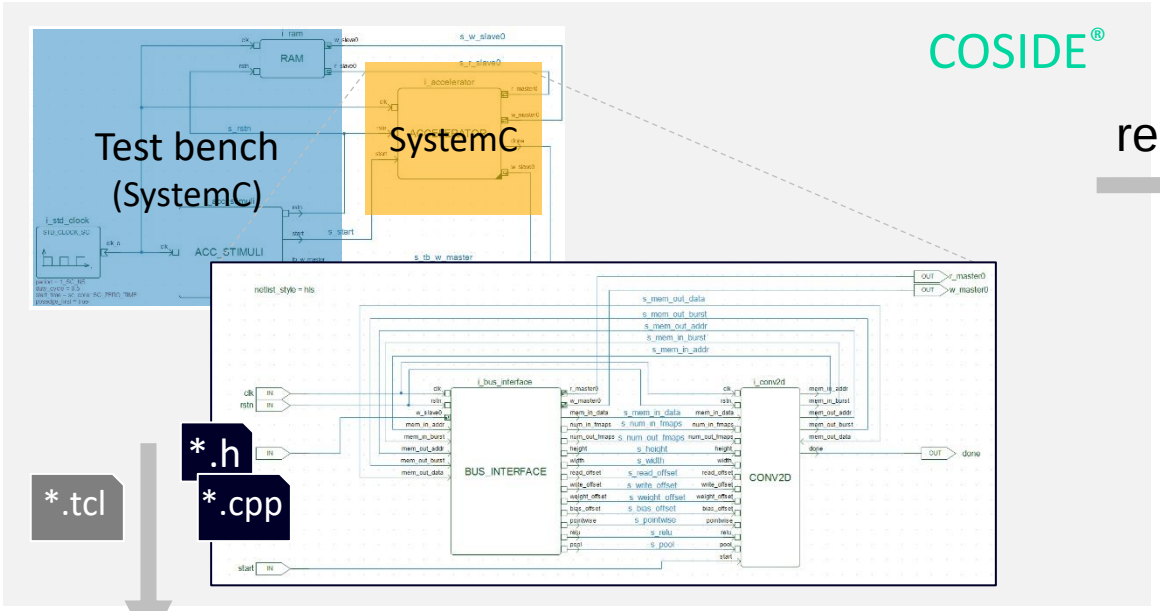
SIEMENS

Analyzing Data Traffic with COSIDE Waveform Viewer

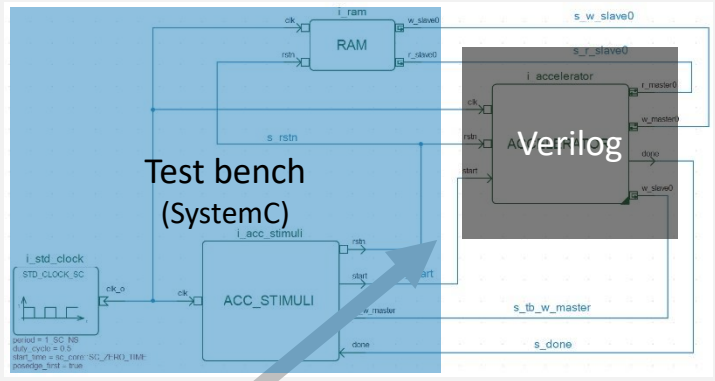
- Mixed signal
- Cursor handling
- Measurement utilities
- MatchLib is cycle accurate
  - RTL-like timing results
- Python post-processing backend
- Stores layout, cascaded execution of post-processing



Integrated High-Level Synthesis Flow

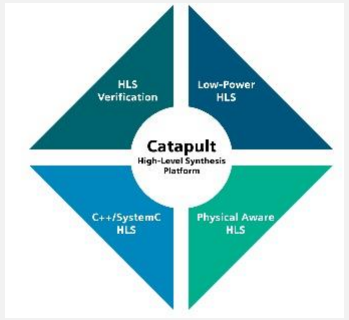


reuse



Verilator (import, built-in)
Questa (SystemC on top)

verify



Catapult® HLS

*.vhd
*.v

Logic synthesis tool

ASIC/ FPGA

SIEMENS

Exporting Accelerator Model to Catapult and Running HLS

Running “Export to Catapult” action – separates in user land and configuration scripts:

Constraints TCL file generation (user land)

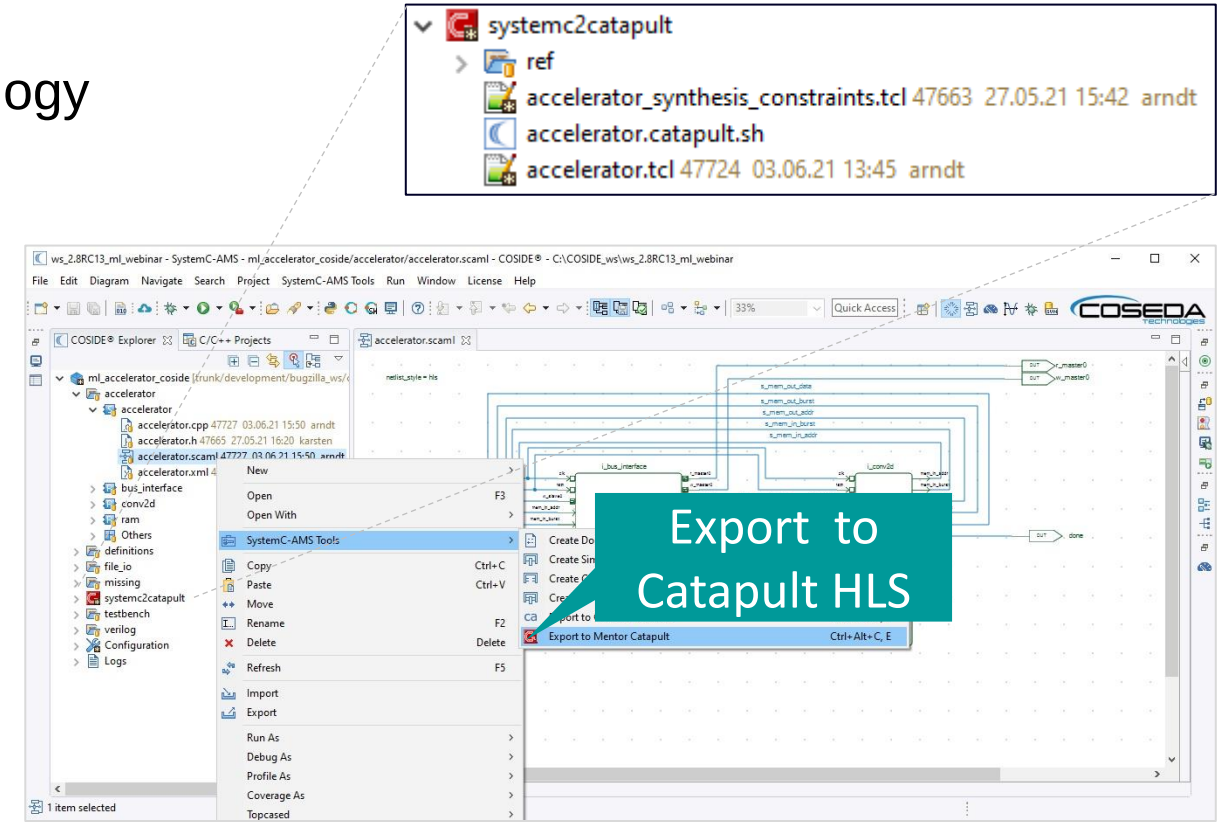
- Constraint template e.g. set directives, technology
- e.g. <mod>_synthesis_contraints.tcl

Configuration TCL file generation (auto)

- Register input files to be synthesized
- Compiler flags and invoke constraints file
- e.g. <mod>.tcl

Catapult starter (with/without GUI)

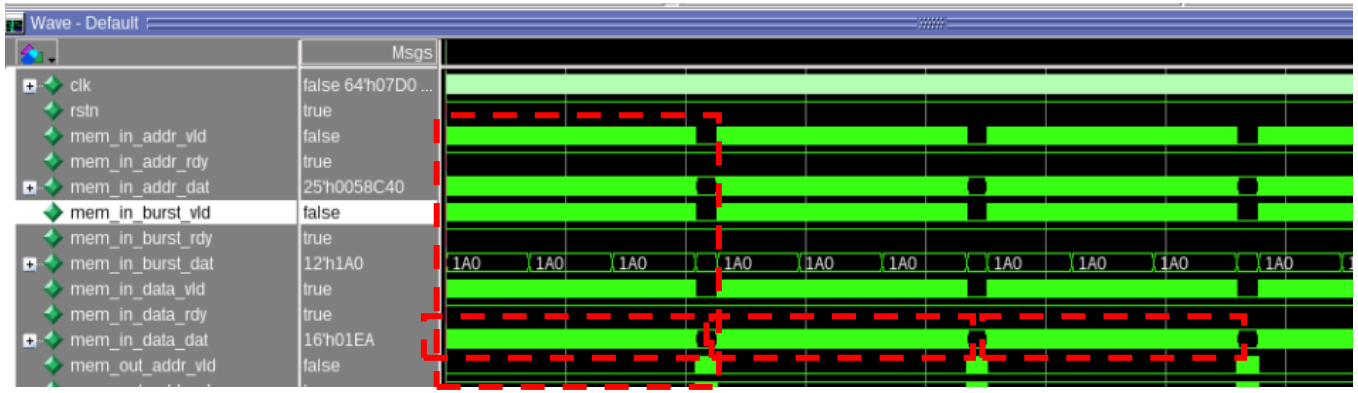
- Shell script to invoke Catapult
- Push-button to start synthesis
- e.g. <mod>.catapult.sh

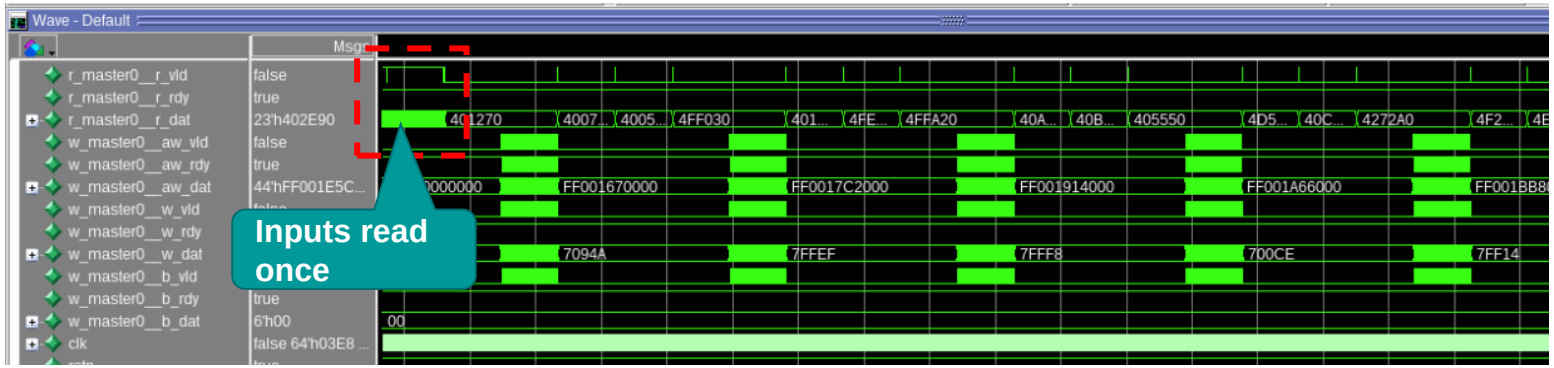


SIEMENS

Optimizing System Performance

- Move Bias, ReLU and max pooling from SW into the accelerator
  - Cost little more in hardware area
  - Reduces data traffic between system memory and accelerator
  - Reduces CPU load
  - Memory bus is ~95% used by accelerator
- Add on-chip feature map buffering
  - Minimizes data traffic to minimum





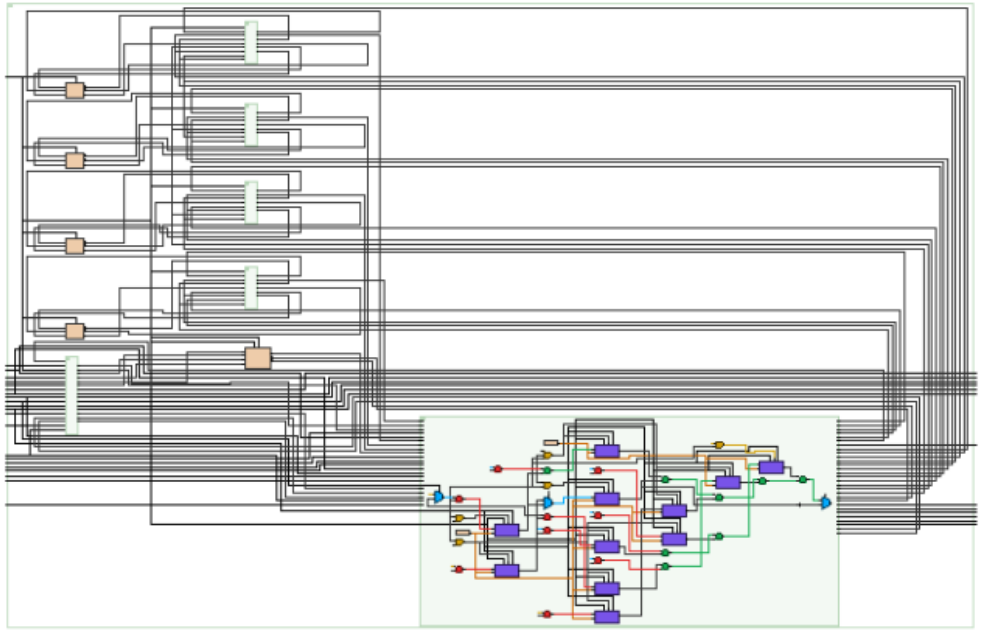
Post-HLS Synthesis RTL Simulation Results

Post-HLS simulation results were very close to the pre-HLS simulation

Post-HLS simulation runtime was over 30x longer than the pre-HLS simulation

- Not practical for simulating multiple frames of video

MatchLib supports also a fast simulation mode, which is ~10x faster than accurate mode used here, but the timing accuracy is not as good.



Simulation Type	Inference Time (secs)	Simulation Wall-clock time(mins)
Pre-HLS	0.93	20
Post-HLS	0.97	630

Conclusions

- Increasing AI/ML algorithm complexity makes design and verification at RTL-level more difficult and time consuming.
- MatchLib and SystemC allow designers to model and verify the true hardware performance, catching bugs early that would normally be exposed during system integration when it’s too late.
- COSIDE SystemC IDE with integrated Catapult High-Level Synthesis provides an efficient design framework for AI/ML designs.
- Using SystemC with MatchLib and High-Level Synthesis removes the need for RTL design and keeps the abstraction level high, but still HW aware.
- MatchLib models can be directly synthesized to RTL and performance of the pre-hls and post-hls results are near identical.

Siemens EDA:

Where electronic innovation meets tomorrow.

SIEMENS