2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Cut Your Design Time in Half with Higher Abstraction

Organizer: Adam Sherer – Accellera Systems Initiative

Speakers: Frederic Doucet - Qualcomm

Mike Meredith – Cadence Design Systems, Inc.

Peter Frey – Mentor Graphics Corp.

Bob Condon – Intel Corp.

Dirk Seynhaeve – Intel Corp.

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

- Introduction – How High-Level Synthesis (HLS) works targeted for hardware designers
- The Proposed Accellera SystemC Synthesizable Subset
- High-Level Synthesis and Verification
- HLS in the Wild – Intel Experience
- HLS for the FPGA/Programmable Market
- SystemC Synthesis Standard: Which Topics for the Next Round?

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

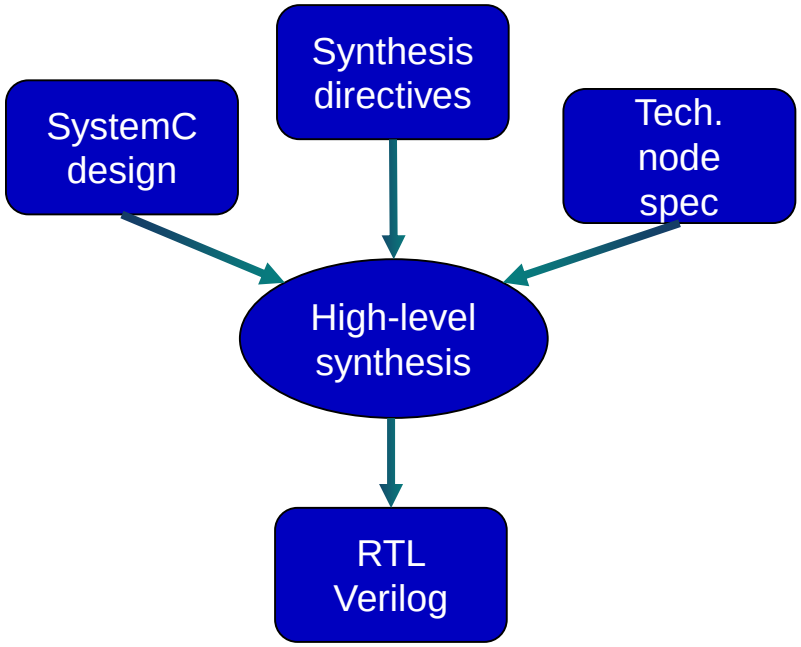
How High-level Synthesis Works:
An Intro for Hardware Designers

Frederic Doucet

Qualcomm Atheros, Inc.

High-level Synthesis

- HLS tool transforms synthesizable SystemC code into RTL Verilog
  1. Precisely characterizes delay/area of all operations in a design
  2. Schedules all the operation over the available clock cycles
  3. Can optionally increase latency (clock cycles) to get positive slack and increase resource sharing (reduces area)
  4. Generate RTL that is equivalent to input SystemC
     - Pipe depths / latencies decided by HLS scheduler



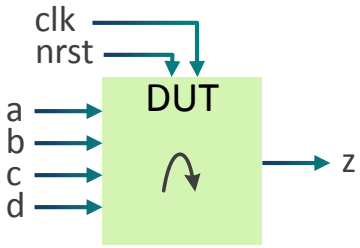
High-level Synthesis

- SystemC HLS has been used in many large semiconductors companies for years, on both control/datapath heavy designs

- Main SystemC HLS usage:
  - Encode and verify all high-level control-flow and datapath functions in SystemC
  - Use HLS tool automatically generate all pipelines and decide latencies resulting in RTL is optimized for specified clk period / tech node

SystemC: Hardware Model in C++

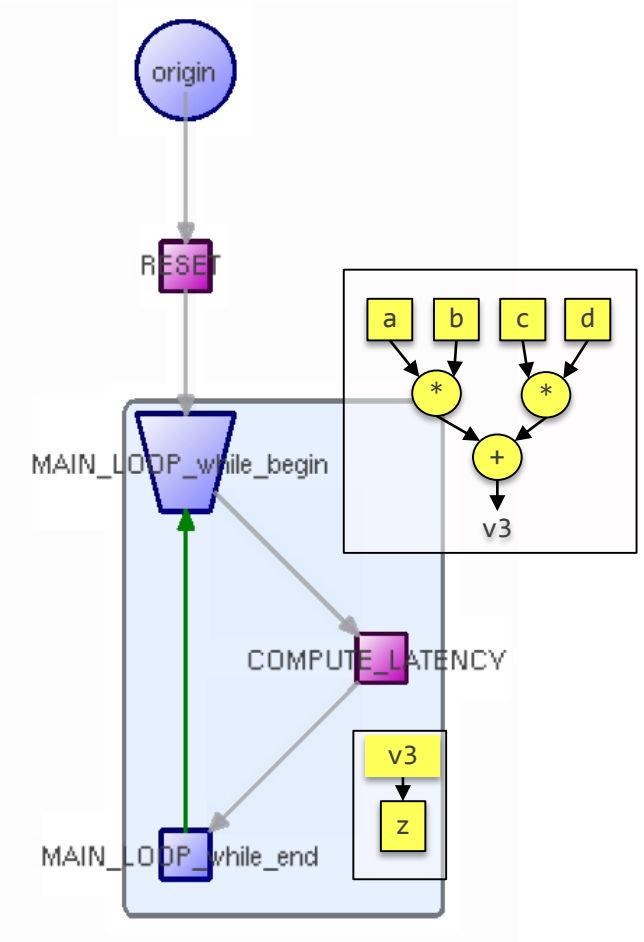
- SystemC: syntax for hardware modeling framework in C++
  - Modules
  - Ports
  - Connections
  - Processes



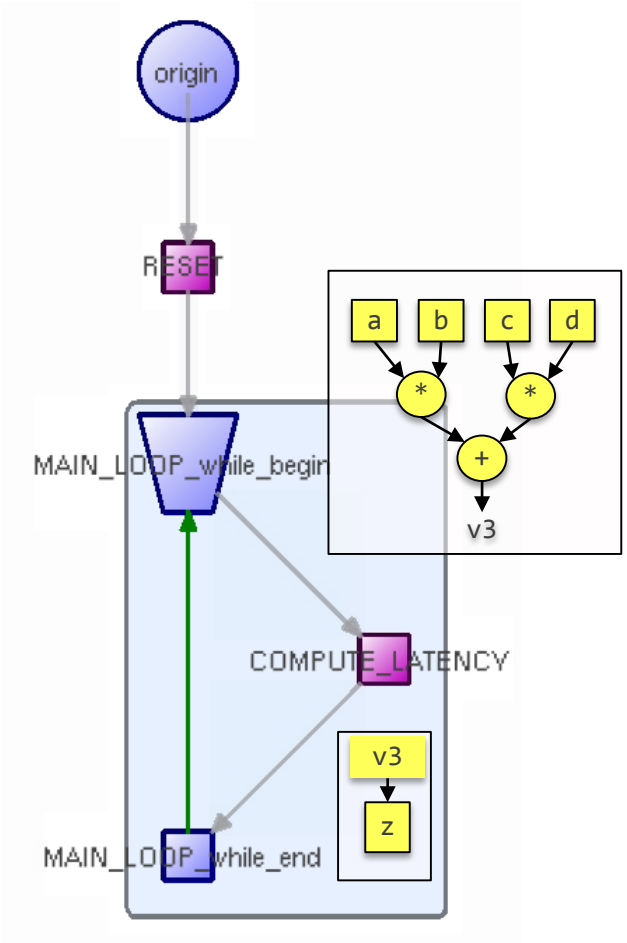
- Inside a process is C++ code describing the functionality
  - DSP processing
  - Control logic
  - Etc.

Example: Synthesizable SystemC

```
SC_MODULE(DUT)
{
  sc_in <bool> clk;
  sc_in <bool> nrst;
  sc_in <int>  a;
  sc_in <int>  b;
  sc_in <int>  c;
  sc_in <int>  d;
  sc_out<int>  z;
  ...
  void process() {
    z = 0;
    RESET:
    wait();

    MAIN_LOOP:
    while (true) {
      int v1 = a * b;
      int v2 = c * d;
      int v3 = v1 + v2;

      COMPUTE_LATENCY:
      wait();

      z = v3;
    }
  }
};
```



Example: High-level synthesis



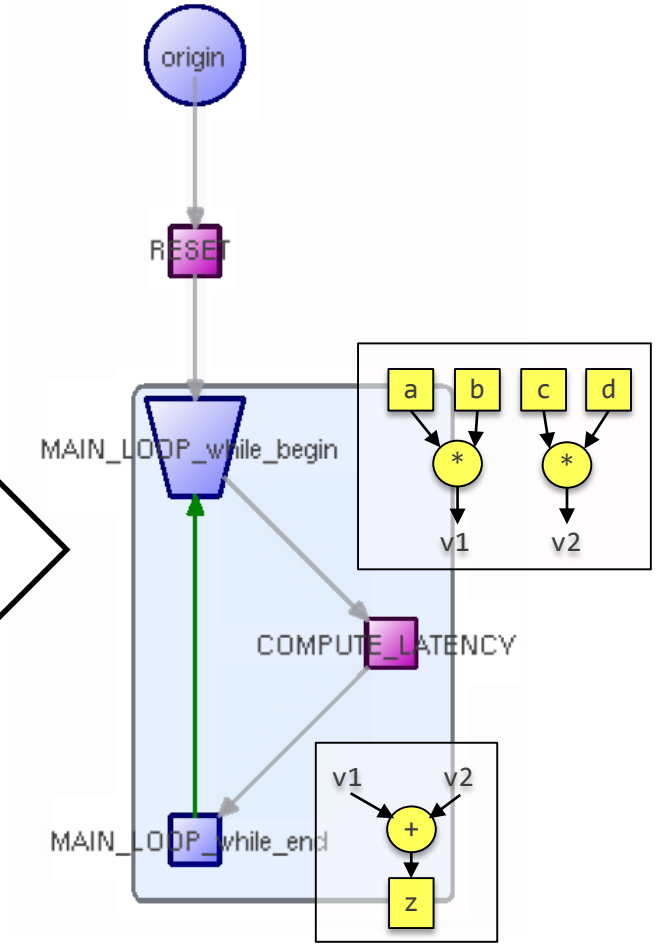
Synthesis directives:

- clk period: 5ns
- tech node: 65lp
- *no micro-arch directive*

Scheduling/resource allocation/binding

Op delays:

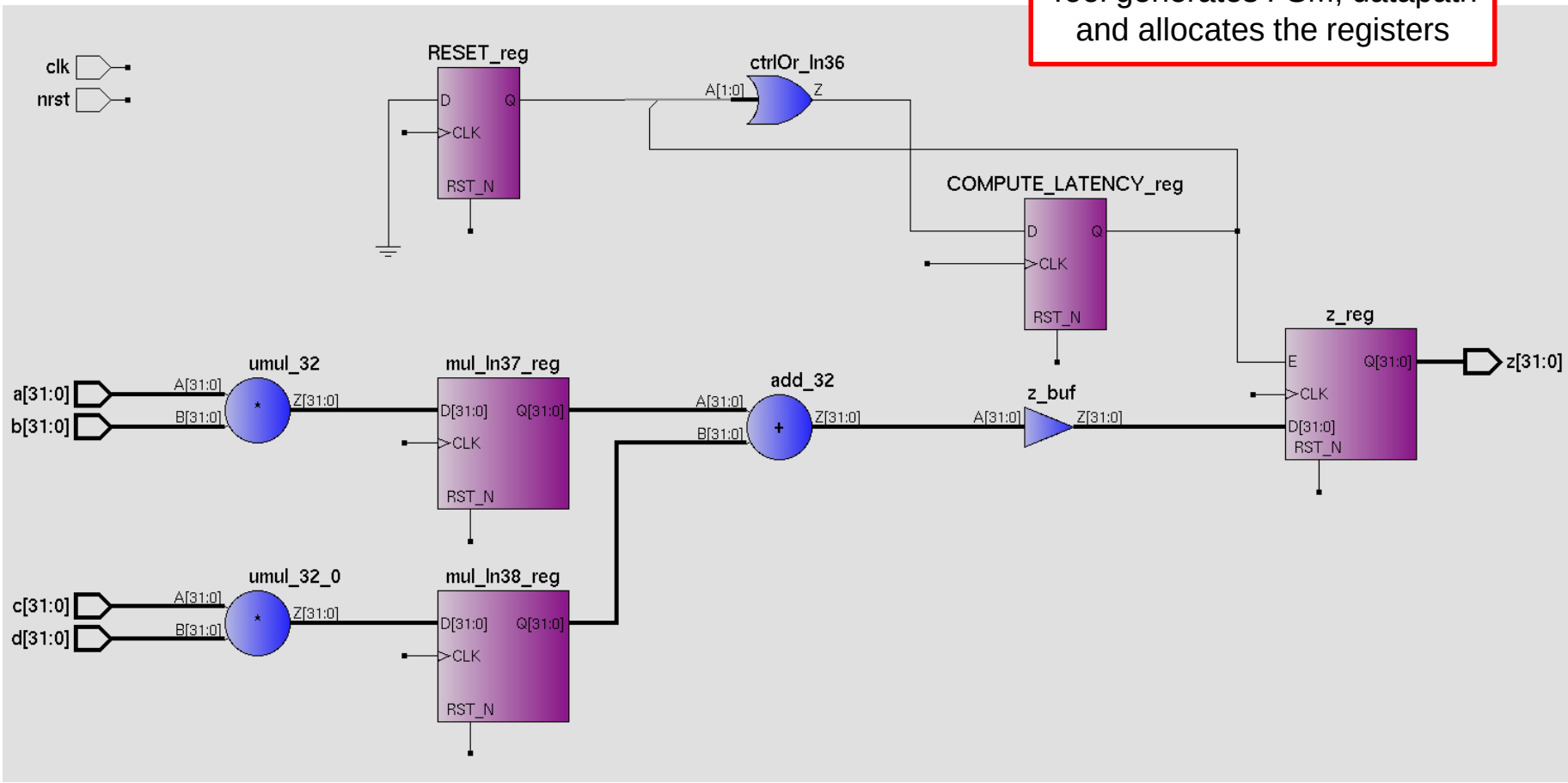
- mul: 4ns
- add: 2ns



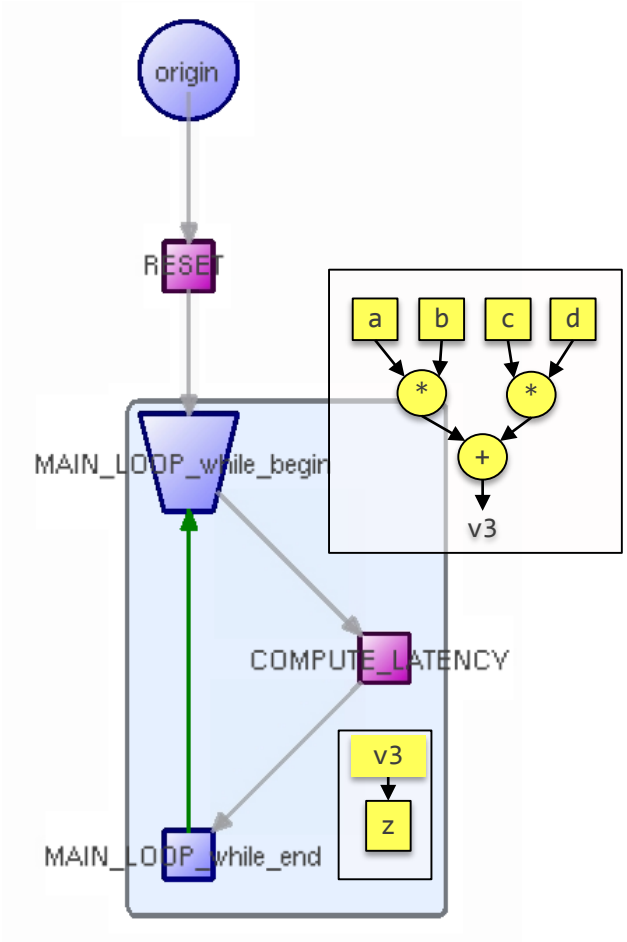
Scheduler moved the addition across the state to get positive slack

Example: High-level synthesis

Tool generates FSM, datapath and allocates the registers



Example: High-level synthesis, second run



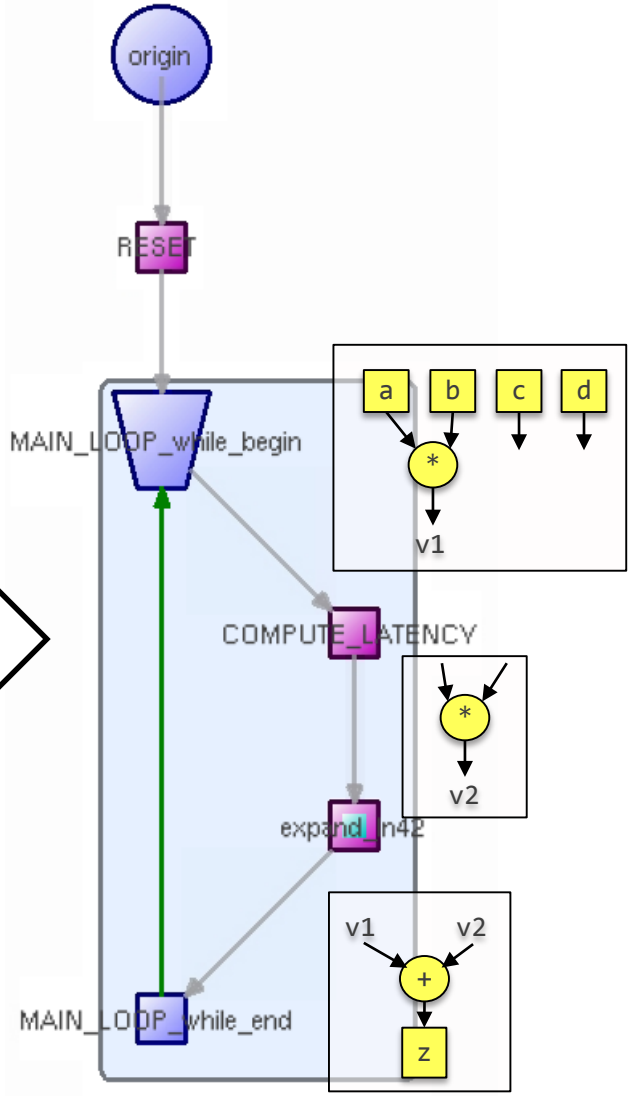
Synthesis directives:

- clk period: 5ns
- tech node: 65lp
- *minimize resources*

Scheduling/resource allocation/binding

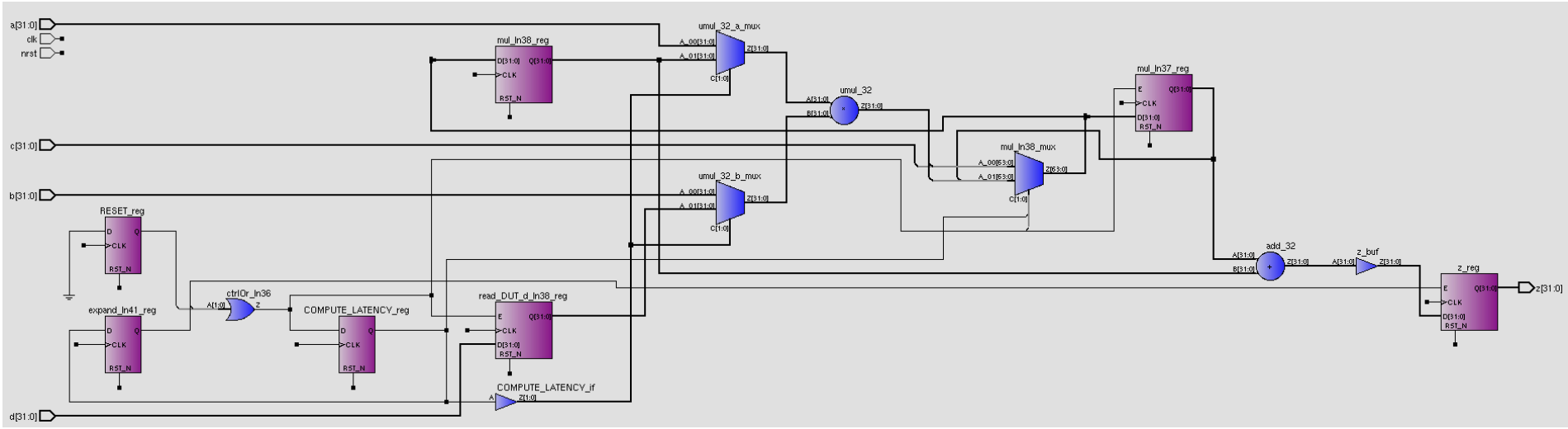
Op delays:

- mul: 4ns
- add: 2ns



Scheduler added a state to share the multiplier

Example: High-level synthesis, second run



- Notice that there is only one multiplier
- Sharing mux/registers are automatically allocated and bound to the generated FSM

HLS and Abstraction

- The tool automatically generates the micro-architecture details
  - latencies, muxes, registers, FSMs

  → *this is what can be abstracted out in the SystemC code*

- Starting from SystemC code, HLS tool does:
  1. Map arithmetic/logical operations to resources
  2. Allocate resources and try to share them as much as possible
  3. Automatically generate FSM and sharing logic
  4. Allocate registers and try to share them as much as possible
  5. Optionally add clock cycles to get positive slack and maximize sharing
  6. Generate RTL

SystemC to Describe Hardware

- Input SystemC code still needs to capture hardware architecture
  - What is the high-level control, data flow and I/O protocols
  - What are the necessary concurrent processes
  - Which are the abstract datapath functions for the tool to refine

→ *Best done by hardware designer*

- Fast turnaround is a big benefit
  - Small changes in the SystemC/synthesis directives can quickly generate new RTL with new and very different micro-architecture
  - Impossible to do with RTL design

SystemC Language

- Designers can use many of the nice C++ features to help write the code
  - Structs/classes, templates, arrays/pointers, functions, fixed/complex classes, etc.
  - Coding patterns/guidelines to separate signal processing code from I/O, etc.

- A standard interpretation of SystemC will help energize the SystemC HLS marketplace and accelerate adoption

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Thank You!