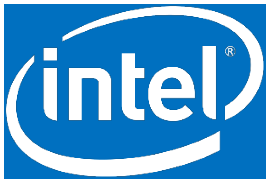
2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

# HLS in the Wild
# -- Intel's Experience

Bob Condon, Intel DTS



# Hi…

- Bob Condon - past 5 years at Intel
  - (Past life HLS, FV, Logic Synthesis at Mentor and Exemplar)
  - Coach new teams adopting HLS adoption
  - HLS-specific tools and libraries
- Disclaimers
  - I won't talk about specific vendor tools
  - I won't talk about specific Intel products
  - "Customers" are internal Intel product groups designing RTL IP which will get integrated into a full SOC

# Spoiler Alert…

- Many production teams at Intel are using SystemC-based High-Level Synthesis to produce the RTL we ship in product
- These designs include both algorithm dominated designs and control dominated designs
- The groups who are happiest report:

  "The HLS flow got us to meet the ___ RTL readiness milestone ___ weeks faster than we estimate with our hand-written RTL approach"

# Why Adopt HLS?

Marketing pitch gives lots of reasons:

- Retarget new process technology
- Automatic (or rapid) design exploration
- Free simulation
- Faster time to validated RTL
- Code is easier to modify
- Eliminates the need for hardware designers
- Provides single source with the VP/Functional model
- Design is “correct by construction”

# Reality Check

- Faster time to validated RTL (the big one)
- Code is easier to modify (pretty big)
- Retarget new process technology (somewhat )
- Provides single source with the VP/Functional model (not really)
  - You can share code but these teams are often very disjoint

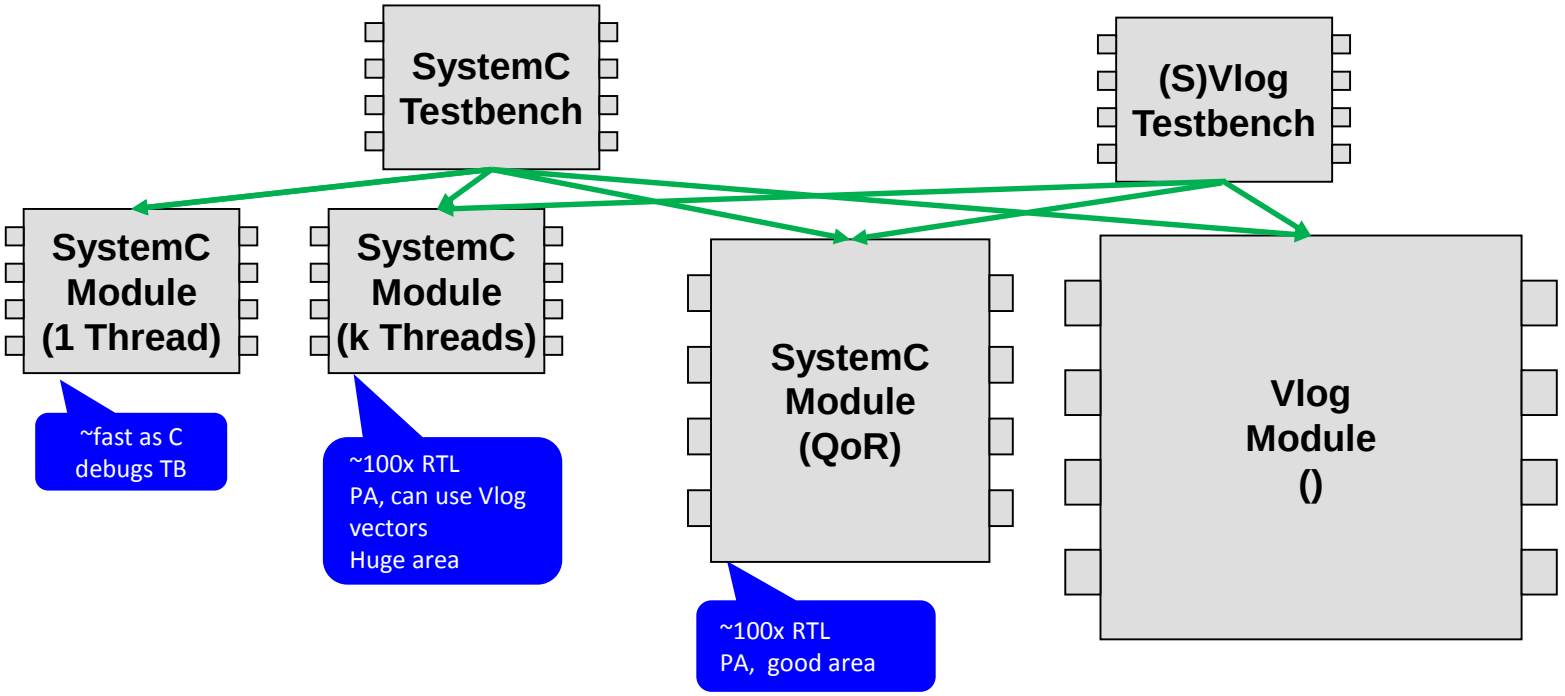
(Not worth it….)

- Automatically do design exploration (not much)
- Free simulation (nope)
- Eliminates the need for hardware designers (nope)
- Design is “correct by construction” (myth)

# HLS Increases Test Velocity

Find bugs with “cheapest” test possible

- HLS designs ready before full SV test ready
- Some flavor of model (vectors, c++ code, matlab exists) – use it
- Find (as many) algo bugs as possible in the fast SystemC simulation
- Mixed language sim to find final communication bugs (and spec changes)



# Plan for Success…

- Project
  - Under time pressure
  - Has a significant amount of new code
  - Has line of sight to a derivative
  - A C/C++ model of some flavor exists
  - The project size corresponds to the “testability” size
- Team
  - >= 4 people with skin in the game
  - At least one of them has decent C++ skills
  - Lined up HLS support
  - Verification and Product build team involved
- The first deliverable is a DOA test Verification team and Build team is involved early

# Who Does the Work?

- 3 Pools of people
  - Verilog coders moving up a level of abstraction
    - Ask them to anticipate a “dreaded” change
    - C++ is often a hurdle
    - Symptom – they write an SC_METHOD in their first design
  - Architects – Our sweet spot
    - “Is overall design better if we tradeoff bus traffic for a bigger RAM?”
  - Algorithm specialists (we don’t really see them doing much HLS)
    - Hardware knowledge is still critical
    - Some software techniques work against HLS

# DataPath vs. Control

We do both and HLS is a win for both

- DataPath designs rely a lot on the HLS tools –
  - Automatic pipelining
  - Common subexpression extraction
- Control based designs rely on lots of use of C++ idioms
  - operator[], Template,
  - Use language to make sure each decision is represented exactly once
- Things that are hard get implemented as library components
  - Start to think of reuse (IP?) differently
  - DataPath: A FIR filter with three taps (traditional “algorithm” IP)
  - Control: A unknown block with Streaming Input, Streaming output, reading coefficients from a RAM and the ability to flush FIFOS on an interrupt

# How Do I Integrate to My Backend Flow?

- HLS output is “generated” RTL (gRTL)
  - Use the same flows as for your h(and)RTL (we relax some lint rules)
- May need a RTL wrapper to leave exactly the same pins as before including things like scan
- The gRTL is uglier -- Minimize the amount of debugging there
  - You do get a waveform and all your vendor tools support mixed language
  - GDB augmented with SC viewers
  - Keep your SystemC test complete on algo-functionality
- Add monitors if you need them
- What about ECOs?
  - We see very few -- ECO modes of the tools are satisfactory

# How Do I Verify?

- Same as today
  - Really, the same way you validated the architectural model against your current RTL
  - RTL still needed for final verification
  - The source is (usually) multi-threaded and not cycle-accurate
    - Formal only works in restricted domains (and with formal expertise)

HLS lets you find and fix your bugs faster but you still need a full testplan to release quality silicon.

# Déjà Vu All Over Again…

- Many production teams at Intel are using SystemC-based High-Level Synthesis to produce the RTL we ship in product
- These designs include both algorithm dominated designs and control dominated designs
- The groups who are happiest report:

  "The HLS flow got us to meet the ___ RTL readiness milestone ___ weeks faster than we estimate with our hand-written RTL approach"

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

# Thank You!