SystemC Suggested Improvements Based On a Formal Flow of HLS Code

Dominik Strasser

VP Engineering

OneSpin Solutions

SYSTEMC™ EVOLUTION DAY
MAY 3, 2016 | MUNICH | GERMANY

Presentation Copyright Permission

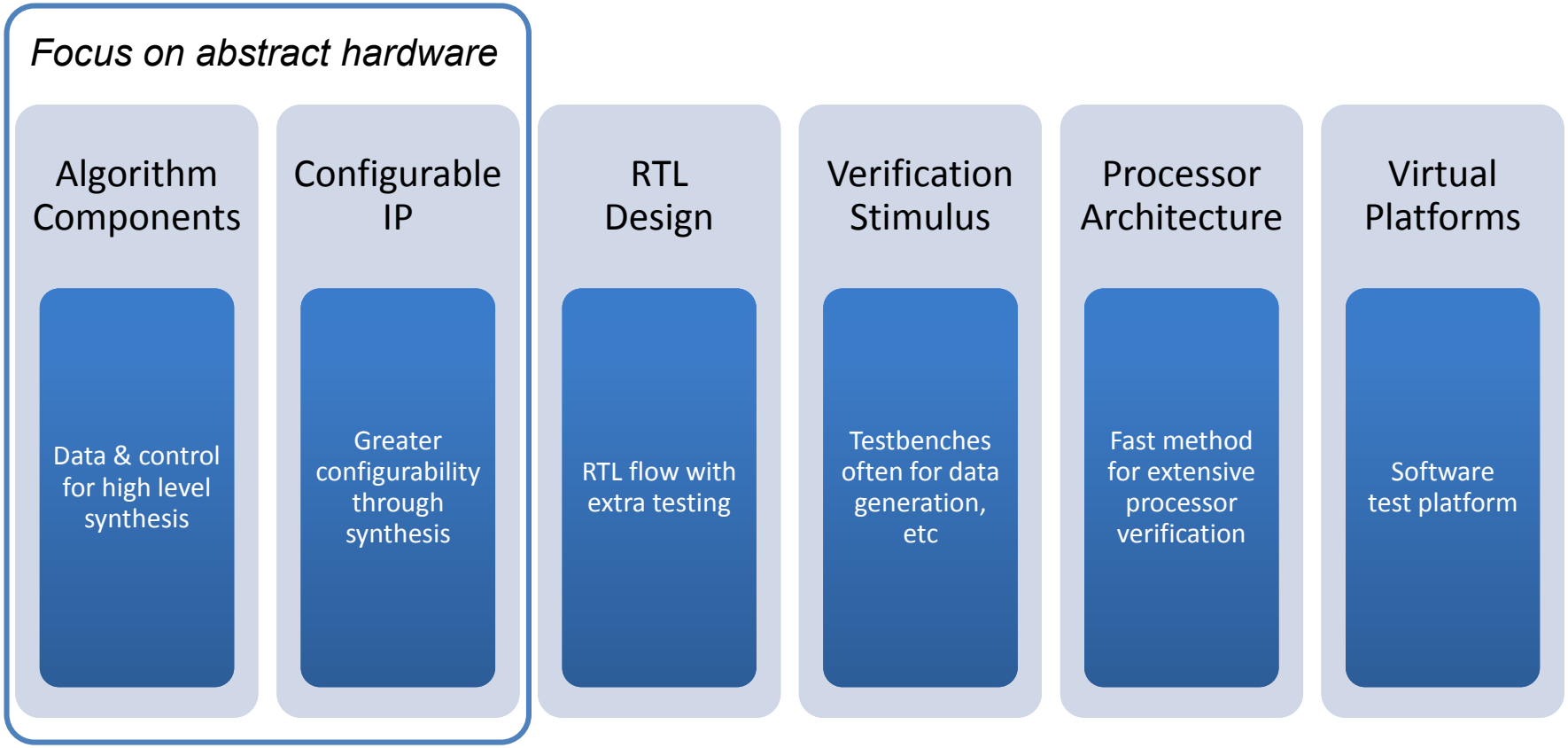
– A non-exclusive, irrevocable, royalty-free copyright permission is granted by **OneSpin Solutions** to use this material in developing all future revisions and editions of the resulting draft and approved Accellera Systems Initiative **SystemC** standard, and in derivative works based on this standard.

Introduction
A Path of SystemC Discovery

- OneSpin working in SystemC space for about 2 years
  - Cooperating with several large users on Formal Verification for the HLS flow
- In this time we have learned a lot about tool development and usage issues in this SystemC segment
- We would like to share some of the things we have seen and make a few suggestions

SystemC Broad Application Space

OneSpin Focus: HLS Segment

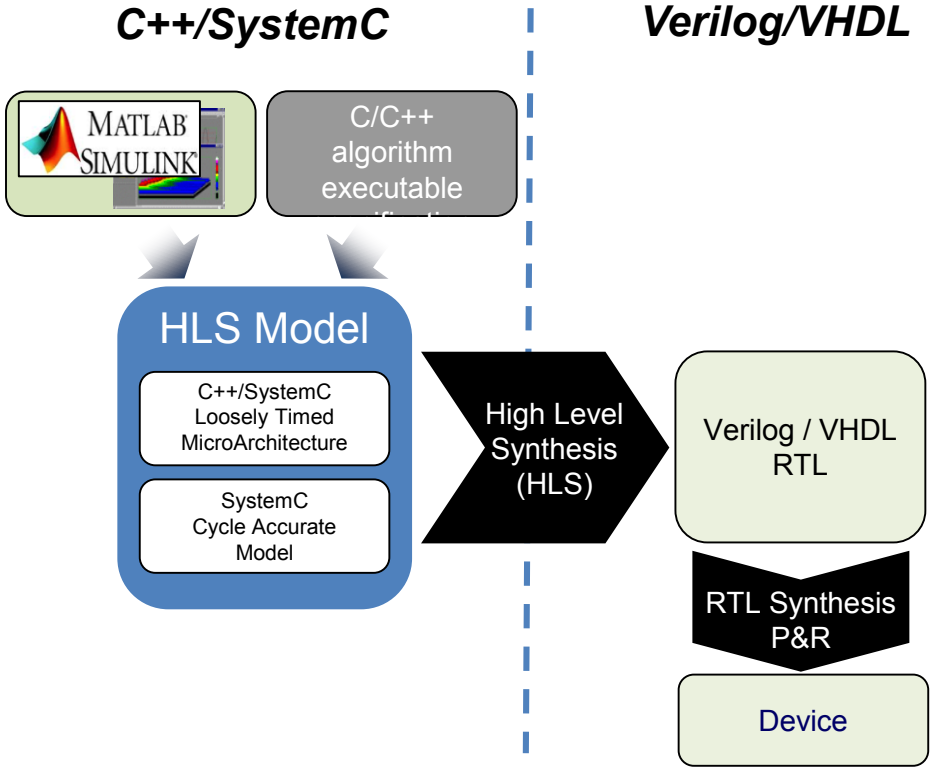


SystemC High Level Synthesis Important SystemC Sweet Spot

- High-Level Synthesis SystemC Market
  - Possibly larger than generally realized
  - Focus:
    Image and communications DSP algorithms
  - SystemC provides benefits over HDLs
  - Appears to be an expanding area
  - Many more doing C refinement without HLS

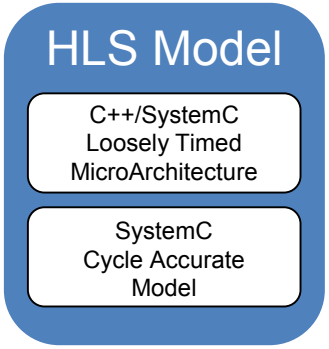
- Companies working in this area
  - Toshiba - NXP (possibly)
  - Fujitsu - ST (possibly)
  - Sony - ARM
  - NEC - Infineon
  - MEE - Samsung
  - Intel - Huawei
  - Qualcomm - Broadcom
  - BlueWire - Canon

  And plenty of others



C++/SystemC Abstraction Range of Issues

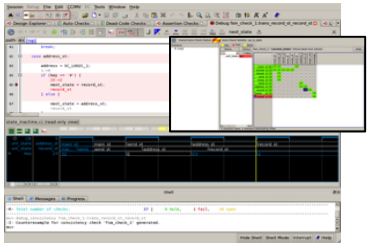
Verification Target



- Proprietary library issues
  - HLS libraries required by other tools to read these designs
- SystemC hardware coding issues, e.g.
  - No undefined (X) signal state
  - Race conditions between threads
- HLS algorithm coding issues, e.g.
  - Number system verification
  - Operation hard to comprehend

OneSpin in the HLS Flow
Formal Being Used to Solve These Issues

Automated Formal Verification



SystemC/C++ Code Apps

Specialized HLS Apps

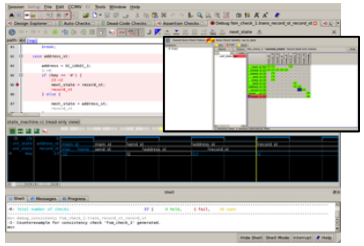
SystemC Golden Model

cadence™

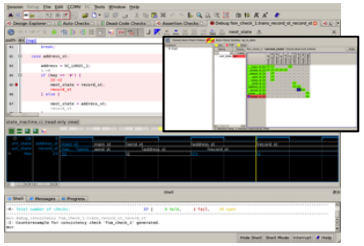
Stratus High-Level Synthesis

Verilog/VHDL RTL Model

Assertion-Based Verification



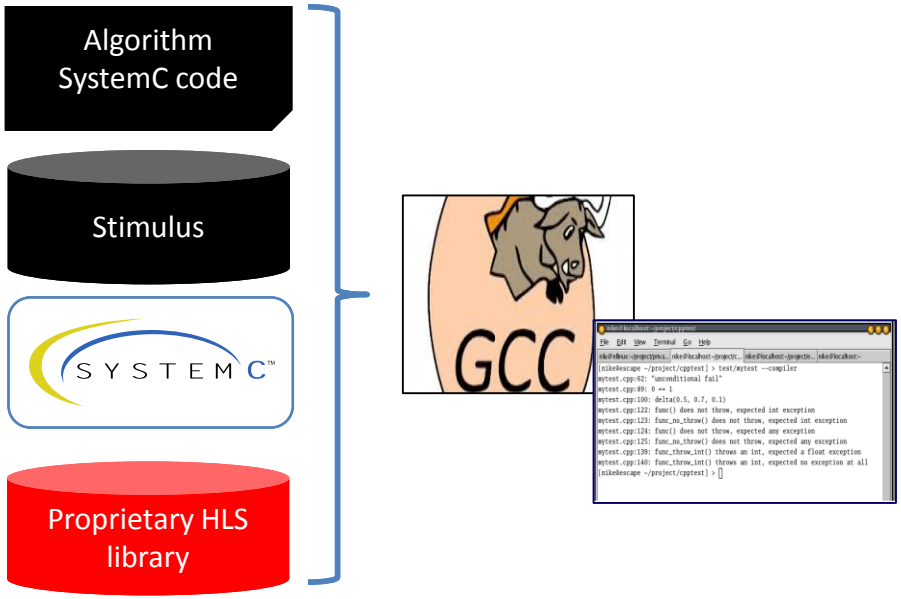
SVA C Asserts



accellera SYSTEMS INITIATIVE

SYSTEMC™ EVOLUTION DAY
MAY 3, 2016 | MUNICH | GERMANY

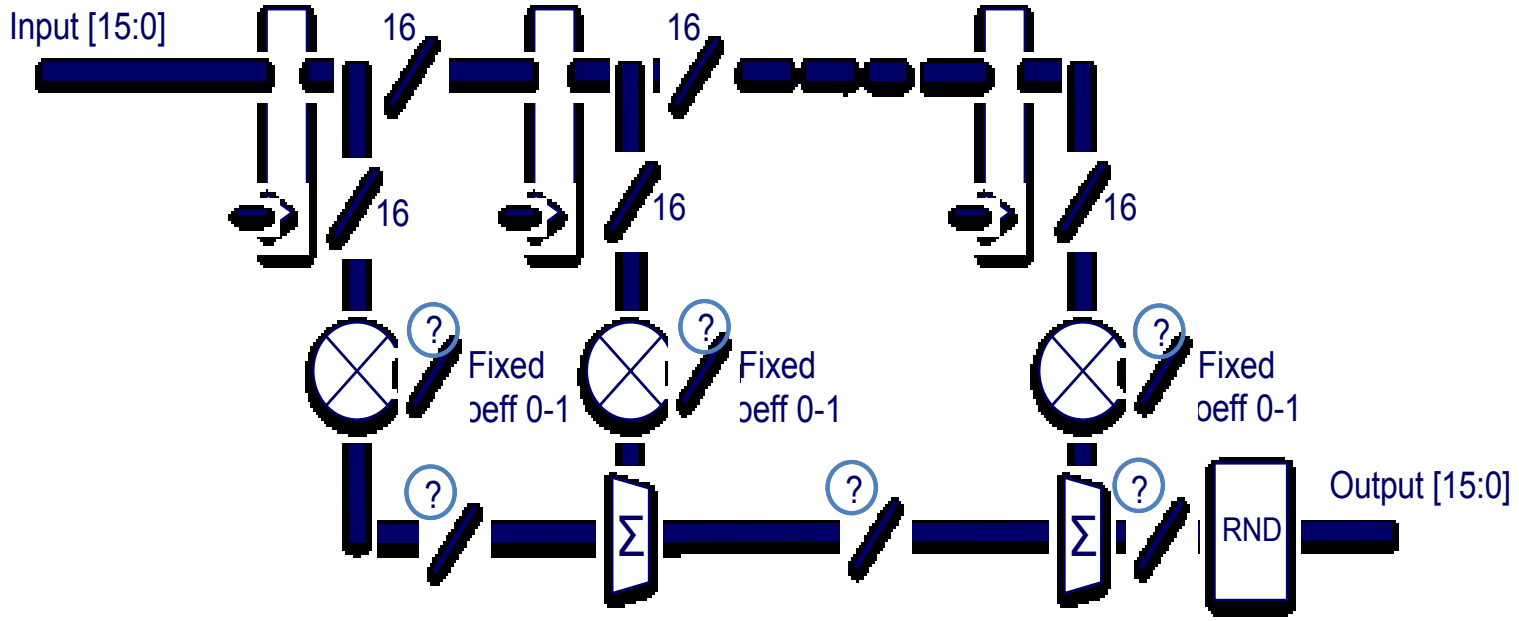
C++/SystemC HLS Relies on Proprietary Libraries Goes Against Ideals of the Standard



- HLS tools require additional language capabilities
- Currently these are being provided in proprietary libraries
  - For good reason; we're not criticizing HLS vendors
- Synthesis subset committee not really providing answer
- For greater HLS ecosystem, this issue needs to be solved

Algorithmic Example
Varied Bit Widths in DSP Function

"Generalized" Filter implementation options
– hardware structure below or MAC working with buffer



Different bit widths due to coefficient sizes may be used to optimize design

Overflow checking for fixed floating points

- **Assigning different width types may loose precision**
  - **Saturation mode can be set, default is to cut (possibly significant) bits**
  - **More confusion than with integer types, because of 2 parameters: bit width and non-fractional part**
  - **Non-fractional part important**
  - **Operations are value preserving**
    - **E.g. 16bit multiplied by 13 bit gives 29 bits**
    - **Converting unsigned to signed adds one bit**
  - **Truncation is done only in one operation: assignment**
  - **Why not place check that non-fractional part is preserved depending on the rounding mode**
  - **Defines could control whether overflows shall be checked, and whether they should result in an abort or just a message**
- **Could be extended to sc_(u)int, too**

Examples

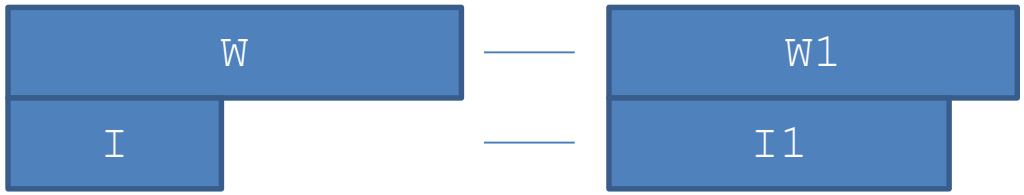
- `sc_(u)fixed<W1,I1>` is assigned to `sc_(u)fixed<W,I>`

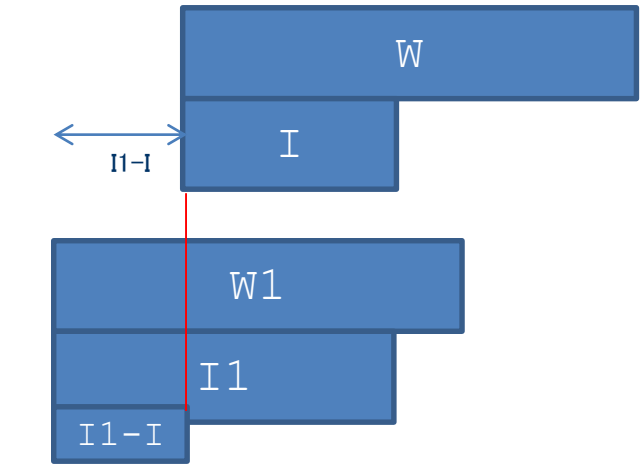
sc_ufixed<8,8> a, b, c; * Default sc_o_mode is SC_WRAP
When using SC_WRAP, we should check for overflow
sc_ufixed<9,9> d;
a = b+c; => FAIL(if b+c > 2^8)
a = sc_ufixed<8,8>(b+c); => NO FAIL
a = sc_ufixed<8,8,SC_TRN,SC_SAT>(b+c); => NO FAIL
d = b+c; => NO FAIL

sc_ufixed<8,8,SC_TRN,SC_SAT> e; * If sc_o_mode is other than SC_WRAP, overflow checking
is not required
e = b+c; => NO FAIL

sc_ufixed<8,8> f(b+c); => FAIL
sc_ufixed<8,8> g = 500; => FAIL

Overflow when assigning to sc_ufixed



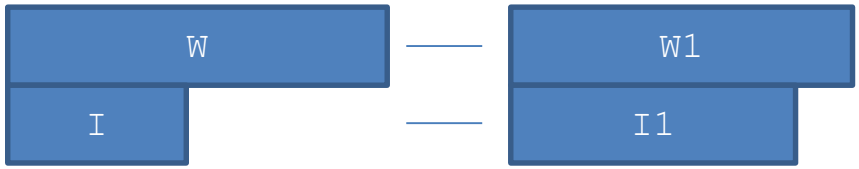


If non-zero, overflow occurs

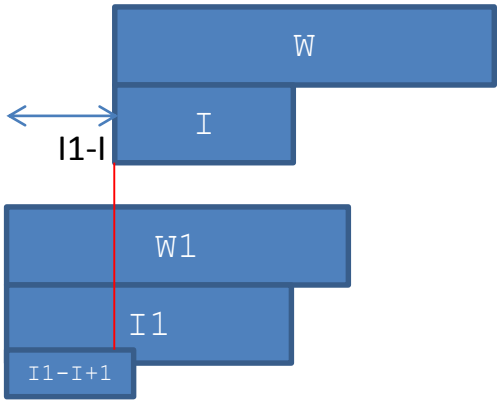
```
if(I<I1 && O_MODE==SC_WRAP) {
  if(a.range(W1-1,W1-(I1-I))!=0) assert(false);
}
```

Bigger than 0

Overflow when assigning to sc_fixed

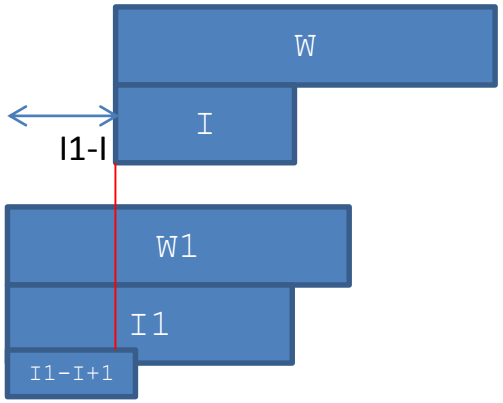


a >= 0



If non-zero, overflow occurs
I1-I+1 >= 2

a < 0



If not all1, overflow occurs
I1-I+1 >= 2

```
if(I<I1 && O_MODE==SC_WRAP) {
  if(a>=0) {
     if(a.range(W1-1,W1-(I1-I+1))!=0) assert(false);
  } else {
     if(a.range(W1-1,W1-(I1-I+1))!=sc_biguint<I1-I+1>(-1))
assert(false);
  }
}
```

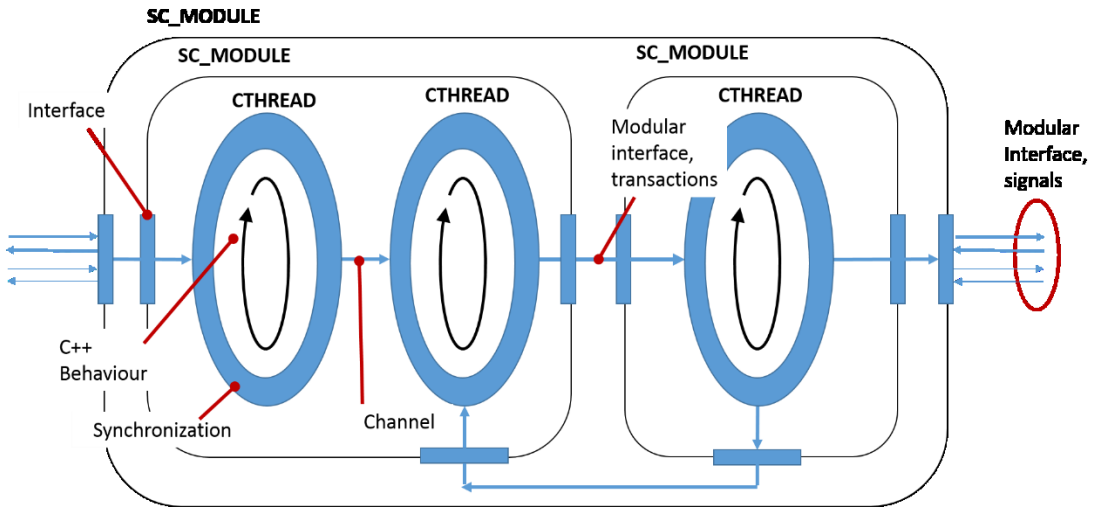
Special cases

- `I > W` or `I1 > W1`
- `I < 0` or `I1 < 0`
  - No overflow possible, as no non-fractional part

- Singularity for `sc_fixed`, `I1 == 1` and `I == 0` (only sign bit)

```
sc_fixed<1,1> a; // W1=1, I1=1, a is 0 or -1
sc_fixed<1,0> b; // W=1, I=0, b is 0 or -0.5
b = a; // If a is -1, OVERFLOW occurs
```

SystemC Imposes Code Structure Can Lead to Hard-to-Verify Issues



Classic SystemC coding issues

- Lack of X state hides potential instability
- Write-write-race between threads
- Arithmetic overflow on data path
- Hard to verify algorithm versus spec
- Port-size mismatch at interface
- Array-out-of-bounds access on buffer
- Dead code and unreachable FSM states
- Packet loss on data transfer

Initialization

- **SystemC has (due to it's mother language C++) automatic initialization**
  - **But: Synthesis semantics specifies that initializations from module constructor shall be ignored**
  - **This leads to simulation-synthesis mismatches**
  - **Why not make this explicit by adding a template parameter to at least `sc_out/sc_signal` specifying that a random value should be used for initialization**
  - **This makes missing initialization problems visible much earlier in the verification flow which are currently difficult to catch**

- **Proposal:**
  - **Add a template parameter to `sc_out`, `sc_signal` specifying the reset behavior**
  - **The default for this parameter shall be the current behavior**
  - **Add `sc_signal_inout_if::write` registering a write**
  - **Add `sc_signal_inout_if::read` delivering a random value before the first write if the reset behavior is set to synthesis semantics**

Arrays

- SystemC has out-of-bounds checking for bit vectors, sc_(u)_int, …
  - But for vectors, good old C-arrays must be used
    - In C++ Software development, C-Arrays are superseded by std::vector
    - Std::vector is dynamic, thus not suitable for hardware
    - bounds checking is available – either by using .at() or by optional debugging code

- **Why no sc_array class ?**
  - **Same interface as C-Arrays**
  - **(Optional) Bounds checking**
  - **Would make hard to find out-of-bounds errors much easier to find**
  - **In-line with other sc_* classes doing bounds checking**
  - **Operator [] asserts on OOB**
  - **Function at throws exception on OOB**

Summary
A Path of SystemC Discovery

- HLS language issues restricting proliferation
- SystemC verification issues driving indirect, post-synthesis verification
  - Eliminates many of the advantages of HLS
- A few enhancements could make a difference
  - Proprietary code
  - Coding restrictions
  - Algorithmic design