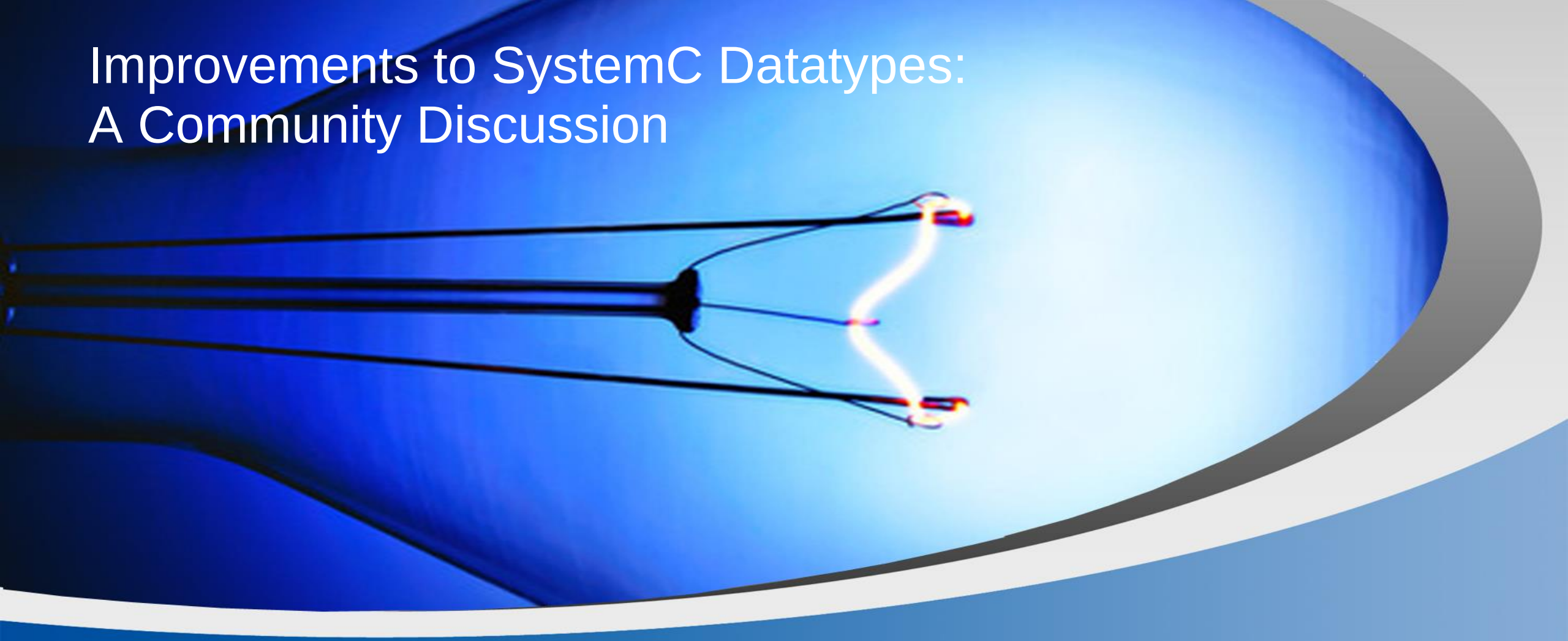
Improvements to SystemC Datatypes: A Community Discussion



SystemC Datatypes Working Group @ SystemC Evolution Day 2017

Presented by Frederic Doucet (WG Chair)

accellera SYSTEMS INITIATIVE

Presentation Copyright Permission

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

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

Agenda

- **Usage of SystemC datatypes and SystemC Datatype Workgroup**
- **Issues in current POC implementation**
- **Proposal:**
  - Improve performance without API changes: implementation improvements by Cadence
- **Proposal:**
  - Improvements with API changes: AC Datatypes by Mentor
- **Open discussion**

accellera SYSTEMS INITIATIVE

SystemC Datatypes

- **Provides a very convenient API for bit exact code and -point code**
  - Very convenience syntax, close to HDL style syntax, much more convenient than standard shift and mask operations
  - Main classes are sc_int<W>, sc_bitgint<W> and sc_fixed<W,I,Q,O>
- **Rich set of functionality**
  - Many functions to access individual bits, bit ranges, arithmetic and logical operations, etc
  - Classes and functions automatically keep track of growing bit widths in expressions
  - Saturation and rounding logic in datatypes

```
sc_int<2>  c =  …
sc_int<8>  v1 = …
sc_int<8>  v2 = …
sc_int<16> v3 = …
sc_fixed<18,2,SC_RND,SC_SAT> v = v1*v2+v3;
if (c[0])      { v >>= 4; }
else if (c[1]) { v >>= 2; }
```

- **Widely used in design code to precisely describe the datapath computation bit widths**
  - i.e. to be synthesized by HLS

SystemC Datatypes in Production

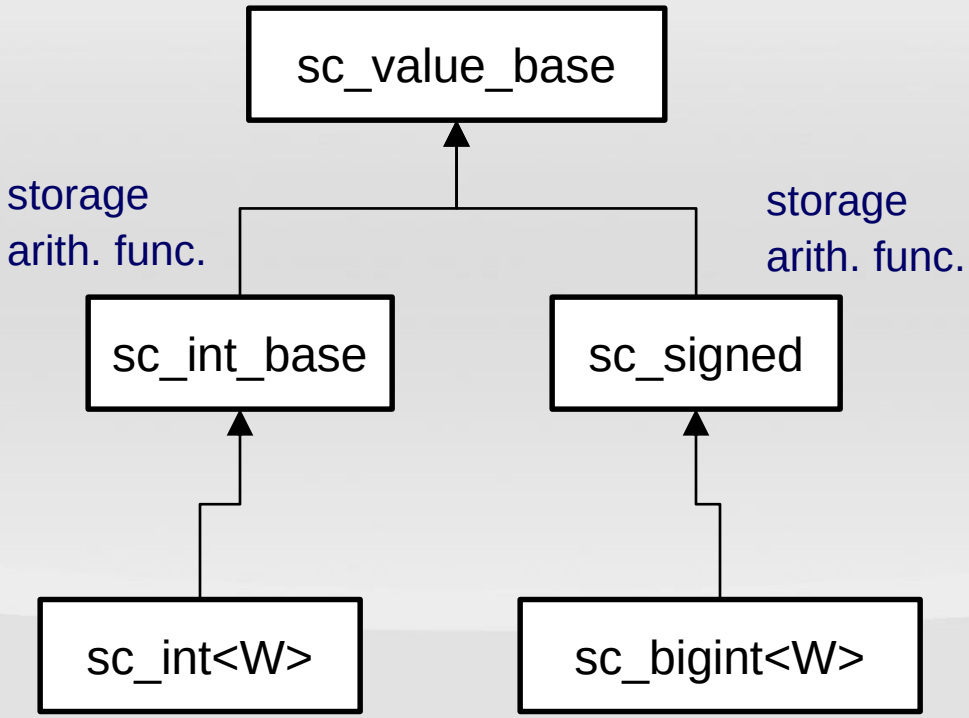
- **Over the last fifteen years, SystemC users and EDA companies have been using the standard SystemC datatypes in different ways:**
  - using the proof-of-concept library as is, or
  - internally customizing it, or
  - completely re-implementing them for simulation speed or synthesizability
- **Main issues in current POC implementation of SystemC datatypes**
  1. Simulation speed is slow, especially for large bit widths
  2. Syntax where bit widths are dynamic can be problematic for HLS
- **The AC datatype library was developed to address these issues (and more)**
  - There was a proposal in SWG workgroup to standardize AC datatypes as a library
  - The proposal was raised to the LWG and …

SystemC Datatypes Workgroup (SCDT)

- **In January 2017, the LWG launched a new sub-workgroup with focus on datatypes**
- **WG Objective:**
  - Define an advanced set of SystemC datatypes, suitable for all SystemC modeling domains and abstraction layers from algorithmic models to synthesis
- **Regular meetings twice a month (2nd and 4th Tuesday of the month)**
- **Active participation from Qualcomm, NXP, Intel, Mentor and Cadence**
  - Currently a heavy focus on design and HLS (we want to change that)
- **The goal of this interactive session is to**
  1. Present the status of the work and results from workgroup
  2. Discuss the datatypes issues and possible solutions with the broader SystemC communities
  3. Collect more feedback and engage the community at large (especially non-HLS users)

accellera SYSTEMS INITIATIVE

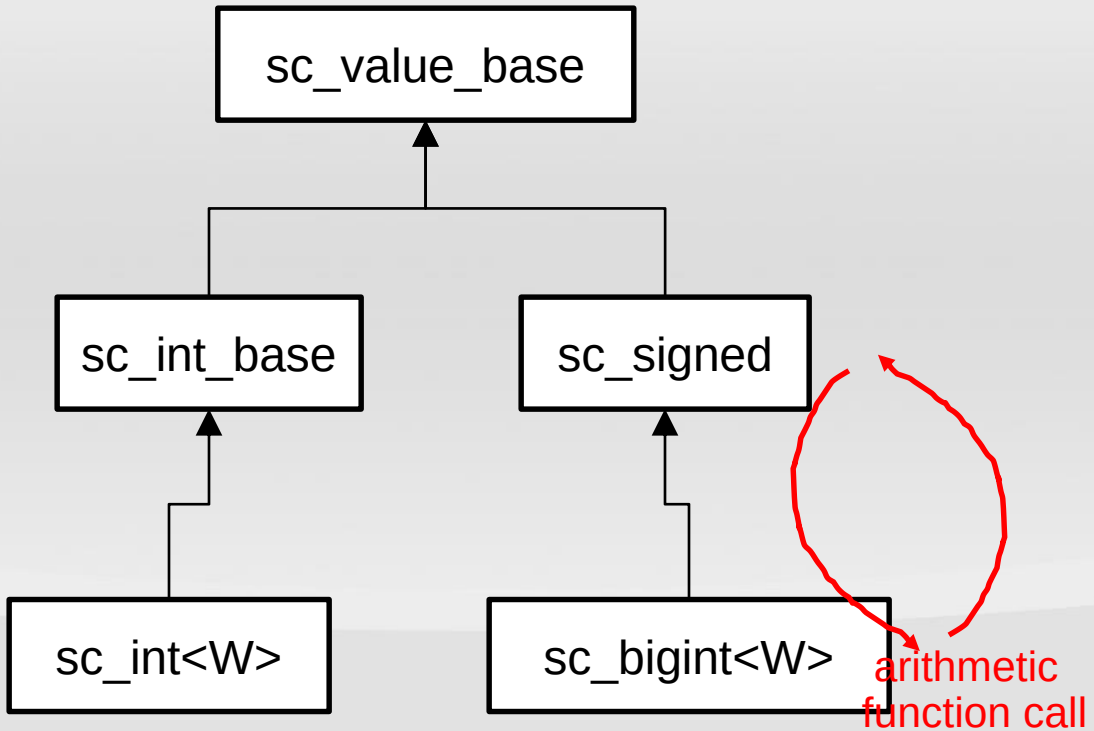
Current POC Datatypes



- **Base classes (sc_signed) bit widths specified at runtime through constructor arguments**
  - Used for modeling, performance (bitwidth) exploration and tuning
- **Leaf classes (sc_bigint) must have bit widths specified at compile time through templates**
  - Use for design with HLS
- **Storage and logical and arithmetic functions are in the base classes**
  - Shared storage in base classes
  - Virtual functions dispatching makes clean code

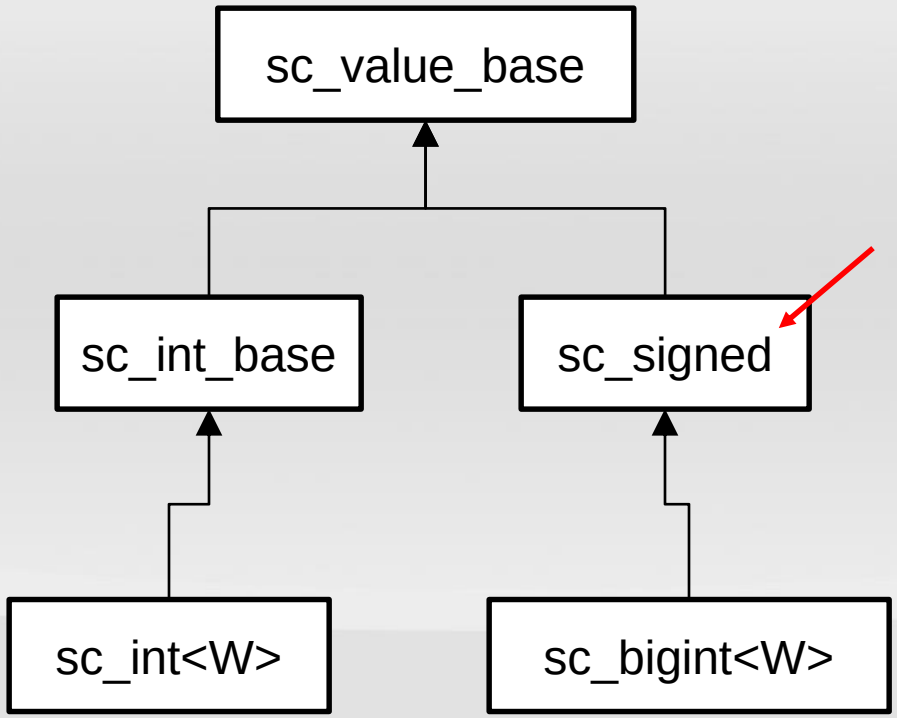
There are other classes too for part-select, concatenation, etc, (for simplicity, we omit those)

API Issues in Current POC Datatypes



- **Arithmetic functions have unnecessary overhead in going back into the base classes**
  - Arithmetic operations go though virtual functions
  - Prevents many compiler optimizations
- **Some operations (concatenation, shifts) calculate return bit widths at runtime**
  - Go into the base class, and dynamically compute the bit widths of the range selection, based on the arguments
  - Big issue for design : HLS tool needs to know all bit widths at compile time
- **Inconsistent semantics in bit shifting**
  - Not matching C++ semantics, negative shift values

Implementation Issues in Current POC Datatypes



1. Significant constructor overhead

- Each time a sc_signed is allocated , a call to new() is done for the internal data storage (and to delete for destructor)
- Data is always initialized to zero

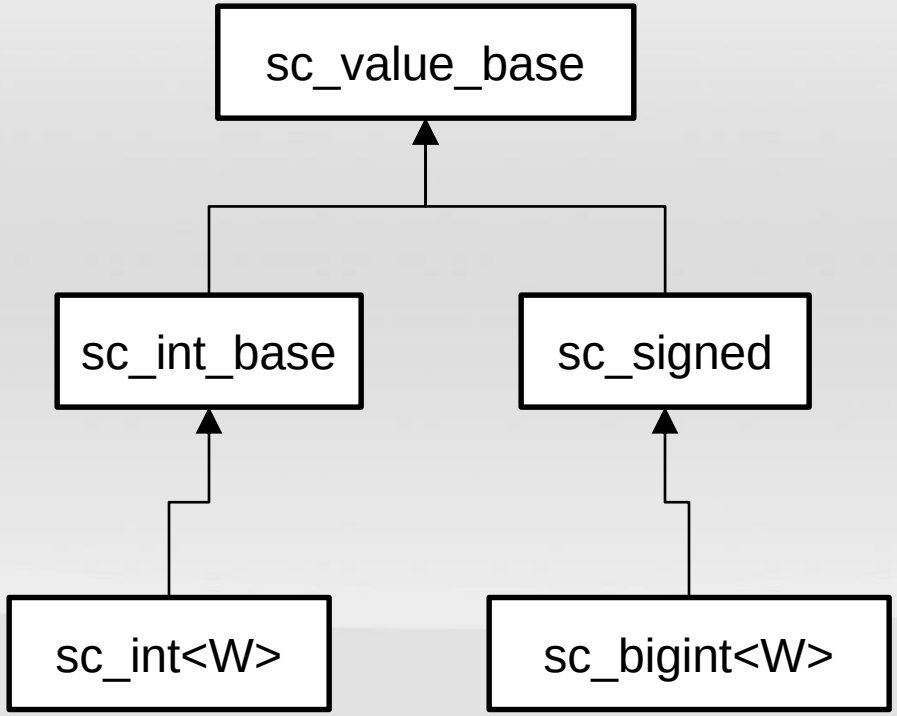
2. Arithmetic performed using signed-magnitude

- Overhead in maintaining separate sign field
- in some calculation requiring conversion to two's complement
- No reason not to use 2's complement

3. 32 bits words used to store 30 bits

- Assignments require shifting, and, or etc.
- Calculating word indices is inefficient (div by 30)
- Cross-type assignments to sc_int done one bit at-a-time

Implementation Issues in Current POC Datatypes



Many of these implementation choices where done more than 20 years ago, when programs and computer memory were much smaller…

…days when 640k of memory was enough for everybody!)

Mike Meredith
Accellera datatype subgroup 10/13/2017

CADENCE PROPOSAL FOR DATATYPE IMPROVEMENT

Context and motivation

- **Context**
  - SystemC is the (only?) C++-based standard that defines arbitrary bit-width computation
  - The existing Accelera implementation has performance problems related to datatypes
- **Goals**
  1. Improve simulation performance for existing SystemC code
     - Users who have written millions of lines of code need a performance improvement
  2. Attract new users e.g. algorithm developers to use SystemC
  3. Avoid expensive LRM development
  4. Avoid proliferation of incompatible classes and APIs inside the standard
     - We need to be able to intellectually defend proposed standardization

Performance evaluation of SystemC 2.3

- **The existing SystemC 2.3 implementation**
  - sc_uint performance is reasonable
  - sc_biguint implementation produces very slow performance
- **Compare SystemC 2.3 datatype performance with ac_int performance**
  - 1-32 bits - sc_uint is 1.3X slower
  - 32-64 bits - ac_int is 1.6X slower
  - >64 bits - sc_biguint is up to 10X slower

width	sc_uint	sc_biguint	ac_int
1-32	1.3	43.8	1
32-64	1	39.6	1.6
>64		Up to ~10	1

accellera SYSTEMS INITIATIVE

Cadence proposal summary

- **Do not change any existing SystemC semantics**
- **Improve performance of inefficient sc_biguint implementation**
  - Cadence has contributed an implementation
- **Performance improvement achieved with biguint**
  - 1-32 bits - 10.2X faster
  - 32-64 bits - 8.7X faster
  - >64 bits - Up to ~6X faster
- **Comparison with ac_int**
  - Assuming sc_uint used for <64 bits
  - 1-32 bits - sc_uint is 1.3X slower
  - 32-64 bits - ac_int is 1.6X slower
  - >64 bits - biguint is 1.6X slower

width	sc_uint	sc_biguint	ac_int	Proposed biguint
1-32	1.3	43.8	1	4.3
32-64	1	39.6	1.6	4.5
>64		Up to ~10	1	1.6

accellera SYSTEMS INITIATIVE

Advantages of this approach

1. **Maintain compatibility with existing designs**
   - Millions of lines of code
2. **Provide performance improvement for existing designs without conversion cost**
3. **Avoid cost and delay of development of new LRM**
4. **Avoid development of new SystemC tests**
5. **Avoid complicating and confusing the standard with additional incompatible classes and APIs**

accellera SYSTEMS INITIATIVE

Presentation of proposal by Mentor

IMPROVEMENTS WITH API CHANGES

accellera SYSTEMS INITIATIVE

ISSUES WITH FIXED-POINT TYPES

accellera
SYSTEMS INITIATIVE

Issues in Fixed-point Types

- **Many SystemC users use the fixed-point datatypes (also has speed issues)**
- **This from post by David Black: http://forums.accellera.org/topic/1638-data-type-sc_int-vs-int/**

```
{
    using data_t = int32_t; //< replace this with
sc_int<32>, sc_fixed<32,32>, float, etc...
    data_t result = 1;
    data_t A = 1103515245;
    data_t C = 12345;



    start_timer(); //< Calls a routine to get CPU time
using C++11 constructs -- portable
    for (size_t loop=loop_count; loop!=0; --loop)
    {
      result = A * result + C;
    }
    cout << "result=" << result << endl; // Ensure
compiler doesn't optimize loop out
    report_time(typeid(data_t).name); //< Calculates
final time -- Be sure to filter with c++filt
  }
```

I saw results for SystemC 2.3.0-ASI --- Jan 6 2014 07:20:27 (on 2.7GHz Intel i7) with no optimization using clang++ compiler:

```
loop_count = 1e8
int                          took  0.343008s
sc_dt::sc_int<32>            took  2.01204s
double                       took  0.755805s
sc_dt::sc_fixed<32, 32>      took 41.3482s
```

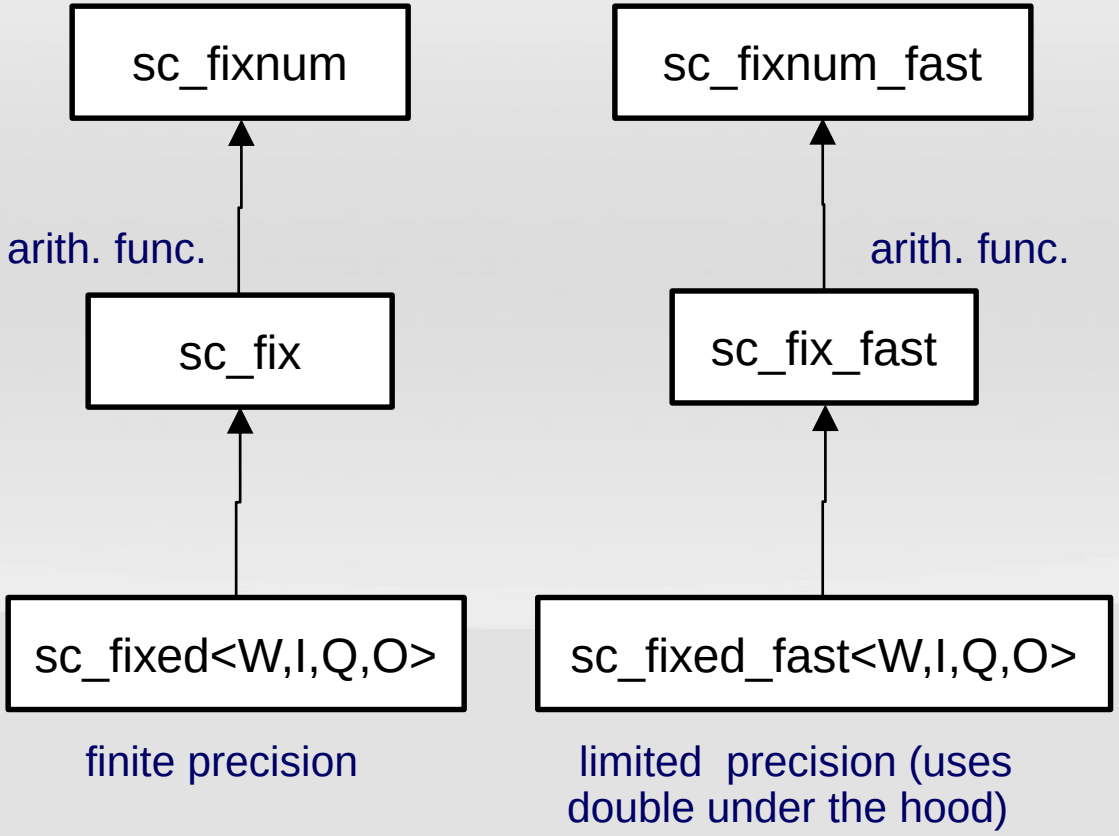
With -O3 optimization:

```
loop_count = 1e8
int                          took  0.125028s
sc_dt::sc_int<32>            took  0.199671s
double                       took  0.364543s
sc_dt::sc_fixed<32, 32>      took 25.6714s
```

So be careful what you select AND what compiler options you use.

accellera SYSTEMS INITIATIVE

Issues in Fixed-point Types



- **Also has similar API issues to bigint**
  - Arithmetic functions in base classes
- **sc_fixed_fast provided to provide faster simulation speed**
  - Equivalent to sc_fixed only for mantissas of bitwidths up to 53 bits
- **API also has implicit cast to double**
  - Possible loss of precision in design because double has only 53 bits mantissa

Status of the Proposals (1/2)

1. Speed improvements bigint without API changes

Cadence has contributed a prototype implementation

https://workspace.accellera.org/apps/org/workgroup/sdtwg/download.php/15963/Fast_systemc-2.3.0.tgz

We begin a 90-day review period of the proposed changes (until January 15th 2018)

Please download the tarball and experiment with it, report feedback on the performance improvements or questions on either the LWG forum in a datatype thread:

http://forums.accellera.org/forum/10-systemc-language/

Please join the working group!
https://workspace.accellera.org/apps/org/workgroup/sdtwg/index.php

accellera SYSTEMS INITIATIVE

Status of the Proposals (2/2)

2. **Speed and syntax improvements with API changes**

   Needs more discussions and experimentations

3. **Improvements to fixed-point datatypes**

   - Discussions and work to start in future

Please join the working group!
https://workspace.accellera.org/apps/org/workgroup/sdtwg/index.php

accellera SYSTEMS INITIATIVE

Open Discussion

Are SC integer datatypes used only by HLS designers?

a) yes
b) no

Are you (or anybody) using sc_signed?

a) If so, why?

Is/would your algo team used SC fixed-point datatypes if fast enough?

a) yes
b) no

Would you like a sc_complex class?

a) yes
b) no

Would you like a sc_float class?

a) yes
b) no

What kind of API changes would you find reasonable?

a) None! We need 100% backward compatibility!
b) Anything! No backward compatibility needed, we'll rewr ite everything!
c) Be cautious! Backward-compatibility for 95% of the code?

Are you using AC Datatypes mixed with SystemC?

a) Standardize AS IS!
b) Standardize, but change a few things!
c) Merge the good stuff from AC into SC improvements

What is the most important thing for you?

a) Bigint speed
b) Fixed-point speed
c) AC datatypes standard
d) Free lunch!

accellera SYSTEMS INITIATIVE