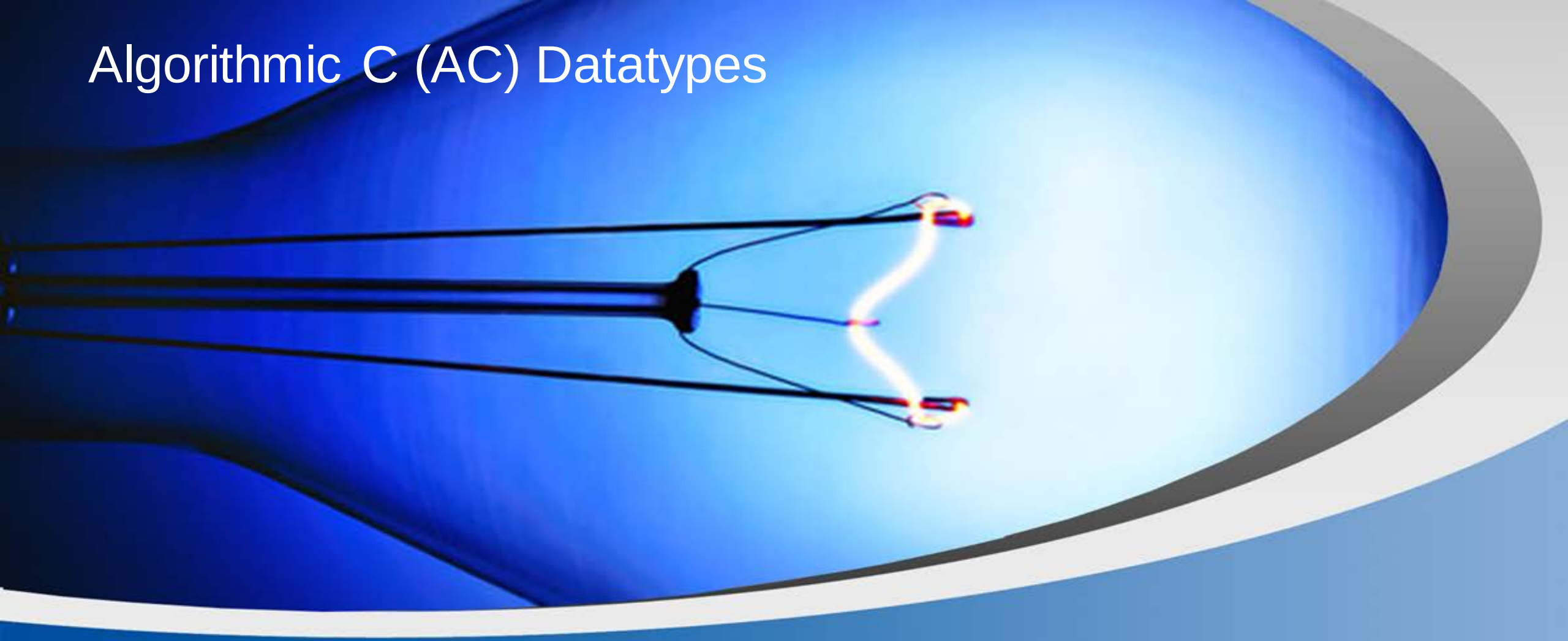
Presentation Copyright Permission

- A non-exclusive, irrevocable, royalty-free copyright permission is granted by **Mentor Graphics(a Siemens business)** 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 the standard.

accellera SYSTEMS INITIATIVE

Algorithmic C (AC) Datatypes



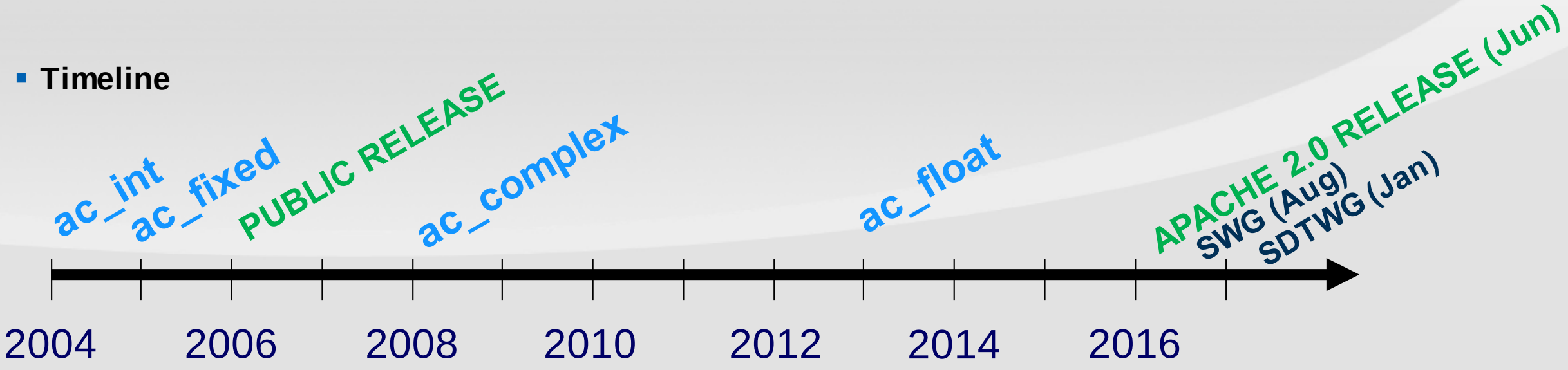
SystemC Datatypes Working Group @ SystemC Evolution Day 2017

Presented by Kunal Bindal (Mentor, a Siemens Business)

accellera SYSTEMS INITIATIVE

Numerical Algorithmic C (AC) Datatypes Overview

- **Integer: ac_int (replaces sc_int, sc_uint, sc_bigint, sc_biguint)**
- **Fixed-point: ac_fixed (replaces sc_fixed, sc_ufixed, sc_fixed_fast, sc_ufixed_fast)**
- **Complex: ac_complex<T> (Not available in SystemC)**
  - Type T can be any AC type or C integer or float/double
- **Float: ac_float (Not available in SystemC)**
- **Timeline**



accellera SYSTEMS INITIATIVE

Apache 2.0 Release (Since Jun 2016)

https://www.mentor.com/hls-lp/downloads/ac-datatypes

https://github.com/andres-takach/ac_types

Motivation For Creating AC Datatypes

- **Identified issues with SystemC Datatypes in fall of 2003**
  - ST and Mentor collaborated and donated work to Synthesis Working Group
  - Findings reported to Language Working Group

- **Issues**
  - Too slow for hardware verification requirements
  - Many issues with inconsistent and not well defined semantics
  - Deep hierarchy of classes all exposed as API.
    - It constitutes Documentation/LRM
    - Hardware designers expect a more formal definition of what to expect

Standardization of AC Datatypes

- **Requested by active members of the SWG over years to be included in the SWG standard**

- **Licensing was changed to Apache 2.0 in June 2016 to enable inclusion in the SWG subset**
  - Just a C++ library of classes: no impact on the SystemC kernel
  - Does not need to be formal part of the SystemC language to be considered in the SWG

- **Language Working Group**
  - Suggested creation of the SystemC Datatype Sub-Working Group (SDTWG)

- **AC Datatypes addresses ALL issues that have been identified**
  - Not just speed of integer types

PROPOSAL FOR ADDING AC DATATYPES TO SYSTEMC

Proposal

- **Keep behavior/API of existing SystemC datatypes unchanged**
  - Legacy IP blocks need to work unchanged
  - APIs changes are disruptive

- **Add AC Datatype Library to SystemC**
  - Headers: include ac_sc.h in systemc.h
    - Includes all AC types
    - Provide conversion functions:
      - to_ac({sc_int, sc_uint, sc_bigint, sc_big(u)int, sc_(u)fixed})
      - to_sc({ac_int, ac_fixed})
    - Provides sc_trace so that AC types can be used for sc_signals

- **For first release of AC Datatypes, keep API as is**
  - Take input from user community for additional API extensions for second release
  - Extensions may leverage C++ 11/14

Available: Interfacing with SystemC

- **Include ac_sc.h**

- **Provides explicit conversions**
  - to_ac (from sc_bigint, sc_biuint, sc_fixed, sc_ufixed)
    - From sc_int and sc_uint could be added
  - to_sc (to sc_bigint, sc_biguint, sc_fixed, sc_ufixed)
  - For example:
    - ac_int<W,true> x = to_ac(x_in.read());
    - auto x = to_ac(x_in.read()); // C++ 11

- **SystemC tracing**

Saturation/Overflow Query

- **Saturation/Overflow handling already part of ac_fixed**
  - Querying whether overflow occurred is currently not available
- **Different use models**
  - Simulation assert
    - Easy to implement, can be made highly customizable
    - Needs definition on how much control is desired
  - Synthesizable assert
    - Needs more formal definition
- **The SDTWG could define**
  - Need some user input and validation
    - Several users have expressed interest

Workarounds for Dynamic Width and Other Differences

- **Static Width in AC Datatype is a positive**
  - Has advantages for synthesis, speed and predictable semantics
- **If Dynamic Width is required**
  - Dynamic versions can be created
- **Operators that return Dynamic Width**
  - range operator (i,j) and range(i,j)
  - Shift operators << and >>
- **to_sc can be used for ostream <<, to_string to keep exact format if required**

Advantages

- **Production Quality:**
  - Validated in production for hardware designs for more than a decade
  - IP such as the VP9 from Google relies on the AC types
- **Fast:**
  - **~100x faster** against existing sc_bigint
  - **~6x faster** than NEW PROPOSED sc_bigint
    - DCT, Slice with 32 bits or less
  - **1.5x** to **2.0x faster** than sc_int
    - Though not comparable since is not sc_int is not arbitrary length
  - **40x to 100x faster** than sc_fixed
- **Consistent: operators are all defined with consistent semantics**
  - Fully defined operator for mixing AC Datatypes and C integers

Advantages

- **Compact and Easy to Use**
  - Implemented as header files.
  - Pretty printing available for the gdb debugger
- **Clean separation of Implementation and Exposed API**
  - In contrast with SystemC datatypes that expose all the class hierarchy as the API
    - SystemC datatypes very hard to improve without impacting users
- **Fully Documented**
  - Return type for all operators (bit-width) fully documented
    - As compared to SystemC which requires to understand details of implementation
- **Traits Defined: enables writing templated IP**
  - SystemC types present many difficulties as dynamic width base classes are returned
- **Interfaces with SystemC Datatypes: Interface with legacy blocks**

Issues That Remain with SystemC

- **Even with proposed improvements to sc_bigint, it is still Still slower. Can speed up by taking more ideas from AC Datatypes, BUT**
  - Will be disruptive to existing users that already have legacy IP
  - Will take time to implement and solidify
  - Why not just use AC Datatypes then?
- **Unresolved Datatypes**
  - How about sc_fixed?
  - Will sc_int/sc_uint be obsoleted?
    - Numerous issues with semantics, undefined behavior etc.
    - sc_uint<W>(1)/ sc_int<W>(-1) = 0 !!!
  - Will the extensive set of base class, helper classes be eliminated?
  - Requires a major rewrite of LRM. Disruptive!
- **Inconsistencies between Datatypes and other gaps**
  - Shift behavior is different between sc_bigint and sc_fixed
  - ~ unary operator is different between sc_bigint and sc_fixed
  - sc_fixed has issues with how it defines the division operator

THANK YOU!