

**NASCUG VI Panel:**  
**SystemC: A User's Perspective**  
**What is Working**  
**and**  
**What Needs to be Improved**

Tor Jeremiassen, Ph.D.

Advanced Architecture & Chip Technology

Texas Instruments

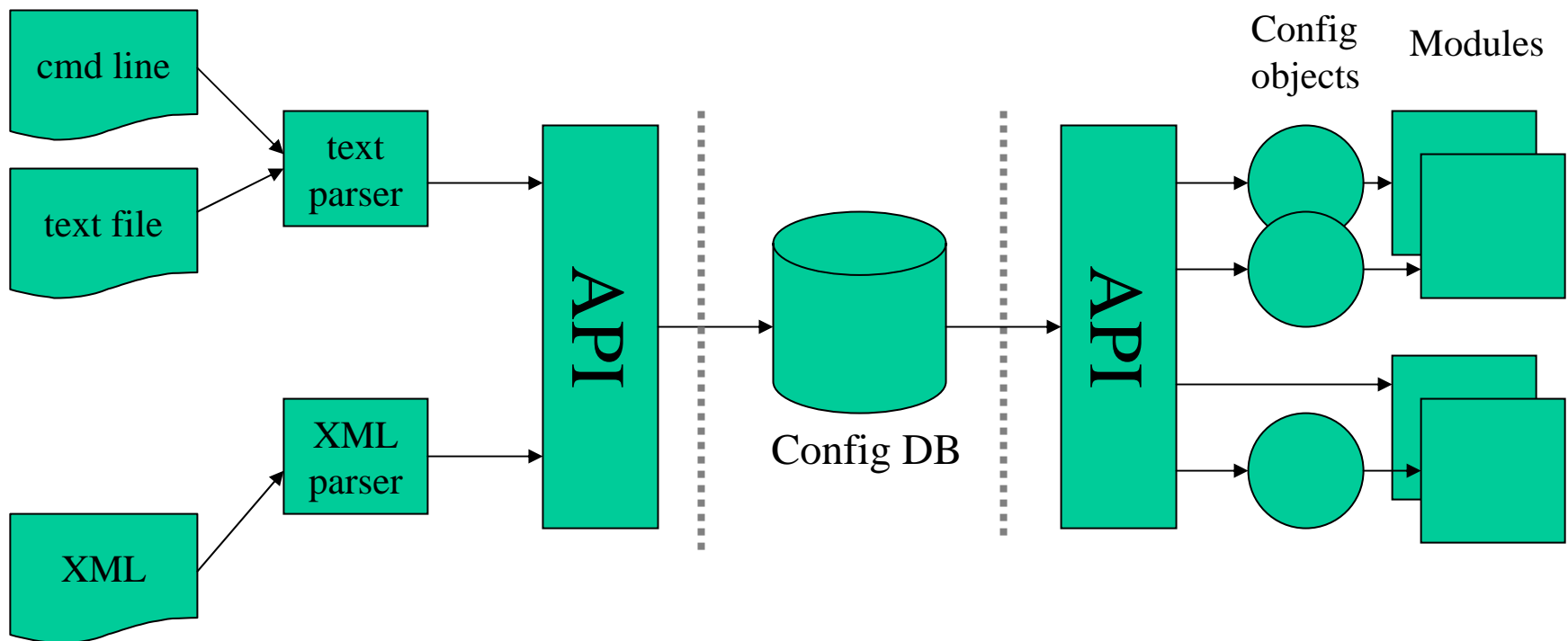
Feb 21<sup>st</sup>, 2007

# Use of SystemC

- Multiple centers of SystemC activity in TI:
  - Different goals, different methodologies:
    - High level architectural modeling (PV/PVT)
    - Low level architectural/SoC modeling (CC)
    - Synthesis (experimental)
  - Different levels of penetration
- Use of SystemC within TI has matured significantly over the last two years.
  - Internal SystemC user group meetings
- Personally: worked extensively with SystemC for 4 years.
  - Framework for modeling TI SoCs for architectural modeling and V&V purposes
    - Bus models, CPU models, peripheral models, configuration, statistics, debug

# Configuration

- Standardization on configuration mechanism is a requirement for interoperability
- Need a consumer (module) driven approach



# Statistics

- Statistics and trace objects should exist as objects visible in the SystemC object hierarchy.
  - Child objects of the enclosing module/channel/port
- Discover, traverse and obtain values using introspection calls.
  - Created locally, collected globally
  - No global a-priori knowledge of particular statistics objects
- Allow selective export to:
  - Human readable text
  - XML
  - Binary?

# Introspection

- SystemC introspection calls are inadequate:
  - Module hierarchy can be inferred
  - But connectivity (module – port – channel) is very hard to determine in a generic fashion.
    - Requires either custom ports/channels/interfaces or custom simulator environment.
- Want to be able to arbitrarily traverse design
  - Compute connectivity and evaluate “goodness”.
    - Visualize design easily.
  - Remember, design can change up until **end\_of\_elaboration()**
  - Automatically compute routing tables/address lookup
    - Which bus masters are connected to which bus slaves
    - Partially self-configuring designs

# Asynchronous Events

- SystemC has poor support for asynchronous events
  - Interrupts/resets/power-downs
  - Cooperative co-routines/threads
  - For full support would require every method and every thread to add async events to their sensitivity lists
    - Possible to use, but very inconvenient
- Need kernel support