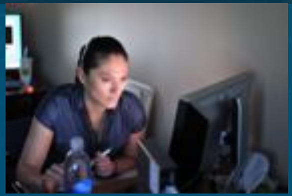
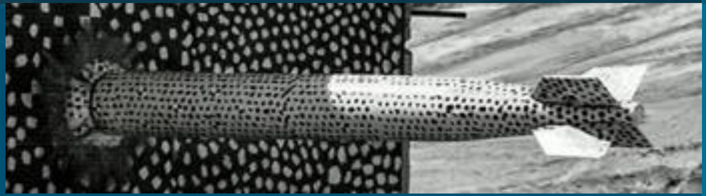
SST-Explorer

Enabling System-level Performance and Reliability Analysis for Designs with Real-World IPs





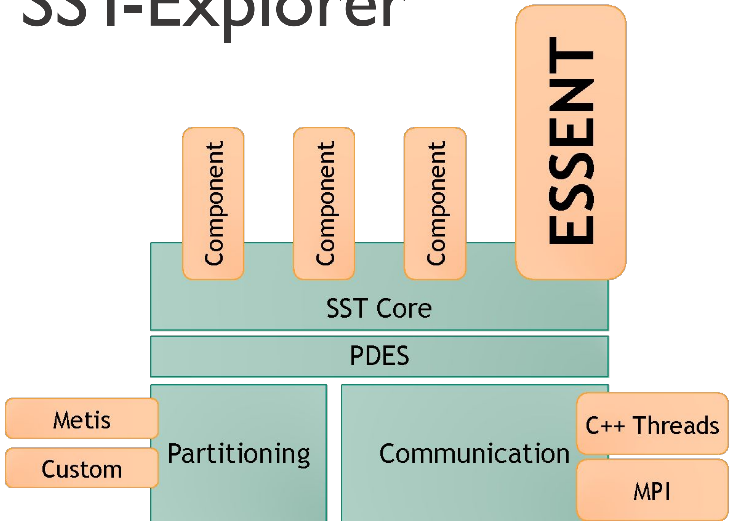
Arun Rodrigues, Amro Awad, Clayton Hughes, Sapan Agarwal, Michael Skoufis, Gwen Voskuilen, Shubham Nema, Rohin Razdan, Alan Gardner, Scott Hemmert, and Simon D. Hammond

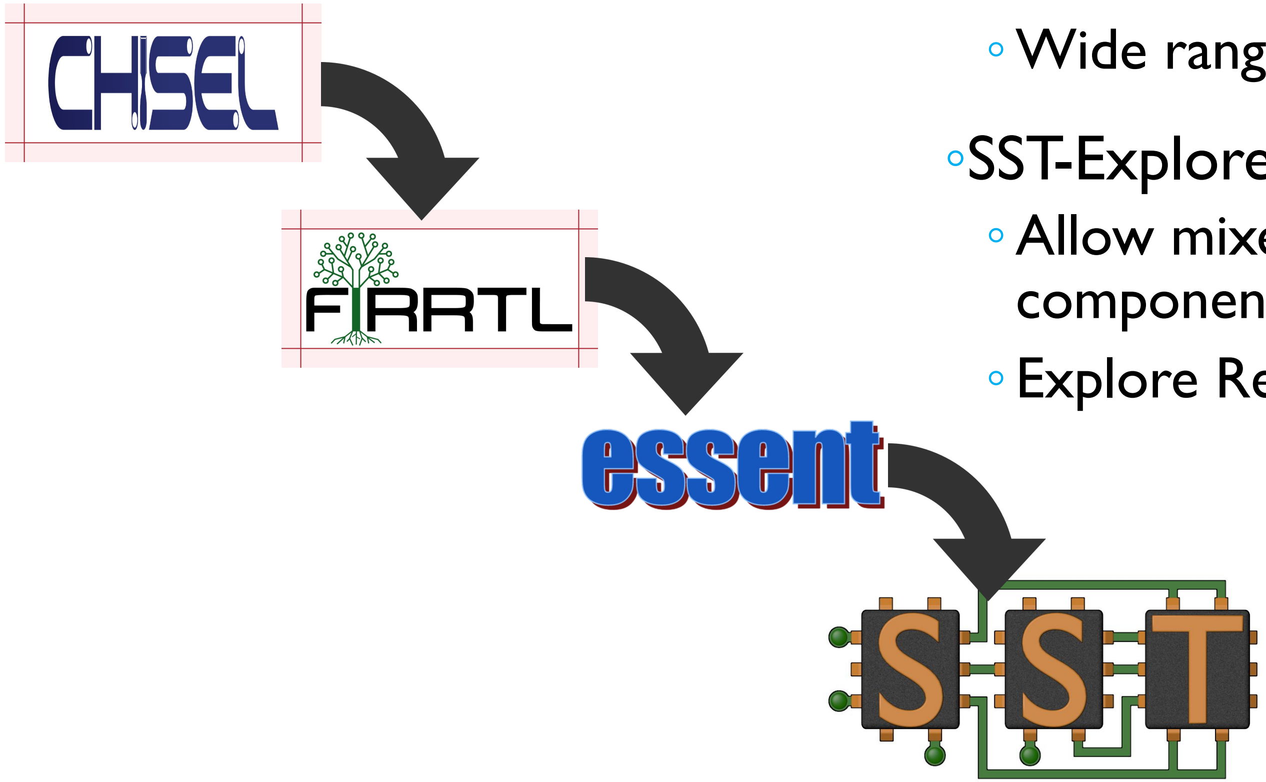
PRESENTED BY

Arun Rodrigues

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

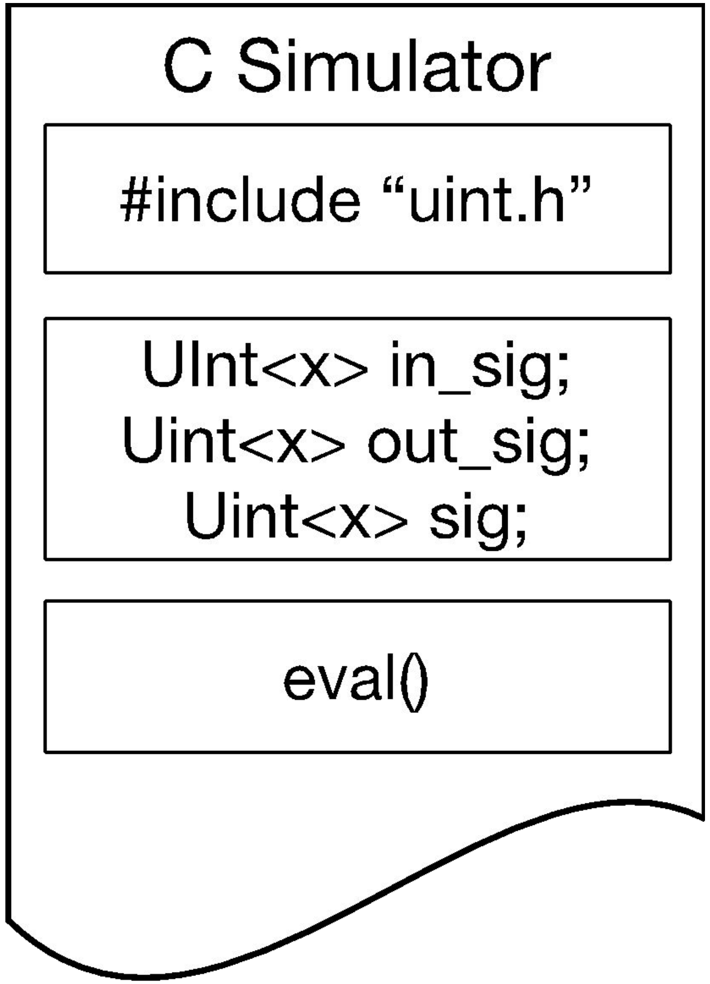
SST-Explorer





- SST: Parallel, Open, Multi-scale, Interoperable
  - SST Core framework: PDES, utilities and interfaces for simulation components
  - SST Element libraries: Libraries of components that perform the actual simulation
- C++ Models: functional to cycle-accurate
  - Wide range of models for network, processor, memory, etc…
- SST-Explorer Goals
  - Allow mixed-mode simulations that combine RTL-level components and high-level components
  - Explore Reliability with fault injection <u>and tracking</u>

ESSENT Output



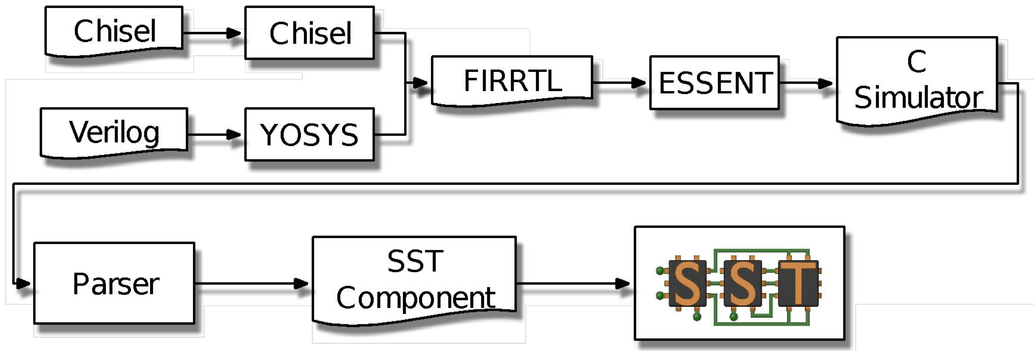
- ESSENT produces output file (.h) includes…
- Headerfile defining basic types (e.g. Uint<T>)
- List of signals (in, out, internal)
- Eval() function which does the actual simulation
- User supplies testbench wrapper code to provide input stimulus

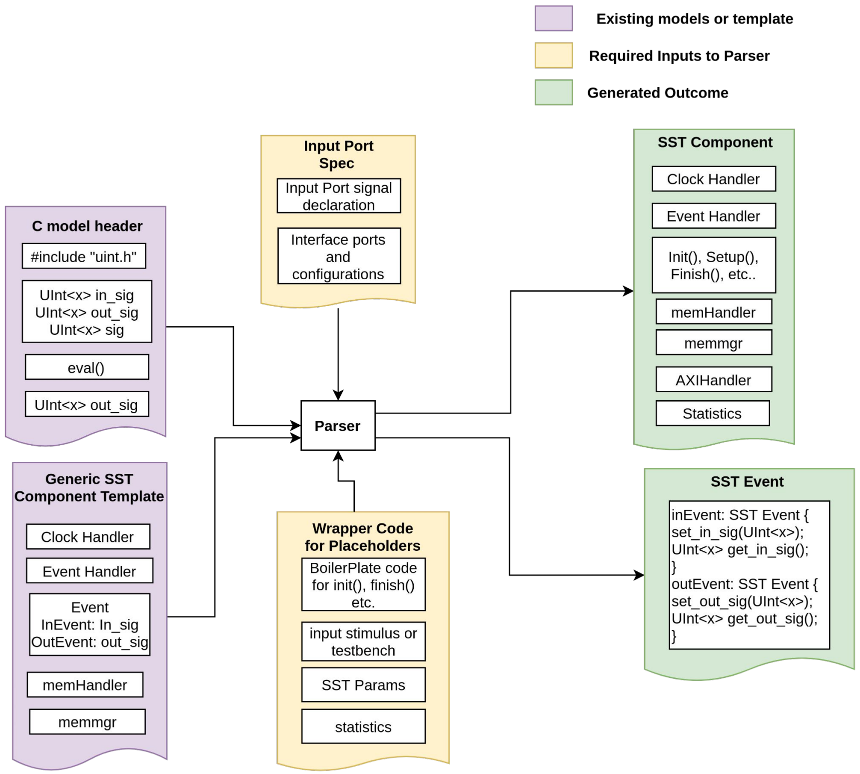
```
module Adder(
  input        clock,
  input        reset,
  input  [7:0] io_in0,
  input  [7:0] io_in1,
  output [7:0] io_out
);
  assign io_out = io_in0 +
io_in1;
endmodule
```

```
#include <uint.h>

typedef struct Adder {
  UInt<8> io_in0;
  UInt<8> io_in1;
  UInt<8> io_out;

  void eval() {
    UInt<9> _T = io_in0 + io_in1;
    io_out = _T.tail<1>();
  }
} Adder;
```

SST/ESSENT: Workflow

- SST-Explorer framework allows a simple workflow which can transform Chisel or Verilog code into an SST component
- SST-Explorer parser reorganizes the C simulator created by ESSENT in to an SST component
  - (optionally) adds fault injection & tracking capabilities
- ESSENT output + Template + user supplied code and port maps → SST Components and Events.
  - Templates: 'generic' components, UART-based, or AXI interfaces.
- Use cases
  - Fast high-level models + slow detailed models = improve simulation speed
  - High-level "placeholder" components + low-level components early in design cycle





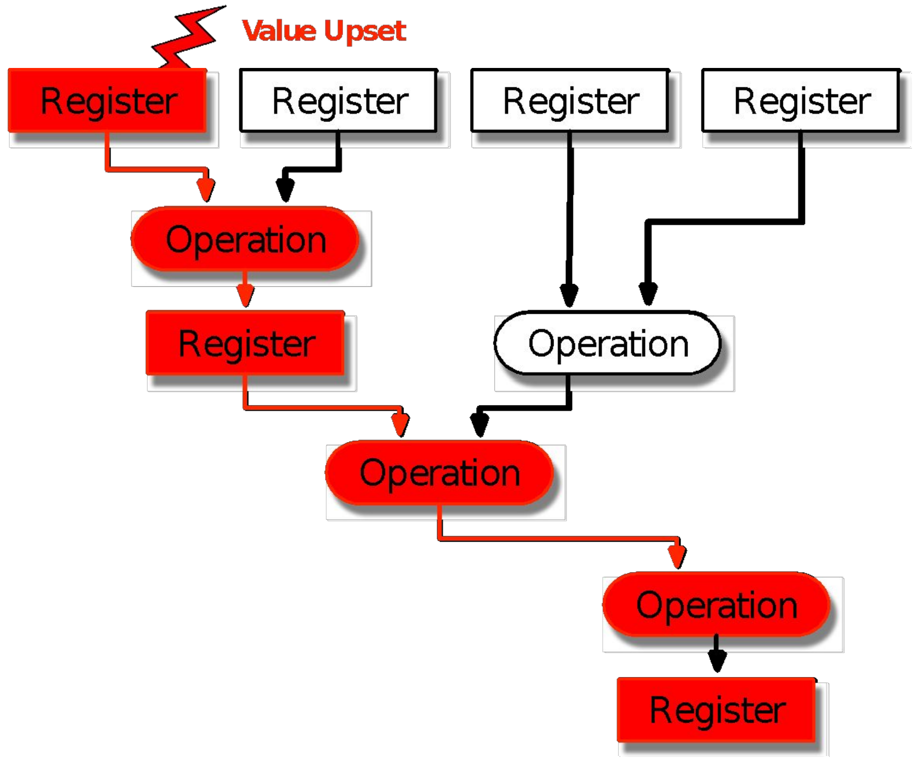
Fault Tracking

◦ SST-Explorer allows fault injection **and tracking**

◦ ESSENT Uint<T> and Sint<T> structures replaced
  - New structure stores original data, "faulty" data, and info on inciting upset
  - Operators overloaded so fault information is propagated
  - Faults are tracked and fault corrections are noted

◦ For each fault
  - Where it started
  - What it affected

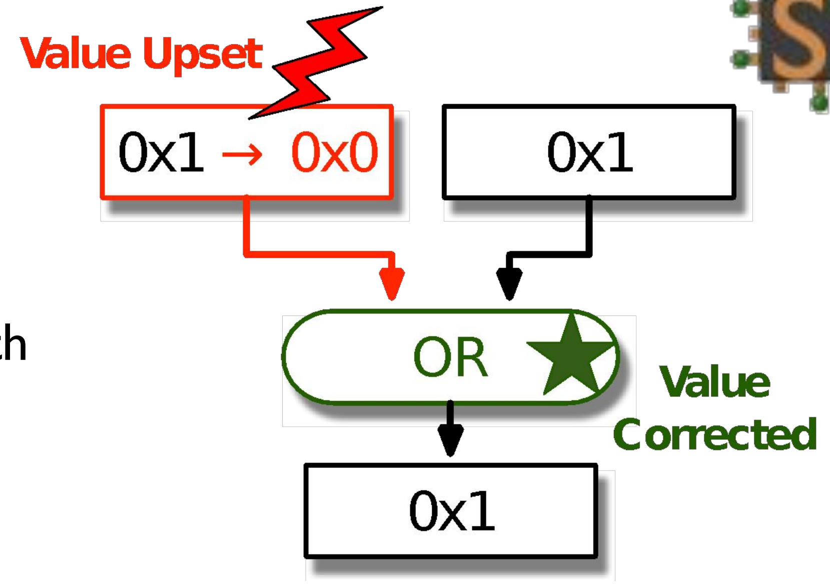
```
template <int N>
class Uint {
  Uint_<N> origData; // correct data
  Uint_<N> data;     // faulted data

  list<upsetDesc> upsets; // fault track
};
```



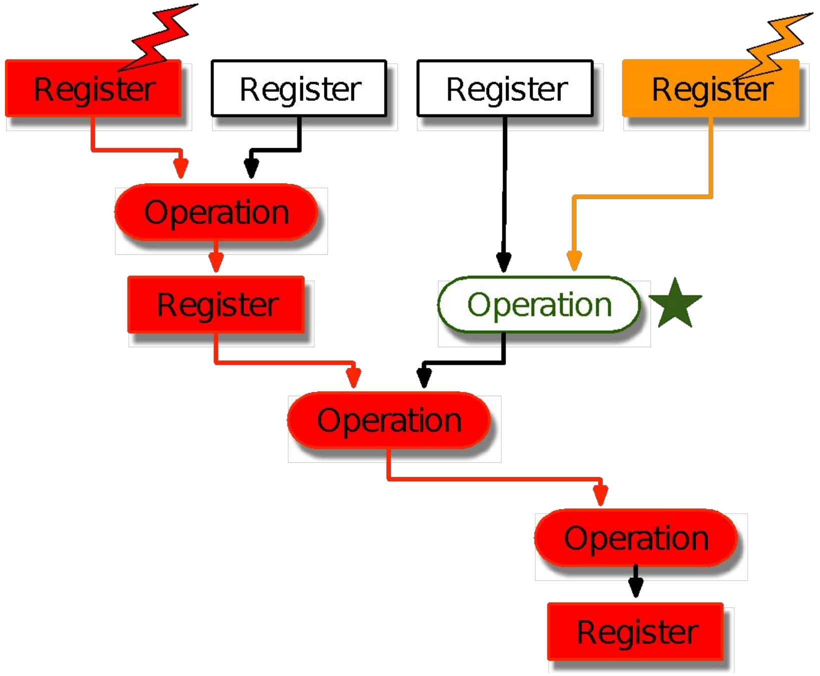
Fault Corrections & Diagnosis

- Other Use Cases
  - Detect fault corrections
    - Data struct carries 'correct' value, can determine if math operations restore faulted to correct
    - Useful for determining where faults squashed
  - Multi-fault diagnosis
    - Origin of each fault is tracked
    - Can determine which upset (of many) caused fault or error

- Summary: SST-Explorer
  - RTL models to be integrated with SST
  - Fault injection <u>& tracking</u>



Fault Correction



Multi-Fault Diagnosis