



SST (micro) Introduction

Arun Rodrigues, Si Hammond, Sue Kelly
 Scalable Computer Architecture Department
 Sandia National Laboratories
 Albuquerque, NM, USA
afrodri@sandia.gov

SST Hack-a-thon

April 11, 2013



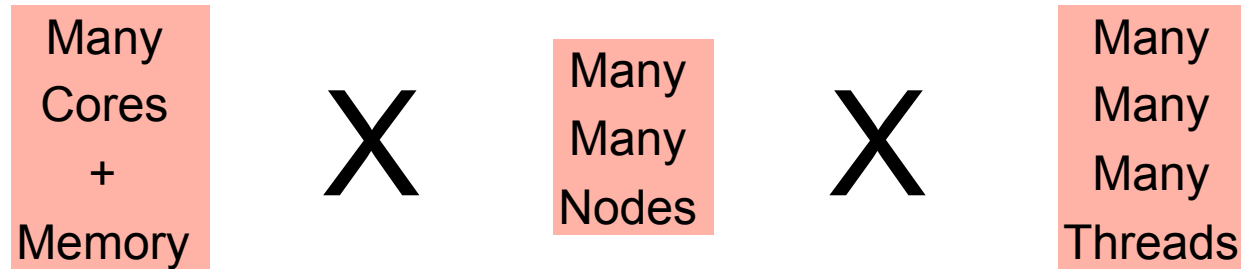
*Exceptional
 service
 in the
 national
 interest*



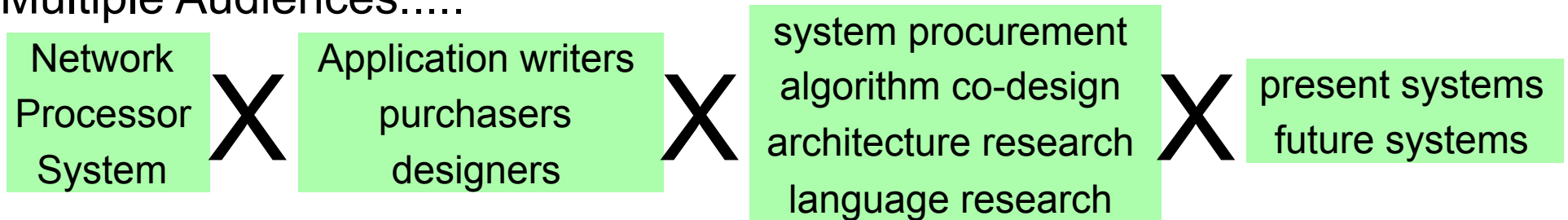
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

View of the Simulation Problem

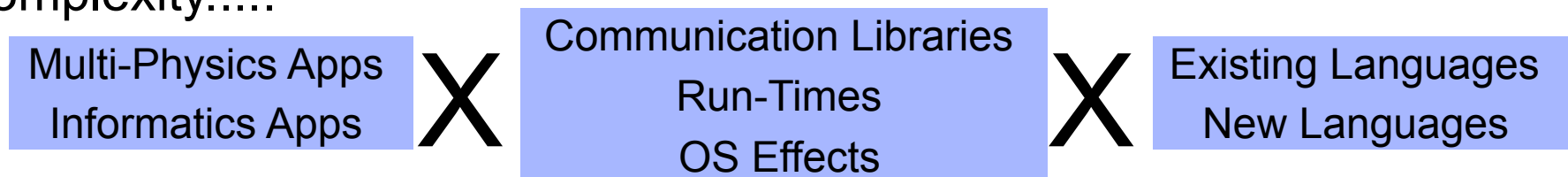
Scale.....



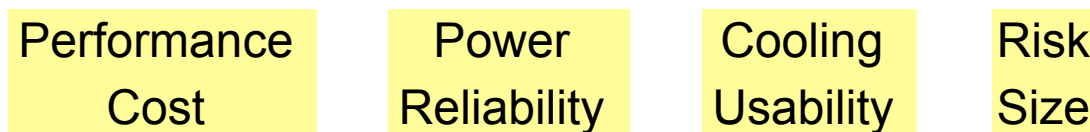
Multiple Audiences.....



Complexity.....



Constraints.....



What Is the SST?

SST Simulation Project Overview



Goals

- Become the standard architectural simulation framework for HPC
- Be able to evaluate future systems on DOE workloads
- Use supercomputers to design supercomputers

Technical Approach

- Parallel Discrete Event core with conservative optimization over MPI
- Holistic
- Integrated Tech. Models for power
- McPAT, Sim-Panalyzer
- Multiscale
- Detailed and simple models for processor, network, and memory
- Open
- Open Core, non viral, modular

Status

- Current Release (2.3) at code.google.com/p/sst-simulator/
- Includes parallel simulation core, configuration, power models, basic network and processor models, and interface to detailed memory model

Consortium

- “Best of Breed” simulation suite
- Combine Lab, academic, & industry



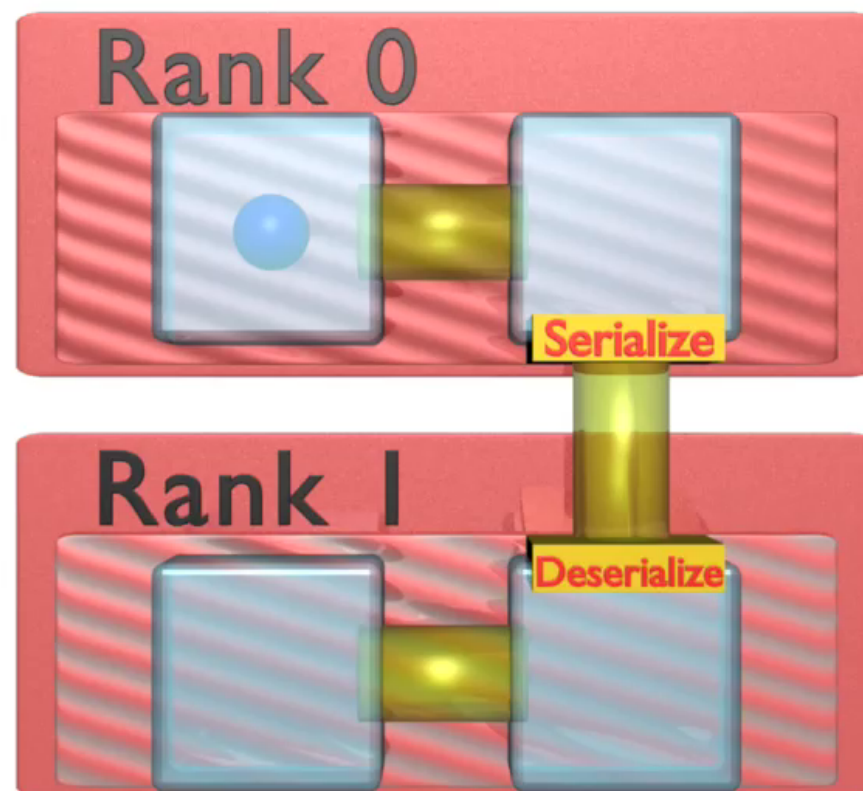
Parallel Implementation

- Implemented over MPI
- Configuration, partitioning, initialization handled by core
- Conservative, distance-based optimization

```
<component>  
<component>  
XML  
Config  
<component>  
<component>  
<component>  
<component>  
<component>
```

Message Handling

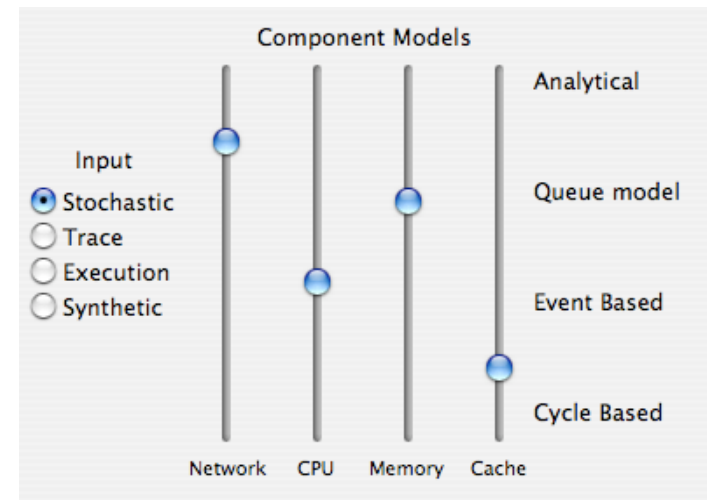
- SST core transparently handles message delivery
- Detects if destination is local or remote
- Local messages delivered to local queues
- Remote messages stored for later serialization and remove delivery
 - Boost Serialization Library used for message serialization
 - MPI used for transfer
- Ranks synchronize based on partitioning



Multi-Scale

- Goal: Enable tradeoffs between accuracy, flexibility, and simulation speed
 - No single “right” way to simulate
 - Support multiple audiences
- High- & Low-level interfaces
 - Allows multiple input types
 - Allows multiple input sources
 - Traces, stochastic, state-machines, execution...

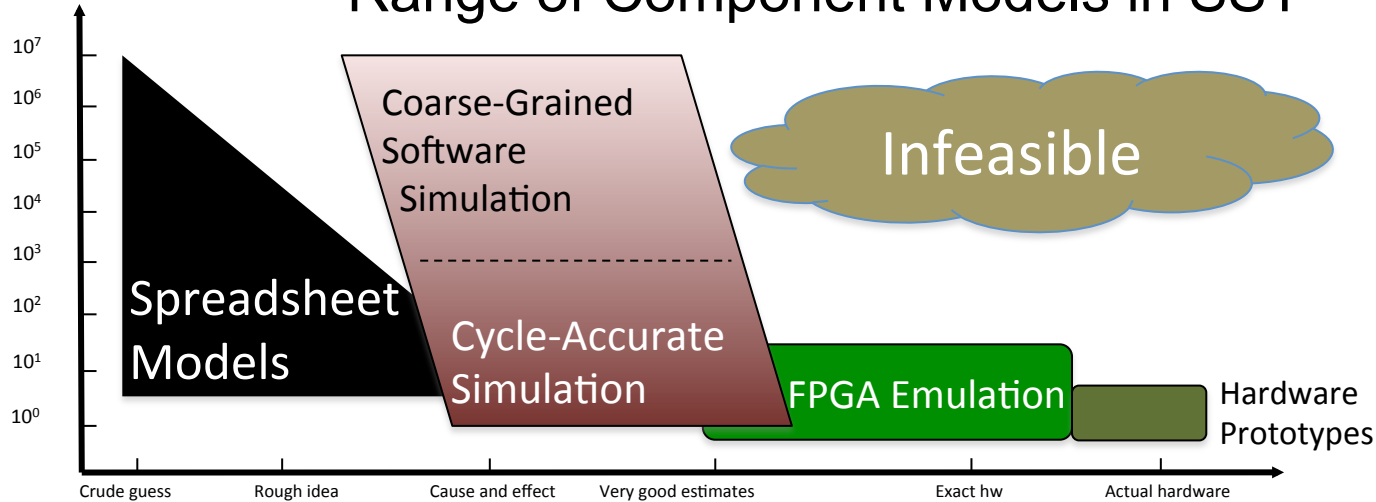
	High-Level	Low-Level
Detail	Message	Instruction
Fundamental Objects	Message, Compute block, Process	Instruction, Thread
Static Generation	MPI Traces, MA Traces	Instruction Trace
Dynamic Generation	State Machine	Execution



Multiscale Parameters

Diversity

Range of Component Models in SST



DRAMSim

M5 O3 Zesto

RS Router

Stochastic

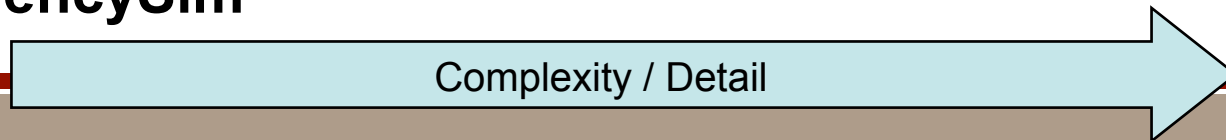
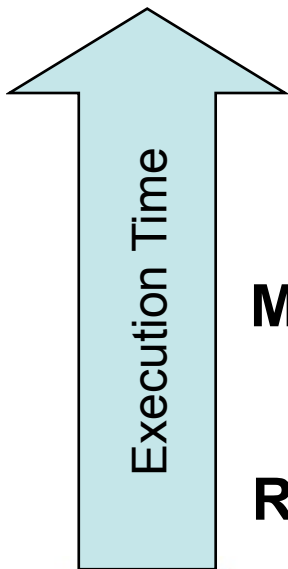
simpleRouter

- Commonalities
- Discrete Event or time stepped
 - Amenable to event counting for power modeling

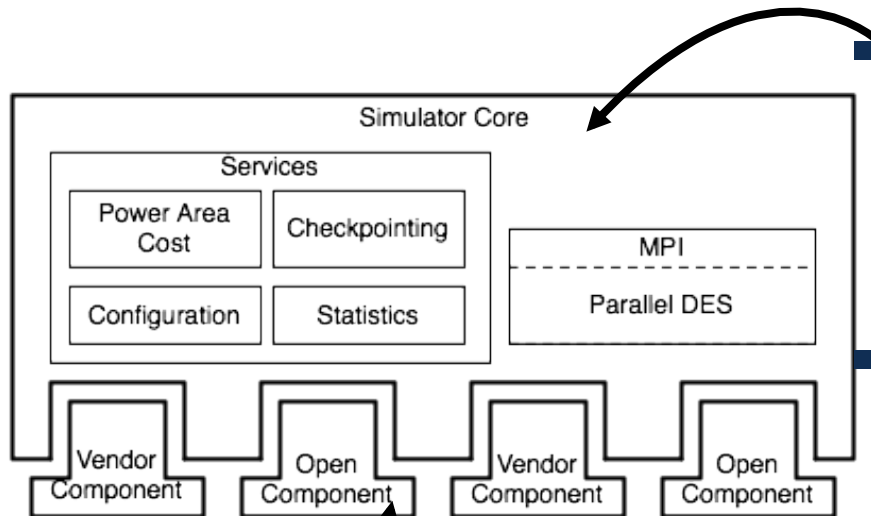
Macro

scheduleSim
ResiliencySim

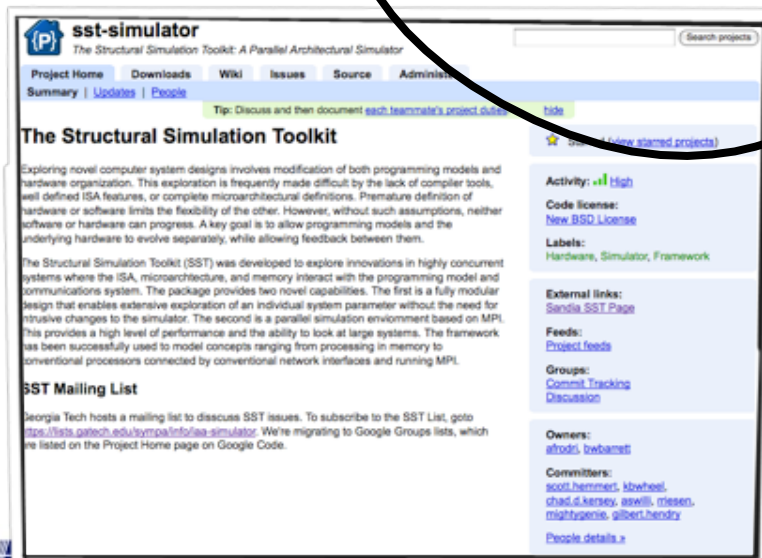
Complexity / Detail



Open Simulator Framework



- Simulator Core will provide...
 - Configuration
 - Parallel Component-Based Discrete Event Simulation
- Components
 - Ships with basic set of open components
 - Industry can plug in their own models
 - Under no obligation to share
- Open Source (BSD-like) license
- SVN hosted on Google Code



What the SST is not

Ugly Truths About Simulation

SST is not...

- Unified
 - Lots of different components, not all of which work together
- Fast
 - Simulating a processor + memory system can cause >10000:1 slowdown (i.e. 1 second = ~3 hours)
- Easy to use
 - Lots of features and knobs to turn, many poorly documented
 - Tall software stack, often changing and poorly documented
 - Designed for 'expert' users who are familiar with underlying concepts
- Easily verified...

Validation Issues

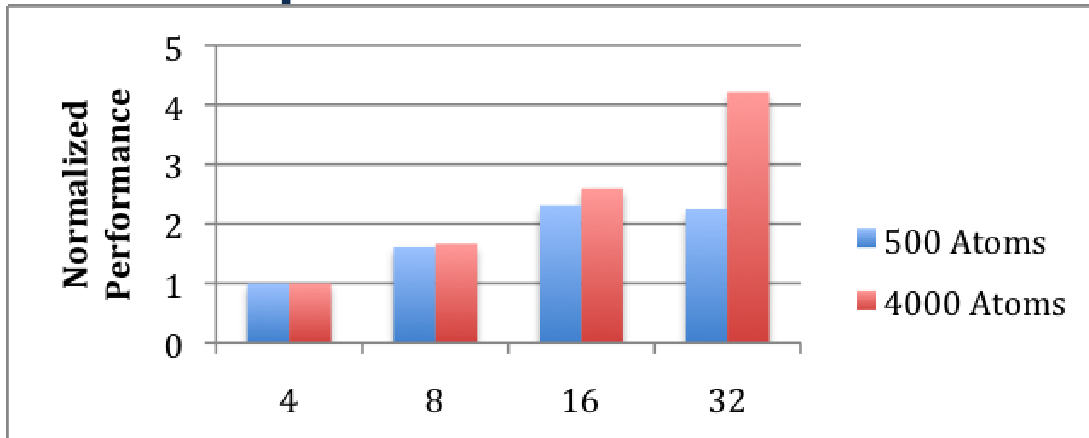
- Vendor IP considerations make building accurate models hard
 - Like simulating a physical system, but you can't know which materials are being used...
- Component & System validation is difficult, too many unknowns for high accuracy
- Current components validated at different levels, with different methodologies
- How accurate is enough?
 - Lots of disagreement
 - What happens when models disagree
- Comparison with current designs
- Use of testbeds, prototypes
- Role of UQ

Role of Simulation

- We can use roadmaps, make sure we are tracking the same trends, but must recognize the specific realization will be different
- It is not our role to simulated exactly what vendors will / should be doing
- We should be developing...
 - Abstract models which inform the vendors of our requirements and some design specifics that can make them possible
 - Models which the application teams can use to understand how future architectures will impact their applications

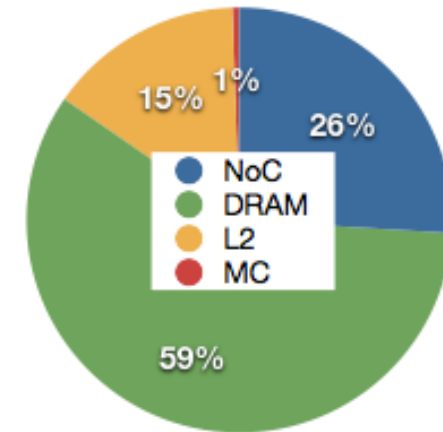
Use Cases

Sample Results & Uses

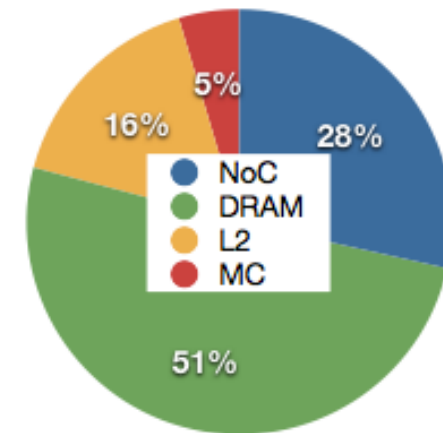


SST Simulation of MD code shows diminishing returns for threading on small data sets

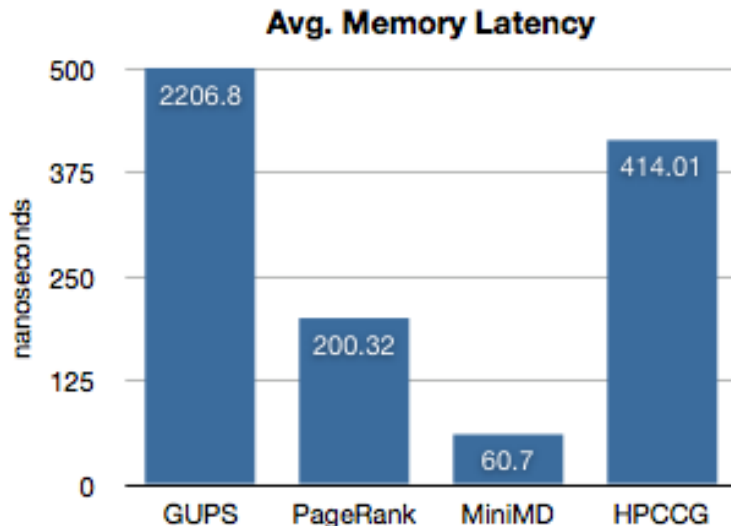
GUPS Memory Power Breakdown



MiniMD Memory Power Breakdown



Power analysis help prioritize technology investments



Detailed component simulation highlights bottlenecks

Which Memory System

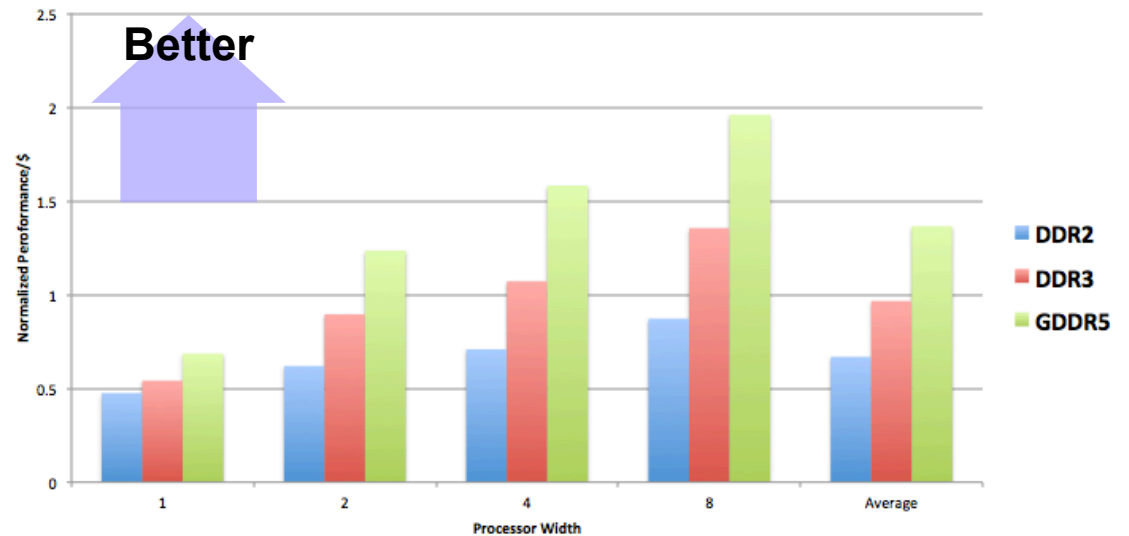
Options

- DDR2: Cheap, low power, antiquated
- DDR3: Higher performance, reasonable power
- GDDR5: Expensive, high power, very fast

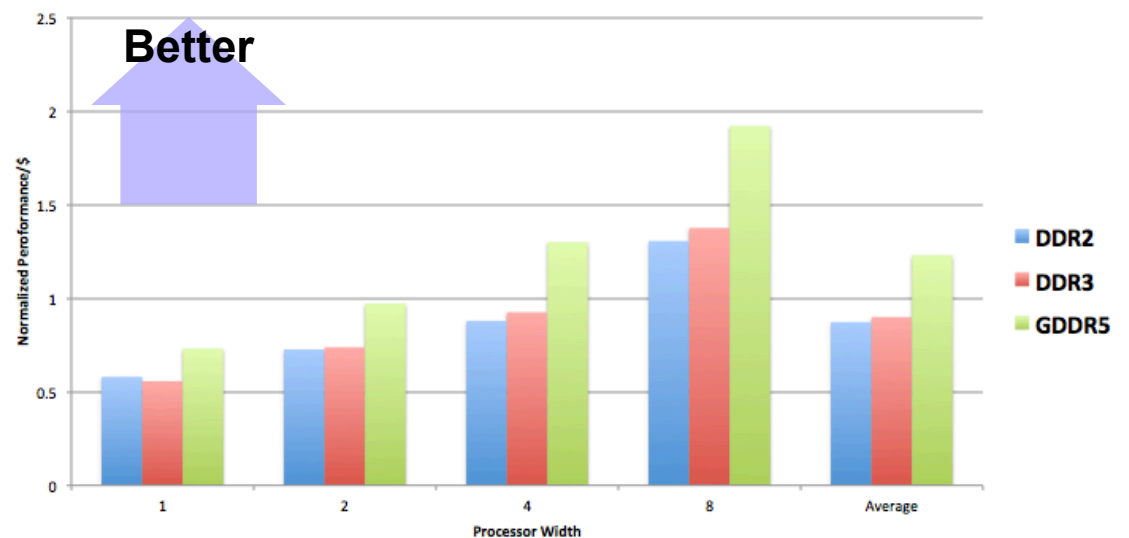
Pure performance:

- GDDR 26-47% faster than DDR3 (Lulesh)
- GDDR 32-41% faster than DDR3 (HPCCG)
- GDDR Wins?

Lulesh Performance

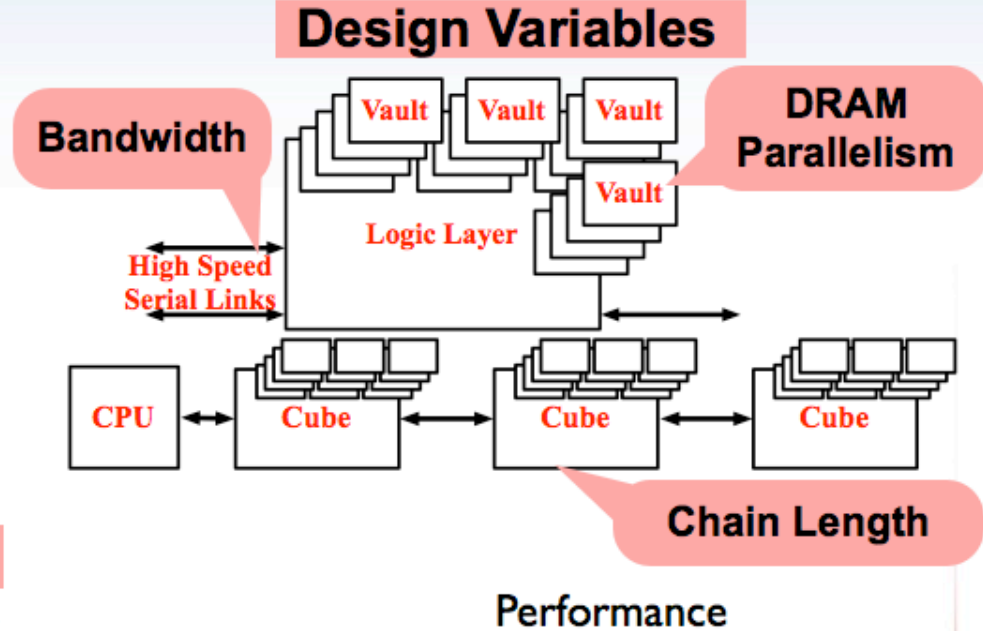
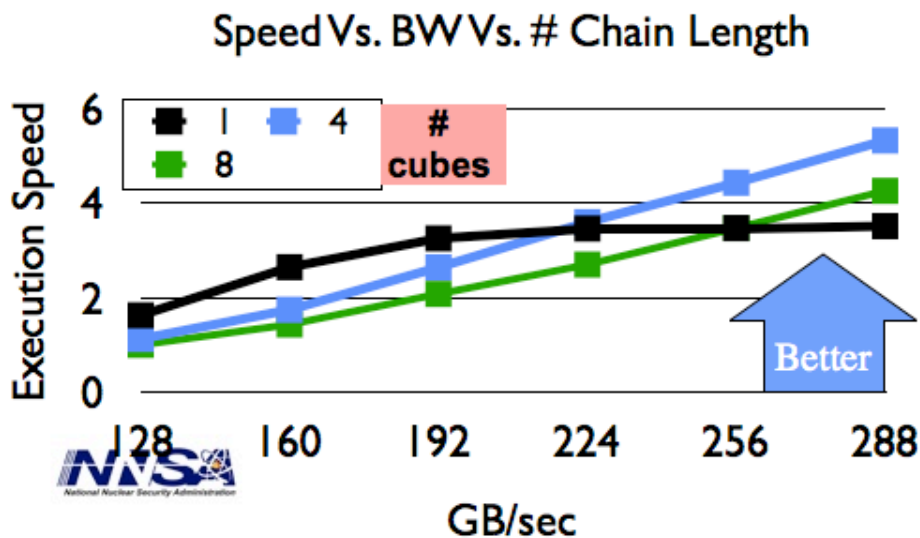
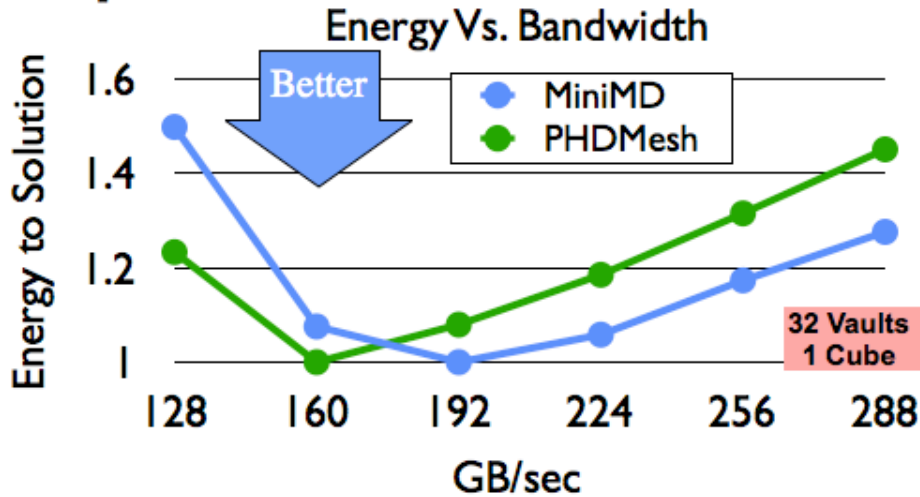


HPCCG Performance



Design Space Exploration: Stacked Memory

- Stacked Memory may offer improvements in power, performance

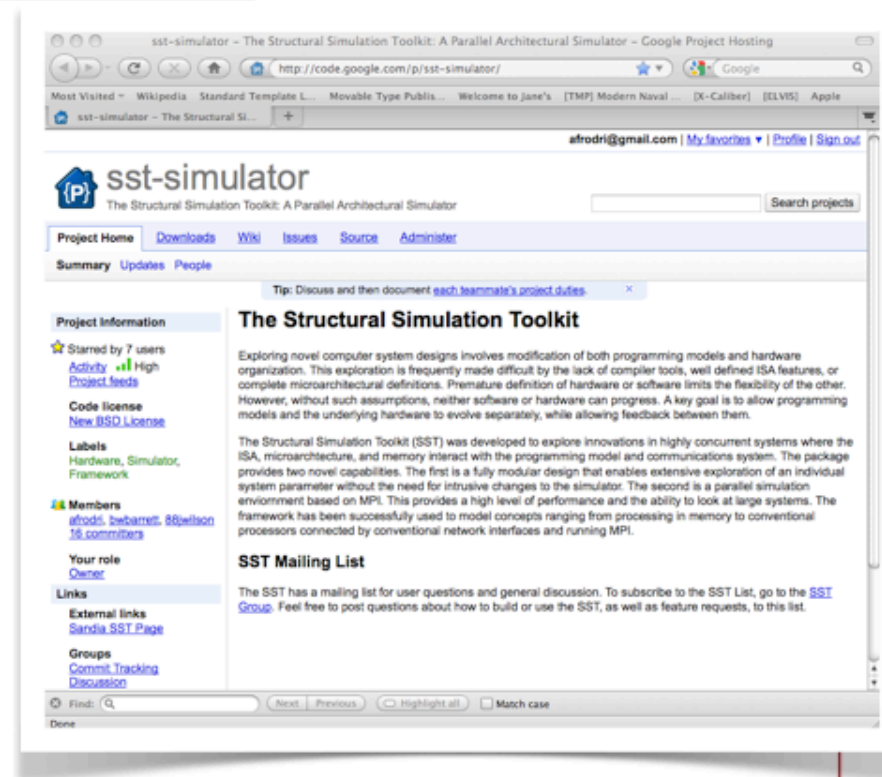



Building SST

Where to Get the SST

- **Google Code**
- **<http://code.google.com/p/sst-simulator/>**
 - Anonymous checkout available
- **BSD-like License**
- **Directory Structure**

/sst-simulator : Top Level
/deps : Scripts to help build dependencies
/doc : Doxygen
/sst : Source code
/core : Simulator Core (DES)
/elements : Simulation models
/DRAMSimC : DRAM Simulator
/M5 : GeM5 Simulator
/SS_Router : Cray SeaStar Simulator
/zesto : Detailed x86
/iris : NoC Simulator
/Many Others...





Dependencies

- **System**
 - 64-bit Linux
 - MacOS >10.6
 - **Libraries**
 - MPI Distribution (Open MPI or MPICH2)
 - Boost 1.50.x
 - ParMETIS 3.1.1 (Optional)
 - Zoltan 3.2 (Optional)
 - **Optional Components**
 - DRAMSim2
 - Qsim
 - SSTMacro
 - Palacios
 - HotSpot
- GeM5 (64-bit Linux Only)
 - **Detailed directions at:**
<http://code.google.com/p/sst-simulator/w/list>

Building

- `cd $HOME/scratch/src`
- `tar xzf sst-2.3.0.tar.gz`
- `cd $HOME/scratch/src/sst-2.3.0`
- `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<MPI library path>:$<other libraries>`
- `./autogen.sh`
- `./configure`
- `make all`
- `make install`
- `export LD_LIBRARY_PATH=$HOME/local/packages/boost-1.50/lib:$LD_LIBRARY_PATH`

Using SST

Running the SST

- Serial
 - `sst.x <SDL File>`
- Parallel (works OK on an SMP, distributed performance depends on application)
 - `mpirun -n <ranks> sst.x <SDL File>`

Sample XML Format

```
<component id="nic1" weight=0.5>
  <nicmodel>
    <params debug=1 />
    <links>
      <link id="cpu1nicmodel">
        <params lat="1" name="CPU" />
      </link>
      <link id="Router1LocalPort">
        <params lat="1" name="NETWORK" />
      </link>
    </links>
  </nicmodel>
</component>
```

Component

- Simple component-based format
- Programmatic construction in testing

```
<component id="router1" weight=0.3>
  <routermodel>
    <params hop_delay="2000" debug=1 />
    <links>
      <link id="Router1LocalPort">
        <params lat="1" name="local" />
      </link>
      <link id="H0">
        <params lat="2" name="west" />
      </link>
    </links>
  </routermodel>
</component>
```

Component Parameters

Link Specification(s)

GeM5 Example

- Two Level Configuration

- SST Configuration

- One M5 SST::Component per rank
 - Simulation variables (stop time, etc...)

- M5 Configuration

- M5 sub-components (processors, caches, busses)
 - Uses same format as SST XML
 - Subcomponent Config parameters same as GeM5

```
<sst>
  <component name=system type=m5C.M5 rank=0 >
    <params>
      <configFile>exampleM5.xml</configFile>
      <debug> 0 </debug>
      <M5debug> none </M5debug>
      <info>yes</info>
      <registerExit>yes</registerExit>
    </params>
  </component>
</sst>
```

```
<component name=nid0.cpu0 type=03Cpu >
  <params include=cpuParams>
    <base.process.cmd>${M5_EXE}</
base.process.cmd>
    <base.process.env.0>RT_RANK=0</
base.process.env.0>
  </params>
  <link name=nid0.cpu-dcache port=dcache_port/>
  <link name=nid0.cpu-icache port=icache_port/>
</component>

<component name=nid0.cpu0.dcache type=BaseCache >
  <params include=cacheParams>
    <size>65536</size>
  </params>
  <link name=nid0.cpu-dcache port=cpu_side/>
  <link name=nid0.dcache-bus port=mem_side/>
</component>
```

Sample XML Format

```
<component id="nic1" weight=0.5>
```

```
  <nicmodel>
```

```
    <params debug=1 />
```

```
    <links>
```

```
      <link id="cpu1nicmodel">
```

```
        <params lat="1" name="CPU" />
```

```
      </link>
```

```
      <link name="Router1LocalPort">
```

```
        <params lat="1" name="NETWORK" />
```

```
      </link>
```

```
    </links>
```

```
  </nicmodel>
```

```
</component>
```

```
<component id="router1" weight=0.3>
```

```
  <routermodel>
```

```
    <params hop_delay="2000" debug=1 />
```

```
    <links>
```

```
      <link name="Router1LocalPort">
```

```
        <params lat="1" name="local" />
```

```
      </link>
```

```
      <link id="H0">
```

```
        <params lat="2" name="west" />
```

```
      </link>
```

```
    </links>
```

```
  </routermodel>
```

```
</component>
```

Link
Name

Link
Local
Name

- Named Links
 - Endpoints specified
 - Minimum Latency specified
 - Used in Partitioning & Optimization
- Variables & Parameter Blocks
- Environment Variables

```
<variables>
  <nic_link_lat> 200ns </nic_link_lat>
  <rtr_link_lat> ${LAT}</rtr_link_lat>
</variables>

<link name=0.cpu2nic
port=cpu
latency=$nic_link_lat/>
```

Gem5 Parameters

Cache Parameters

```
<L3CacheParams>  
  <size> 33554432 </size>  
  <assoc> 16 </assoc>  
  <latency> 50 </latency>  
  <mshrs> 24 </mshrs>  
</L3CacheParams>
```

Core Parameters

```
<o3cpu.fetchWidth> 4 </o3cpu.fetchWidth>  
<o3cpu.decodeWidth> 4 </o3cpu.decodeWidth>  
<o3cpu.renameWidth> 4 </o3cpu.renameWidth>  
<o3cpu.dispatchWidth> 4 </o3cpu.dispatchWidth>  
<o3cpu.issueWidth> 4 </o3cpu.issueWidth>  
<o3cpu.wbWidth> 4 </o3cpu.wbWidth>  
<o3cpu.wbDepth> 1 </o3cpu.wbDepth>  
<o3cpu.commitWidth> 4 </o3cpu.commitWidth>  
<o3cpu.squashWidth> 4 </o3cpu.squashWidth>  
<o3cpu.issueToExecuteDelay> 2 </o3cpu.issueToExecuteDelay>  
<o3cpu.numIQEntries> 64 </o3cpu.numIQEntries>
```

DRAMSim Parameters

In SST xml

```
<systemIniFilename> system.ini </systemIniFilename>  
<deviceIniFilename>  
  DDR3_micron_32M_8B_x4_sg125.ini  
</deviceIniFilename>  
<pwd> ../../DRAMSim2/ </pwd>
```

System.ini

```
NUM_CHANS=1  
TRANS_QUEUE_DEPTH=32  
CMD_QUEUE_DEPTH=32  
ROW_BUFFER_POLICY=open_page  
ADDRESS_MAPPING_SCHEME=scheme2
```

Device File

```
NUM_BANKS=8  
  
A series of Byzantine timing parameters
```

Design Space Exploration Example

- Design Space Exploration

- Inputs

- Memory technology (DDR3, GDDR5)
 - Core width (1,2,4,8- wide issue)
 - Cache size (4i+4d/8K, 2i+4d/8K, 4i+2d/8K)

Absurdly small for demo purposes

- Methodology

- Performance models: GeM5/x86, DRAMSim2

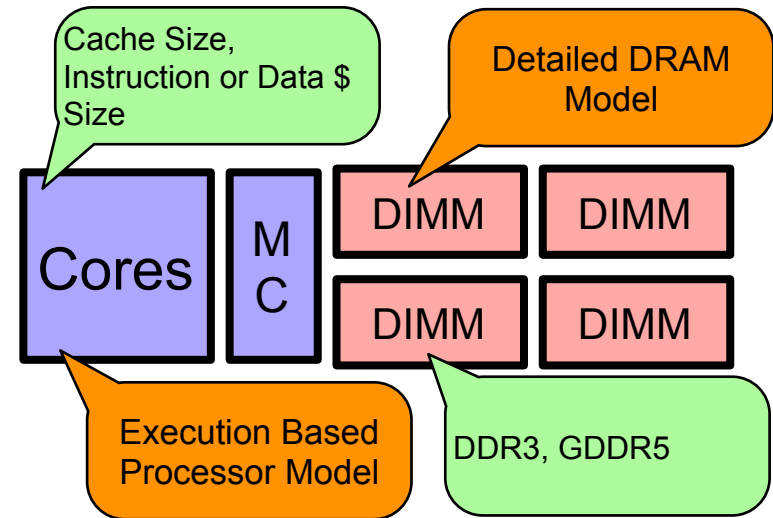
- Key Questions

- What is good cache size? Core Width?
 - Which DRAM technology?

- Example of sorts of questions simulation

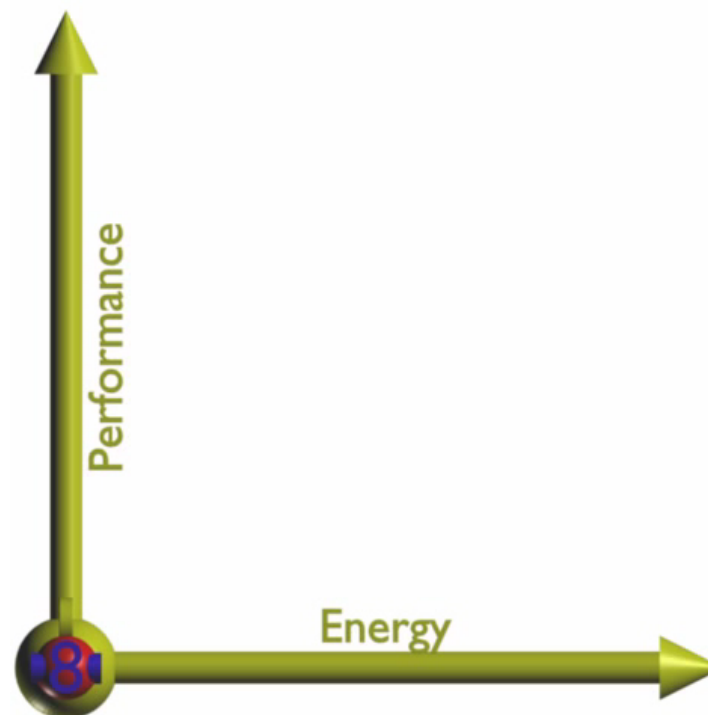
can answer

Core Width	1	2	4	8
*Width	1	2	4	8
issueToExecuteDelay	2	4	4	5
numIQEntries	32	32	64	64



Running an Experiment

- `/home/projects/sst/explore/configs`
- Edit `explore.sh` (set variables to executable, arguments)
 - `makeCfg.pl` : Makes configuration directories
 - `runCfgs.pl` : Runs 'em (takes a while)
 - `gatherResults.pl` : gets results (IPC, power, cost estimates)
 - `pareto` : finds pareto-optimal configurations



Results

Pareto	Config	power	perf	cost
Pareto	1-ddr3-large	33	32.2	238.60
Pareto	1-ddr3-smallI	32	31.9	236.04
Pareto	1-ddr3-smallD	32	31.2	230.28
Pareto	1-gddr-large	48	33.0	411.40
Pareto	1-gddr-smallI	47	32.7	408.84
Pareto	1-gddr-smallD	47	32.1	403.08
Pareto	2-ddr3-large	41	35.5	249.19
Pareto	2-ddr3-smallI	40	35.2	246.62
Pareto	2-ddr3-smallD	40	34.6	240.82
Pareto	2-gddr-large	56	36.3	421.99
Pareto	2-gddr-smallI	55	36.1	419.42
Pareto	2-gddr-smallD	55	35.6	413.62
Pareto	4-ddr3-large	53	52.0	279.06
Pareto	4-ddr3-smallI	53	51.2	276.45
Pareto	4-ddr3-smallD	52	50.6	270.56
Pareto	4-gddr-large	68	53.7	451.86
Pareto	4-gddr-smallI	68	52.7	449.25
Pareto	4-gddr-smallD	67	52.1	443.36
Pareto	8-ddr3-large	84	47.4	374.61
Pareto	8-ddr3-smallI	83	46.2	371.90
Pareto	8-ddr3-smallD	83	46.4	365.78
Pareto	8-gddr-large	99	48.4	547.41
Pareto	8-gddr-smallI	98	47.4	544.70
Pareto	8-gddr-smallD	98	47.4	538.58

DRAMSim Results

Total Return Transactions : 5141 (329024 bytes) aggregate average bandwidth
2.451GB/s

-Rank 0 :

-Reads : 2886 (184704 bytes)

-Writes : 2255 (144320 bytes)

-Bandwidth / Latency (Bank 0): 0.311 GB/s 101.213 ns

-Bandwidth / Latency (Bank 1): 0.313 GB/s 101.549 ns

-Bandwidth / Latency (Bank 2): 0.313 GB/s 96.461 ns

-Bandwidth / Latency (Bank 3): 0.304 GB/s 91.569 ns

-Bandwidth / Latency (Bank 4): 0.302 GB/s 109.342 ns

-Bandwidth / Latency (Bank 5): 0.305 GB/s 110.077 ns

-Bandwidth / Latency (Bank 6): 0.302 GB/s 97.368 ns

-Bandwidth / Latency (Bank 7): 0.301 GB/s 102.829 ns

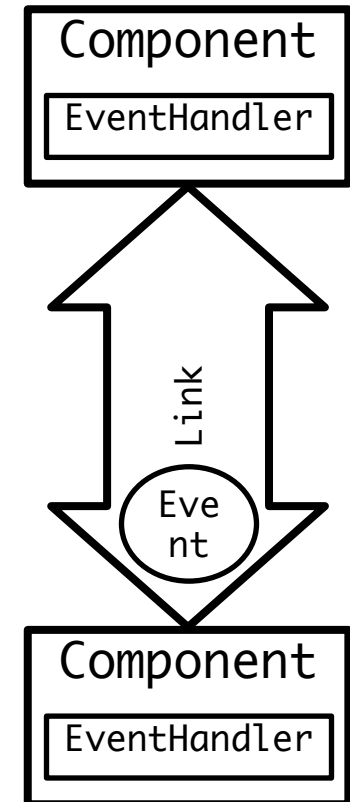
Other Applications

Graph 500 (seq-csr sz3)	Graph 500 (size 5)	Stream
1-ddr3-large	1-ddr3-small1D	1-ddr3-small1D
1-ddr3-small1D	2-ddr3-large	1-gddr-small1D
2-ddr3-small1D	2-ddr3-small1D	2-ddr3-large
4-ddr3-large	4-ddr3-large	4-gddr-large
		4-gddr-small1D
		8-gddr-large
<ul style="list-style-type: none"> Higher width processor (8) actually slower due to pipeline length DDR memory sufficient due to very small problem size Smaller caches can save \$, power, but better to decrease data than instruction cache 	<ul style="list-style-type: none"> Similar to size 3 Able to exploit slightly more parallelism 8-wide core still not worth it 	<ul style="list-style-type: none"> GDDR memory gives better performance Uses a lot of power, \$ Bigger processors = better performance Fewer dependencies mean long pipeline is OK

SST Internals

Key Objects & Interfaces

- Goal: Simplicity
- Objects
 - SST::Component: A model of a hardware component
 - SST::Link: A connection between two components
 - SST::Event: A discrete Event
 - SST::EventHandler: Function to handle an incoming event or clock tick
- Events
 - SST::Component::ConfigureLink(): Registers a link and (optionally) handler
 - SST::Link::Recv(): Pull an event from a link
 - SST::Link::Send(): Send an event down a link
 - SST::Component::registerClock(): Register a clock and handler



Components

- Built as (dynamic) library which is loaded into SST at runtime
- Constructor
 - Passed parameters
 - Initializes clock handlers, links
 - registerExit()
- Registration
 - ElementInfoComponent: Allows SST to query which components are in a library
 - Create_XXX() : C function called to create a component instance
- Look at simpleComponent, simpleRNGComponent

Simple Event Handler

```
bool Routermodel::handle_events(Time_t time, Event *event)
{
    NicEvent *e= static_cast<NicEvent *>(event);

    e->router_delay += hop_delay;
    if (e->routeX > 0)    {
        // Keep going East
        e->routeX--;
        east_port->Send(e);
    } else if (e->routeX == 0)    {
        if (e->routeY > 0)    {
            // Go South
            e->routeY--;
            south_port->Send(e);
        } else if (e->routeY == 0)    {
            // We have arrived!
            local_port->Send(e);
        } else /* e->routeY < 0 */    {
            // Go North
            e->routeY++;
            north_port->Send(e);
        }
    } else /* e->routeX < 0 */    {
        // Keep going West
        e->routeX++;
        west_port->Send(e);
    }

    return false;
}
```

Multiple Ways of Handling Events

•“Poll” a Link

- Link::Recv()
- Usually called from a clock
- Allows multiple events to be pulled in same function

•Register event handler with link

- handler called whenever event arrives

•Register a Clock

- Called at regular interval

```
Constructor {  
    cpu= LinkAdd( "port0" );  
    eventHandler = new EventHandler< Xbar, bool, Time_t, Event* >  
                    ( this, &Xbar::processEvent );  
    nic= LinkAdd( "port1", eventHandler );  
    clockHandler = new EventHandler< Xbar, bool, Cycle_t, Time_t >  
                    ( this, &Xbar::clock );  
    ClockRegister( frequency, clockHandler );  
}  
....  
CompEvent *e = cpu->Recv();
```

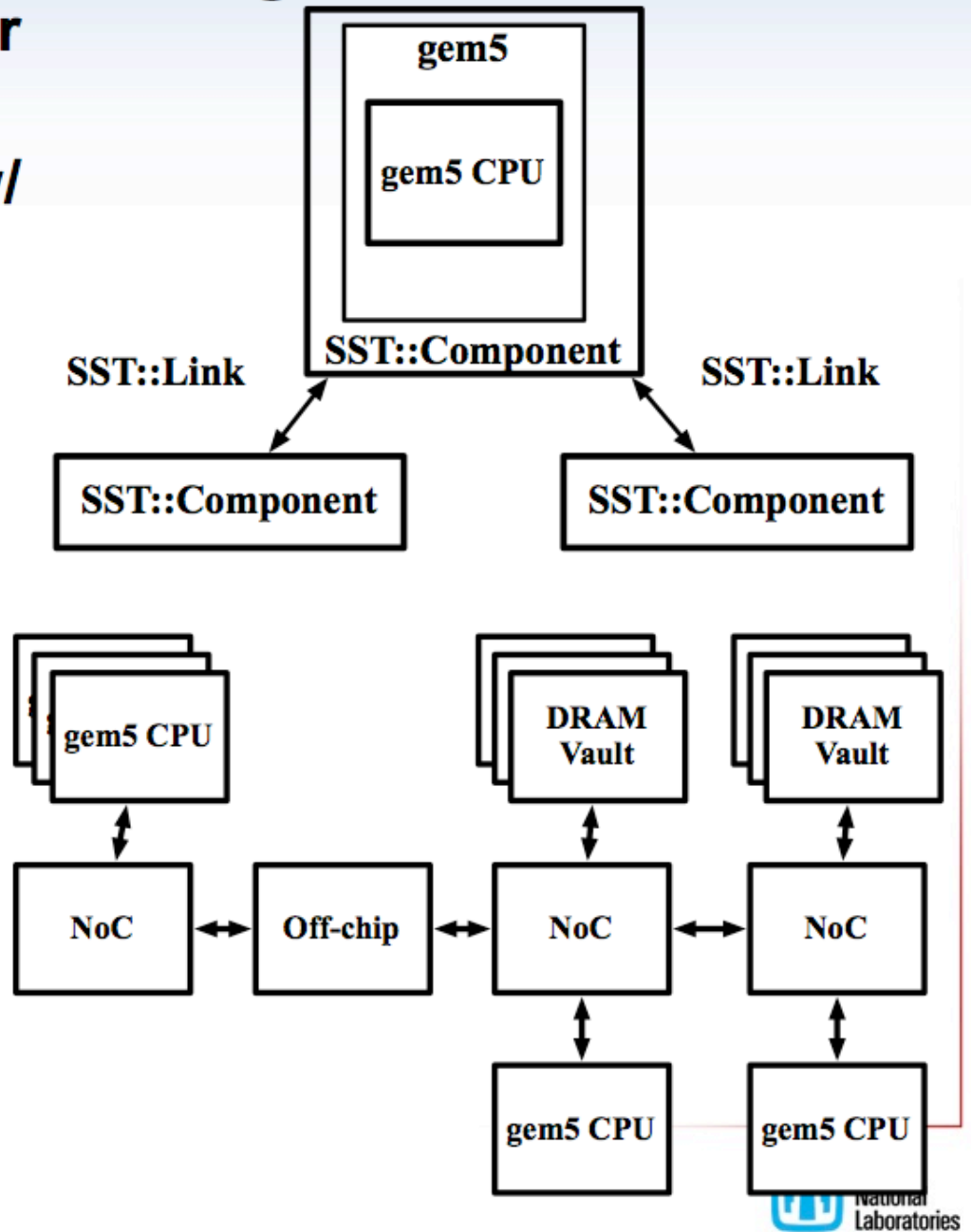
Bonus Slides & Future Directions

Parallelism

- MPI
 - Waiting on new network components, gem5 reorganization
- Thread-level
 - Some pthreads support
 - Issues with atomics
 - Qthreads support planned
 - Will allow OpenMP

Future gem5 Integration

- Encapsulate gem5 at lower level
- Allow tighter integration w/ DRAM, network models
- Enable PIM Experiments
- Issues
 - functional accesses for system calls, thread management
 - Turn into timed accesses?
 - Cache coherency model
 - Need flexibility, parallelism
 - Roll our own?



Future Network Efforts

- Merlin NIC / Router
- Portals-like NIC Interface
 - Allows protocol offload experiments
 - Good low-level interface
 - Leverage existing Portals-based MPI implementations
 - Coherent Interface to gem5 Models
- Router
 - Models internal queues, link and cross-bar contention
 - Multiple topologies
 - Adaptive Routing

MacSim Example

- Each MacSim instance is an SST::Component
- Clock speed, Paths to traces, configuration specified in SDL
- Two Modes
 - Standalone
 - w/ DRAMSim2

```
<component name=gpu0
  type=macsimComponent.macsimComponent>
  <params>
    <paramPath></paramPath>
    <tracePath></tracePath>
    <outputPath></outputPath>
    <clock>1.4Ghz</clock>
  </params>
  <link name=membus port=bus latency=1ns />
</component>
```

```
<component name=mem0 type=DRAMSimC >
  <params>
    <clock> 1066 Mhz </clock>
    <systemini> system-2.ini </systemini>
    <deviceini>GDDR5_hynix_1Gb.ini</
deviceini>
  </params>
  <link name=membus port=bus latency=1ns />
</component>
```

MacSim+DRAMSim in SST