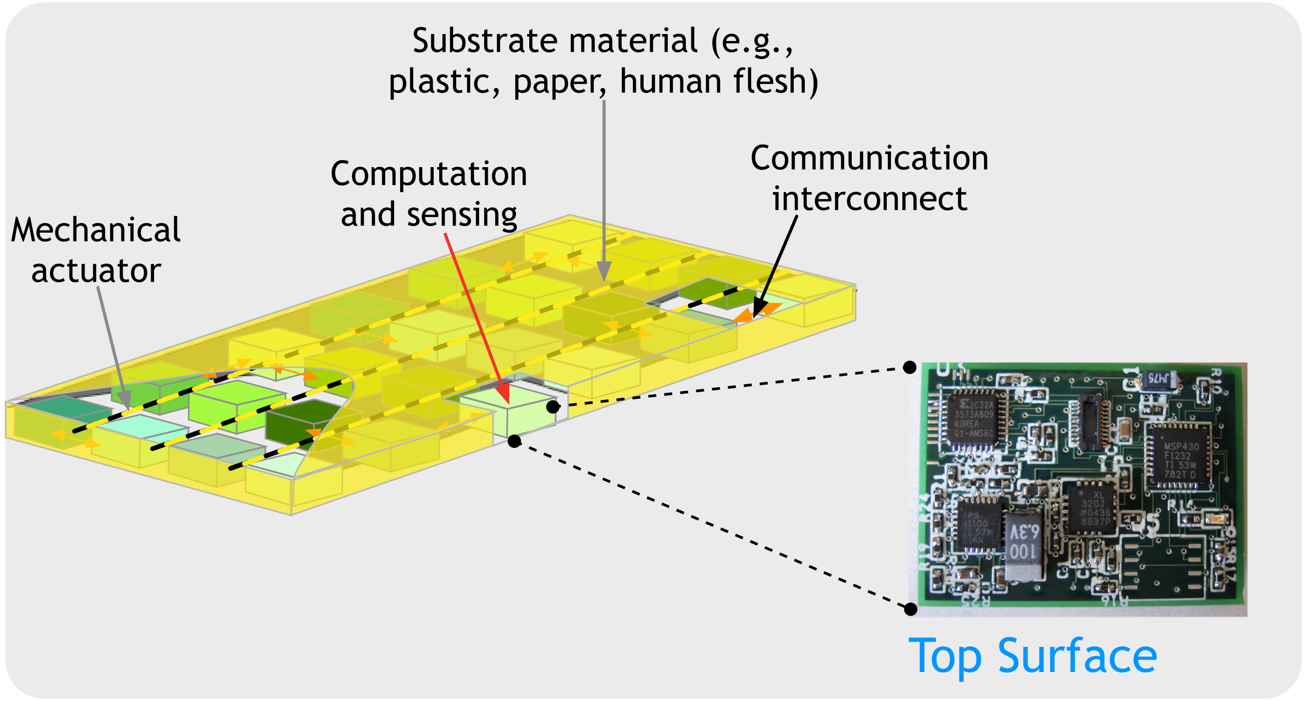
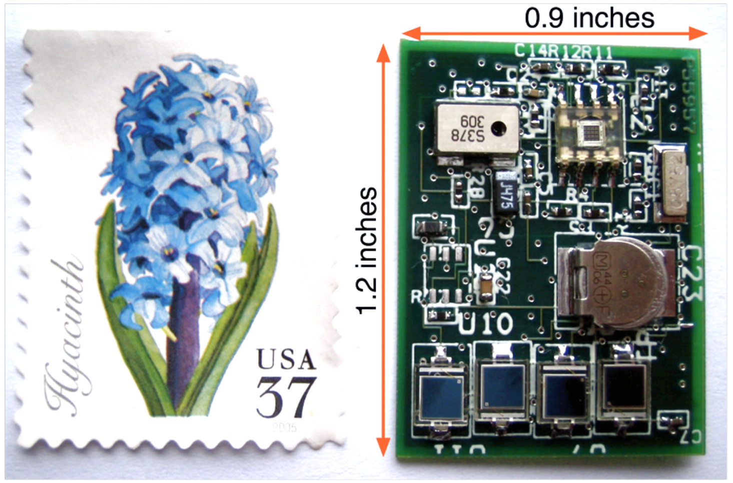
Implementation of a Distributed Full-System Simulation Framework as a Filesystem Server

Phillip Stanley–Marbell
Carnegie Mellon University

Motivation


Substrate material (e.g., plastic, paper, human flesh)
Computation and sensing
Communication interconnect
Mechanical actuator

Top Surface



Bottom Surface

[“An 0.9x1.2 Energy-Harvesting System with Custom Multi-Channel Communication Interface”, *IEEE DATE’07*, Nice, France]

- Context
  - Investigating highly-integrated networks of compute/sensing/actuation systems
  - We currently cannot afford ($) to build systems with thousands of nodes
  - Simulation permits the investigation of large scale systems
  - Simulation is not a substitute for actual hardware
- Challenge: simulating large scale (thousand+ node) systems

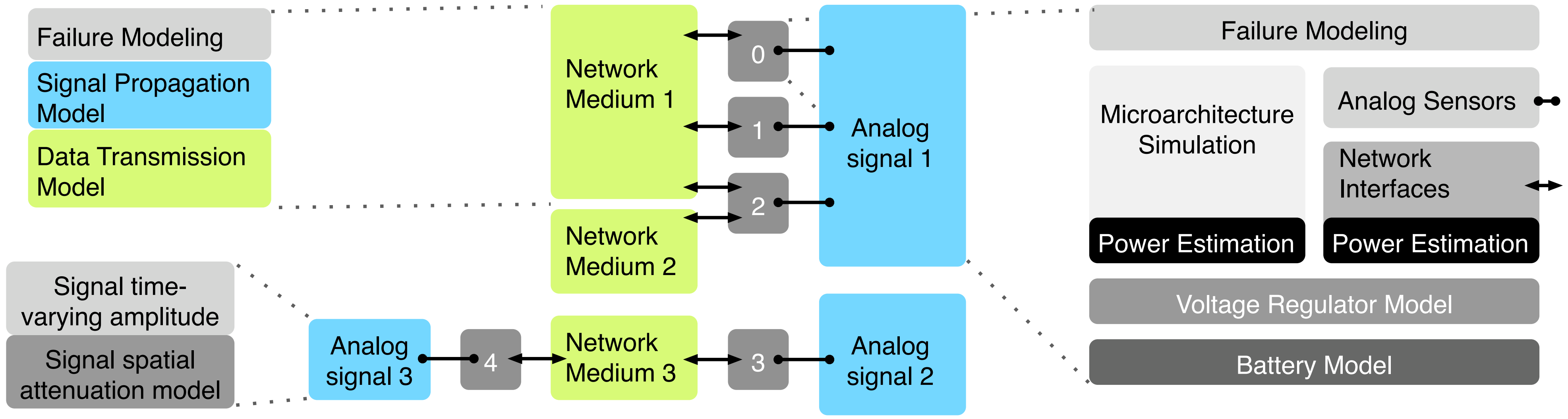
Outline

- Motivation
- Simulation Framework Overview
- Distributed Simulation
- Multi-Platform Packaging
- Summary

Sunflower Full-System Simulator

- Simulation Engine Models:
  - Computation — at instruction execution level, for two different ISAs
  - Communication — at the MAC and PHY levels
  - Compute & network power dissipation — includes instruction-level power models
  - Batteries and voltage regulators — models for several batteries and regulators
  - Device and networking faults — bit-level logic upsets and node / network failures
  - Physical phenomena external to hardware — location in three-space, attenuation

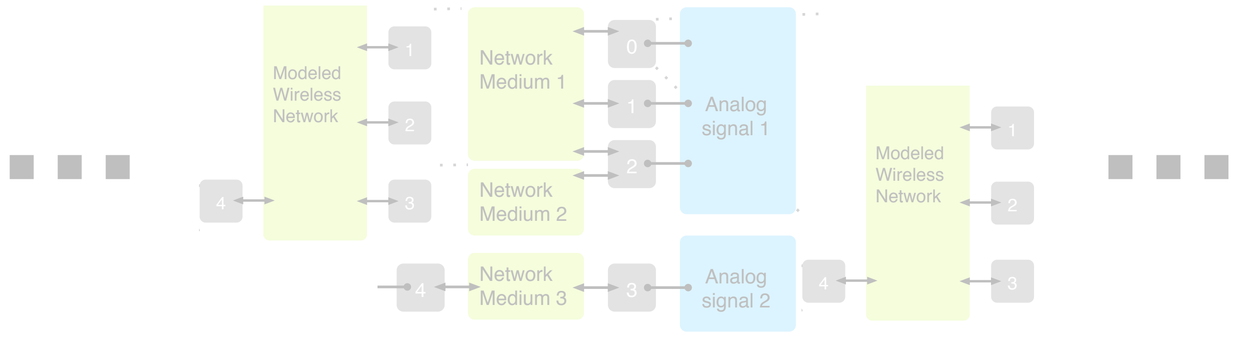
- Example composition of a simulated system:



["Sunflower: Full-System Embedded Microarchitecture Evaluation", *HiPEAC '07*, Ghent, Belgium]

Sunflower Full-System Simulator

- Simulation engine implemented in C
  - Originally implemented as a standalone application on Unix
- Challenges
  - Simulating large systems (thousands of nodes) can be tasking, even on a high-end workstation
  - Especially so, when modeling all the aforementioned components



  - Can split simulation across multiple workstations
  - Nodes simulated on separate hosts may communicate : must handle that
  - Must keep passage of time synchronous across portions of partitioned simulation

Outline

- Motivation
- Simulation Framework Overview
- Distributed Simulation
- Multi-Platform Packaging
- Summary

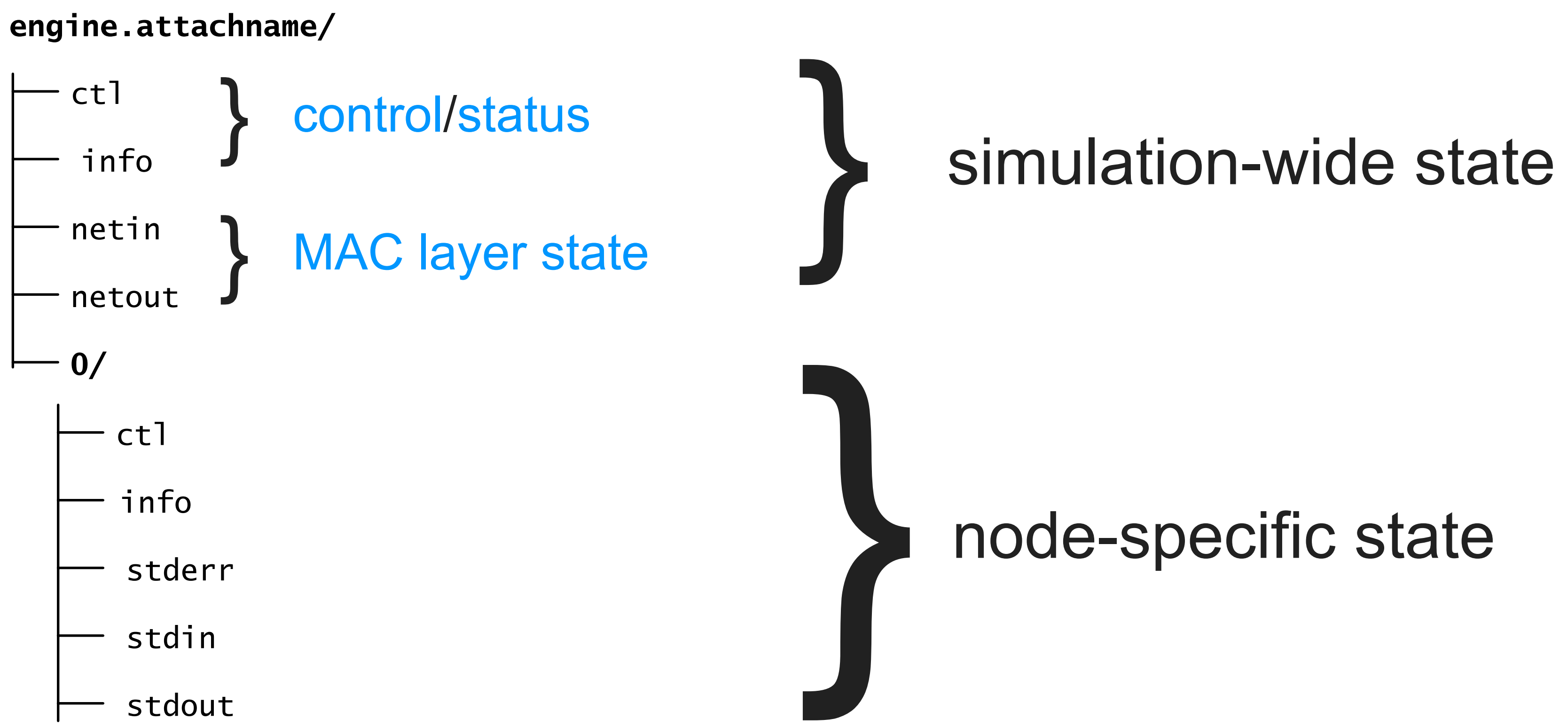
Distributed Simulation

- Challenges
  - An efficient / easy way to connect state of portions of sim across hosts
  - A means of ensuring the state of simulation across hosts is consistent
  - An interface to keep track of hosts, global simulation state, etc.

- Approach
  - *There are many ways you could do this...*
  - Chose to take advantage of ease of connecting systems with Inferno

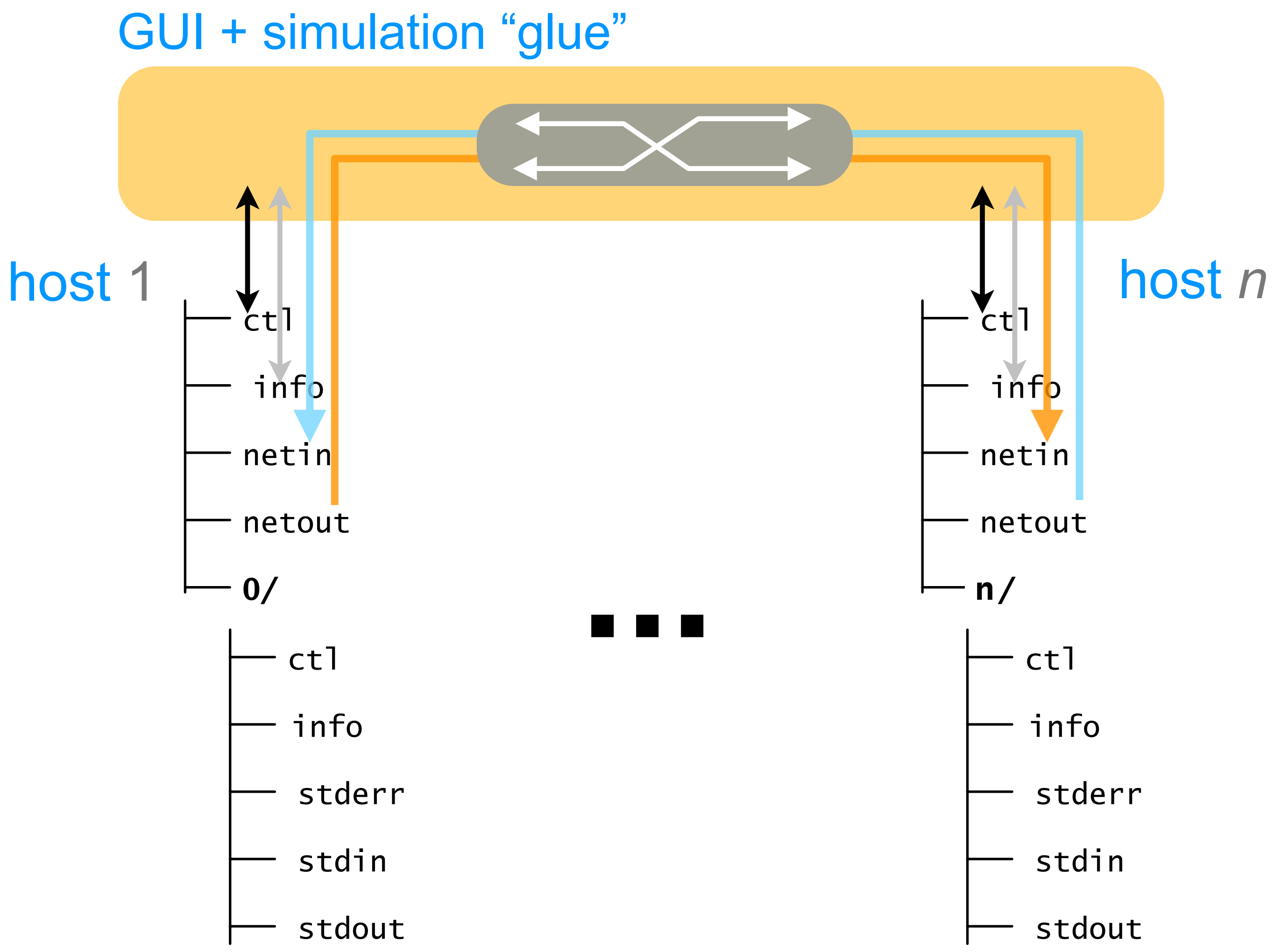
- Implementation
  - Simulation engine compiled as a library, linked against emu (chose not to use libstyx)
  - All relevant state on each host exposed as a filesystem, via a device driver interface
  - “Glue” application connects together name spaces, keeps engine state consistent

Simulation Engine Interface



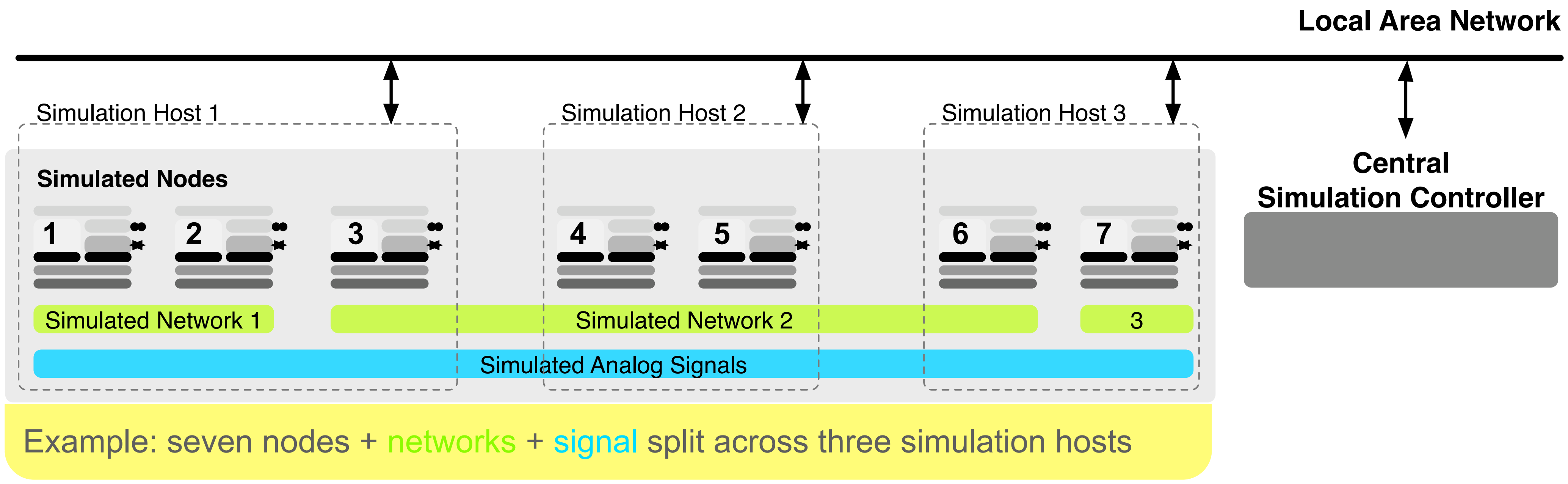
- Interface to simulation engine is a device driver, `devsf`, **#j**
  - Dynamic filesystem interface w/ one line directory per simulated node (processor+batt, etc)
  - Exposes simulation engine state (e.g., simulated MAC-layer frames via `netin`, `netout`)

Simulation Glue Logic



- "Glue" application (implemented in Limbo)
  - Engine filesystems from different simulation hosts mounted in name space of glue app
  - Interconnects simulated networks on different hosts
  - Implements facilities for synchronizing virtual time across portions of simulated system

Distributed Simulation



- Synchronization issues
  - Simulation rates across hosts may vary, but global timebase must remain synchronized
  - Well known issues, already investigated in the area of parallel discrete-event simulation
- Synchronization facilities
  - Time synchronization
  - Simulation rate synchronization

GUI + Glue Logic Application

Pull-down menu with shortcuts for common commands

Button shortcuts for "glue" commands

Message output window

clicking on a node makes it the current

node # @ host #

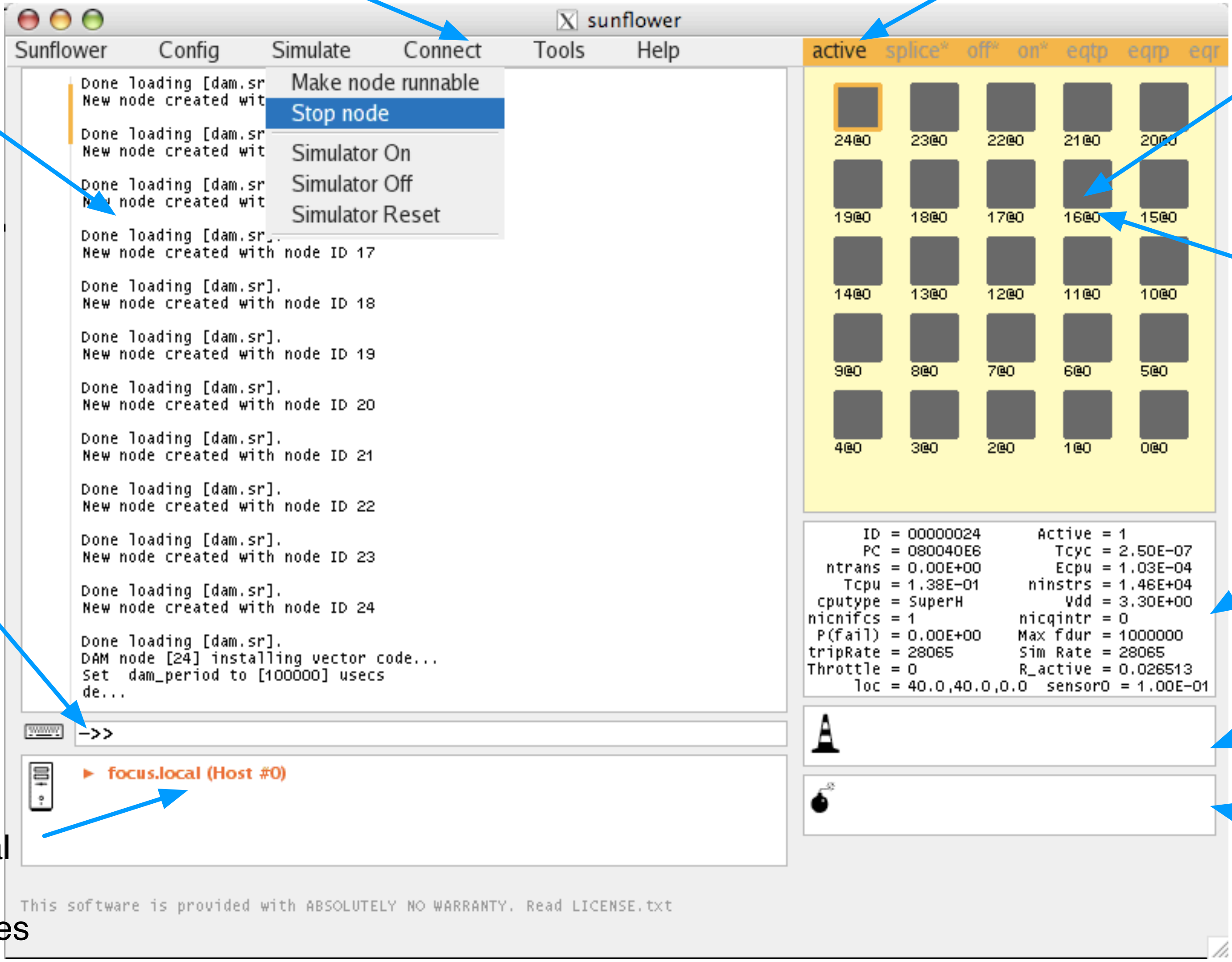
Command input

Summary of statistics for current node

Warning messages

Error messages

Connected local and remote simulation engines



```
Done loading [dam.sr
New node created wit

Done loading [dam.sr
New node created wit

Done loading [dam.sr
New node created wit

Done loading [dam.sr].
New node created with node ID 17

Done loading [dam.sr].
New node created with node ID 18

Done loading [dam.sr].
New node created with node ID 19

Done loading [dam.sr].
New node created with node ID 20

Done loading [dam.sr].
New node created with node ID 21

Done loading [dam.sr].
New node created with node ID 22

Done loading [dam.sr].
New node created with node ID 23

Done loading [dam.sr].
New node created with node ID 24

Done loading [dam.sr].
DAM node [24] installing vector code...
Set  dam_period to [100000] usecs
de...
```

```
      ID = 00000024          Active = 1
      PC = 080040E6            Tcyc = 2.50E-07
  ntrans = 0.00E+00            Ecpu = 1.03E-04
    Tcpu = 1.38E-01         ninstrs = 1.46E+04
 cputype = SuperH               Vdd = 3.30E+00
nicnifcs = 1                nicqintr = 0
 P(fail) = 0.00E+00         Max fdur = 1000000
tripRate = 28065            Sim Rate = 28065
Throttle = 0                R_active = 0.026513
     loc = 40.0,40.0,0.0   sensor0 = 1.00E-01
```

```
This software is provided with ABSOLUTELY NO WARRANTY. Read LICENSE.txt
```

Outline

- Motivation
- Simulation Framework Overview
- Distributed Simulation
- Multi-Platform Packaging
- Summary

Multi-Platform Packaging

- Goal — Implementation transparency
  - A single executable binary, indistinguishable from a native app. on host platform
  - Should not require (explicit or automatic) installation of Inferno
- Implementation
  - All the necessary dis executables, fonts, etc. compiled into in-memory root filesystem
  - Simulation engine library & driver interface (`#j`)
  - Server mode or client mode, w/ GUI or server app. instead of `emuinit.dis` → `sh.dis`
  - Reads / writes from host filesystem like any other host application (via `#U`)
  - On most platforms, binary is ~2—5 MB; self-contained executable
- Alternative — don't roll filesystem into emu binary image
  - Distribute emu + Inferno filesystem tree (acme-sac does this)
  - I currently think a single-executable approach is cleaner <u>for our purposes</u>

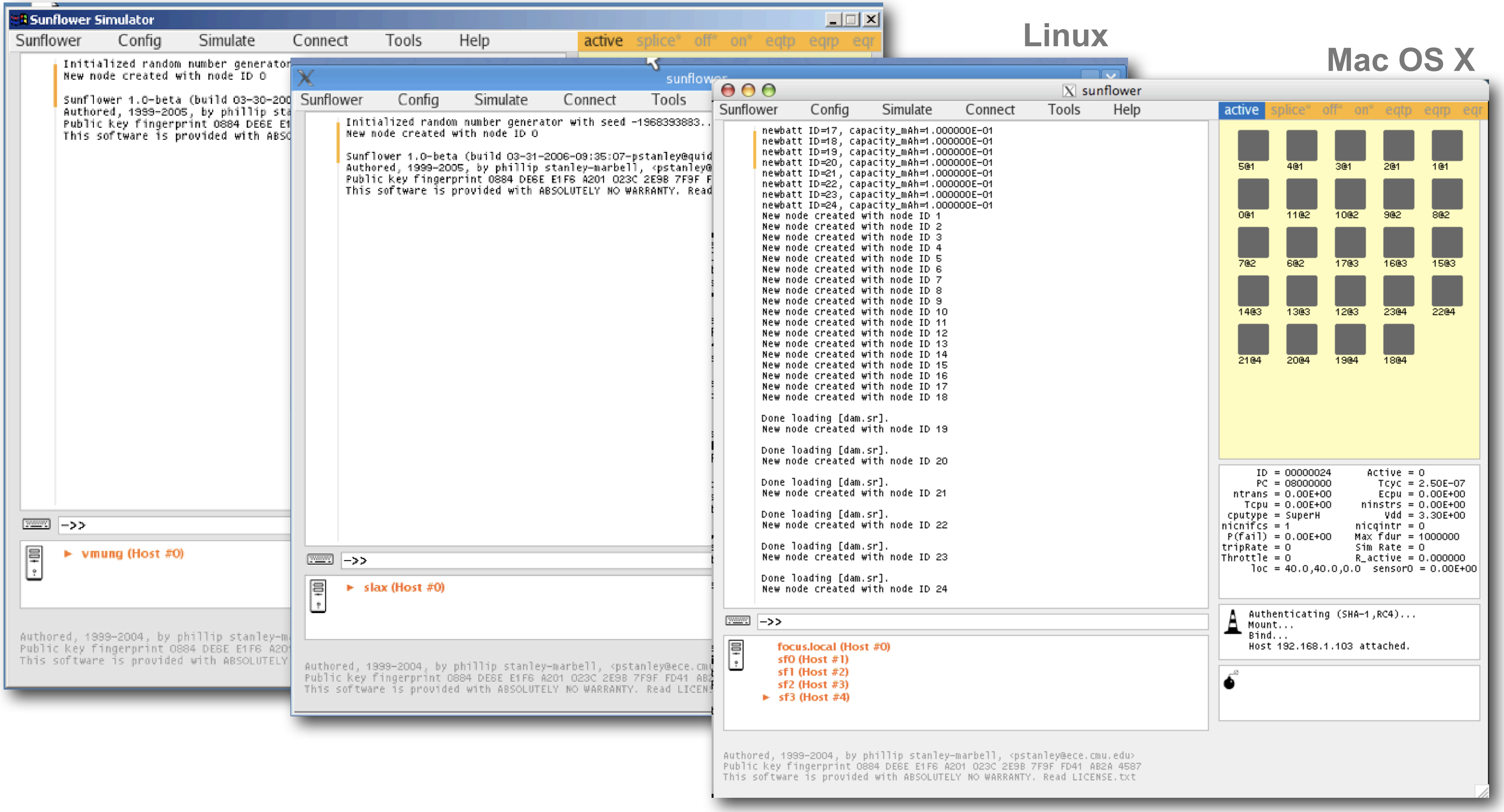
↩

Multi-Platform Packaging

Windows 2000

Linux

Mac OS X



```
Initialized random number generator
New node created with node ID 0

Sunflower 1.0-beta (build 03-30-200
Authored, 1999-2005, by phillip sta
Public key fingerprint 0884 DE6E E1
This software is provided with ABSO
```

vmung (Host #0)

```
Authored, 1999-2004, by phillip stanley-m
Public key fingerprint 0884 DE6E E1F6 A20
This software is provided with ABSOLUTELY
```

```
Initialized random number generator with seed -1968393883..
New node created with node ID 0

Sunflower 1.0-beta (build 03-31-2006-09:35:07-pstanley@quid
Authored, 1999-2005, by phillip stanley-marbell, <pstanley@
Public key fingerprint 0884 DE6E E1F6 A201 023C 2E98 7F9F F
This software is provided with ABSOLUTELY NO WARRANTY. Read
```

slax (Host #0)

```
Authored, 1999-2004, by phillip stanley-marbell, <pstanley@ece.cm
Public key fingerprint 0884 DE6E E1F6 A201 023C 2E98 7F9F FD41 AB2
This software is provided with ABSOLUTELY NO WARRANTY. Read LICEN
```

```
newbatt ID=17, capacity_mAh=1.000000E-01
newbatt ID=18, capacity_mAh=1.000000E-01
newbatt ID=19, capacity_mAh=1.000000E-01
newbatt ID=20, capacity_mAh=1.000000E-01
newbatt ID=21, capacity_mAh=1.000000E-01
newbatt ID=22, capacity_mAh=1.000000E-01
newbatt ID=23, capacity_mAh=1.000000E-01
newbatt ID=24, capacity_mAh=1.000000E-01
New node created with node ID 1
New node created with node ID 2
New node created with node ID 3
New node created with node ID 4
New node created with node ID 5
New node created with node ID 6
New node created with node ID 7
New node created with node ID 8
New node created with node ID 9
New node created with node ID 10
New node created with node ID 11
New node created with node ID 12
New node created with node ID 13
New node created with node ID 14
New node created with node ID 15
New node created with node ID 16
New node created with node ID 17
New node created with node ID 18

Done loading [dam.sr].
New node created with node ID 19

Done loading [dam.sr].
New node created with node ID 20

Done loading [dam.sr].
New node created with node ID 21

Done loading [dam.sr].
New node created with node ID 22

Done loading [dam.sr].
New node created with node ID 23

Done loading [dam.sr].
New node created with node ID 24
```

focus.local (Host #0)
sf0 (Host #1)
sf1 (Host #2)
sf2 (Host #3)
sf3 (Host #4)

```
      ID = 00000024       Active = 0
      PC = 08000000         Tcyc = 2.50E-07
  ntrans = 0.00E+00         Ecpu = 0.00E+00
    Tcpu = 0.00E+00      ninstrs = 0.00E+00
 cputype = SuperH            Vdd = 3.30E+00
nicnifcs = 1            nicqintr = 0
 P(fail) = 0.00E+00     Max fdur = 1000000
tripRate = 0            Sim Rate = 0
Throttle = 0            R_active = 0.000000
     loc = 40.0,40.0,0.0  sensor0 = 0.00E+00
```

```
Authenticating (SHA-1,RC4)...
Mount...
Bind...
Host 192.168.1.103 attached.
```

```
Authored, 1999-2004, by phillip stanley-marbell, <pstanley@ece.cmu.edu>
Public key fingerprint 0884 DE6E E1F6 A201 023C 2E98 7F9F FD41 AB2A 4587
This software is provided with ABSOLUTELY NO WARRANTY. Read LICENSE.txt
```

- Distributed as a single executable, no installation reqd.
  - 2.5MB binary on MacOS, 5.2MB binary on Linux, 2.7MB .exe on Windows

Outline

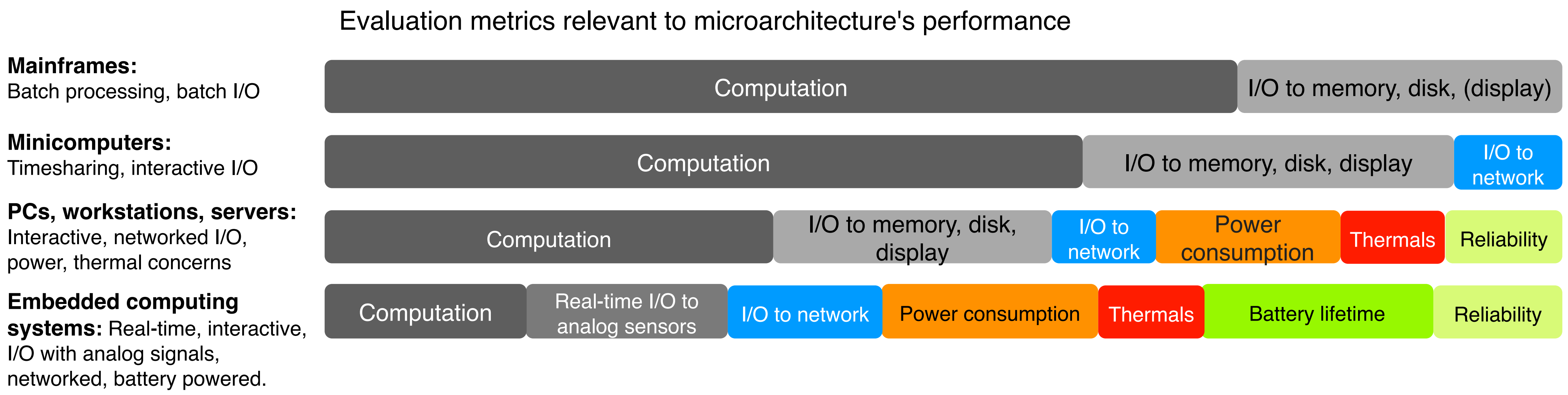
- Motivation
- Simulation Framework Overview
- Distributed Simulation
- Multi-Platform Packaging
- Summary

Summary

- Sunflower
  - A full-system simulator for networks of embedded systems
- Problem
  - Simulation of large networks (thousand+ nodes) is compute- and memory intensive
  - Simulation can be split at the level of individual nodes (with some added work)
- Distributed simulation
  - Simulation engine compiled as a Inferno emulator library
  - Device driver (`#j`) interface to simulation state
  - GUI+glue application interconnects simulated state across multiple simulation hosts
  - Integration into Inferno enables easy multi-platform packaging/GUI
- Sources, binaries, documentation
  - `http://www.ece.cmu.edu/~pstanley/sunflower`

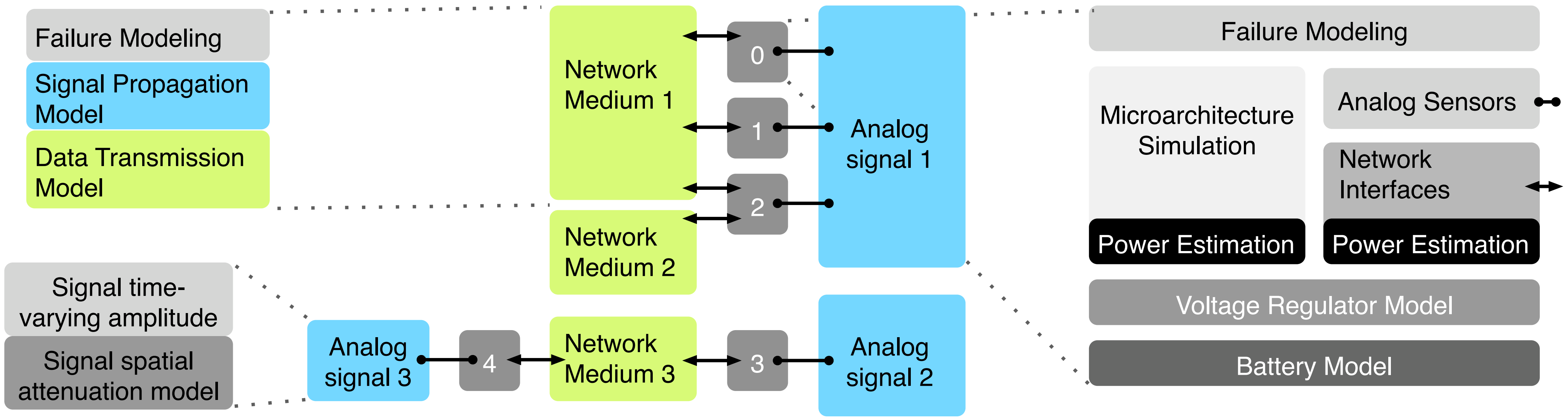
Backup Slides

Hardware Emulation Environment – Sunflower Simulator


Evaluation metrics relevant to microarchitecture's performance
Mainframes: Batch processing, batch I/O
Computation
I/O to memory, disk, (display)
Minicomputers: Timesharing, interactive I/O
Computation
I/O to memory, disk, display
I/O to network
PCs, workstations, servers: Interactive, networked I/O, power, thermal concerns
Computation
I/O to memory, disk, display
I/O to network
Power consumption
Thermals
Reliability
Embedded computing systems: Real-time, interactive, I/O with analog signals, networked, battery powered.
Computation
Real-time I/O to analog sensors
I/O to network
Power consumption
Thermals
Battery lifetime
Reliability

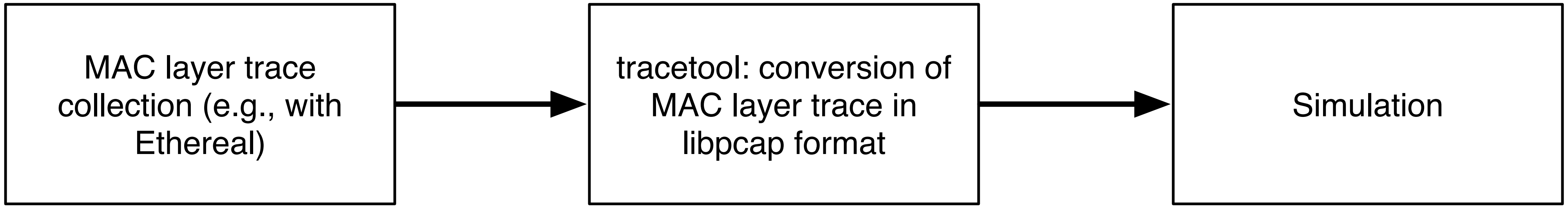
Backup Slides

Hardware Emulation Environment – Sunflower Simulator

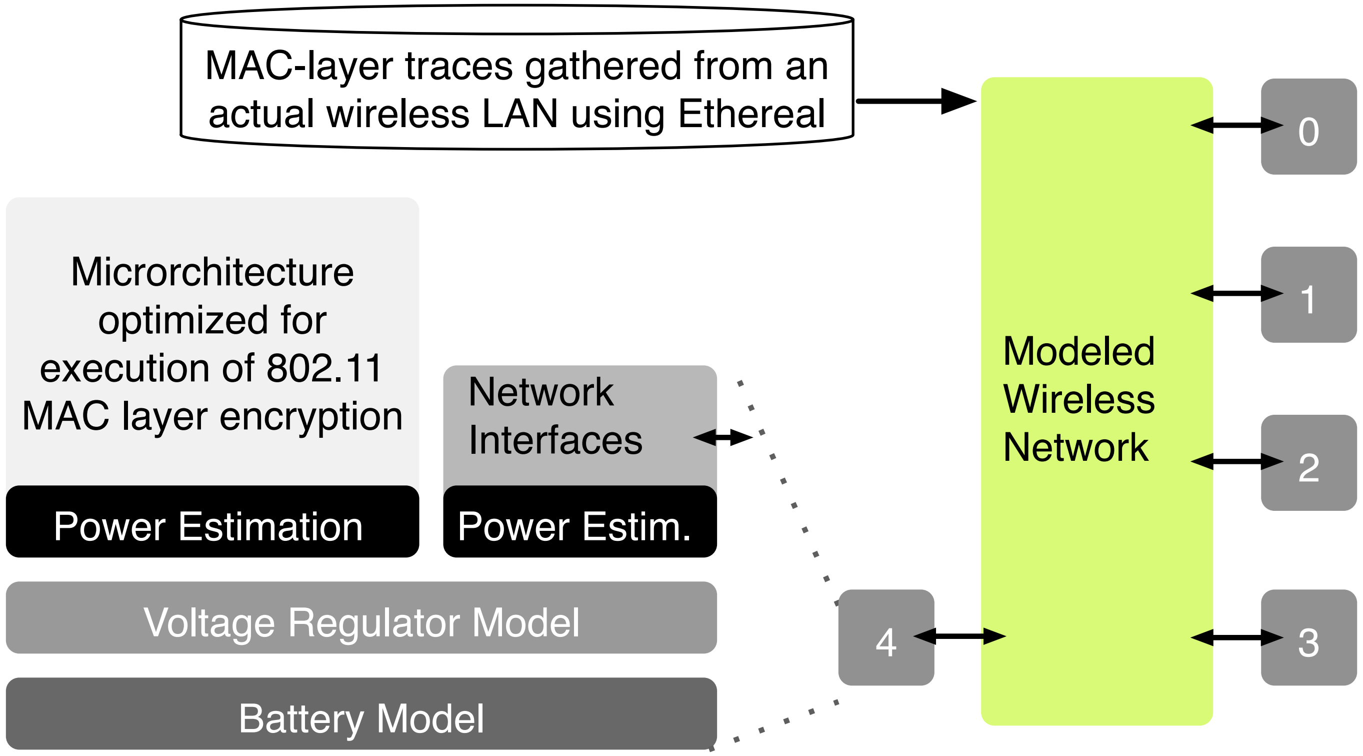


Backup Slides ↩

Using Real Network Traces

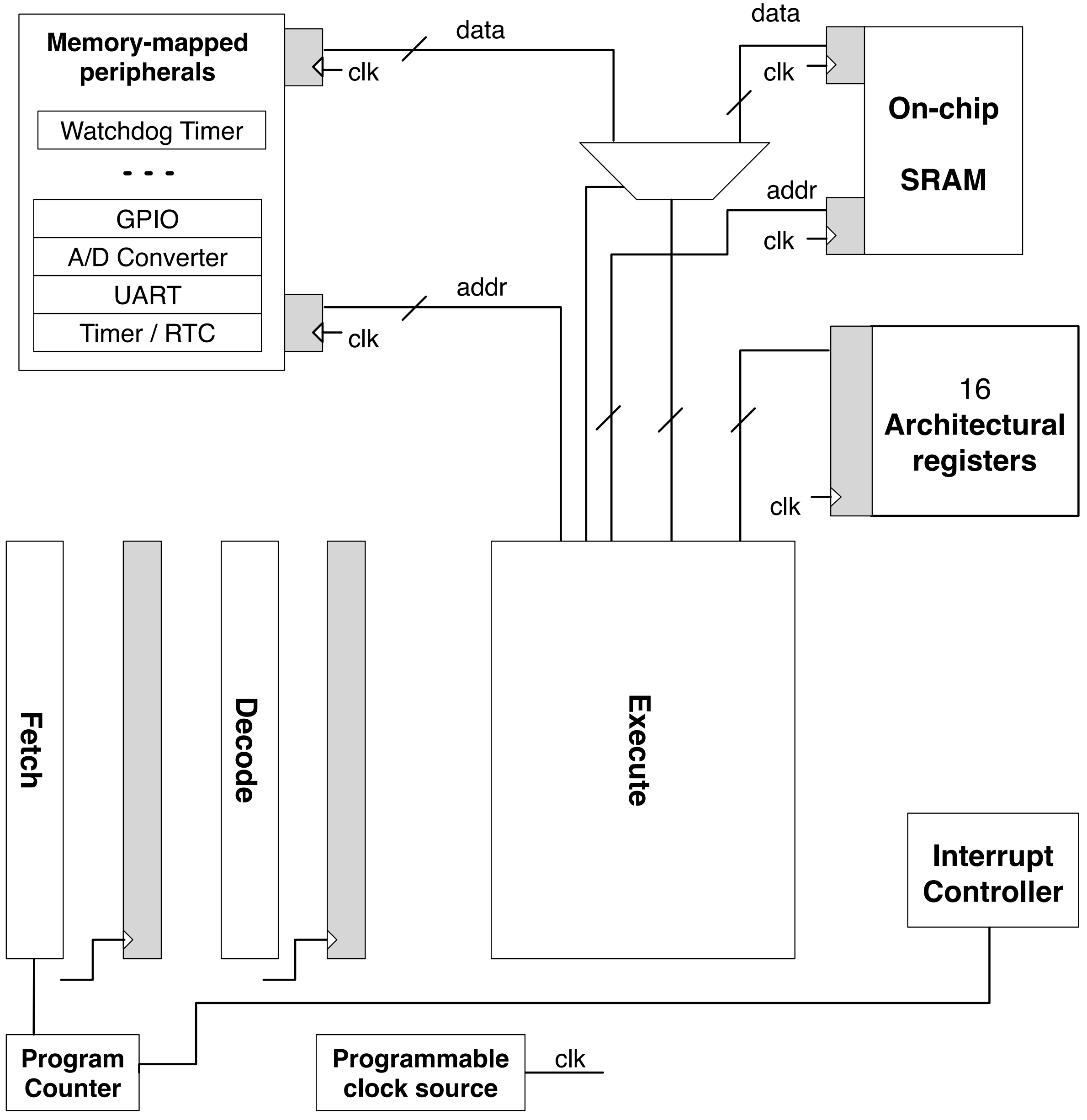

MAC layer trace collection (e.g., with Ethereal)
tracetool: conversion of MAC layer trace in libpcap format
Simulation

Example: Using Real Network Traces


MAC-layer traces gathered from an actual wireless LAN using Ethereal
Microrchitecture optimized for execution of 802.11 MAC layer encryption
Network Interfaces
Power Estimation
Power Estim.
Voltage Regulator Model
Battery Model
Modeled Wireless Network
0
1
2
3
4

Backup Slides ↩

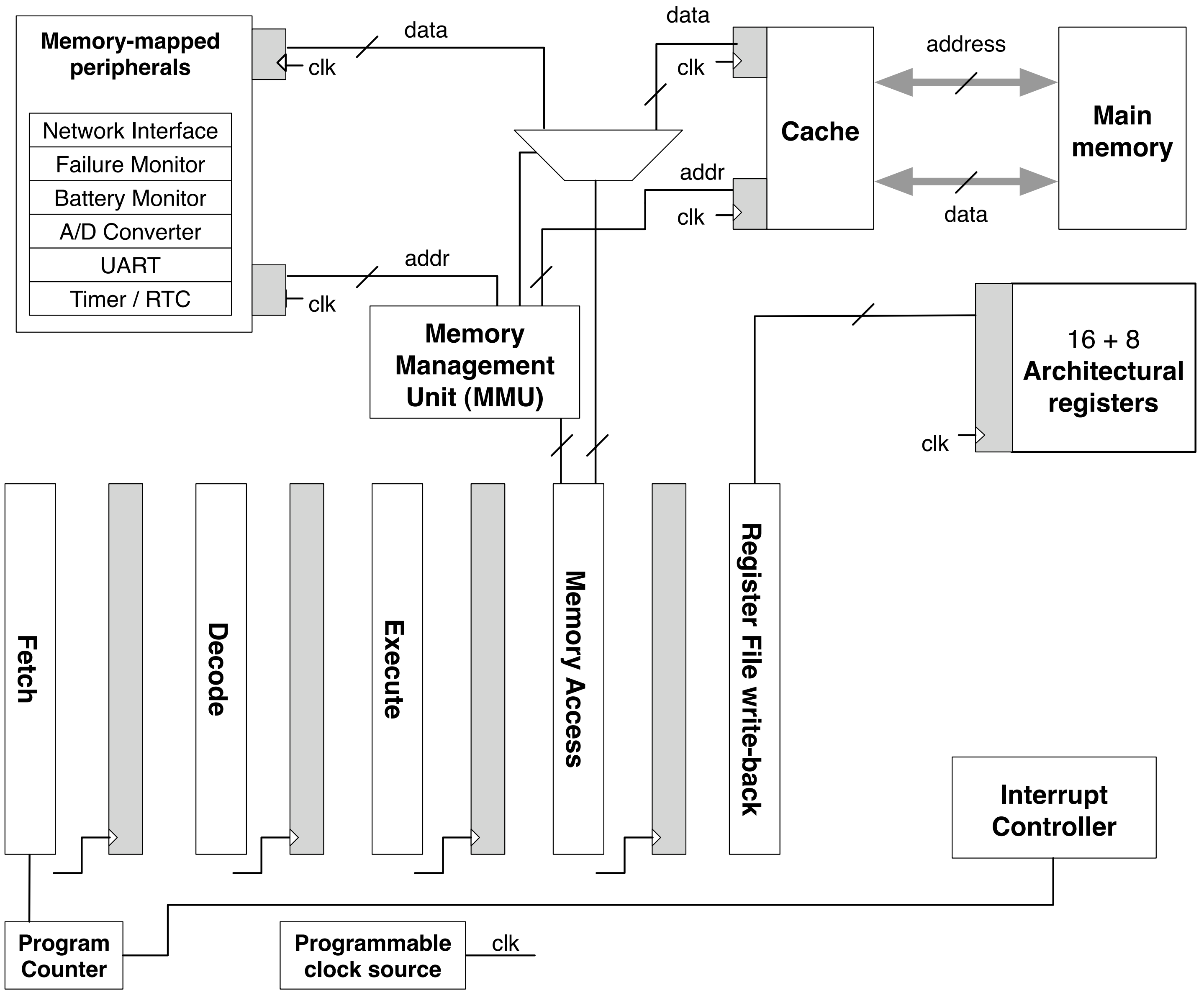
Modeled 16-bit Microarchitecture (TI MSP430)



▨ = Structures modeled at bit-level, enabling signal transition activity and SEU modeling

Backup Slides

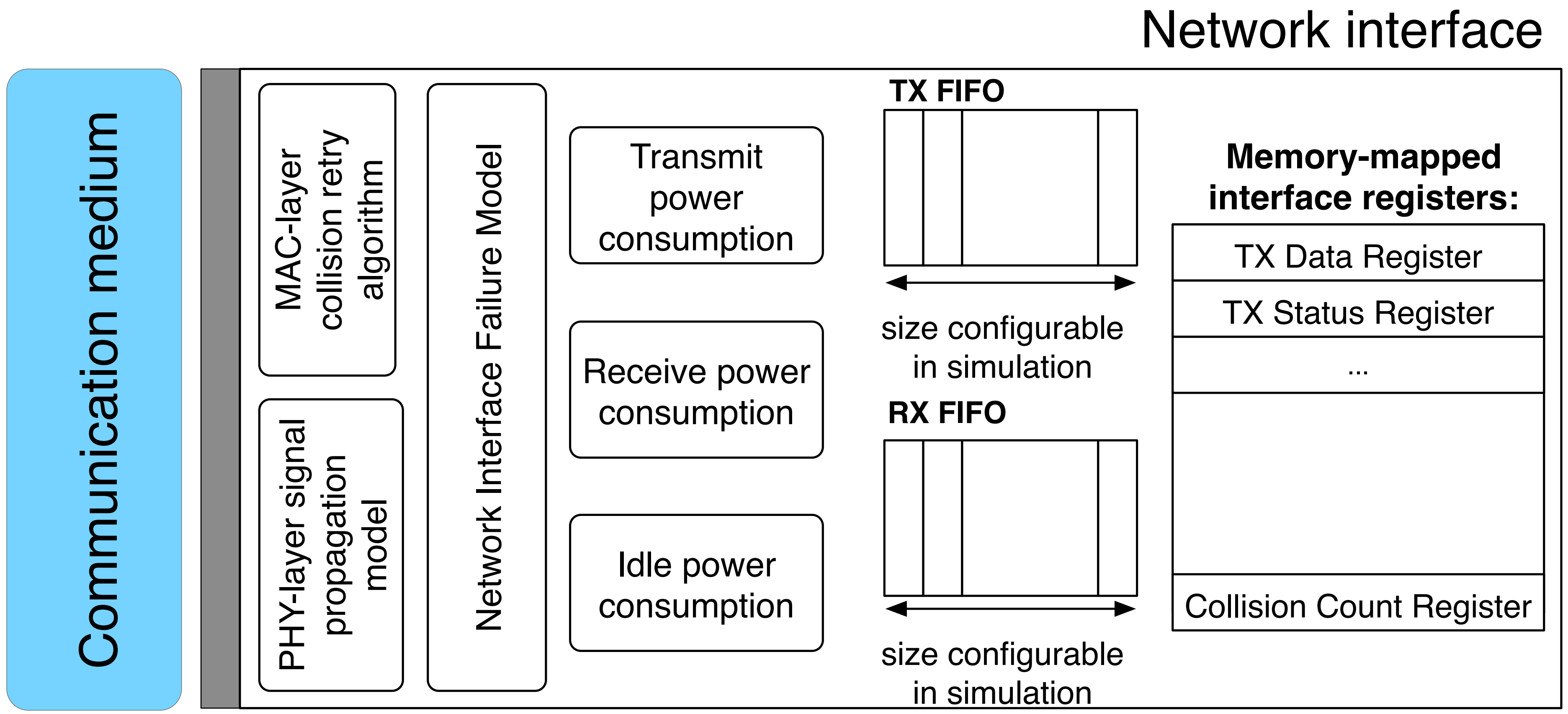
Modeled 32-bit Microarchitecture (Hitachi SH)



▨ = Structures modeled at bit-level, enabling monitoring of signal transition activity and SEUs during simulation

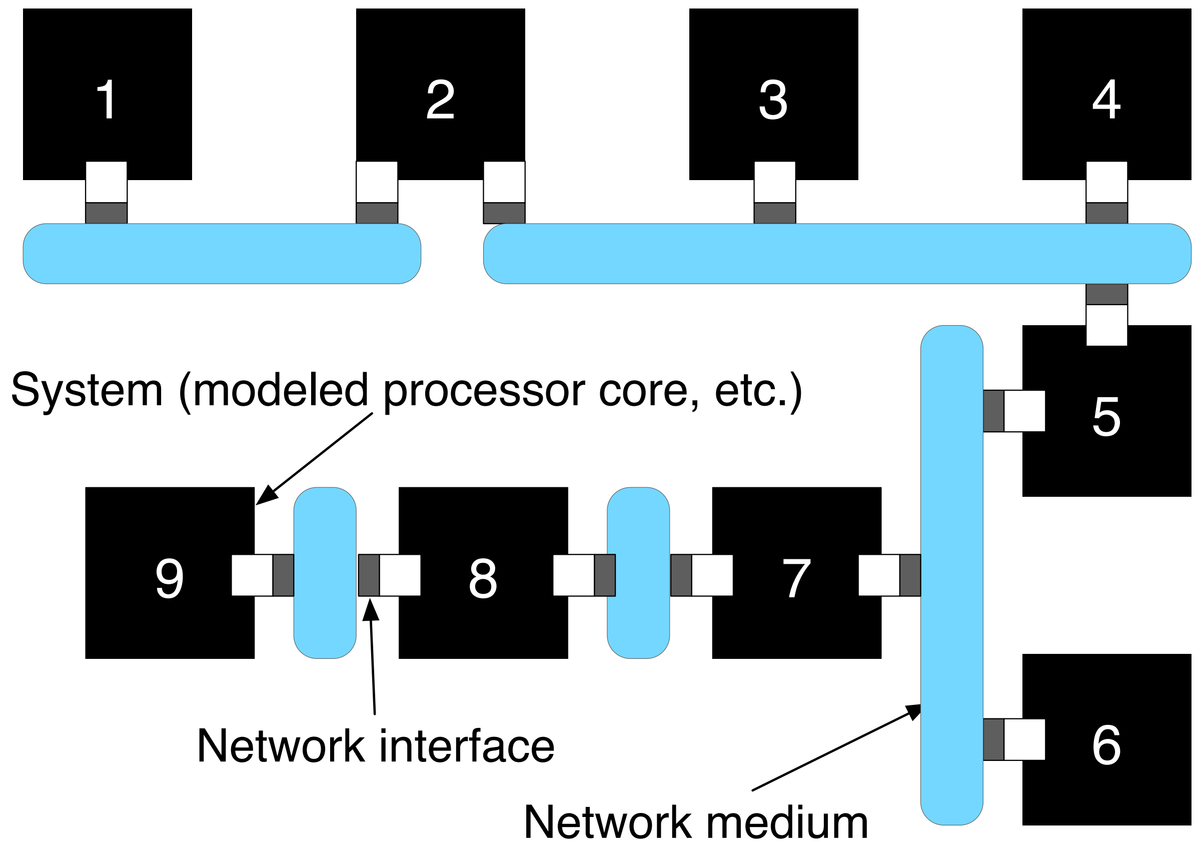
Backup Slides

Communication Interface Modeling


Network interface
Communication medium
MAC-layer collision retry algorithm
PHY-layer signal propagation model
Network Interface Failure Model
Transmit power consumption
Receive power consumption
Idle power consumption
TX FIFO
size configurable in simulation
RX FIFO
size configurable in simulation
Memory-mapped interface registers:
TX Data Register
TX Status Register
...
Collision Count Register

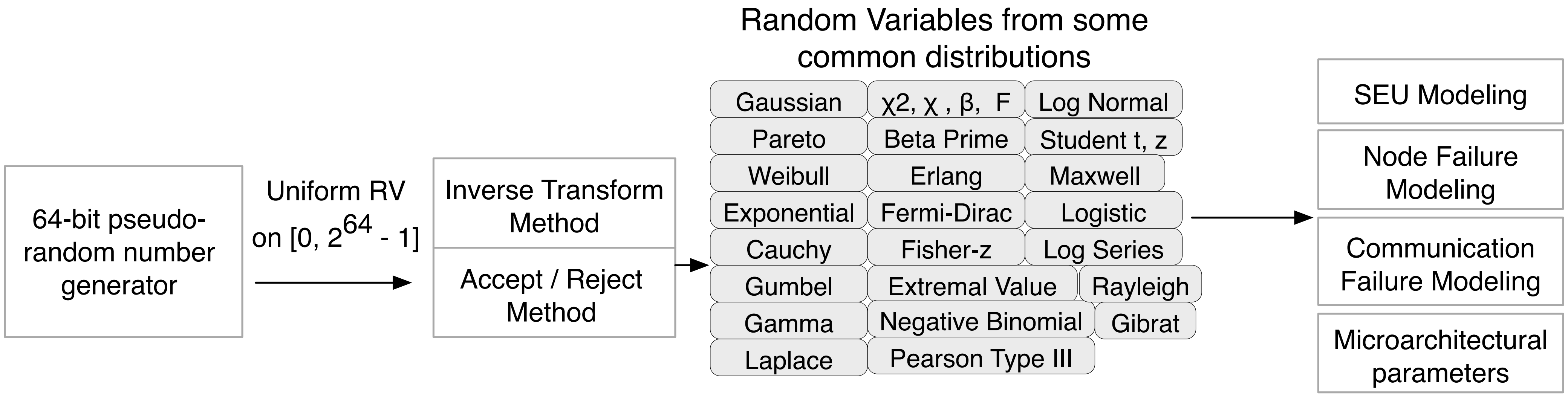
Backup Slides ↩

Creating Arbitrary Topologies



Backup Slides ↩

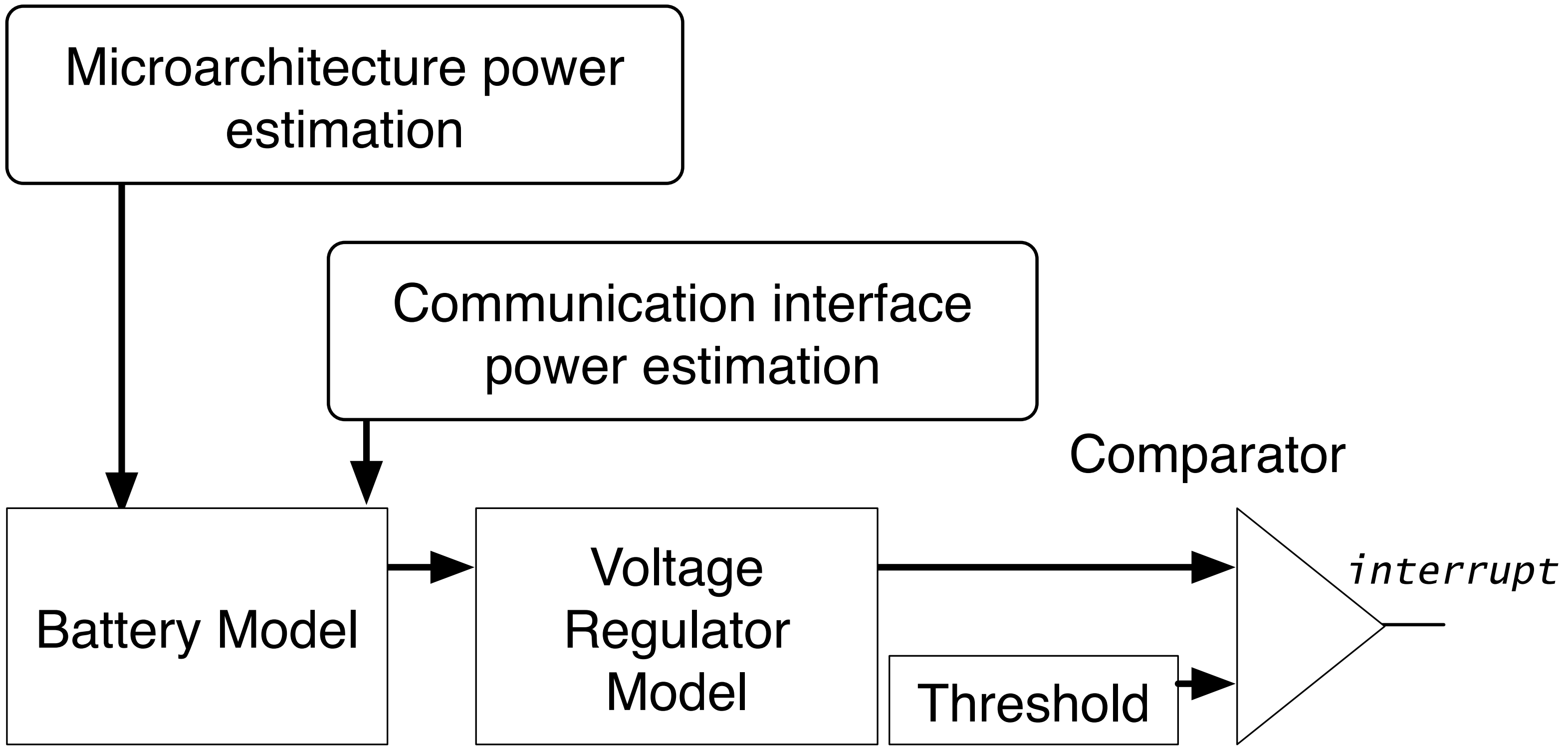
Rich Set of Stochastic Distribution Generators



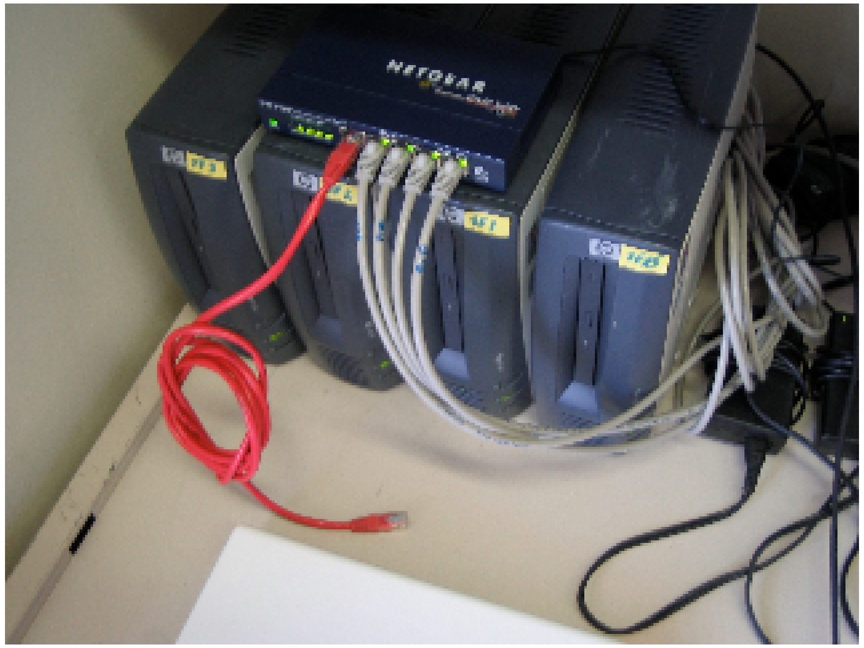
- All built upon a 64-bit pseudo-random number generator with very large period [Nishimura]

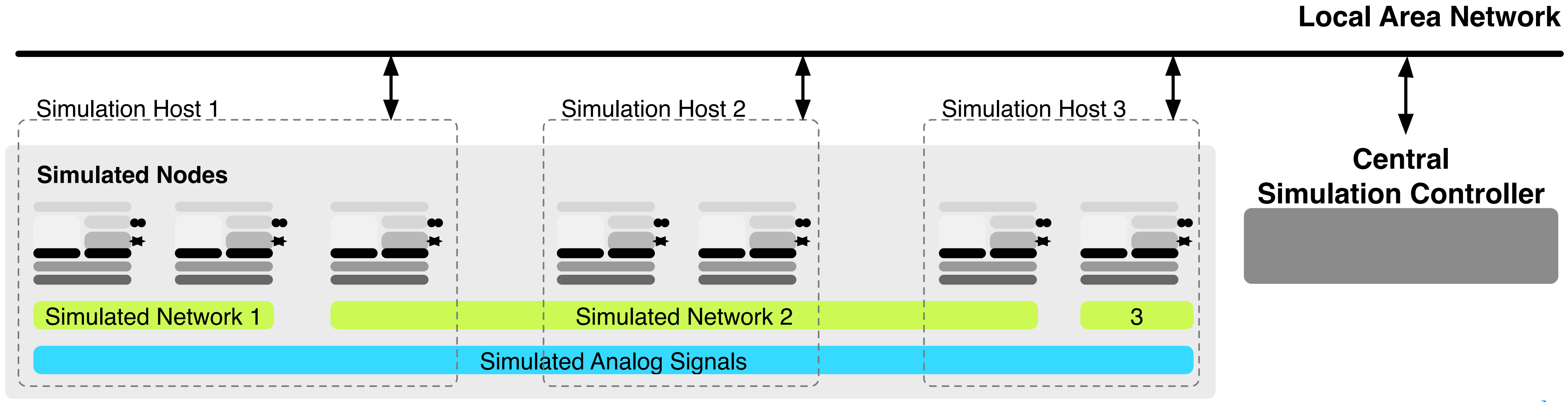
Backup Slides ↩

Battery Subsystem Model


Microarchitecture power estimation
Communication interface power estimation
Comparator
Battery Model
Voltage Regulator Model
Threshold
interrupt

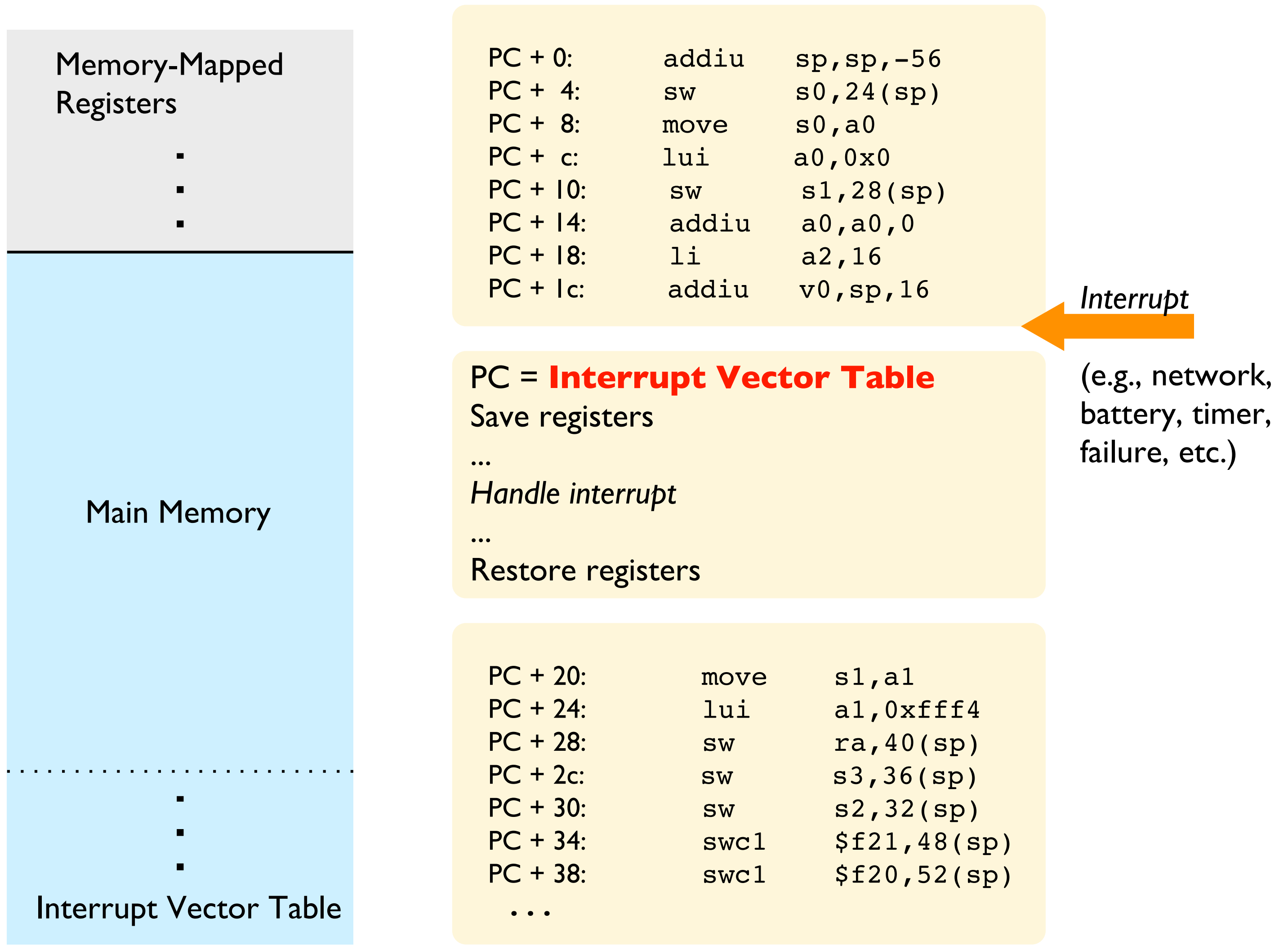
Distributed Simulation





Backup Slides

Example Memory Map



```
PC + 0:     addiu   sp,sp,-56
PC +  4:    sw      s0,24(sp)
PC +  8:    move    s0,a0
PC +  c:    lui     a0,0x0
PC + 10:    sw      s1,28(sp)
PC + 14:    addiu   a0,a0,0
PC + 18:    li      a2,16
PC + 1c:    addiu   v0,sp,16
```

PC = **Interrupt Vector Table**
Save registers
...
Handle interrupt
...
Restore registers

(e.g., network, battery, timer, failure, etc.)

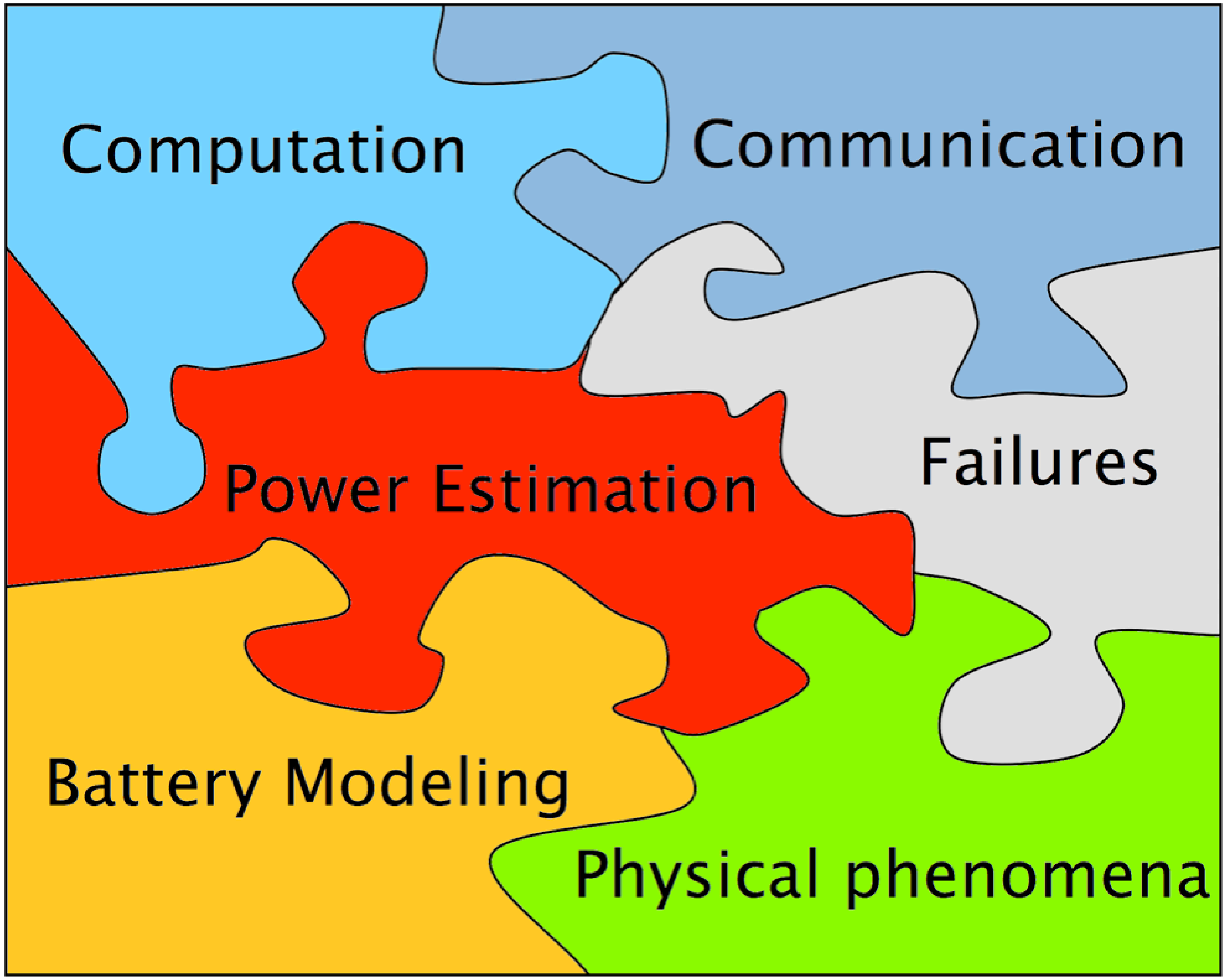
```
PC + 20:       move     s1,a1
PC + 24:       lui      a1,0xfff4
PC + 28:       sw       ra,40(sp)
PC + 2c:       sw       s3,36(sp)
PC + 30:       sw       s2,32(sp)
PC + 34:       swc1     $f21,48(sp)
PC + 38:       swc1     $f20,52(sp)
  ...
```

(a) Memory map of simulated processing elements.

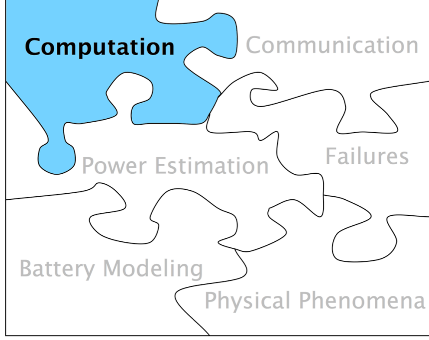
(b) Example flow of instruction execution in the presence of interrupts from modeled peripherals

Backup Slides

Sunflower Simulator / HW EMulator



Modeling Computation



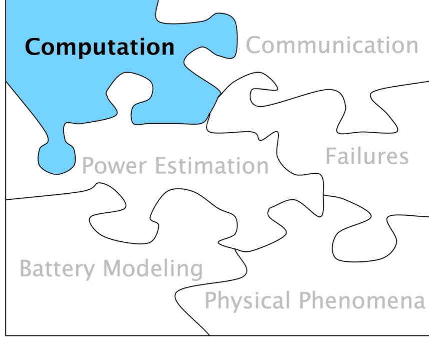
- Modeled ISA
  - Hitachi SH ISA, support for new ISAs being added
  - Applications (e.g., SPEC CPU 2000) compiled with GCC 3.x toolchain, Hitachi/Renesas HEW compiler, Microsoft VC for Hitachi SH, ...
  - Should be able to compile apps in any source language that GCC or any of the compilers support (C, C++, FORTRAN, Java, Chill, Ada)

- Microarchitecture
  - Detailed simulation of SH3 family pipeline

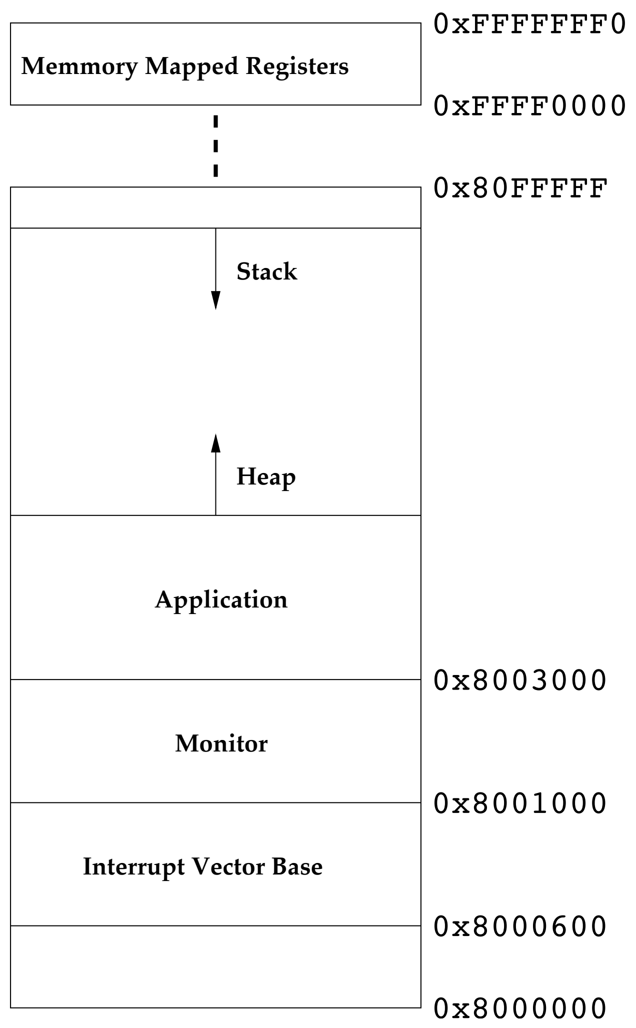
- Processor peripherals and interrupt sources
  - SH3's UART, Timer Unit
  - Added new peripherals: Network Interface, Sensors, Random Number Source, Logging, Simulator Control Interface

Backup Slides

Application's View



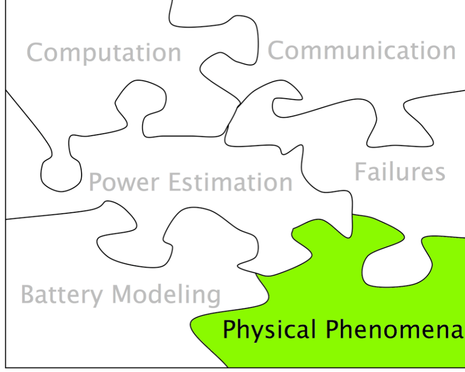
Memory Map



- Peripherals and interrupt sources
  - Interaction with modeled peripherals and simulation control and is through memory-mapped registers
  - Interrupt sources: network, device failures, battery status, timer
- Typical application style:
  - `main()+interrupt handler`
  - E.g., received network frames handled in network intr handler, periodically scheduled tasks triggered by timer interrupts
  - Rudimentary device drivers and interrupt handling code is often reused

Backup Slides ↩

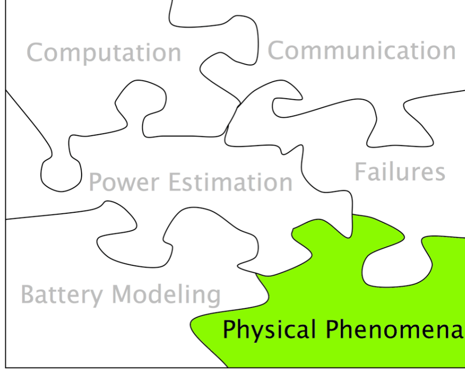
Modeling Physical Phenomena



- Modeling both computation and the physical phenomena that drive computation is important

- If we're modeling a sensor network, would like to model the physics of signal propagation
  - Attenuation of signals in space
  - Interference between signals
  - Location and motion of signal sources in space

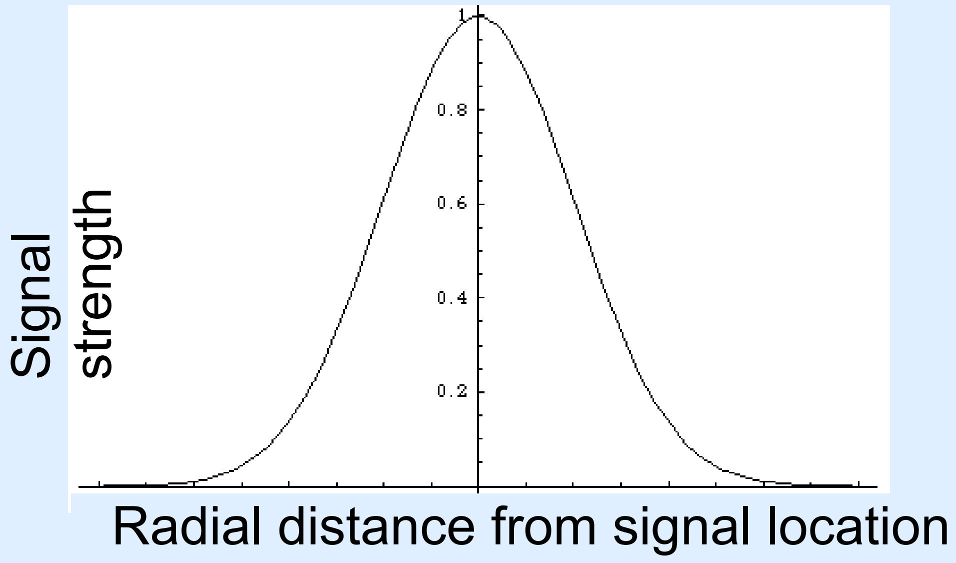
Backup Slides ↩



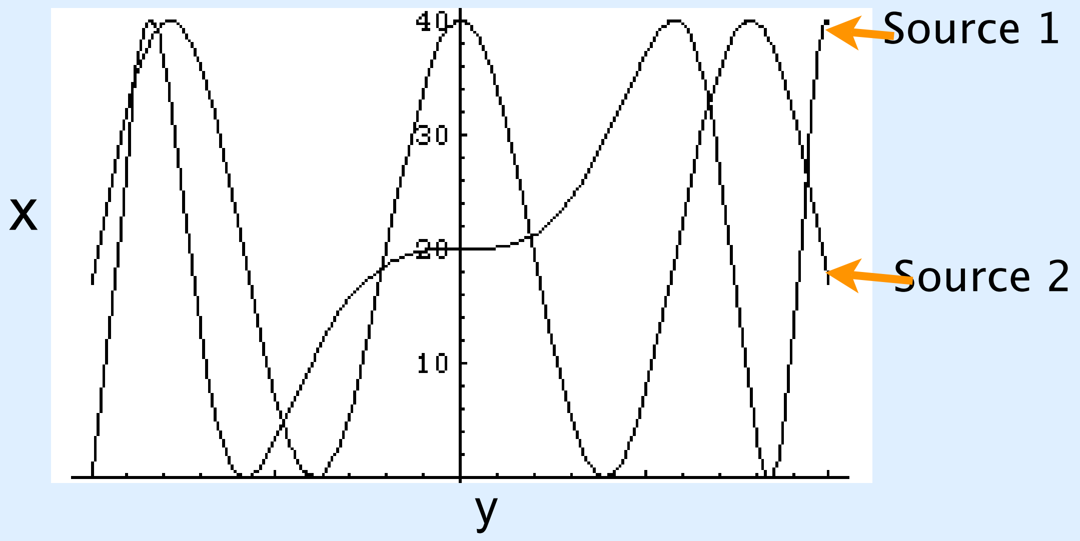
Modeling Physical Phenomena

- Example
  - 2 light sources with Gaussian spread of intensity from peak
  - The light sources move according to trajectories specified by a LUT

Example user-specified signal source attenuation

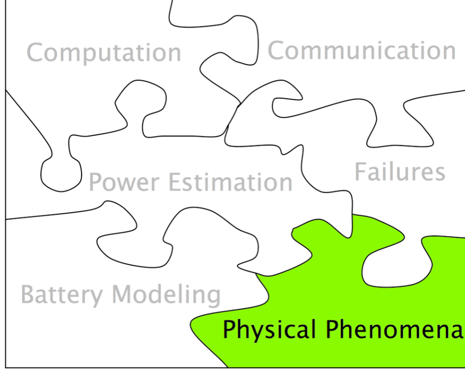


Example user-specified signal source trajectories



Backup Slides

Modeling Physical Phenomena



- Signal attenuation
  - Attenuation with radial distance, x, specified by providing coefficients for

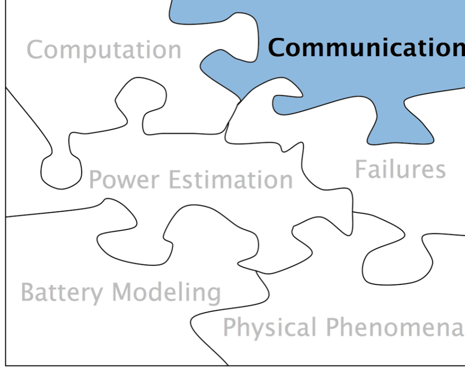
$$S(Ax^{m} + Bx^{n} + Cx^{o} + Dx^{p} + EK^{(Fx^{q}+ Gx^{r} + Hx^{s} + Ix^{t})})$$

  - Example, for Gaussian spread light source with peak intensity S, E=1, K= **e**, F = -0.5, q = 2, all other coefficients are 0

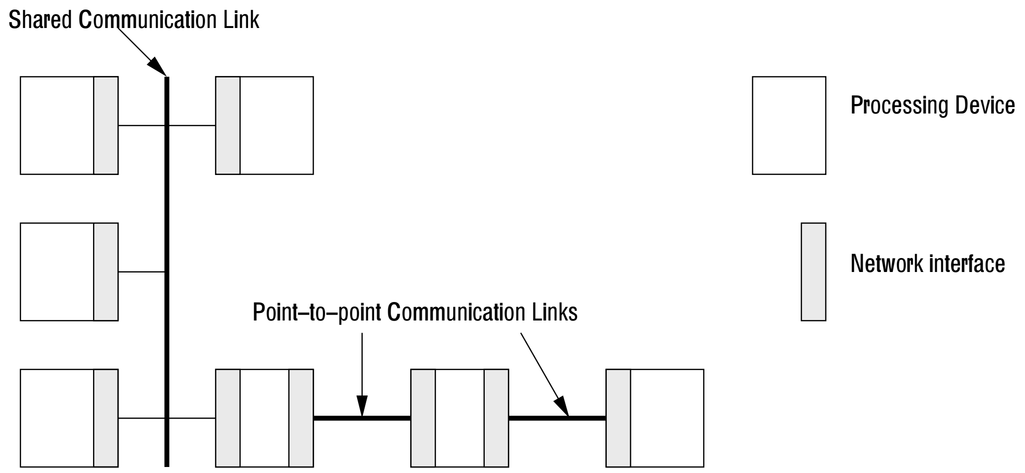
- Signal trajectory
  - Trajectory and speed specified as a list of way-points and sampling rate
  - Notion of time in physical models is synchronized with ISA simulation, communication modeling, battery models, etc.

Backup Slides ↩

Modeling Communication



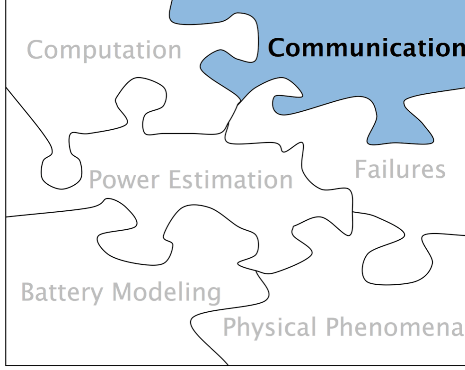
- Model
  - Each processor can have multiple **network interfaces** (NICs) instantiated
  - Within a simulation, multiple **network segments** are instantiated
  - NICs are connected to network segments to create arbitrary topologies



- Application's view
  - Code running over simulator sees NIC as a memory-mapped peripheral
  - Interrupts generated for TX/RX and various errors

Backup Slides ↩

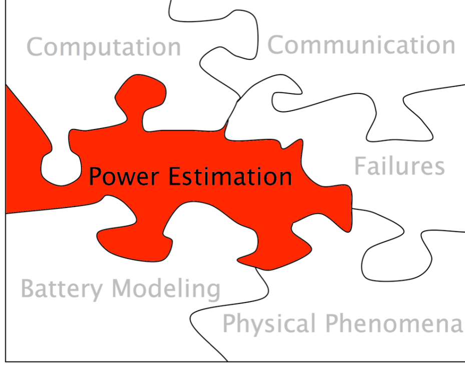
Modeling Communication



- Network Segment (Defines properties of physical link)
  - Bit rate
  - Frame size
  - Number of simultaneous transmissions that can be accommodated
  - Rate and distribution of intermittent failures
  - Signal propagation properties, same model as described for phy. sources
  - Minimum SNR before introducing bit errors
- Network Interface
  - Transmit, receive and idle power consumption
  - Number of TX and RX FIFO entries

Backup Slides ↩

Power Estimation



- 3 different methods for power estimation

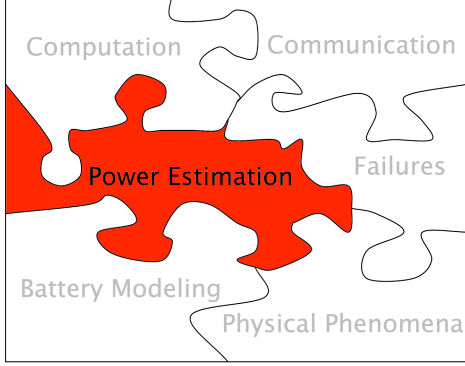
- Requires most effort, slower, not always practicable:
  - Signal transition activity reported per cycle for pipeline latches, buses, register file, cache ports, usage of FUs
  - Requires capacitance values to obtain power estimates

- Better tradeoff:
  - Characterized instruction-level power model
  - Requires empirical measurements on actual hardware

- Simplest:
  - Specify average active and sleep power consumption, e.g., from a device manufacturer's data sheet

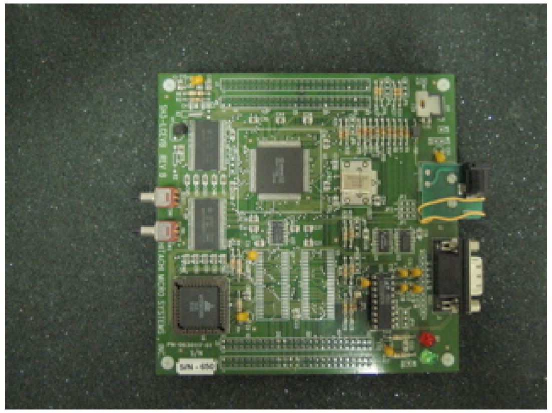
Backup Slides ↩

Power Estimation



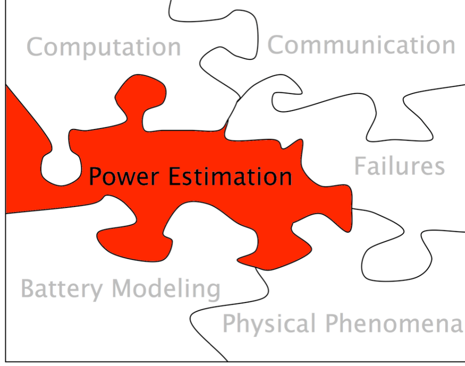
- Simulator includes an instruction-level power model for the Hitachi SH7708

- Empirical measurements using SH7708 eval. board
  - Measured average current draw for a fixed set of operand values, for each instruction in the ISA (~160 instructions / different addressing modes)
  - No inter-instruction effects
  - Non-idempotent instructions like TRAP can't be subjected to this method



Backup Slides

Power Estimation

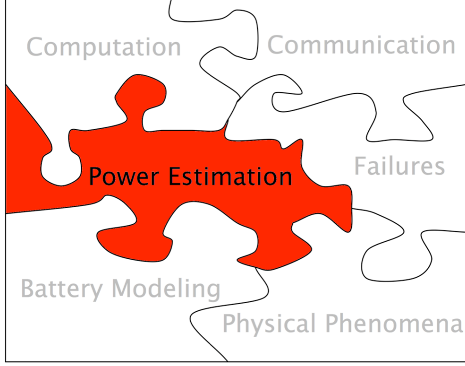


- Signal transition activity estimation
  - Signal transitions as encoded/decoded instructions move through pipeline
  - Register file, program counter
  - Data and address buses
  - Cache read/write ports
  - No modeling of internals of functional units

- When is transition activity estimation useful ?
  - Comparative studies of signal transition activity in the modeled structures
  - If you have capacitance values; this may not be available until after floor-planning and layout

Backup Slides

Power Estimation



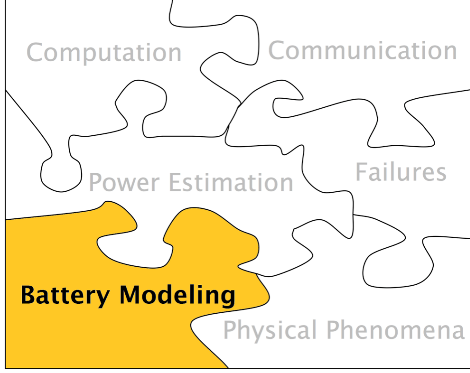
- Other details

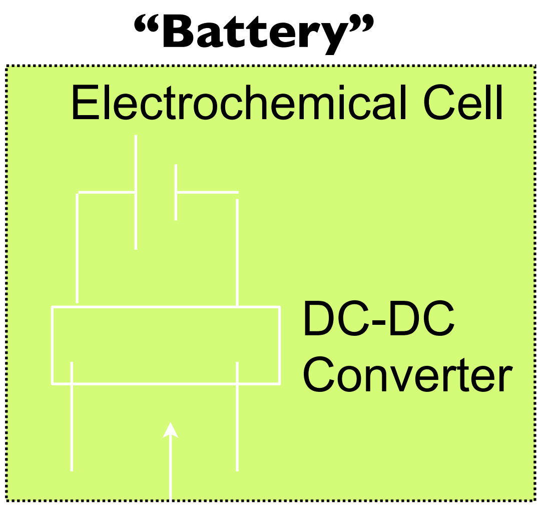
- Sleep mode
  - Reduced power consumption when application executes a sleep instruction (until next interrupt)

- Voltage and frequency scaling during simulation
  - Under simulated application's control via simulator control register
  - Interactively via command prompt
  - Independent voltage and frequency scaling or scale VDD and freq. in tandem for a specified Vt and technology-dependent α

Backup Slides ↩

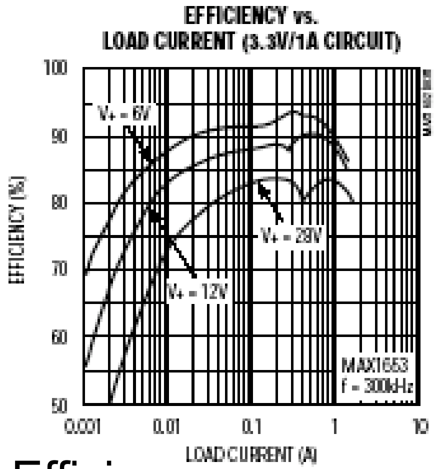
Battery Subsystem





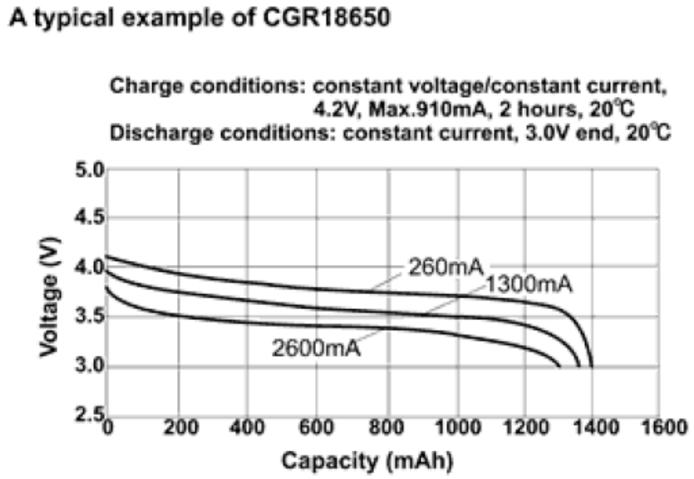
Network interface power consumption + Processor power consumption

DC-DC Converter
LUT from data sheet



Efficiency vs. current for Maxim MAX1652

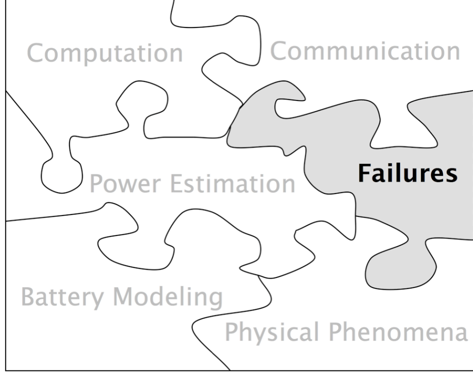
Electrochemical cell discharge profile



- One or more devices can be attached to a battery system
- Sampled current draw periodically supplied to battery subsystem
- Current passed to a model for a DC-DC converter, then to model for electrochemical cell
- Models based on discrete-time battery model from [L. Benini et al., TVLSI '01]

Backup Slides

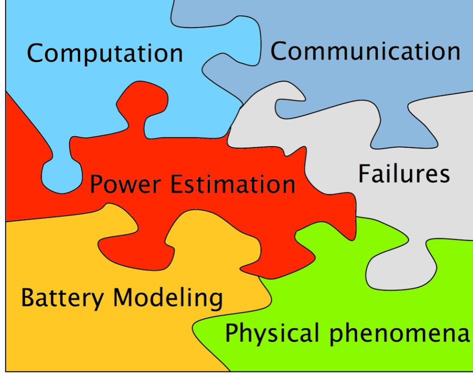
Modeling Failures



- Computation Failures
  - Event upsets within microarchitecture or treat each processor as a unit that may fail *en-masse* with some probability
  - Intermittent failures with a specified rate and distribution for which processor is temporarily inactive
  - Correlated failures between processing elements and network segments
- Communication failures
  - Also permit configuration of intermittent failures of network segments
  - If a network is associated with a physical signal model, induce bit errors when SNR drops below a specified threshold

Backup Slides ↩

Implementation



- Simulator core written in C, interactive command parser and built-in assembler specified in Yacc
- Runs on *BSD, Linux, MacOS, Windows
- Console application or with GUI
- GUI and simulation parallelization implemented with the Inferno OS and Limbo programming language

Backup Slides ↩