Overview
ESESC Tutorial

Speaker: Jose Renau

ESESC

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu

UC SANTA CRUZ

Logistics

09:00 - 09:30: Overview
09:30 - 10:30: Code Structure and Tools
10:30 - 11:00: Morning Break
11:00 - 12:00: Timing Model
12:00 - 12:30: Sampling Methods Part 1
12:30 - 13:30: Lunch
13:30 - 14:15: Sampling Methods Part 2
14:15 - 15:00: Power Model
15:00 - 15:30: Afternoon Break
15:30 - 16:30: Thermal Model
16:30 - 17:00: Wrap-up

Logistics

- ESESC blog has these slides
  http://masc.soe.ucsc.edu/esesc
- ESES forum
  https://groups.google.com/forum/#!forum/esesc
- ESESC repository at github
  https://github.com/masc-ucsc/esesc
  - To get the code

```
git clone https://github.com/masc-ucsc/esesc.git
```

What is ESESC?

- Cycle accurate chip multiprocessor
- In-order and out-of-order processors
- Performance/Power/Thermal models
- Fast simulator

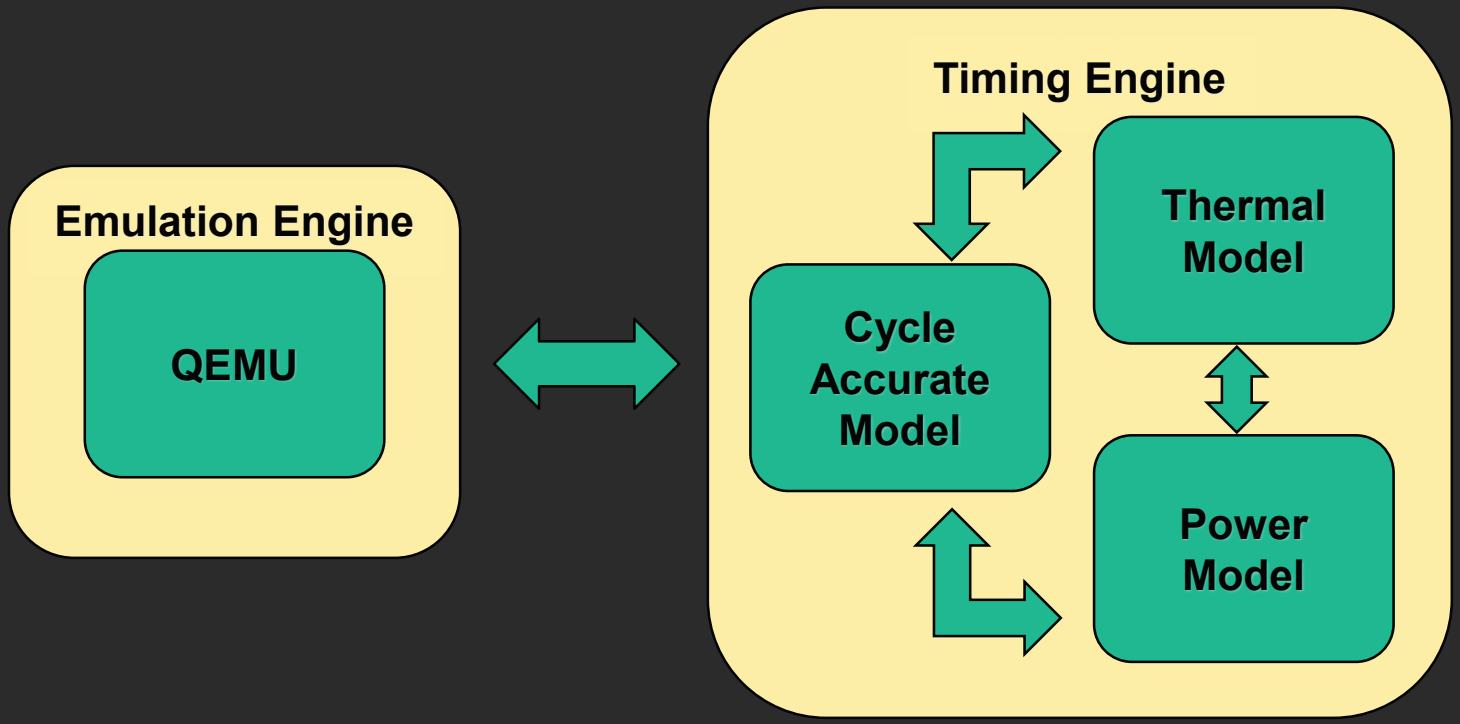
ESESC = Enhanced SESC

- Same goals as SESC
  - Fast cycle-accurate simulator
  - Easy to understand and extend
  - Multiple configurations available
- Many enhancements…

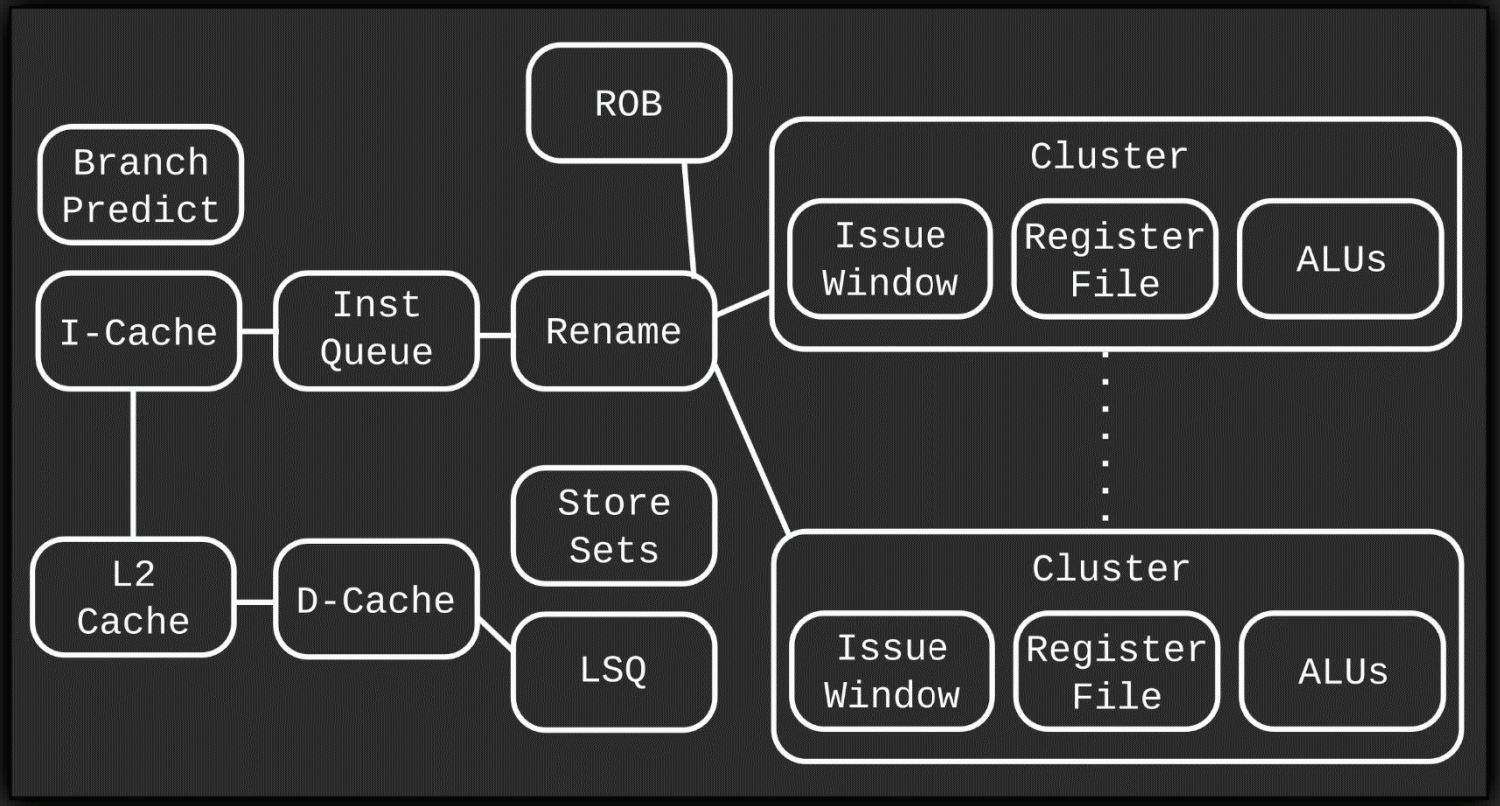
ESESC vs other Cycle-Accurate Sims

Simulator	Models	ISA	Full-System	Key-Feature	Speed
ESESC	Perf/Pwr/Temp	ARMv7	In-Progress	Sampling	~50MIPS
SESC	Per/Pwr/Temp	MIPS	No	Fast	~1MIP
gem5	Perf/Pwr	X86/ARM	Yes		
MARSSx86	Perf	X86	Yes		~200KIPS
Flexus	Perf?/Pwr?	SPARC	Yes	Sampling	
Multi2sim	Perf	X86	No	Heterogenous	
Snipper	Perf	X86	No (pin)	Multithreaded	

Execution-Driven Simulation



Out-of-order Core



Timing Model Verification

None of these class projects required to change a line of code.
Just configuration parameters

ESESC Enhancements

- Runs unmodified ARM binaries
- Statistical sampling
- New memory hierarchy design
- Integrated thermal model
- McPAT power model

Unmodified ARM binaries

- SESC
  - Custom MIPS-based compilation flow
- ESESC
  - Unmodified ARM Linux binaries
  - Cracks ARM instructions to ESESC uOPs

Executable Compilation Platform

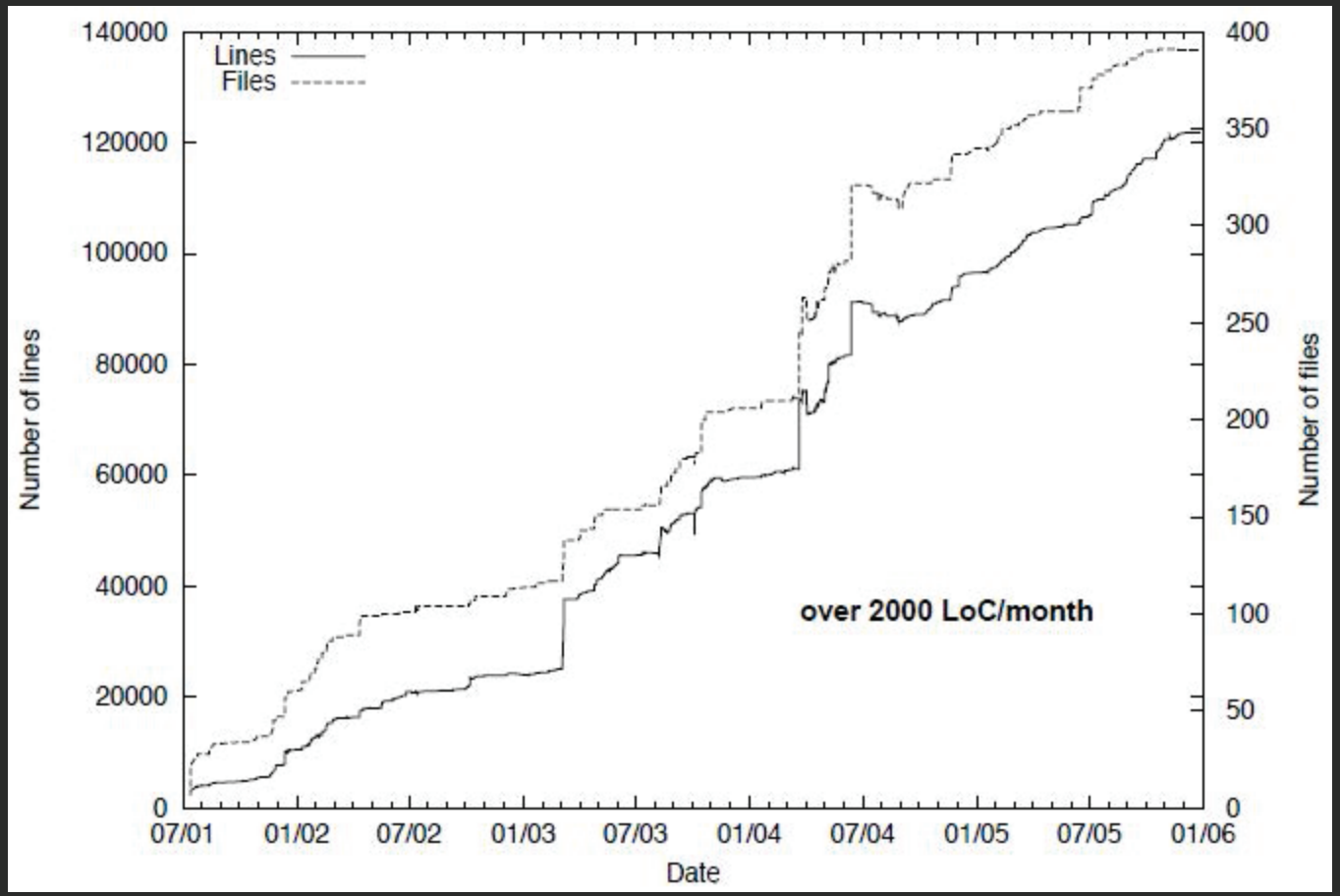
Fast Simulation

- ESESC achieves over 50MIPS
  - Significant effort creating an efficient timing
- Many sampling techniques available

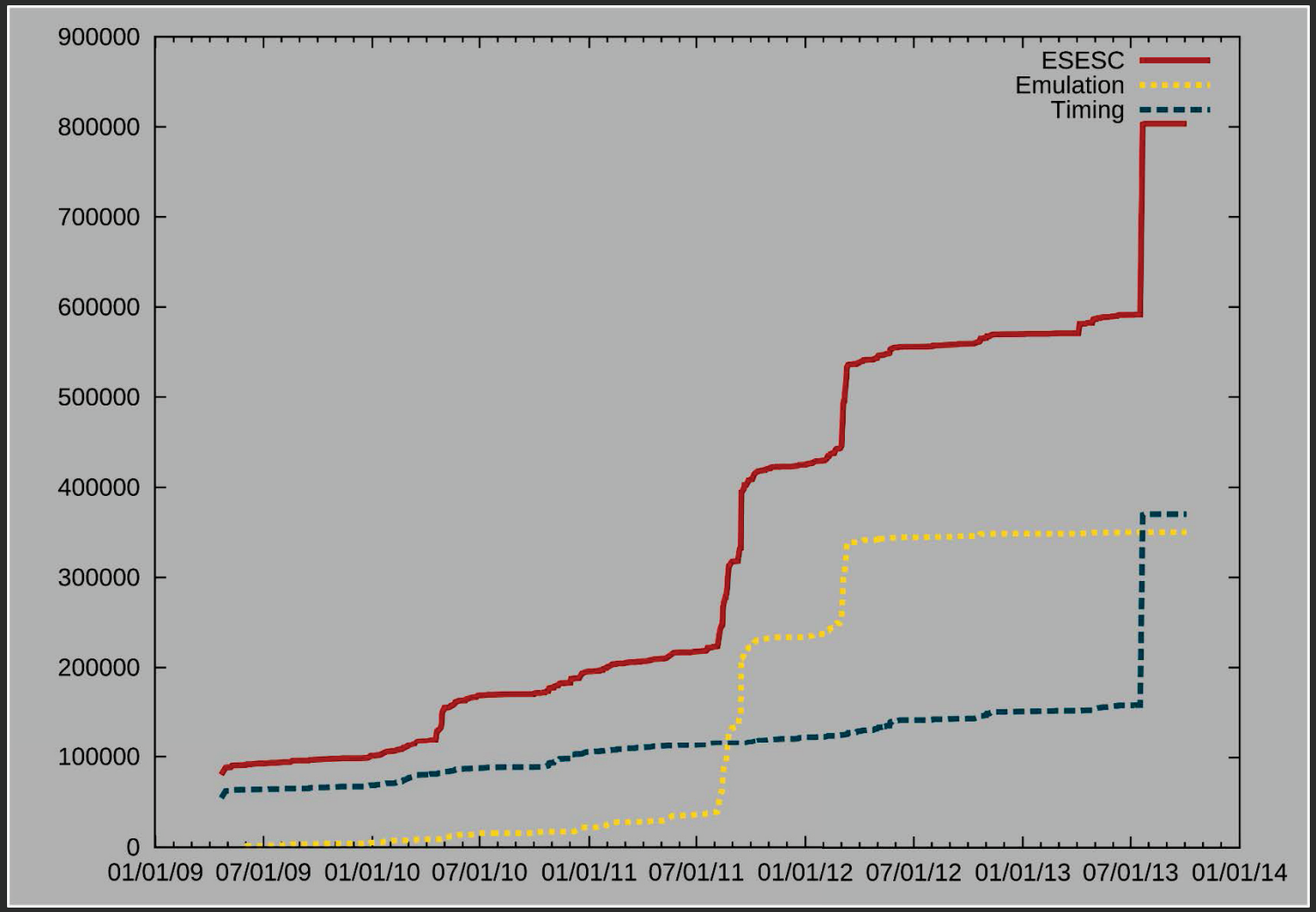
ESESC Usage Sample

```
Aggr.uIPC ProgressedTime globalClock
esesc_testing.U0vLxz AggrR 2.98  0.000000e+00 4.453447e+06
-------------------------------------------------------------------------------------------------------------
#############################################################################
Cache           Occ  AvgMemLat MemAccesses  MissRate  (  RD ,    WR,    BUS) %%DMemAccMB/s
IL1(0)          0.0  2.9       4146594       6.29%    ( 93.7%,  0.0%,  0.0%)     101.50%
-----------------------------------------------------------------------------
DL1(0)          0.0  5.5       3706526       0.36%    ( 99.6%, 99.8%,  0.0%)      90.84%
-----------------------------------------------------------------------------
L2(0)           0.0  14.8      278147       10.47%    ( 89.5%, 89.8%,  0.0%)       6.73%
L3              0.0  46.4      37691         4.56%    ( 95.3%, 58.3%,  0.0%)       0.76%
-----------------------------------------------------------------------------
mada3:~/build/release/run$
mada3:~/build/release/run$rm esesc_*
mada3:~/build/release/run$ll
total 2.0M
-rw-r--r-- 1 renau faculty  14K Sep 27 10:12 cpu2000spMaxk100.conf
-rw-r--r-- 1 renau faculty 8.7K Sep 27 10:12 cpu2000spMaxk10.conf
-rw-r--r-- 1 renau faculty 9.6K Sep 27 10:12 cpu2000spMaxk30.conf
-rw-r--r-- 1 renau faculty 6.6K Sep 27 10:12 cpu2006sp.conf
-rwxr-xr-x 1 renau faculty 780K Sep 27 10:12 crafty.armel
-rw-r--r-- 1 renau faculty  413 Sep 27 10:12 crafty.in
-rwxr-xr-x 1 renau faculty 757K Sep 27 10:12 crafty_sparc32
-rw-r--r-- 1 renau faculty 4.4K Sep 27 10:12 esesc.conf
-rw-r--r-- 1 renau faculty 236K Sep 27 10:12 flp.conf
-rw-r--r-- 1 renau faculty    0 Sep 27 10:13 game.001
-rw-r--r-- 1 renau faculty 3.8K Sep 27 10:12 gpu.conf
-rw-r--r-- 1 renau faculty  847 Oct  4 10:39 memory-arch.dot
-rw-r--r-- 1 renau faculty 1.4K Sep 27 10:12 peq1.conf
-rw-r--r-- 1 renau faculty  24K Sep 27 10:12 shared.conf
-rw-r--r-- 1 renau faculty  18K Sep 27 10:12 therm.conf
-rw-r--r-- 1 renau faculty  26K Sep 27 10:12 therm_lp.conf
-rw-r--r-- 1 renau faculty  14K Sep 27 10:12 tsample.conf
-rw-r--r-- 1 renau faculty  13K Sep 27 10:12 tsample_lp.conf
mada3:~/build/release/run$../main/esesc <^C
mada3:~/build/release/run$ll
total 2.0M
-rw-r--r-- 1 renau faculty  14K Sep 27 10:12 cpu2000spMaxk100.conf
-rw-r--r-- 1 renau faculty 8.7K Sep 27 10:12 cpu2000spMaxk10.conf
-rw-r--r-- 1 renau faculty 9.6K Sep 27 10:12 cpu2000spMaxk30.conf
-rw-r--r-- 1 renau faculty 6.6K Sep 27 10:12 cpu2006sp.conf
-rwxr-xr-x 1 renau faculty 780K Sep 27 10:12 crafty.armel
-rw-r--r-- 1 renau faculty  413 Sep 27 10:12 crafty.in
-rwxr-xr-x 1 renau faculty 757K Sep 27 10:12 crafty_sparc32
-rw-r--r-- 1 renau faculty 4.4K Sep 27 10:12 esesc.conf
-rw-r--r-- 1 renau faculty 236K Sep 27 10:12 flp.conf
-rw-r--r-- 1 renau faculty    0 Sep 27 10:13 game.001
-rw-r--r-- 1 renau faculty 3.8K Sep 27 10:12 gpu.conf
-rw-r--r-- 1 renau faculty  847 Oct  4 10:39 memory-arch.dot
-rw-r--r-- 1 renau faculty 1.4K Sep 27 10:12 peq1.conf
-rw-r--r-- 1 renau faculty  24K Sep 27 10:12 shared.conf
-rw-r--r-- 1 renau faculty  18K Sep 27 10:12 therm.conf
-rw-r--r-- 1 renau faculty  26K Sep 27 10:12 therm_lp.conf
-rw-r--r-- 1 renau faculty  14K Sep 27 10:12 tsample.conf
-rw-r--r-- 1 renau faculty  13K Sep 27 10:12 tsample_lp.conf
mada3:~/build/release/run$../main/esesc <crafty.in
```

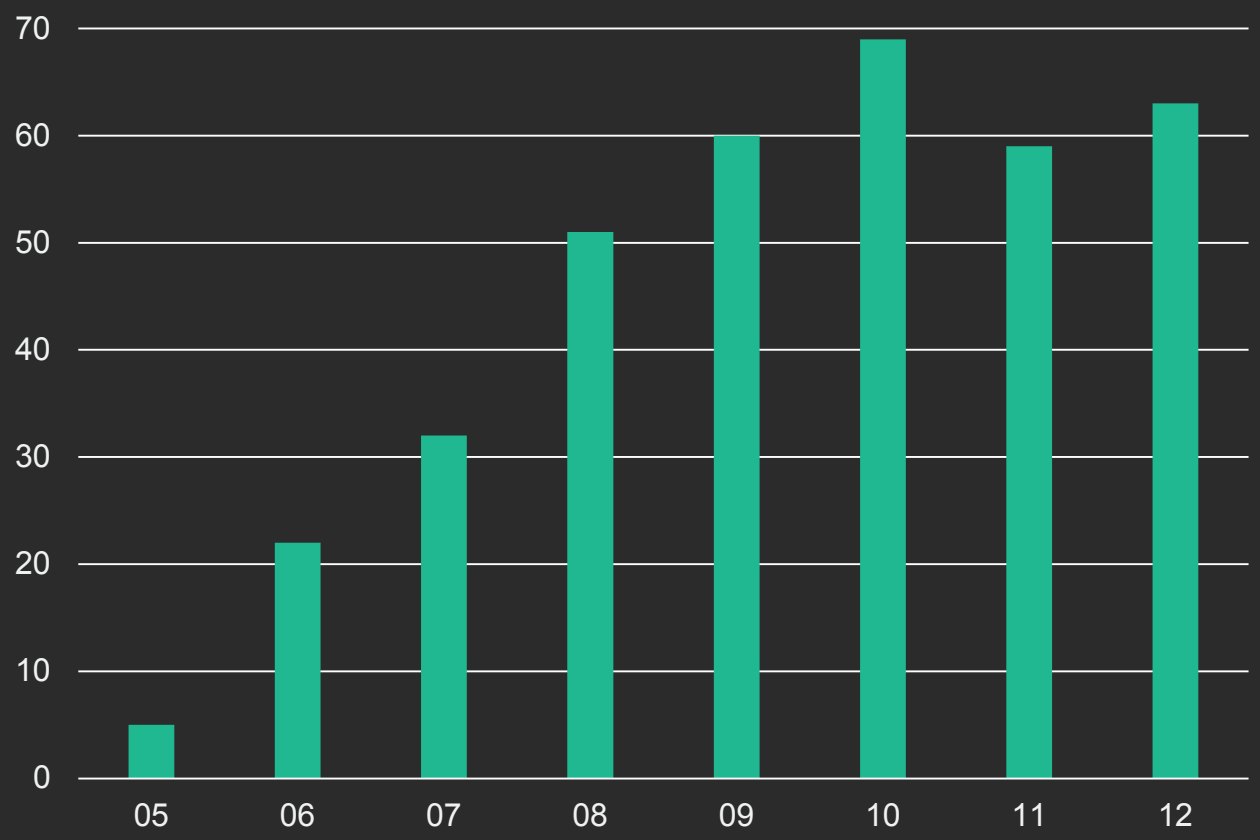
SESC was a large project



ESESC has over 800KLoC Changes

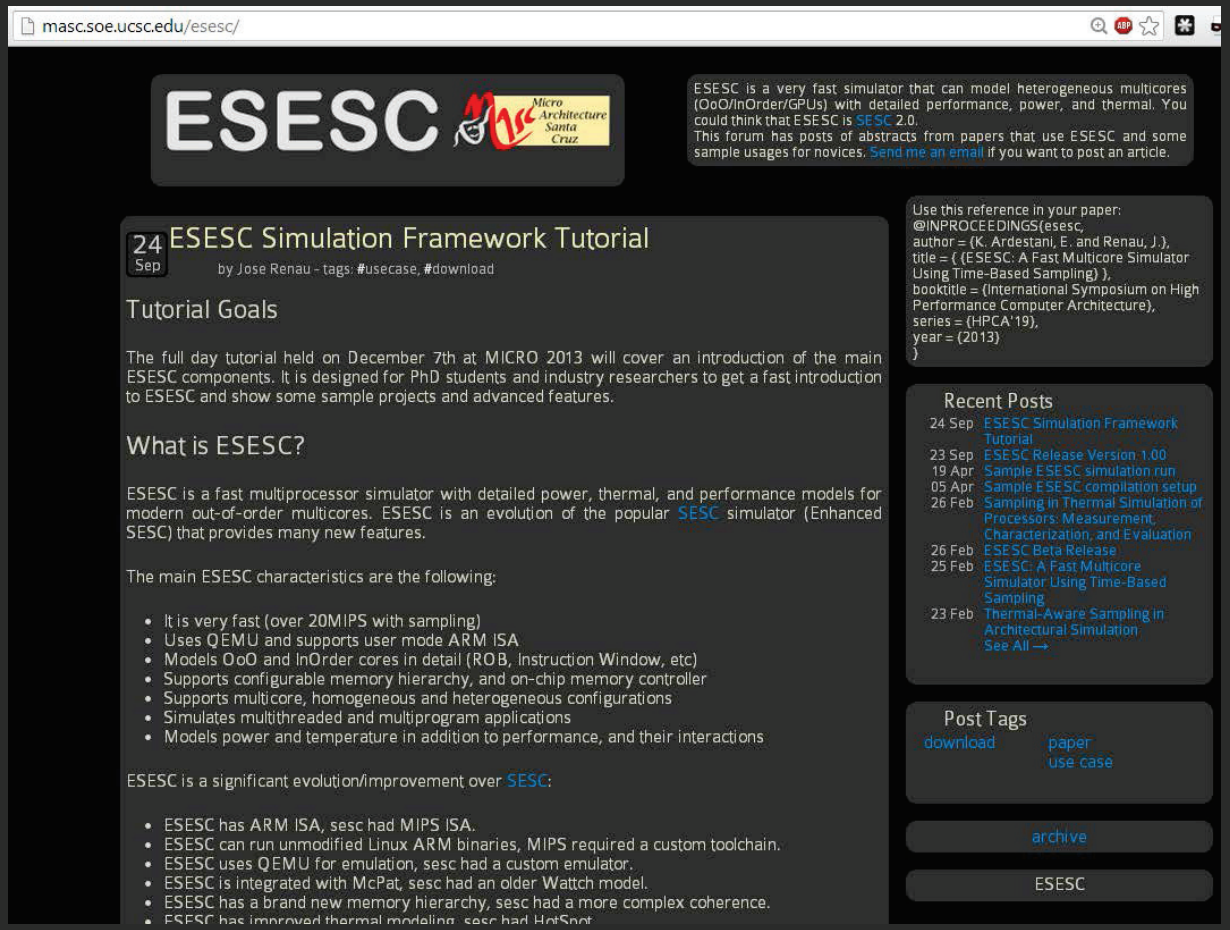


Papers using SESC



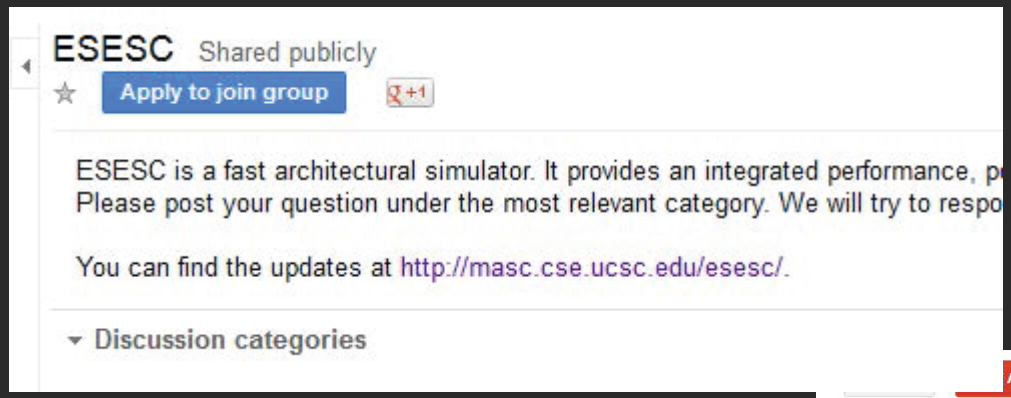
ESESC Blog

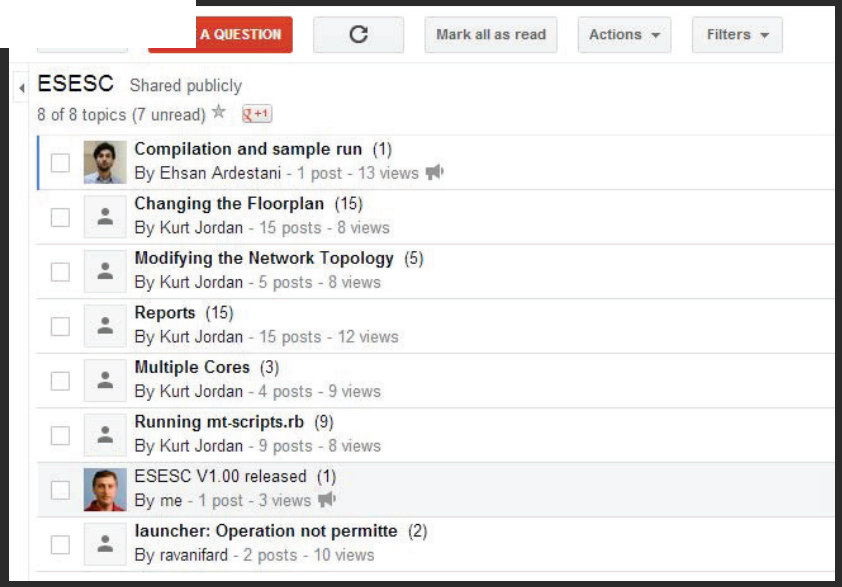
http://masc.soe.ucsc.edu/esesc



ESESC Forum

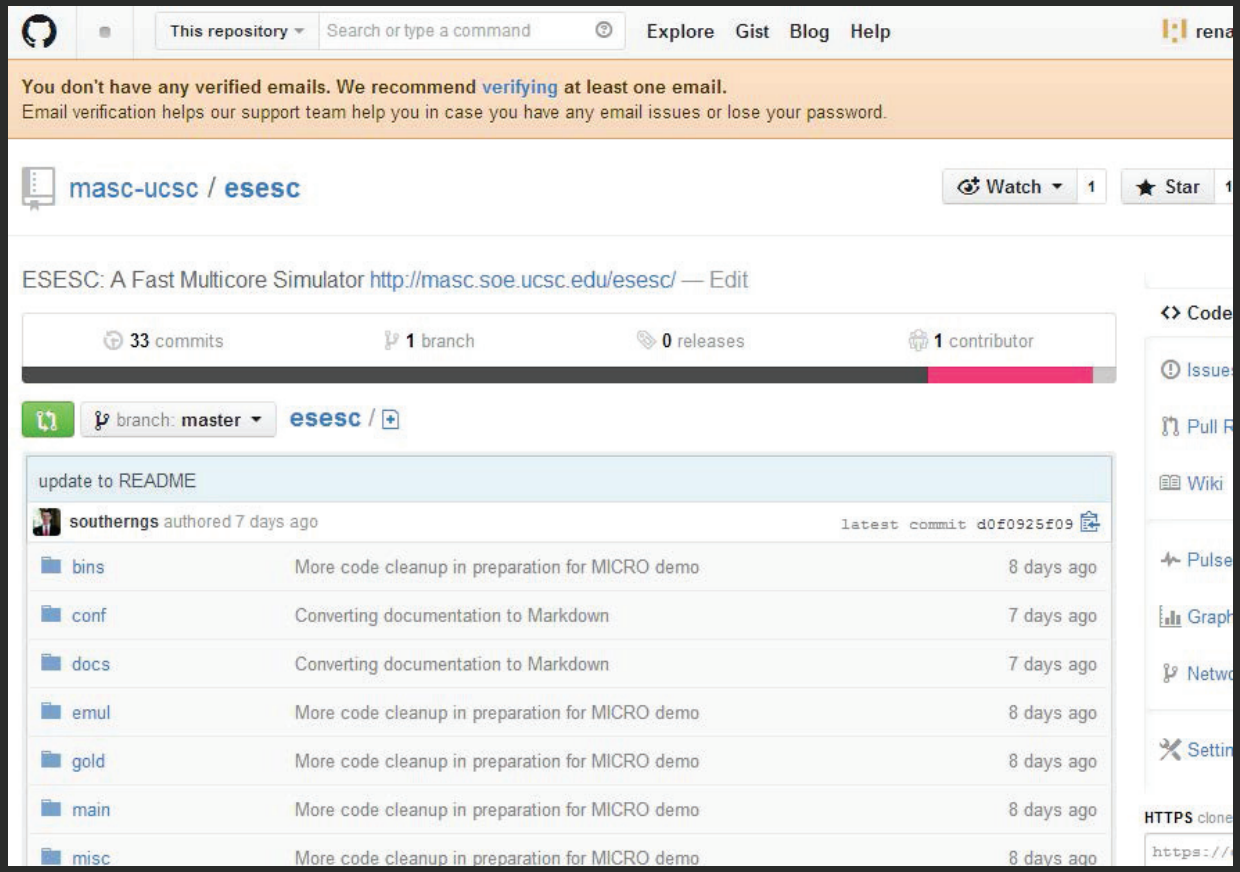
https://groups.google.com/forum/#!forum/esesc





ESESC Public Repository

https://github.com/masc-ucsc/esesc



Remember to cite

- If you use ESESC, cite this paper:
- ESESC: A Fast Multicore Simulator Using Time-Based Sampling, Ehsan K.Ardestani, and Jose Renau, International Symposium on High-Performance Computer Architecture (**HPCA**), February 2013.

Tutorial Outline

- ~~Overview~~
- Code structure and tools
- Timing model internals
- Sampling methods
- Power
- Thermal
- Open questions

Code Structure and Tools

- You will learn:
  - To compile ESESC
  - High level view of code structure
  - Run a simple single threaded application
  - High level view of esesc.conf
  - Simple analysis of statistics dumped

Timing Model Internals

- You will learn:
  - The main timing blocks for a single core
  - The timeline of an instruction in ESESC
  - Some debugging tricks

Sampling Methods

- You will learn:
  - Compare SMARTs vs SimPoint vs TBS
  - Run multithreaded simulation with TBS
  - How to add a new statistics counter

Power

- You will learn:
  - High level view of ESESC power model
  - Run a power simulation
  - Use report.pl to view power numbers
  - Power model options (e.g. LibPeq)

Thermal

- You will learn:
  - High level view of ESESC thermal model
  - Run a thermal simulation
  - Automatically create a floorplan
  - Comparing thermal runs with different thermal management policies

Questions

- Remember to ask questions during talks

Code Structure and Tools
ESESC Tutorial

Speaker: H. Blake Skinner
Matheus Ogleari

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu

Code Structure and Tools

- You will learn:
  - To compile ESESC
  - High level view of code structure
  - Run a simple single threaded application
  - High level view of esesc.conf
  - Simple analysis of statistics dumped

Getting ESESC

Repo:

- https://github.com/masc-ucsc/esesc

Online tutorials:

- http://masc.soe.ucsc.edu/esesc

Building

- Directory structure
  - ~/projs/esesc – source directory

```
ls ~/projs/esesc
mkdir -p ~/build/debug
mkdir -p ~/build/release
```

Building

- Two modes
  - Debug
    - Slower, more information
  - Release
    - Faster, less information

Building

- Build

```
cd ~/build/release
cmake ~/projs/esesc
make
```

Building

- Create a run directory

```
cd ~/build/release
mkdir run
cd run
```

- Copy configuration files

```
cp ~/projs/esesc/conf/* .
```

- Copy binaries to simulate

```
cp ~/projs/esesc/bins/* .
```

Run Release Mode

- From the release build directory, run:

```
~/build/release/main/esesc
```

- Check results:

```
~/projs/esesc/conf/scripts/report.pl -a
```

Demo: Building ESESC

- Build ESESC in Debug and Release modes

Code Structure

- `<esesc root>/conf`
  - Configuration files
- `<esesc root>/docs`
  - READMEs
- `<esesc root>/emul`
  - Source code and libraries for the emulator
- `<esesc root>/main`
  - Top level code directory
- `<esesc root>/simu`
  - Source for simulator

Top level configuration file: esesc.conf

- benchName parameter:
  - Point to an unmodified binary
  - Pass arguments

```
benchName  = "myProgram  myArguments"
```

Launcher

- Simplifies running benchmarks
- Suites
  - CPU 2000/2006

- Usage:

$ launcher [-- rloop] [-- stdin <file>] -- <benchname> [args]

One or more times

report.pl

- `report.pl` is executable script for displaying stats from the ESESC run, using a dump

```
Specify the trace to process
./report.pl [options] <sescDump>
        -a              : Reports for all the stat files in current directory
        -last           : Reports the newest stat file in current directory
        -table          : Statistics table summary (good for scripts)
        -help           : Show this help
```

- The “./report.pl -a” or “./report.pl -last” commands most common to use

report.pl

- Memory Read/Writes, Caches, IPC, Instruction counts, Cycles
- Note: All time units are in **cycles**
- What various fields mean
  - AALU: Arithmetic, Logic (execute stage)
  - BALU: Branching
  - CALU: Control Unit
  - LALU: Loads
  - SALU: Stores
  - B*, br*, or *Br*: Branch-related statistics

Stats from report.pl

```
*****************************************************************************************************
# File  : esesc_testing.9WTcHH : Thu Oct 17 22:39:09 2013
*****************************************************************************************************
Sampler 0 (Procs  0)
         Rabbit Warmup  Detail  Timing  Total  KIPS
  KIPS    40884   7628     431     424   18030
  Time    36.8%   37.8%    5.6%   19.8%          : Sim Time (s) 83.189 Exe   1.475 ms Sim (3000MHz)
  Inst (M)   83.4%       16.0%    0.1%    0.5%  : Approx Total Time  1.475 ms Sim (3000MHz)
*****************************************************************************************************
Proc : Avg.Time : BPType :  Total  :         RAS         :  BPred  :         BTB          :  BTAC
   0 :   11.293 :  ogehl :  95.12% : (100.00% of 7.96%) :  95.55% : ( 98.38% of 38.16%) :  0.04%
-----------------------------------------------------------------------------------------------------
Proc :  rawInst  :   nCommit   :   nInst   :  AALU   :  BALU   :  CALU   :  LALU   :  SALU   :  LD Fwd :
   0 :   6997050 :   13646447  :  13646459 :  61.48% :   5.57% :   0.16% :  17.96% :  14.83% :  0.00%  :
-----------------------------------------------------------------------------------------------------
Proc  IPC  uIPC Active       Cycles  Busy   LDQ   STQ  IWin   ROB  Regs    IO  maxBr  MisBr Br4Clk brDelay
   0  1.58 3.08 0.938       4425069  77.1   0.2   0.2   0.0   0.7   0.0   0.0   0.0    2.2    0.0     2.4
*****************************************************************************************************
Cache           Occ  AvgMemLat MemAccesses  MissRate  (  RD ,    WR,    BUS)   Pow_dyn   Pow_lkg
IL1(0)          0.0  2.6       4597569       3.66%    ( 96.3%,  0.0%,  0.0%)         0         0
-----------------------------------------------------------------------------------------------------
DL1(0)          0.0  5.6       4197060       0.45%    ( 99.4%, 99.8%,  0.0%)         0         0
-----------------------------------------------------------------------------------------------------
L2(0)           0.0  16.1      194347       14.42%    ( 85.5%, 89.3%,  0.0%)         0         0
L3              0.0  43.3      34753         4.90%    ( 95.1%, 66.7%,  0.0%)         0         0
*****************************************************************************************************
                Proc      RF          ROB         fetch        EXE          RNU         LSU          Membus
Dynamic Power   0           0           0           0           0           0           0           0
Leakage Power   0           0           0           0           0           0           0           0
*****************************************************************************************************
```

Summary

- Run ESESC for the first time
- Gather some statistics
- A high level idea of the code structure

Timing Model
ESESC Tutorial

Speakers: Rafael Possignolo
Daphne Gorman

ESESC

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu

Introduction

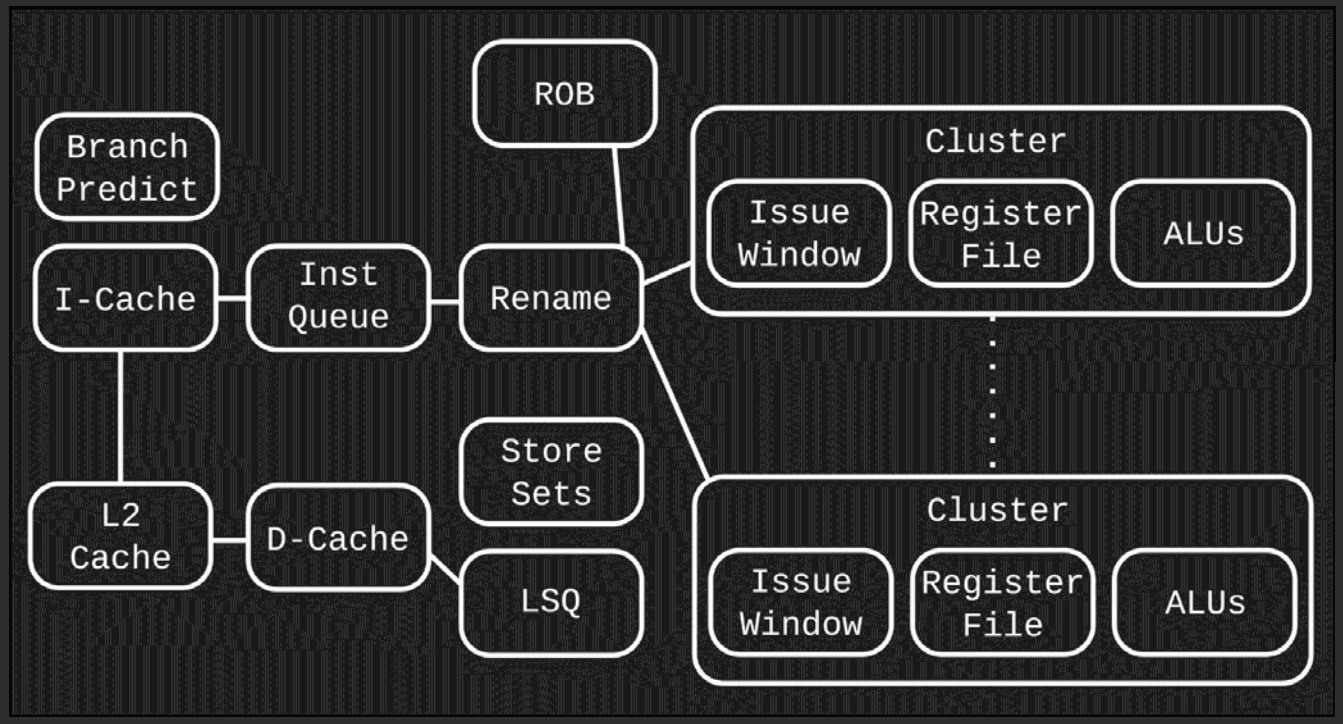
- You will learn:
  - The main timing blocks for a single core
  - The timeline of an instruction in ESESC
  - Memory hierarchy and cache coherence
  - Some debugging tricks

Outline

- OoO Structures in ESESC
- The concept of ports in ESESC
- Memory Hierarchy
- Cache Coherence and Consistency
- Debugging ESESC (Demo)

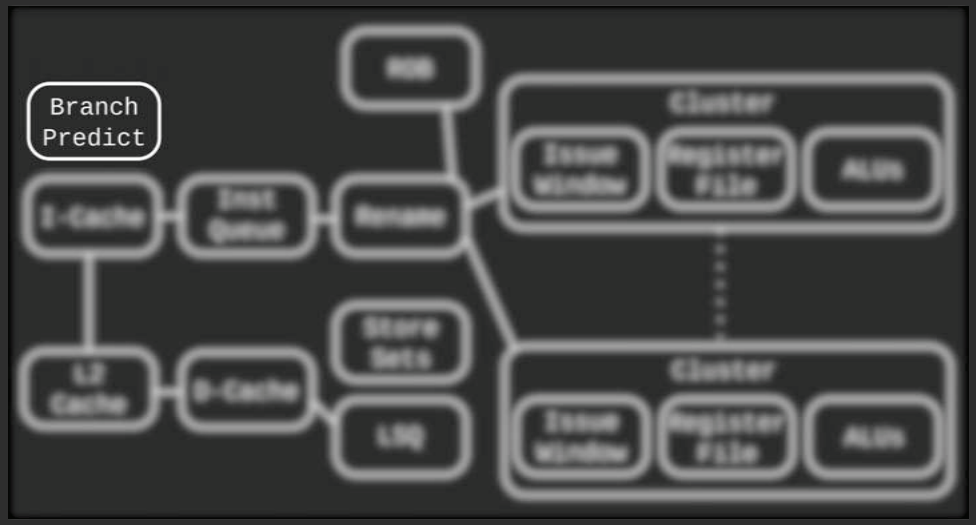
Main OoO Structures

All the parameters for the core structure are in the simu.conf file.



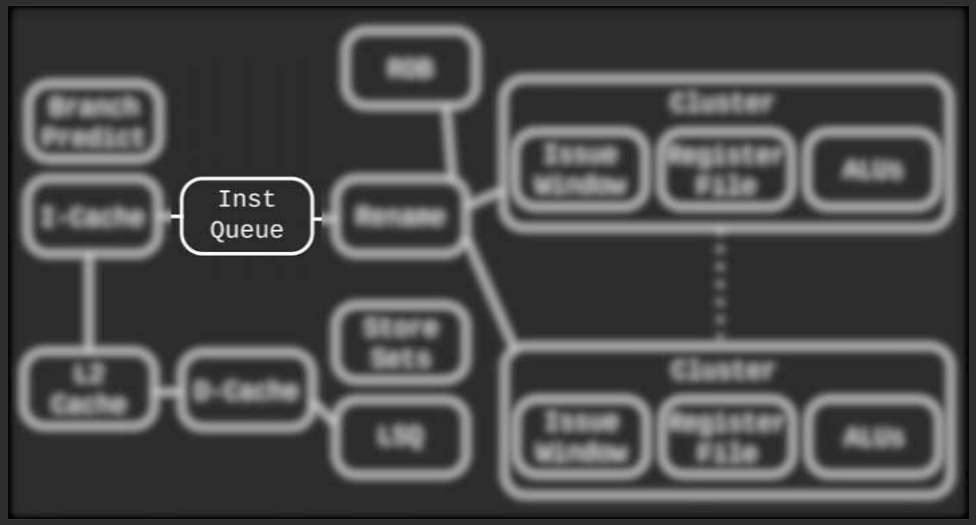
Internals: Branch Predictor

```
[tradCORE]
# not showing all parameters
...
bpredDelay     = 2
bpred          = 'BPredIssueX'
...

[BPredIssueX]
type           = "ogehl"
#type          = "taken“ / "nottaken"
#type          = "oracle“ / hybrid
btbSize        = 4096
btbBsize       = 1
btbAssoc       = 4
btbReplPolicy  = ‘LRU’
rasSize        = 0
nBanks         = 1
...
```



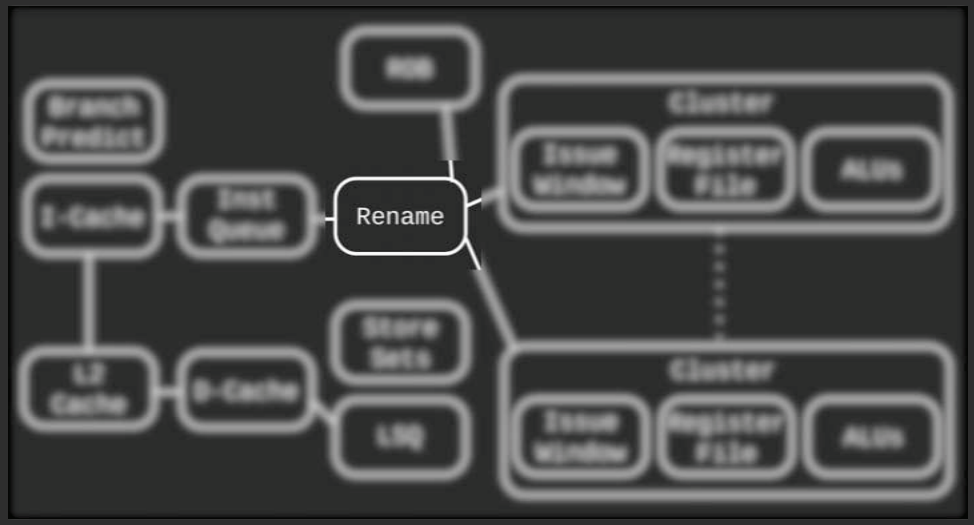
Internals: Instruction Queue

```
[tradCORE]
# not showing all parameters
...
instQueueSize     = 16
...
```



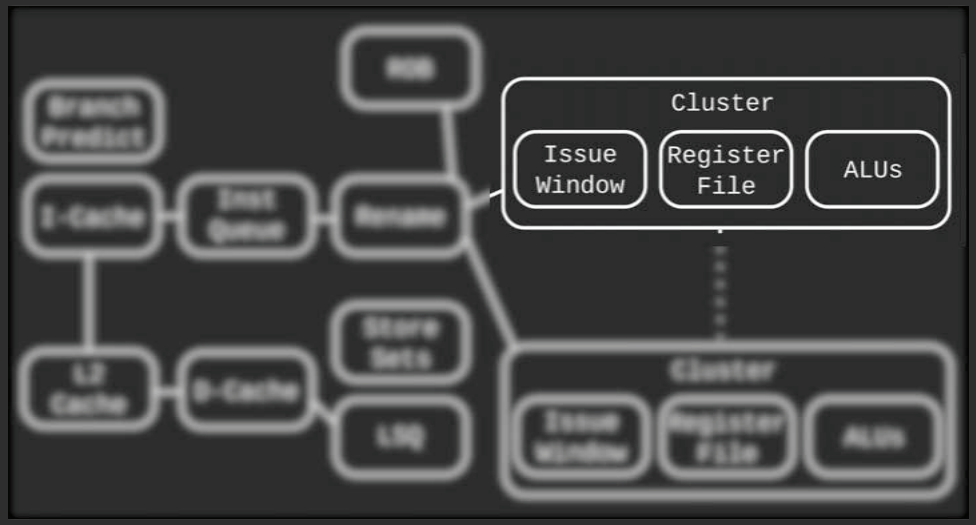
Internals: Rename

```
[tradCORE]
# not showing all parameters
...
renameDelay       = 2
nArchRegs         = 32
...
```



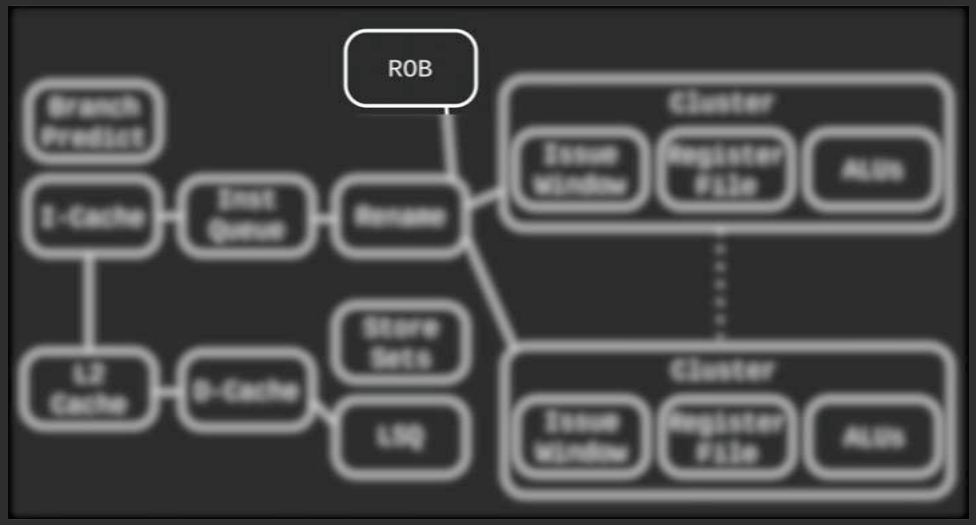
Internals: Cluster

```
[tradCORE]
# not showing all parameters
...
clusterScheduler = 'RoundRobin'
cluster[0]       = 'Aunit'
...
cluster[N]       = 'Nunit'
...


[Aunit]
# Ports will be explained later
Num              = 1
Occ              = 1
```



Internals: ROB

```
[tradCORE]
# not showing all parameters
...
robSize             = 256
retireWidth         = 4
...
```



Internals: LSQ

```
[tradCORE]
# not showing all parameters
...
stForwardDelay      = 256
...
maxLoads            = 48
maxStores           = 32
...
LFSTSzie            = 512
StoreSetSize        = 8192
noMemSpec           = false
...
```

Outline

- OoO Structures in ESESC
- The concept of ports in ESESC
- Memory Hierarchy
- Cache Coherence and Consistency
- Debugging ESESC (Demo)

Ports

- Functional units are modeled as ports
  - `Num` is the number of instances of the unit
  - `Occ` is the occupancy – the number of cycles between new instructions in the unit
    - This is not the same as the latency

```
[port]
Num      = 1
Occ      = 3
```

Ports: Example

Fully pipelined unit (3 stages)

Num = 1
Occ = 1

Stage1	Stage 2	Stage 3
	In1	
In2		

Not-fully pipelined unit :
Each stage takes 2 cycles

Num = 1
Occ = 2

Stage1	Stage 2	Stage 3
	In1	
	In1	
In2		
In2		

Multiple instances:

Num = 2
Occ = 1

Stage1	Stage 2	Stage 3
In1		
In2		

Timing Execution Example

```
Fetch
        Num: 2
        Lat: 1
Decode
        Num: 2
        Lat: 2
Instruction Queue
        Num: 1
        Lat: 1
Cluster
        Num: 2
        Occ: 1
        Lat: 4
```

Cycles	1	2	3	4	5	6	7	8	9
i1	F	D	D	IQ	C	C	C	C	
i2	F	D	D		IQ	C	C	C	C
i3		F		D	D	IQ		C	
i4		F		D	D		IQ		C
i5				F		D	D	IQ	

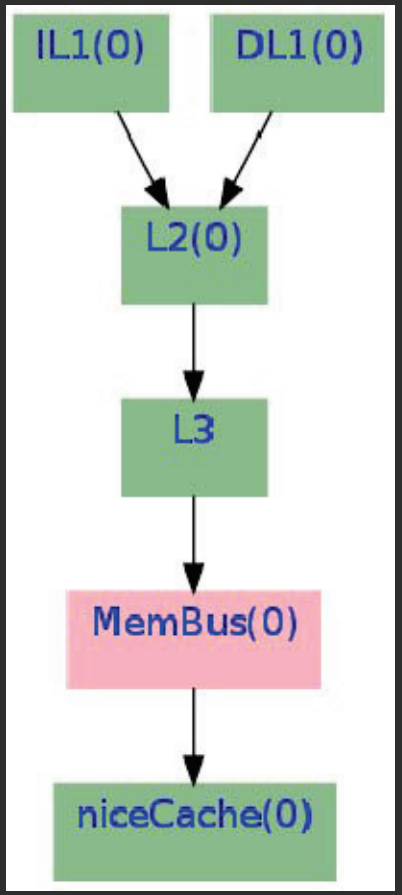
Outline

- OoO Structures in ESESC
- The concept of ports in ESESC
- Memory Hierarchy
- Cache Coherence and Consistency
- Debugging ESESC (Demo)

Memory Hierarchy

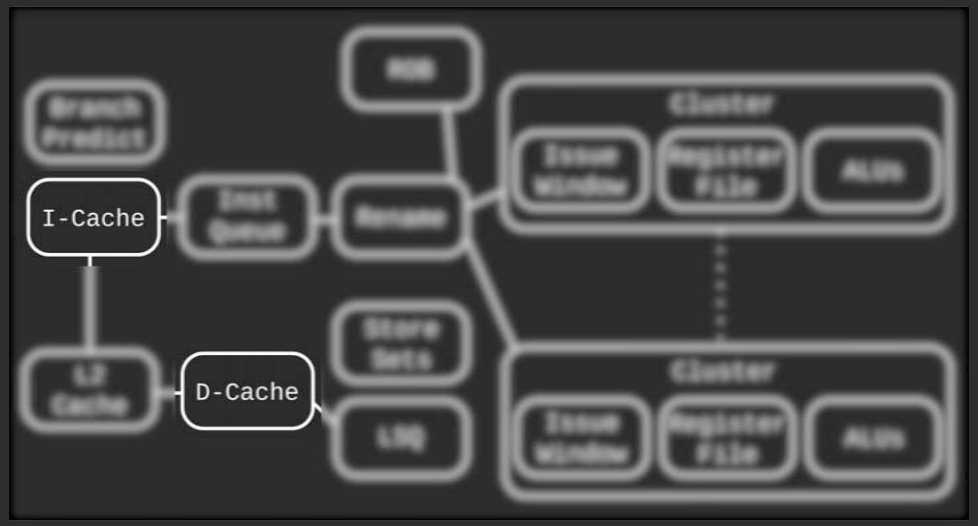
Parameters for the memory hierarchy are also in the shared.conf file.

```
# not showing all parameters

[tradCORE]
IL1 = "IL1_core IL1" # <Class Name>
DL1 = “DL1_core DL1”

[IL1_core]
deviceType = “cache”
lowerLevel = “PrivL2 L2”

[DL1_core]
deviceType = “cache”
lowerLevel = “PrivL2 L2”

[PrivL2]
...

[...]
```



Internals: L1 Cache

```
[DL1_core]
#not showing all the parameters
deviceType          = 'cache'
...
hitDelay            = 4
missDelay           = 4
MSHR                = “DL1_MSHR”
size                = 32*1024
assoc               = 4
bsize               = 64
writePolicy         = 'WB'
replPolicy          = 'LRU‘
??NumPorts          = 0
??PortOccp          = 0

...
```

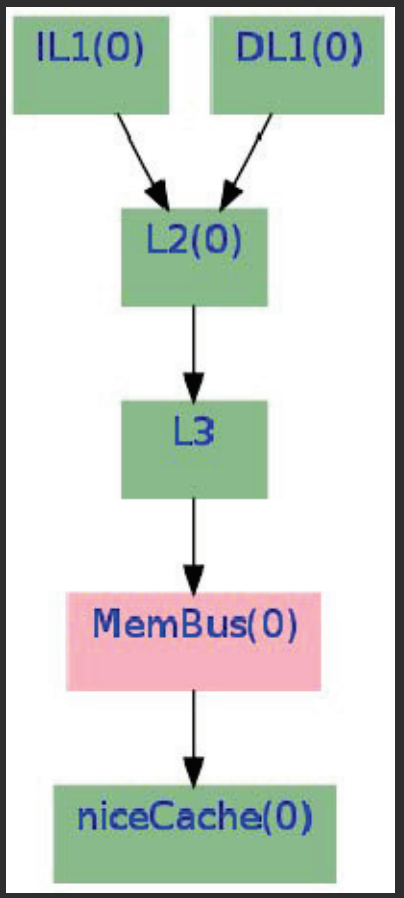


Where ?? is one of following:

- rd
- wr
- bk
- ll

Memory Hierarchy: MSHR

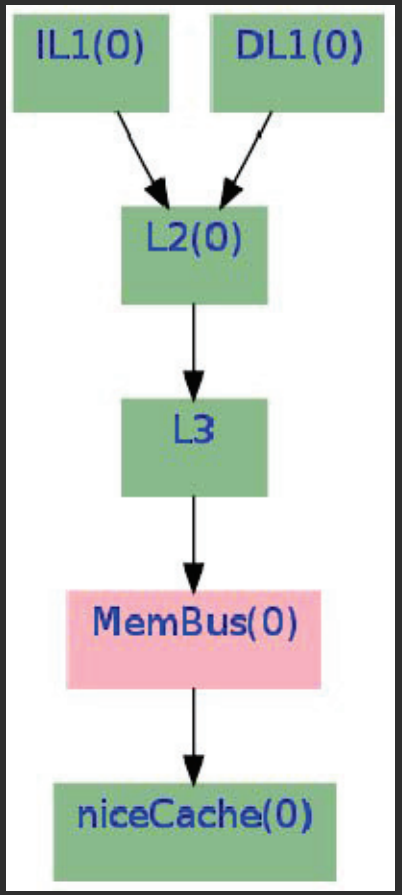
- MSHR – miss status handling register
  - An MSHR instance can be associated to a cache instance
  - Allows multiple ways to handle cache misses




```
[DL1_MSHR]
type
size
subentries
```

Memory Hierarchy: TLBs

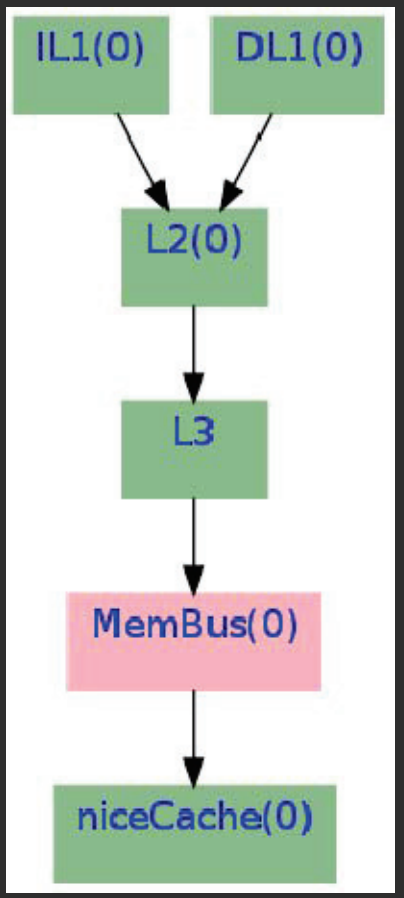
- TLBs
  - Used the same way as caches
  - Can be placed at any level




```
[tlb]
Devicetype = 'tlb'
MSHR = 'mshr_name'
```

Memory Hierarchy: NiceCache

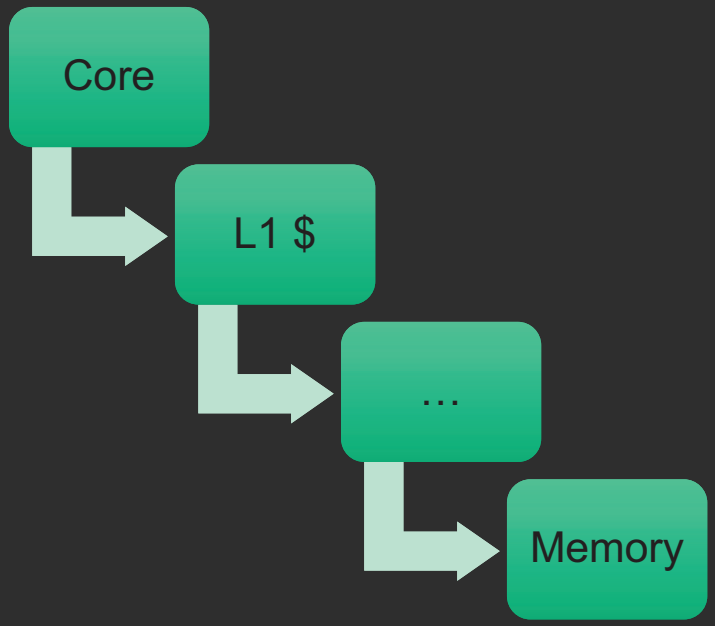
- Guaranteed 100% hit rate
- Models main memory (no misses)

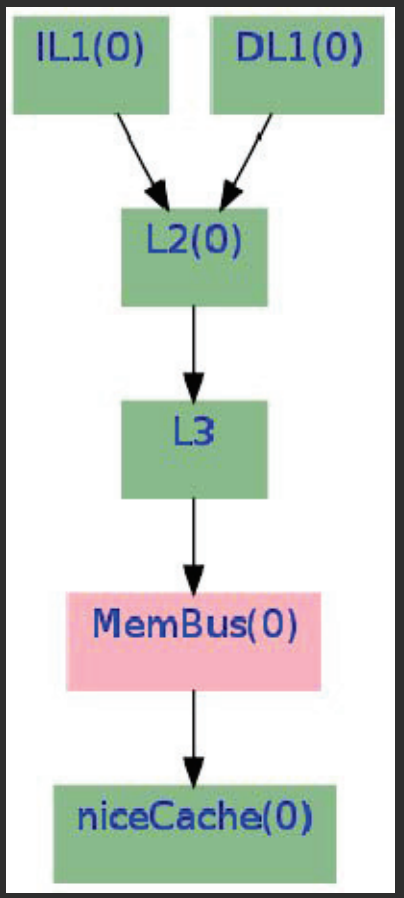



```
[memory]
device      = 'niceCache'
hitDelay    = 200
```

Memory Hierarchy

- Different possible outcomes:
  - Hit in L1
  - Miss in L1 / Hit in L2
  - Miss both L1 and L2 / Hit L3
  - so forth...

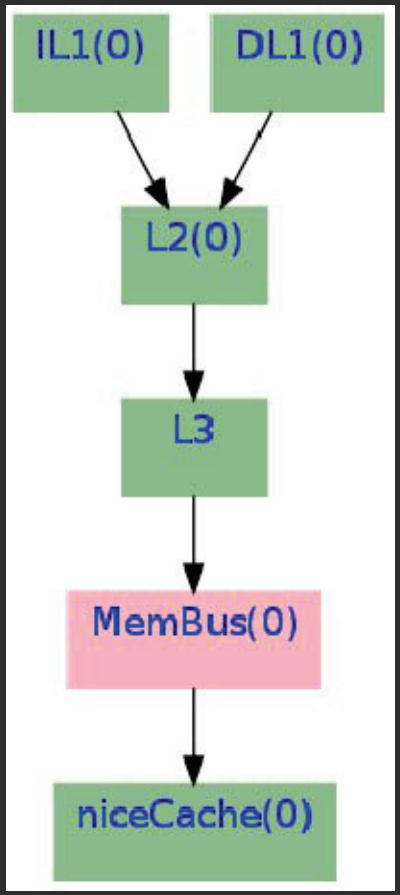




Memory Hierarchy

- When you run ESESC a cool graph of the memory hierarchy is generated in the running folder, using Graphviz.
- You can transform it to png by executing:

```
dot -Tpng memory-arch.dot > memory-arch.png
```



Timing Models Internals

- Possible outcomes:

Hit / Miss	Total latency
Hit in L1	L1 hit latency
Hit in L2	L1 miss latency + L2 hit latency
Hit in L3	L1 miss latency + L2 miss latency + L3 hit latency
Hit in memory	L1 miss latency + L2 miss latency + … + memory latency

Memory Speculation

- ESESC uses aggressive memory speculation schemes
  - Full Load Store Queues
    - Out of order
  - Store Set
    - Memory dependency for memory speculation

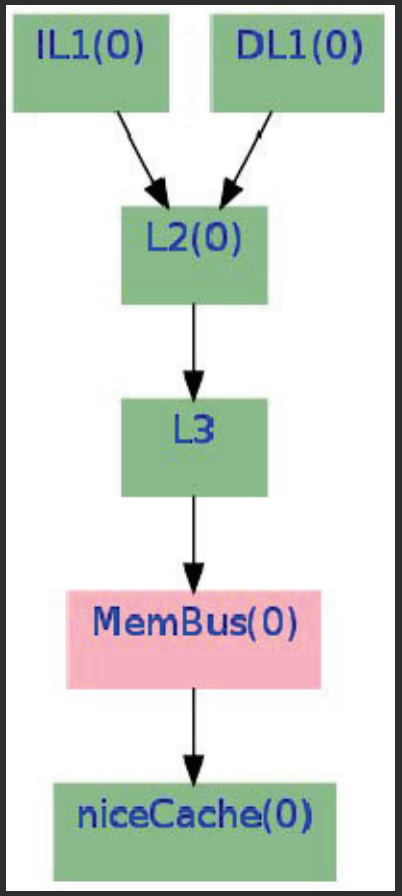
Outline

- OoO Structures in ESESC
- The concept of ports in ESESC
- Memory Hierarchy
- Cache Coherence and Consistency
- Debugging ESESC (Demo)

Multicore Memory

- To enable cache coherence protocols, when instantiating a cache, specify how many cores are sharing the cache:

```
[core1]
...
DL1 = “DL1_core DL1_1”

[core2]
...
DL1 = “DL1_core DL1_2”

[DL1_core]
deviceType = “cache”
lowerLevel = “PrivL2 L2 sharedby 2”
```



Outline

- OoO Structures in ESESC
- The concept of ports in ESESC
- Memory Hierarchy
- Cache Coherence and Consistency
- Debugging ESESC (Demo)

Summary

- The main timing blocks for a single core
- The timeline of an instruction in ESESC
- Memory hierarchy and cache coherence
- Some debugging tricks

Sampling Methods
ESESC Tutorial

Speaker: Gabriel Southern

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu

Sampling Methods

- You will learn:
  - Compare SMARTs vs. SimPoint vs. TBS
  - Run multithreaded simulation with TBS
  - How to add a new statistics counter

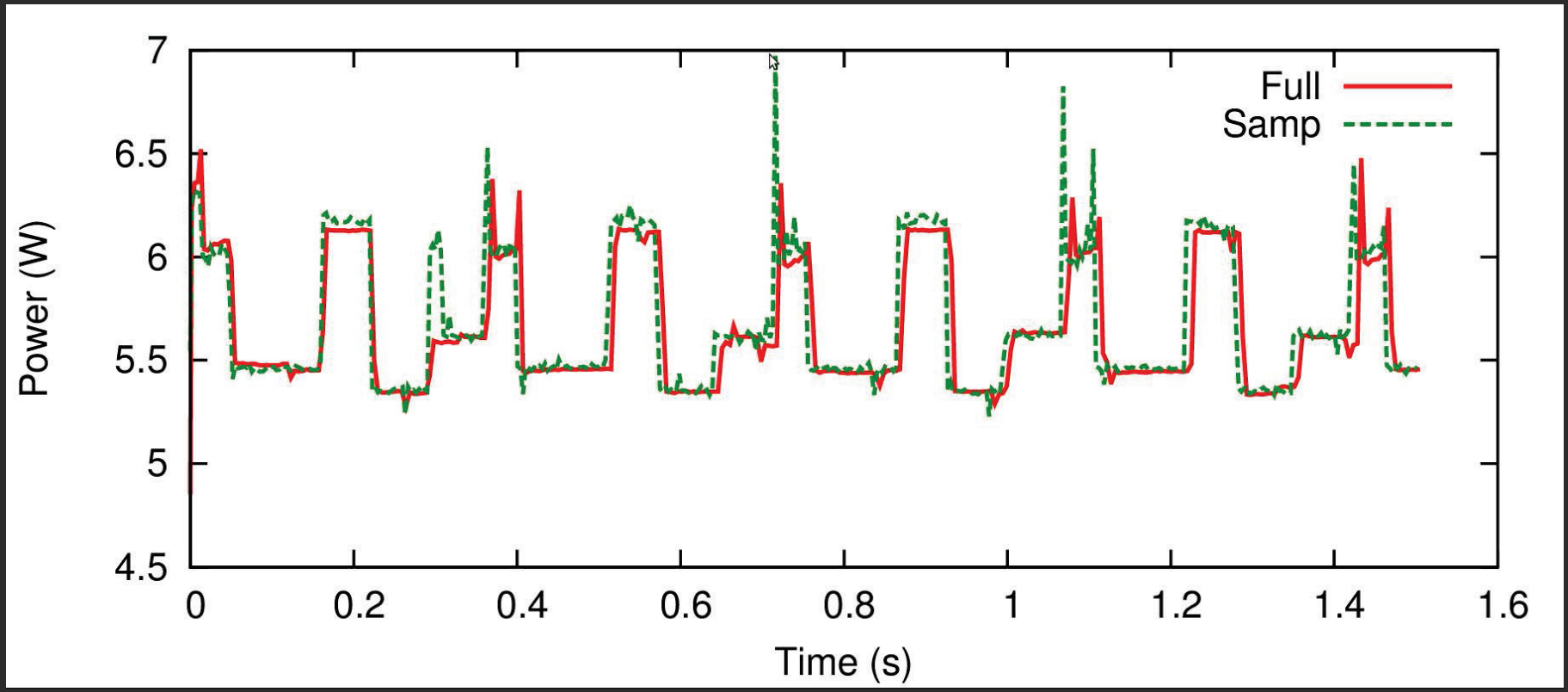
Outline

- Statistical sampling overview
  - SMARTS
  - SimPoint
  - Time Based Sampling (TBS)
- ESESC sampling parameters
- Demo single threaded sampling
- *Lunch Break*
- Demo multithreaded sampling
- Collecting statistics in ESESC

Sampling References

- SimPoint
  - *Automatically Characterizing Large Scale Program Behavior,* Sherwood, et al., ASPLOS 2002.
- SMARTS
  - *SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling,* Wunderlich, et al., ISCA 2003.
- Thermal
  - *Thermal-Aware Sampling in Architectural Simulation,* Ardestani, et al., ISPLED 2012.
- Multithreaded
  - *ESESC: A Fast Multicore Simulator Using Time-Based Sampling,* Ardestani, et al., HPCA 2013

Sampling Overview



SMARTS

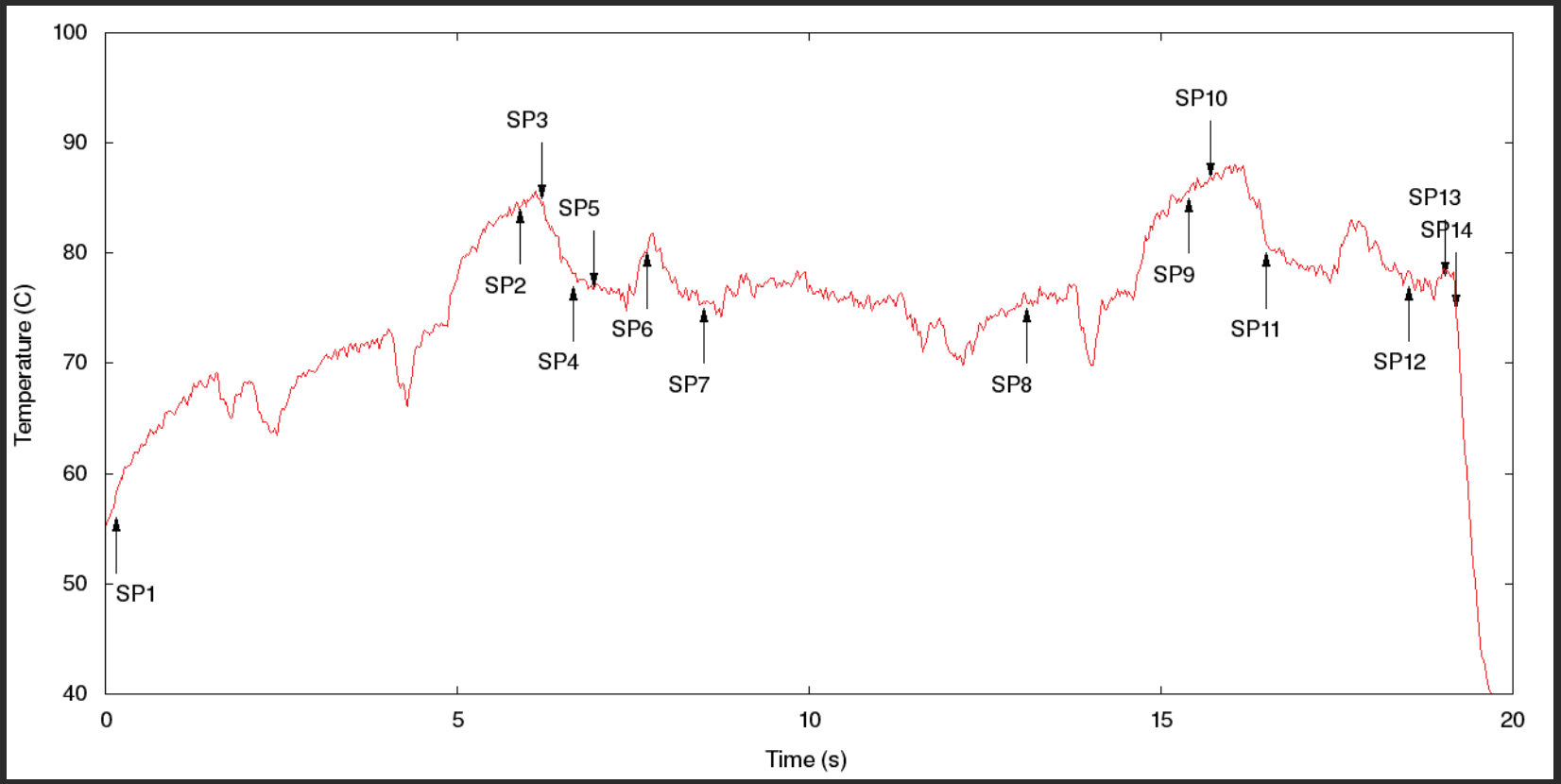
- Applies statistical sampling theory to computer architecture simulation
  - Periodic sampling of instructions
  - Sampling defined with intervals and samples
- Simulation modes
  - Functional warming
    - Always simulate caches and branch predictor
  - Detailed warming
    - Full simulation but discard statistics
  - Sampling unit
    - Full simulation and keep statistics

Warmup

Detailed

Timing

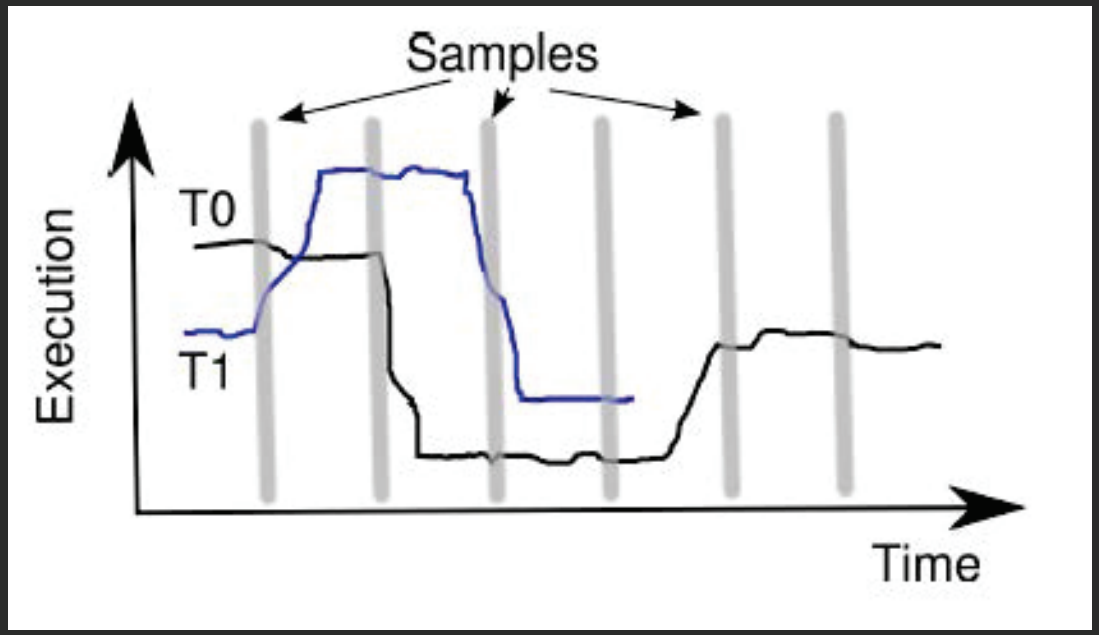
SimPoint



- Programs have phases
- Execute single sample for each phase
- Weight statistics by phase frequency

Thermal and Multithreaded Sampling

- SMARTS and SimPoint do not work for thermal and multithreaded sampling
- Use Time Based Sampling (TBS)

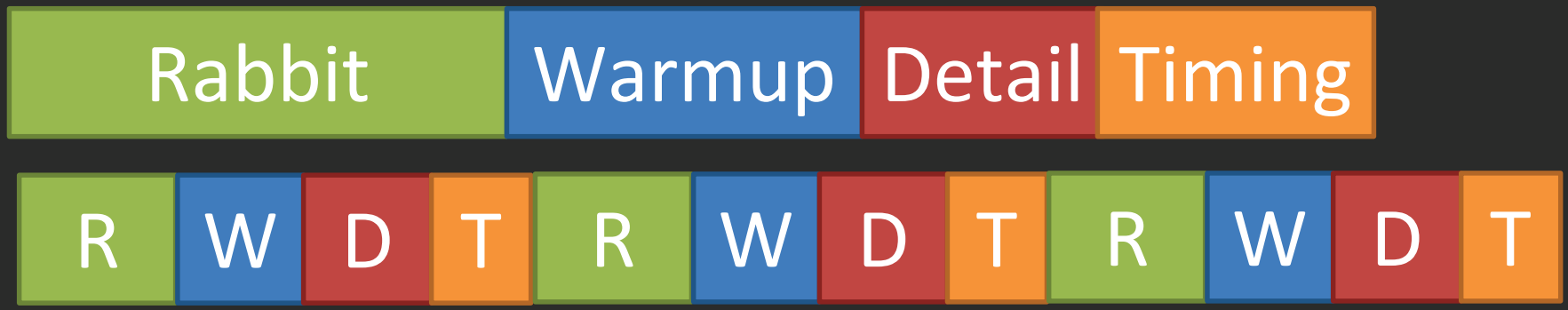


Outline

- Statistical sampling overview
  - SMARTS
  - SimPoint
  - Time Based Sampling (TBS)
- ESESC sampling parameters
- Demo single threaded sampling
- *Lunch Break*
- Demo multithreaded sampling
- Collecting statistics in ESESC

ESESC Sampling Modes

- Rabbit
  - Emulation only
- Warmup
  - Update cache
- Detail
  - Full simulation but discard statistics
- Timing
  - Full simulation and keep statistics



Statistics in ESESC

- Unified statistics management with GStats classes
- Raw GStats output is processed by report.pl script
- Only collect statistics during timing simulation
- How to add a new GStat explained in sampling demo #3

ESESC Sampling Parameters

- `type`
  - `SkipSim, SMARTS, SPoint, Periodic`
- `nInstSkip`
  - Number of instructions main thread skips
- `nInstSkipThreads`
  - Number of instructions spawned threads skip
- `maxnsTicks`
  - Maximum simulation time in ns
- `nInstMax`
  - Maximum number of instructions to simulate
- `nInstRabbit`
  - Emulation only
- `nInstWarmup`
  - Emulation plus cache
- `nInstDetail`
  - Detailed warm-up of full pipeline
- `nInstTiming`
  - Collect statistics

```
[PeriodicMode]
type                = "Periodic"
nInstSkip           = 1e9
nInstSkipThreads    = 1e9
maxnsTicks          = 1e9
nInstMax            = 2e9
nInstRabbit         = 0
nInstWarmup         = 493e4
nInstDetail         = 2e4
nInstTiming         = 5e4
PowPredictionHist = 3   #for power
doPowPrediction   = 1   #for power
TPR                 = 1e0 #for power/thermal
```

SimPoint in ESESC

- Sampling module supports SimPoints with SPoint sampler type
  - `spointSize`
  - `spoint`
  - `spweight`
- Limitations
  - No multithreaded or thermal with SimPoint in ESESC
  - Need to collect BBVs and generate SimPoint output
    - bbv-editor.rb can help
- Time Based Sampling is preferred

```
# Example SimPoints from crafty for SPARC
[SPointMode186crafty]
type              = SPoint
spointSize        = 1e8
spoint[0]         = 67e8
spweight[0]       = 0.152
spoint[1]         = 436e8
spweight[1]       = 0.294
spoint[2]         = 466e8
spweight[2]       = 0.028
spoint[3]         = 643e8
spweight[3]       = 0.189
spoint[4]         = 779e8
spweight[4]       = 0.226
spoint[5]         = 838e8
spweight[5]       = 0.111
doPowPrediction   = 0
nInstSkip         = 1e1
nInstSkipThreads  = 1e1
```

Outline

- Statistical sampling overview
  - SMARTS
  - SimPoint
  - Time Based Sampling (TBS)
- ESESC sampling parameters
- Demo single threaded sampling
- *Lunch Break*
- Demo multithreaded sampling
- Collecting statistics in ESESC

Demo 1: Single-threaded with TBS

- Run crafty with TBS
- Single core configured

Sampling Report

```
**************************************************
Sampler 0 (Procs  0)
          Rabbit Warmup  Detail  Timing  Total  KIPS
  KIPS    96431  15842    1268    1272   42610
  Time    36.9%   43.0%    4.5%   15.6%
  Inst (M)   83.4%         16.0%    0.1%    0.5%   :
**************************************************
```

- Time and instruction percentage in each mode

Lunch Break

Outline

- Statistical sampling overview
  - SMARTS
  - SimPoint
  - Time Based Sampling (TBS)
- ESESC sampling parameters
- Demo single threaded sampling
- *Lunch Break*
- Demo multithreaded sampling
- Collecting statistics in ESESC

ESESC Multicore Configuration

- Homogenous or heterogeneous multicore configuration
- Homogenous
  - cpusimu[0:NUM_CORES-1]
- Heterogeneous
  - cpusimu[0] = ‘coreType1’
  - cpusimu[1] = ‘coreType2’

```
#Single core
cpuemul[0]  = 'QEMUSectionCPU'
cpusimu[0]  = "$(coreType)"

#Homogenous multicore
#cpuemul[0:4]  = 'QEMUSectionCPU'
#cpusimu[0:4]  = "$(coreType)"

#Heterogenous multicore (example only)
#cpuemul[0:1]  = 'QEMUSectionCPU'
#cpusimu[0]  = "$(coreTypeA)"
#cpusimu[1]  = "$(coreTypeB)"
```

- Scripts for running benchmarks
  - spec-scripts.rb
  - mt-scripts.rb

Demo 2: Multithreaded Sampling

- Run blackscholes with TBS
- Multicore configuration

Outline

- Statistical sampling overview
  - SMARTS
  - SimPoint
  - Time Based Sampling (TBS)
- ESESC sampling parameters
- Demo single threaded sampling
- Demo multithreaded sampling
- Collecting statistics in ESESC

Statistics with Sampling

- Use GStats class for simulation statistics
- Each GStat must have a unique name
  - All stats stored in a hash map
  - Need to use state of instruction when updating counter
- `GStatsCntr`
  - Counter that supports
    - `add` – add specified amount to counter
    - `inc` – increment by 1
    - `dec` – decrement by 1
- `GStatsAvg`
  - Average value
- `GStatsMax`
  - Number of samples and max value

Example GStat: nCommitted

- GStatsCntr nCommitted counts number of committed instructions
  - Defined in GProcessor.h as part of GProcessor class
- Used in OoOProcessor which inherits from GProcessor
- Reads state of dinst->getStatsFlag()

```
436     if (!flushing) {
437       nCommitted.inc(dinst->getStatsFlag());
438     }
```

Demo 3: Add a Counter to ESESC

- Add counter to ESESC
- Run and explain GStat output

Summary

- ESESC has built-in support for statistical sampling
  - Instruction based (SMARTS)
  - Phase based (SimPoint)
  - **Time Based Sampling (TBS)**
    - Recommended choice
    - Works with thermal and multithreaded simulation
- Use GStats for simulations statistics
  - Weights statistics based on sampling mode

Power Model
ESESC Tutorial

Speaker: Alamelu Sankaranarayanan
Meeta Sinha

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu

Power Model

- You will learn:
  - High level view of ESESC power model
  - Run a power simulation
  - Use report.pl to view power numbers
  - Power model options (e.g. LibPeq)

References

CACTI

1. Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P. Jouppi. CACTI 6.0: A tool to model large caches. *Technical Report, HP Laboratories*, 2009.
2. Sheng Li et al. CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques. *IEEE/ACM International Conference on Computer-Aided Design*, pages 694–701, 2011.

McPAT

3. Sheng Li et al. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. *IEEE/ACM International Symposium on Microarchitecture*, pages 469–480, 2009.

LibPeq

4. Elnaz Ebrahimi. Pareto-optimal methodology for cache and SRAM modeling. *MS thesis, University of California, Santa Cruz*, 2011.
5. Meeta Sinha. Equation-based power model integration in ESESC. MS thesis, University of California, Santa Cruz, 2013.

Outline

- Overview of the ESESC power model
  - High level description of the power model
  - Structure & components
  - Configuration
- Power Model Demo & Reported values
- Power Model options
- LibPeq
- Code structure

Computation of Power

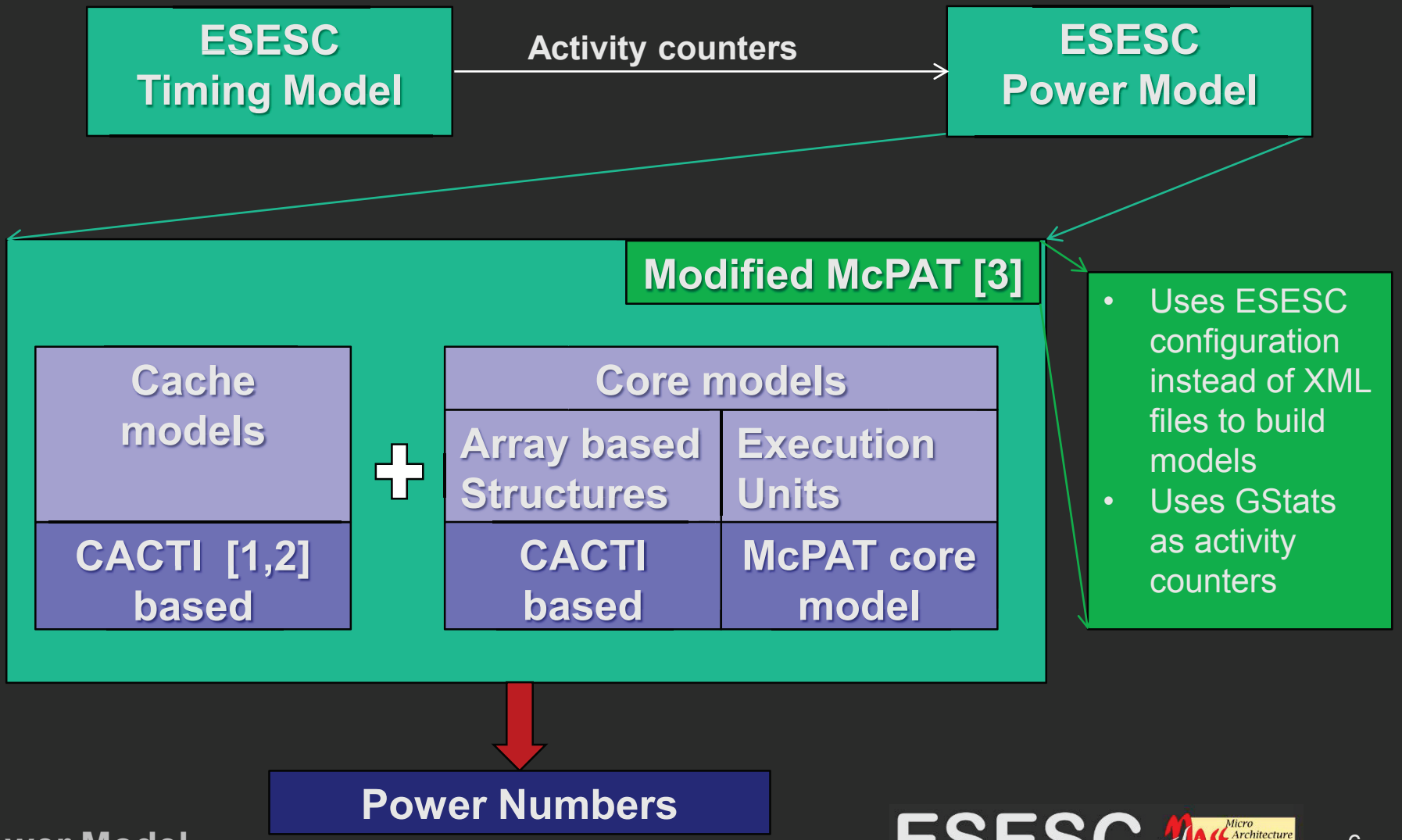
- *Power =*

Event Counters	×	Energy associated with each event.

• Tag Array Reads • Miss buffer accesses • Total instructions • Branch instructions • ALU accesses Etc.	×	• # memory ports • Linesize • Core tech node • Issue width • Branch mispreds Etc.

MODEL

Power Model - Structure



Power Model - Components

Renaming Unit (RNU)

iFRAT
iRRAT
ifreeL
fFRAT
fRRAT
ffreeL

Fetch Unit

global BPT
L1_local BPT
chooser
RAS
BTB
IB

Icache

mmu-itlb
ifu-icache

dcache

mmu-dtlb
lsu-dcache
lsu-dcachecc

Load Store Unit (LSU)

lsu-LSQ
lsu-LoadQ
lsu-ssit
lsu-lfst

Execution Unit (EXE)

EXEU
fp_u
int_inst_window
fp_inst_window

Register File

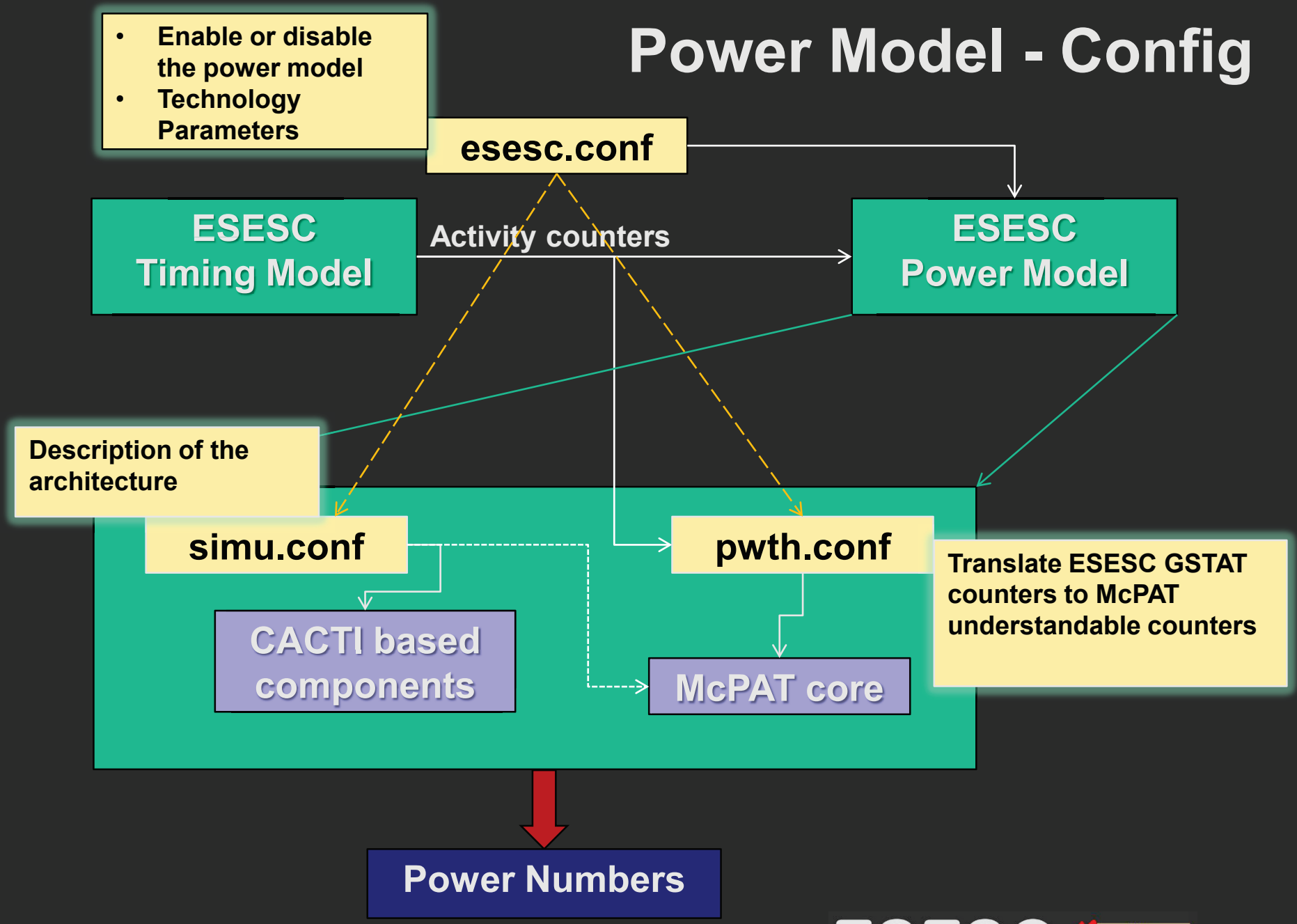
IRF
FRF

Reorder Buffer(ROB)

Array-based structures modeled by CACTI for McPAT

McPAT Core Model

Power Model - Config



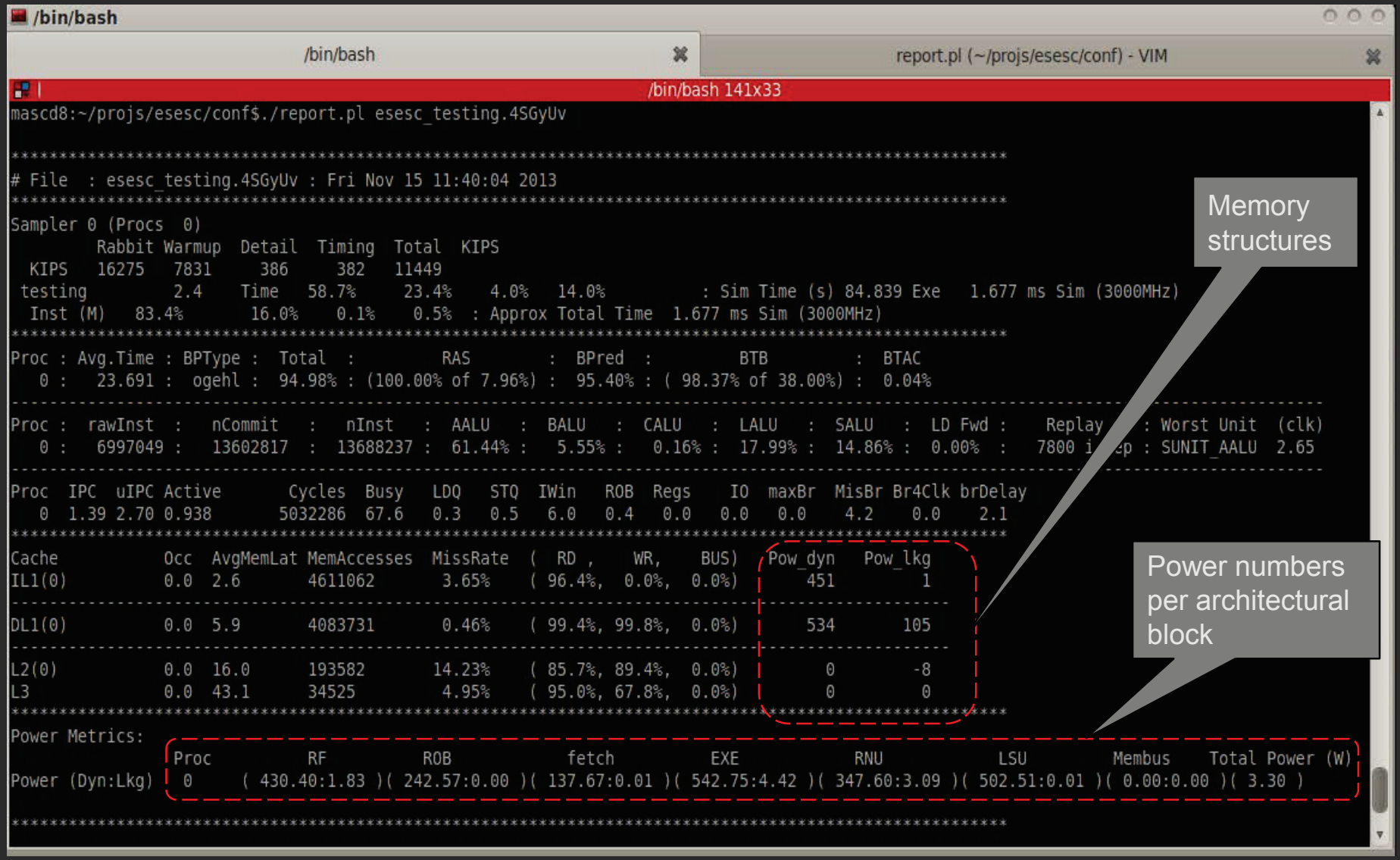
Outline

- Overview of the ESESC power model
  - High level description of the power model.
  - Structure & components
  - Configuration
- Power Model Demo & Reported values
- Power Model options
- LibPeq
- Code structure

Power Model - Demo

- Enable the power model and run a benchmark
- Use report.pl to view the power numbers

Reported numbers



Outline

- Overview of the ESESC power model
  - High level description of the power model.
  - Structure & components
  - Configuration
- Power Model Demo & Reported values
- Power Model options
- LibPeq
- Code structure

Power Model – Options

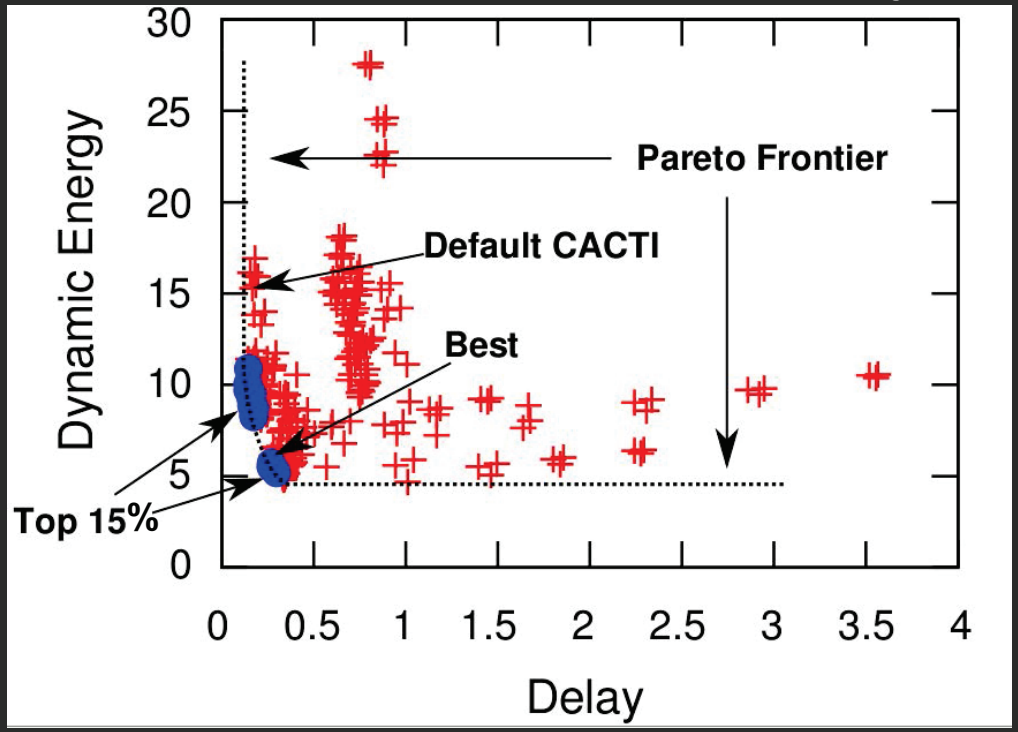
- Turbo Mode
  - Simulate Intel turbo mode.
- DVFS
  - To be covered in the next section
- LibPeq

Outline

- Overview of the ESESC power model
  - High level description of the power model.
  - Structure & components
  - Configuration
- Power Model Demo & Reported values
- Power Model options
- LibPeq
- Code structure

Power Model – LibPeq (alpha)

- Problem : Dependence on CACTI by McPAT

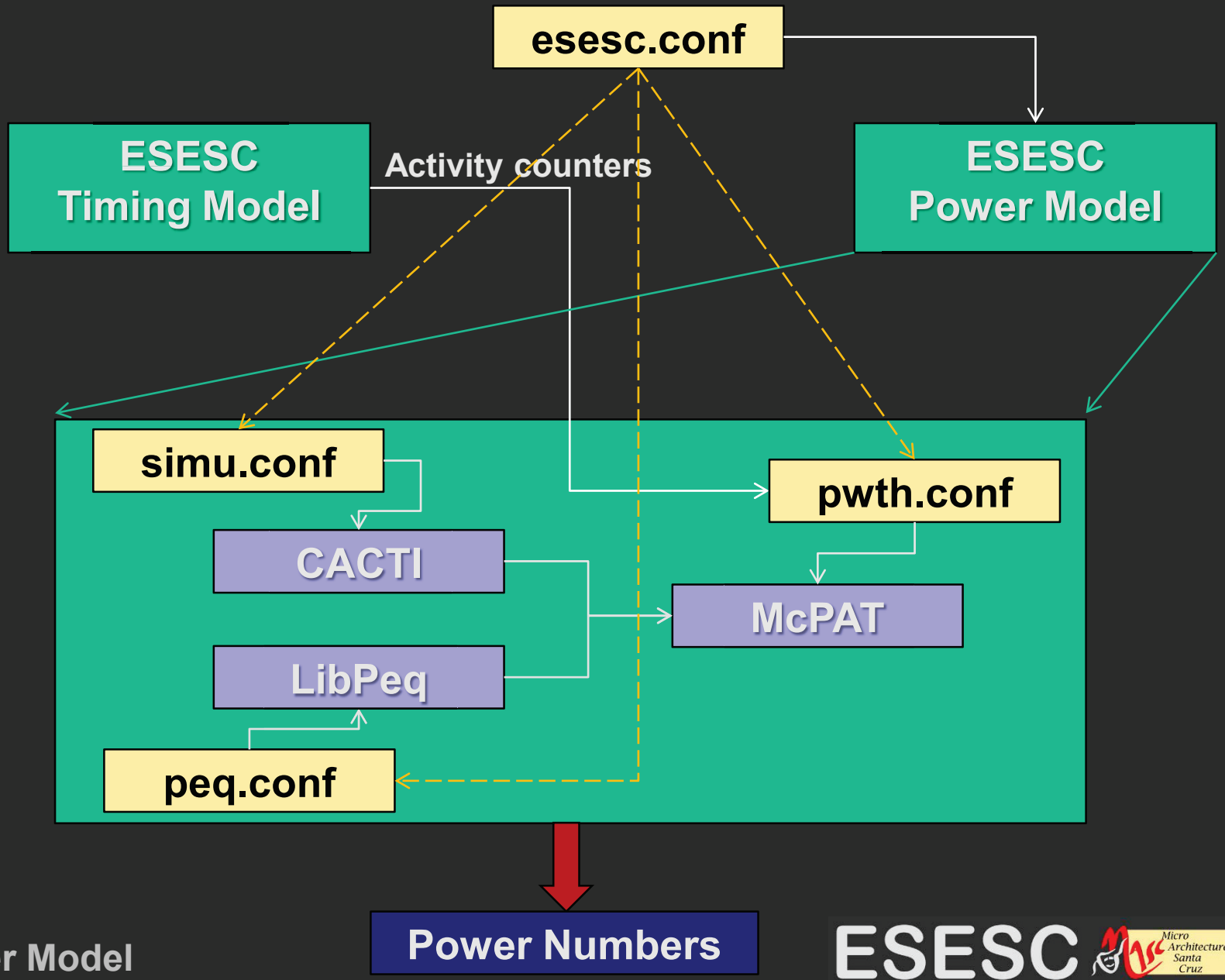


- CACTI → complex power model
- Very slow initialization
- Large Design Space to check
- Slower for complex architectures, multicores

Power Model – LibPeq

- LibPeq [4,5]
  - **Alpha** version
  - Analytical model
  - Developed from statistical analysis of thousands of CACTI simulations covering exhaustive design search space
  - Significantly faster
- Only models SRAM structures
- Does not model leakage

LibPeq - Structure



peq.conf

- Equations defined for array based structures

```
[SRAM_Small1]
dynamic ="exp(-4.982+2.196* ln(tech)+0.4961* ln(ports)-0.00986*
sqrt(size)+0.5464* ln(size)-0.016961* width+0.4027* sqrt(width))* (10^(-9))"


[SRAM_Large1]
dynamic = "exp(-5.446+2.094* ln(tech)+0.886* ln(ports)+0.000458*
sqrt(size)+0.5296* ln(size)-0.011965* width+0.31001* sqrt(width))* (10^(-9))"
```

- Included by esesc.conf
- Should not be modified (unless you know what you are doing!)

LibPeq Speedup

- Running crafty with the default parameters **without** LibPeq

```
time esesc

real    1m0.626s
user    1m25.870s
sys     0m26.930s
```

- Running crafty with the default parameters **with** LibPeq

```
time esesc

real    0m37.680s
user    0m43.160s
sys     0m25.720s
```

Outline

- Overview of the ESESC power model
  - High level description of the power model.
  - Structure & components
  - Configuration
- Power Model Demo & Reported value
- Power Model options
- LibPeq
- Code structure

Code Structure

Files / Directories	Significance
simu/libsampler/Powermodel.*	Main hook to the ESESC power model
pwth/libmcpat	McPAT source code (modified to support ESESC)
pwth/libmcpat/pwth_parser.cpp	Reading pwth.conf, translating ESESC GSTATS to McPAT counters
pwth/libmcpat/processor.cpp	McPAT models for various blocks in a processor
pwth/libmcpat/core.cpp	McPAT models for various blocks in a processor
pwth/libpwrmodel	Wrapper for the power model
pwth/libpeq	Library to parse PEQ

Summary

Important Input Files

- esesc.conf : Enable/Disable Power Model
- simu.conf : Description of architecture
- pwth.conf : Translate GStat counters to McPAT understandable counters. (Don't modify unless you know what you are doing)
- peq.conf: Contains equation for SRAM and caches. CAM equation and leakage equation can be added in future.

Thermal Model
ESESC Tutorial

Speaker: Elnaz Ebrahimi

ESESC

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu

UC SANTA CRUZ

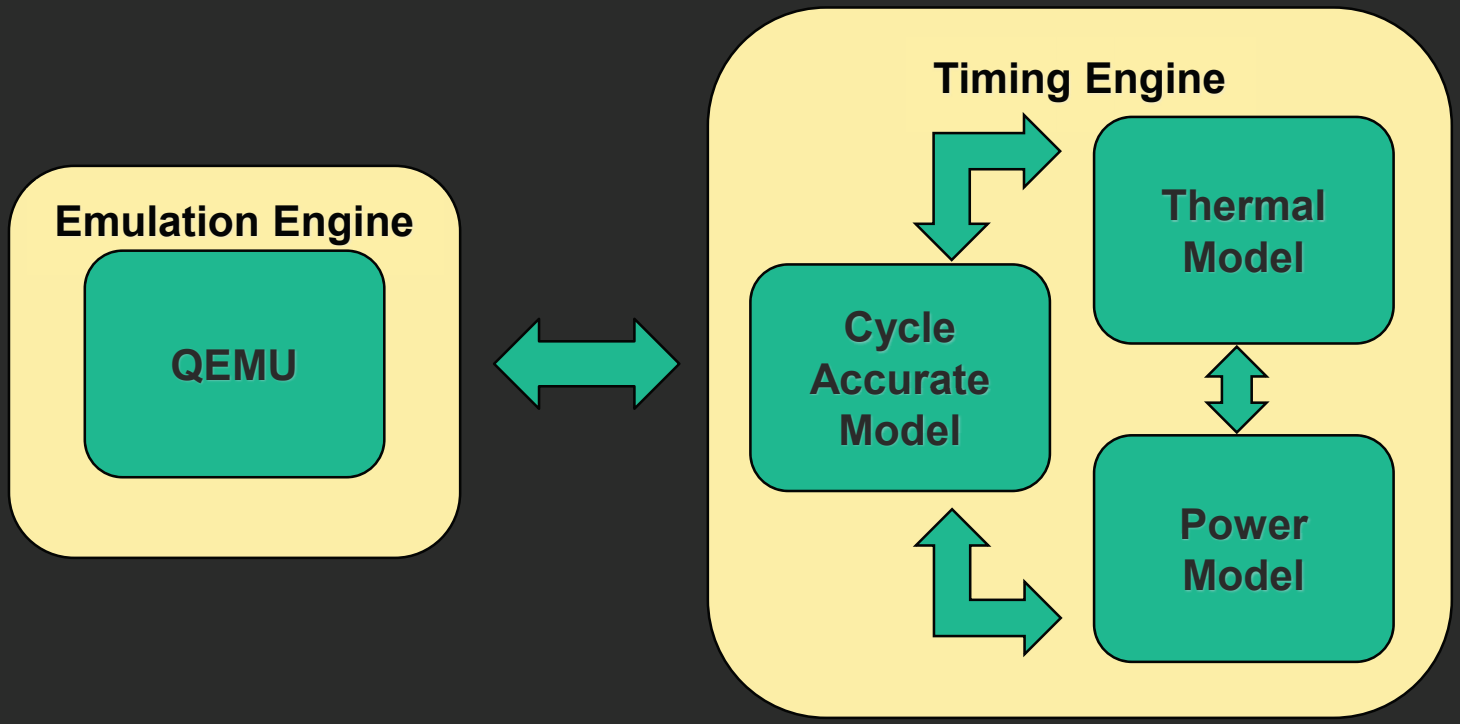
Thermal Model

- You will learn:
  - High level view of ESESC thermal model
  - Run a thermal simulation
  - Automatically create a floorplan
  - Change thermal management policies

Outline

- Thermal Model
- Output Files (Demo 1)
- Setting up the Floorplan (Demo 2)
- DVFS and Thermal Throttling (Demo 3)
- Detailed Package Configurations

ESESC – Thermal Model



SESCTherm

- Thermal model is a modified version of SESCTherm
- Scales leakage according to
  - Temperature
  - Device Properties

- J. N.-Battilana and J. Renau, "*SOI, Interconnect, Package, and Mainboard Thermal Characterization*," in Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2009, pp. 327–330.

SescTherm Main Files

- `SescTherm.cpp`
  - `esesc/pwth/libsesctherm/SescTherm.cpp`

  - Computes the temperature
  - Dumps temperature trace

SescTherm Main Files

- `ThermTrace.cpp`
  - `esesc/pwth/libsesctherm/ThermTrace.cpp`

  - Reads floorplan mapping
  - Reads energy numbers
  - Scales leakage based on temperature

SescTherm Main Files

- `ThermModel.cpp`
  - `esesc/pwth/libsesctherm/ThermModel.cpp`

  - Extracts layer information from `pwth.conf`
  - Partitions the floorplan
  - Creates solution matrices
  - Re-computes material properties

SescTherm Main Files

- `ChipFloorplan.cpp`
  - `esesc/pwth/libsesctherm/ChipFloorplan.cpp`
    - Reads and processes the floorplan based on floorplan information specified in `pwth.conf`

Thermal Modeling Requirements

- Power
- Performance
- Floorplan information and configuration
- Package information
- Thermal management policy

Outline

- Thermal Model
- Output Files (Demo 1)
- Setting up the Floorplan (Demo 2)
- DVFS and Thermal Throttling (Demo 3)
- Detailed Package Configurations

Thermal Model Output Files

- Thermal model related output files in `~/build/release/run`
  - ESESC configurations and statistics
    - `esesc_microdemo.?????`
  - Temperature trace
    - `temp_esesc_microdemo.?????`
  - Total Power
    - `totalpTh_esesc_microdemo.?????`

Thermal Model Output Files

- `esesc_microdemo.?????`
  - Overall chip thermal related statistics
    - Dynamic power
    - Leakage Power
    - Gradient Temperature Across Chip
    - Average Temperature
    - Maximum Temperature
    - etc.
  - Extract thermal statistics using `report.pl`

```
~/projs/esesc/conf/report.pl -last
```

Thermal Model Output Files

- `temp_esesc_microdemo.?????`
  - Columns show temperature (K) vs. time (s)
- `totalpTh_esesc_microdemo.?????`
  - Time (s) and corresponding total power
  - Total power = Dynamic power + leakage power scaled by temperature

Thermal Config File

`pwth.conf`

- Floorplan
- Layers (die, interconnect, cooler)
- Model config (temp and equation solver)
- Cooling Solution (air, oil)
- Package Configuration/Dimension
- Graphical thermal map
- Other layer configurations

Demo 1

- Assume floorplanning and device parameters are set
- Enable power and thermal
- Full thermal run with Crafty benchmark
- Extract thermal statistics
- Explain thermal related output files
- Plot power and temperature

Outline

- Thermal Model
- Output Files (Demo 1)
- Setting up the Floorplan (Demo 2)
- DVFS and Thermal Throttling (Demo 3)
- Detailed Package Configurations

How to Use an Existing Floorplan

- In `~/build/release/run`
  - `esesc.conf`
    - `enablePower = true`
    - `enableTherm = true`

How to Use an Existing Floorplan

- In ~/build/release/run
  - flp.conf
    - floorplan_2C
    - layoutDescr_2C
  - pwth.conf
    - [SescTherm] #section
    - floorplan[0] = 'floorplan_2C'
    - layoutDescr[0] = 'layoutDescr_2C'

Generate Floorplan

- `floorplan.rb`
  - Change power, thermal, refloorplan flags
  - Run esesc to generate block connectivity and power estimation
  - Run `hotfloorplan` to generate floorplan
  - Convert the format for `pwth.conf`
  - Update `pwth.conf` with new floorplan
  - Update `esesc.conf` with floorplan link

How to Generate a New Floorplan

- Change single core to dual core
  - esesc.conf
    - cpuemul[0:1] = 'QEMUSectionCPU'
    - cpusim [0:1] = "$(coreType)"
- In build directory
  - ~/projs/build/release/
- Run
  - make floorplan

How to Generate a New Floorplan

- In run directory
  - `~/projs/build/release/run`
- Run
  - `~/projs/esesc/conf/floorplan.rb BuildDir_Path SrcDir_Path RunDir_Path NameMangle`

- `BuildDir_Path`
  - Path to the ESESC build directory
- `SrcDir_Path`
  - Path to the ESESC source directory
- `RunDir_Path`
  - Path to the configuration files in run directory
- `NameMangle`
  - A string to attach to 'floorplan' and 'layoutDescr' sections

How to Generate a New Floorplan

Example command:

```
~/projs/esesc/conf/floorplan.rb
    ~/projs/build/release/
    ~/projs/esesc/
    ~/projs/build/release/run/
    2C
```

How to Generate a New Floorplan

- New links in `pwth.conf`
  - `floorplan[0]    = 'floorplan2C'`
  - `layoutDescr[0] = 'layoutDescr2C'`

- New layout and floorplan definitions in `flp.conf`
  - `[layoutDescr2C]` …
  - `[floorplan2C]` …

- New floorplan is called `new.flp`

Demo 2

- Change from single core to dual core
- Generate a new floorplan
- Go over the changes in conf files

Outline

- Thermal Model
- Output Files (Demo 1)
- Setting up the Floorplan (Demo 2)
- DVFS and Thermal Throttling (Demo 3)
- Detailed Package Configurations

DVFS Configuration

- esesc.conf
  - enablePower = true
  - enableTherm = true
  - thermTT = 373.15
- pwth.conf
  - enableTurbo = true
  - turboMode = dvfs_t

DVFS Control Code

- Frequency changes based on temperature

~/projs/esesc/simu/libsampler/PowerModel.cpp

```
int PowerModel::updateFreqDVFS_T() {
if (maxT > K(90)) {
    dvfsFreq = 0.7*getFreq();
    …
  } else if (maxT > K(88.5)) {
    dvfsFreq = 0.7*getFreq();
    …
  } else
    …
```

Thermal Throttling Configuration

- esesc.conf
  - thermTT = 373.15
- pwth.conf
  - enableTurbo = false

Temperature Map Graphics

In `pwth.conf`

- `[graphics_config]`
  - Enable thermal map image dump
    - `enableGraphics = true`
  - Set the image resolution
    - `resolution_x  = 1024 #1440x900`
    - `resolution_y  = 768`
  - Link the floorplan layer
    - `grpahics_floorplan_layer = 2`

DEMO 3

- Use floorplan for dual core
- Enable thermal map graphics
- Complete 2 thermal runs with FFT
  - DVFS
  - Thermal Throttling
- Create a short video of thermal maps

Outline

- Thermal Model
- Output Files (Demo 1)
- Setting up the Floorplan (Demo 2)
- DVFS and Thermal Throttling (Demo 3)
- Detailed Package Configurations

Detailed Package Configurations

- Defining chip layers
  - Add or define layers in `pwth.conf`

```
[SescTherm]
layer[0]= 'mainboard0'        #mainboard
layer[1]= 'interconnect0'     #metal
layer[2]= 'die_transistor0'   #transistor
layer[3]= 'bulk_silicon0'     #substrate
layer[4]= 'air_layer0'        #air
```

Detailed Package Configurations

- `[die_transistor0] #power layer`
  - `granularity = 'x'  #(m)`
  - `floorplan = 2  #layer index`
  - `lock_temp = -1`
- `[air_layer0]`
  - `lock_temp = 25+273.15 #ambient T`
  - `floorplan = -1`
- `floorplan = -1`
  - for all layers except `die_transistor0`

Detailed Package Configurations

- Package specific configuration sections
  - Model configuration
    - `Model = 'model_config'`
  - Thermal map image dump
    - `Graphics = 'graphics_config'`
  - Air or oil cooling solution
    - `Cooling = 'air_cooling_config`
  - Chip and package size and dimensions
    - `Chip = 'chip_config'`

Detailed Package Configurations

- `[model_config]`
  - matrix solver
    - `useRK4 = true`
  - cycles per sample
    - `CyclesPerSample = 100000`
  - initial temperature
    - `initialTemp = 35+273.15`
  - ambient temperature
    - `ambientTemp = 35+273.15`

Detailed Package Configurations

- [chip_config]
  - Chip dimensions: based on based on floorplan information (x, y)
    - chip_width
    - chip_height
    - chip_thickness
  - Package size: architectural decision
    - package_height
    - package_width
    - package_thickness

Detailed Package Configurations

- Cooling solutions
  - `[air_cooling_config]`
  - `[oil_cooling_config]`
  - Related code
    - `esesc/pwth/libsesctherm/ChipMaterial.cpp`
- For other `pwth.conf` configurations
  - Compare with default `pwth.conf` settings
  - Check source code

Summary

- High level view of ESESC thermal model
- Run a thermal simulation
- Automatically create a floorplan
- Change thermal management policies

Backup Slides

Turbo Mode

- Frequency changes based on temperature

~/projs/esesc/simu/libsampler/PowerModel.cpp

```
int PowerModel::updateFreqTurbo()
…
// Decide on the actual turbo frequency based on temperature
  if (maxT > K(100)) {
    turboFreq = getFreq();
    state = 4;
  } else if (maxT > K(90)) {
    turboFreq = maxF - 3*(maxF - getFreq())/4;
    state = 3;
  } else if (maxT > K(80)) {
    turboFreq = maxF - 2*(maxF - getFreq())/4;
    state = 2;
  } else ….
```