5MD00

Assignment Introduction

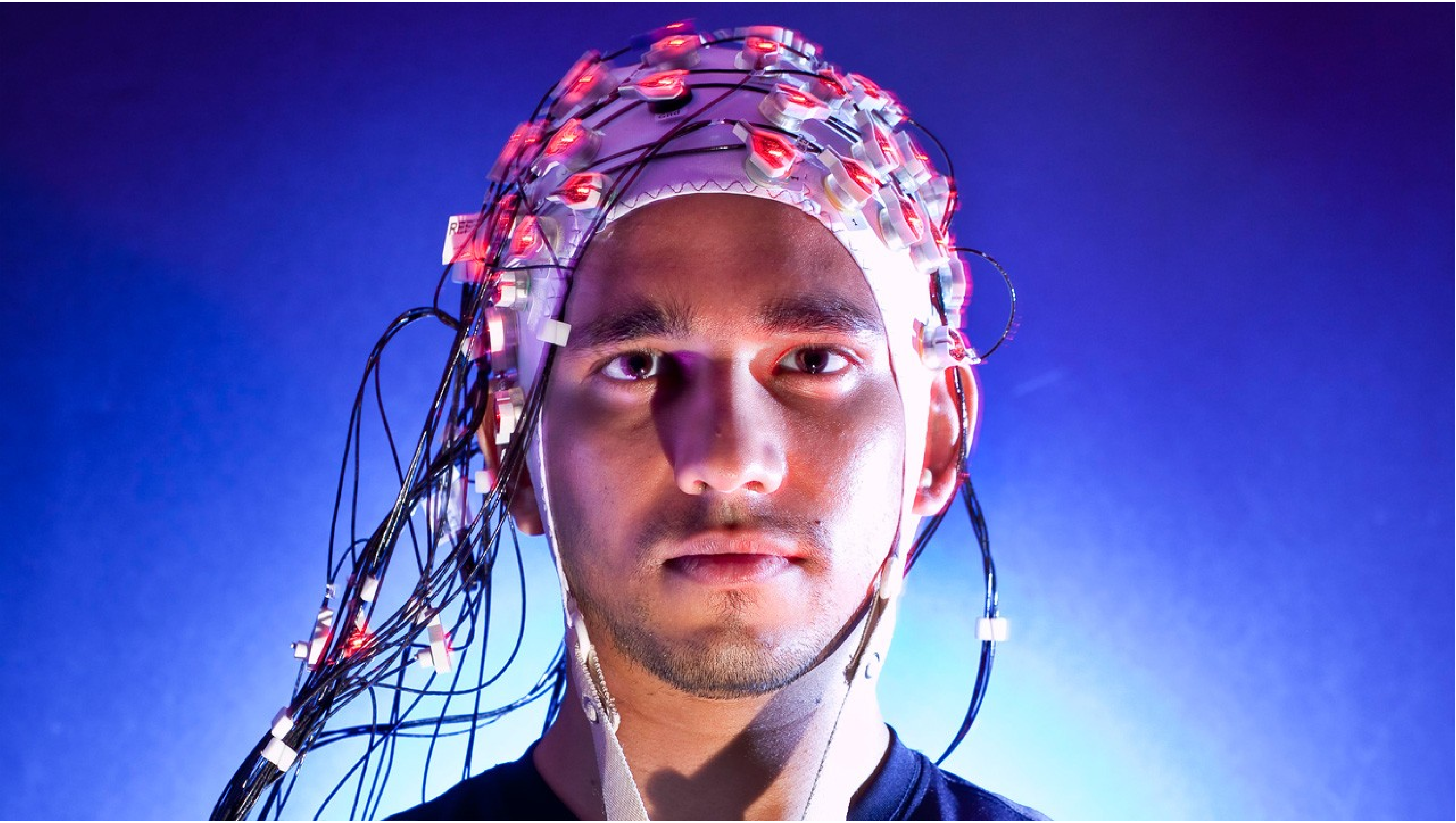
Luc Waeijen

16-12-2014

Contents

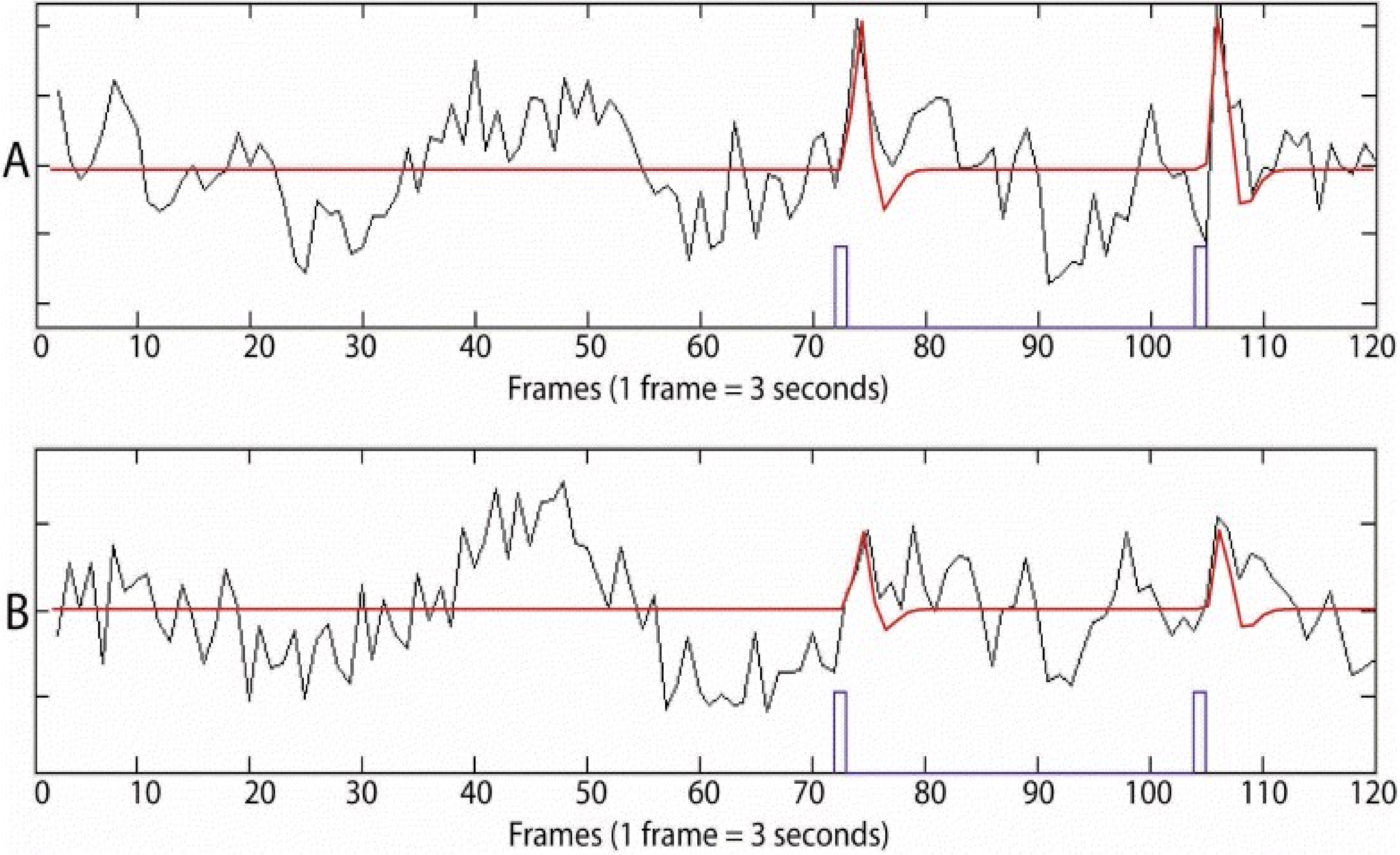
- EEG application
  - Background on EEG
  - Early Seizure Detection Algorithm
  - Implementation Details
- Super Scalar Assignment
  - Description
  - Tooling (simple scalar & wattch)
- Multi Core Assignment
  - Description
  - Tooling (sniper sim, McPat & OpenMP)

Electroencephalography (EEG)



Lets plug some wires into a brain!!

EEG Signals


A
0
10
20
30
40
50
60
70
80
90
100
110
120
Frames (1 frame = 3 seconds)
B
0
10
20
30
40
50
60
70
80
90
100
110
120
Frames (1 frame = 3 seconds)

Uses

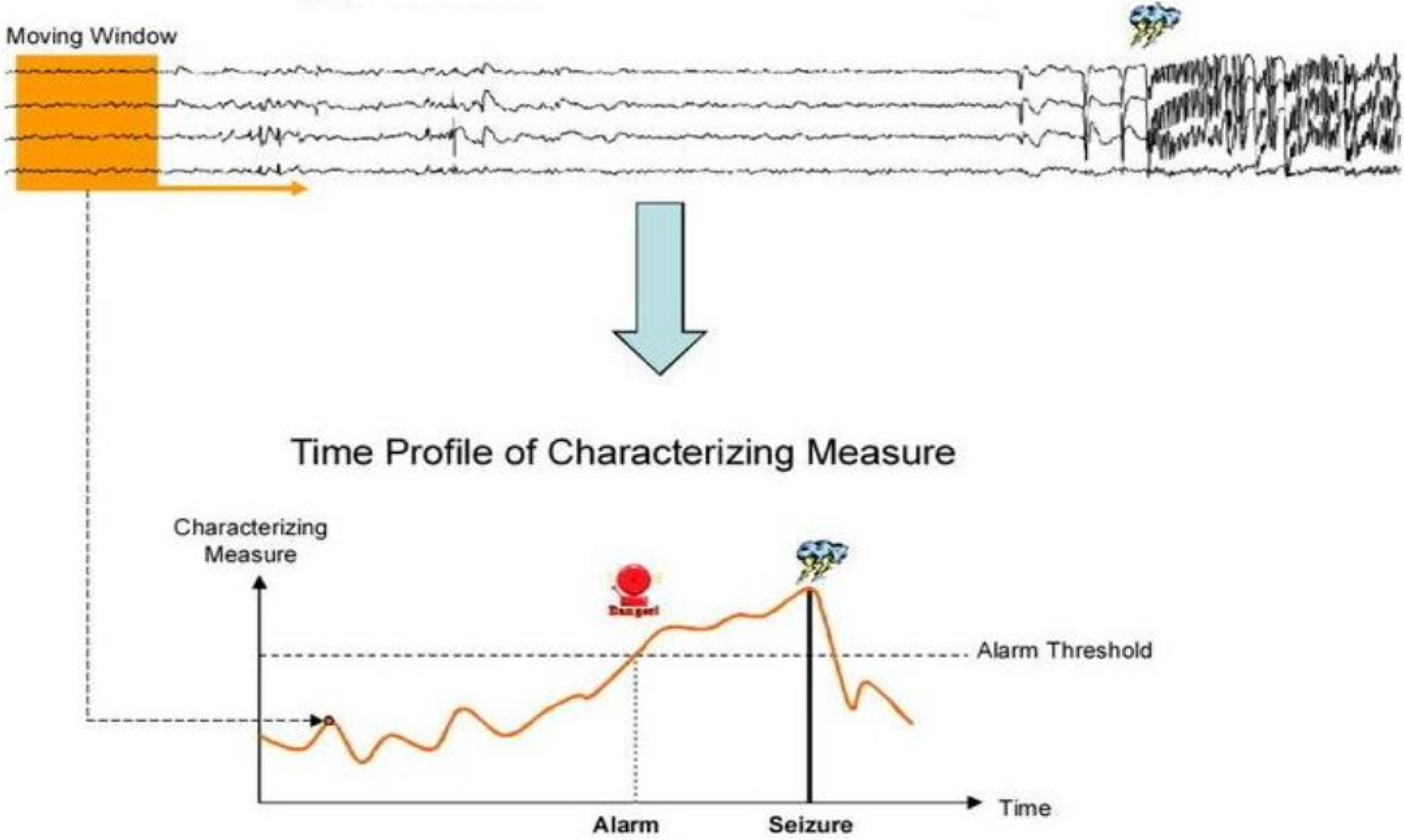
- Research
  - Determine functionality of brain
  - Observe physical reaction to various stimuli
- Clinical
  - Monitoring during operations
  - Diagnosis and classification of epilepsy
  - Prognosticate coma patients
  - Test for brain damage
- Brain computer interface
  - Paralyzed people
  - Gaming

Uses

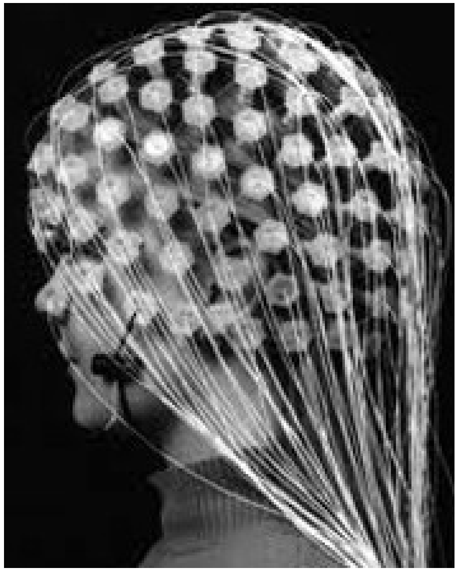
- Research
  - Determine functionality of brain
  - Observe physical reaction to various stimuli
- Clinical
  - Monitoring during operations
  - Diagnosis and classification of epilepsy
  - Prognosticate com
  - Test for brain dar
- Brain computer ir
  - Paralyzed people
  - Gaming

Can even predict seizures before they occur!

Early Seizure Detection



EEG devices are large and make you look like something from the matrix!



EEG devices are large and make you look like something from the matrix!

Epilepsy patients cannot walk the streets like this!

EEG devices are large and make you look like something from the matrix!

Social Stigma

But we are working on it!

Mobile EEG

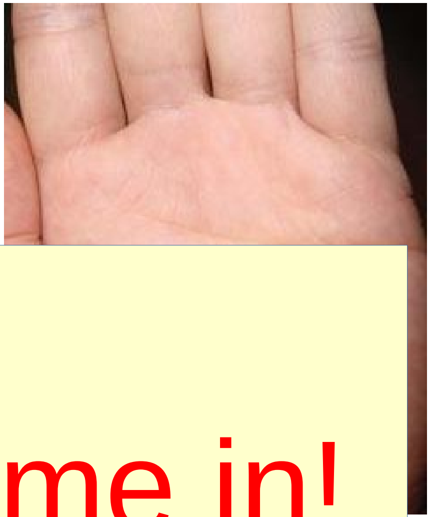
We are developing mobile EEG devices that can be implanted under the skin of the head.

Result is practically invisible sensors, but with severe energy constraints!

Early seizure detection has to be computed on nodes, cannot afford to transmit all data to off site processing unit

Mobile EEG

We are developing mobile EEG devices that can be implanted under

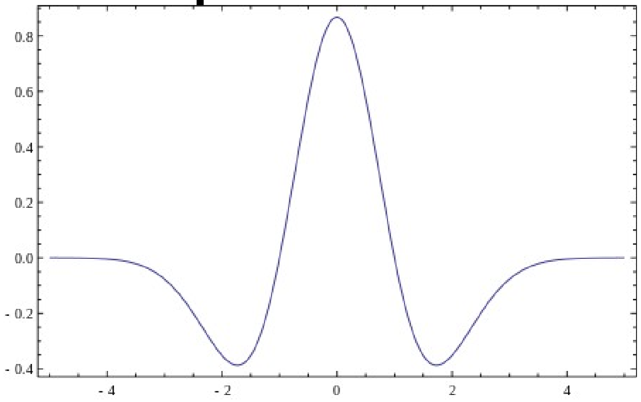


This is where you guys come in!

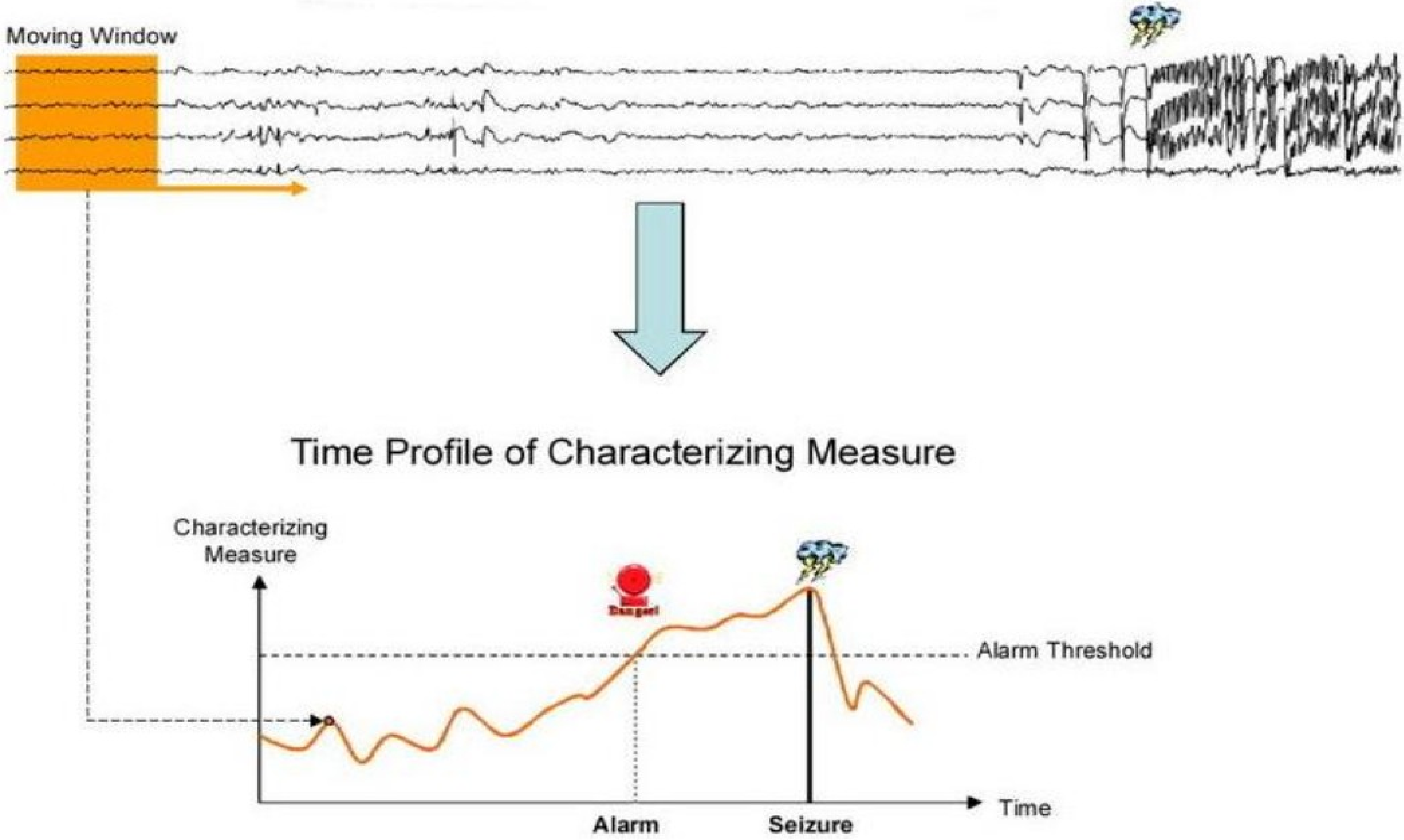
Early seizure detection has to be computed on nodes, cannot afford to transmit all data to off site processing unit

The Algorithm

- Analysis of EEG time signal using discrete wavelet transform over a time window (convolution of time series with a wavelet function)
- Mathematically a wavelet will correlate with the signal if the unknown signal contains information of similar frequency
- Conceptually similar to a Fourier transform (actually the Fourier transform is a special case of the wavelet transform)



Sliding Window

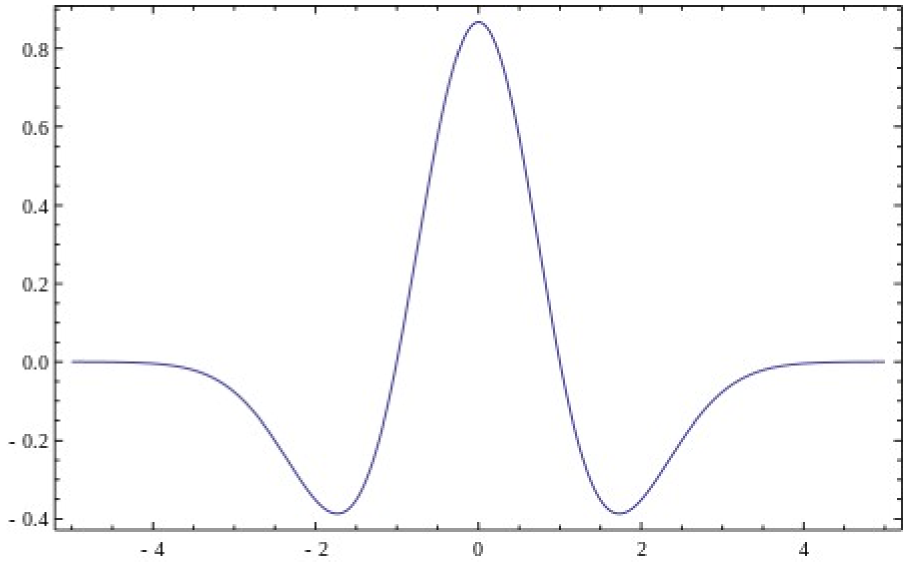


Time Profile of Characterizing Measure

Mother Wavelet

In this algorithm we use the Mexican hat mother wavelet:

$$\psi(t) = \frac{1}{\sqrt{2\pi}}(1 - t^2)\,e^{\frac{-t^2}{2}}$$



Daughter Wavelets and Coefficients

- Mother wavelet is scaled (s) and shifted (tau) to generate daughter wavelets

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right)$$

- Can now calculate wavelet coefficients for different s and tau by convolution with daughter wavelet

$$W(\tau, s) = \int x(t)\frac{\psi\left(\frac{t-\tau}{s}\right)}{\sqrt{s}}\mathrm{d}t.$$

Normalizing and selecting Max

- Normalized Wavelet Energy

$$\tilde{W}(\tau, s) = \sqrt{\frac{W^2(\tau, s)}{\sigma^2}}$$

- Select index of maximum, which is used as indicator value for a given time window (that particular scale and shift dominated the window)

$$\zeta = \operatorname{argmax}_s \left(\sum_{\tau} \tilde{W}(\tau, s) \right)$$

Implementation Trick

- The convolution theorem states that under suitable conditions the Fourier transform of a convolution is the pointwise product of Fourier transforms

  (http://en.wikipedia.org/wiki/Convolution_theorem)

$$f * g = \mathcal{F}^{-1}\big\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\big\}$$

- Fast Fourier Transform can be implemented pretty efficiently, let's use that instead of convolution in time domain!

Discrete Wavelet Transform using the Fast Fourier Transform (FFT)

- Fourier-transformation input signal y(t) with FFT
- Generate Daughter Wavelet for chosen scaling factor s and FFT this wavelet
- Multiply FFT'd input and daughter wavelet (Multiplication in frequency domain is convolution in time domain!)
- Inverse FFT (iFFT) gives Wavelet Coefficients for different shifts (tau) and chosen scaling s
- Back to step 2, until all discrete scaling values s are processed

Input data

- EEG signal sampled at 200 Hz
- Examine windows of 2 seconds
- Shift window with steps of 1 second
- Wavelet transform and measurements taken from:

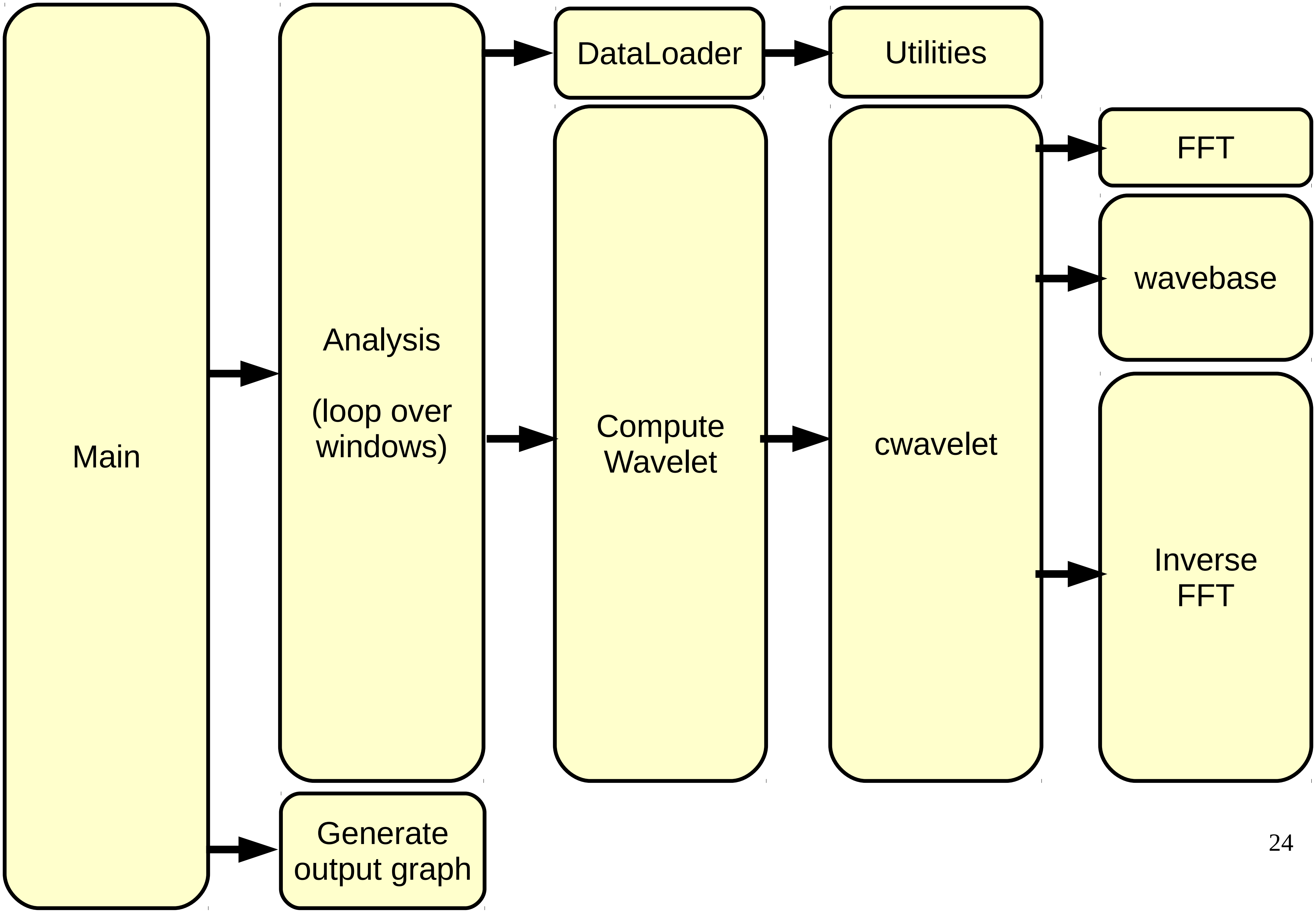
  Non-parametric early seizure detection in an animal model of temporal lobe epilepsy - Sachin S. Talathi et al.

Electrically induced seizures in rats

Implementation

- You are given pure C-code
- Ported from MATLAB code provided with *Sachin S. Talathi* paper
- Function and variable names are not always very clear (taken directly from matlab code)
- Code is functional, but not at all optimized!

Call Graph



Main

Analysis

(loop over windows)

Generate output graph

DataLoader

Utilities

Compute Wavelet

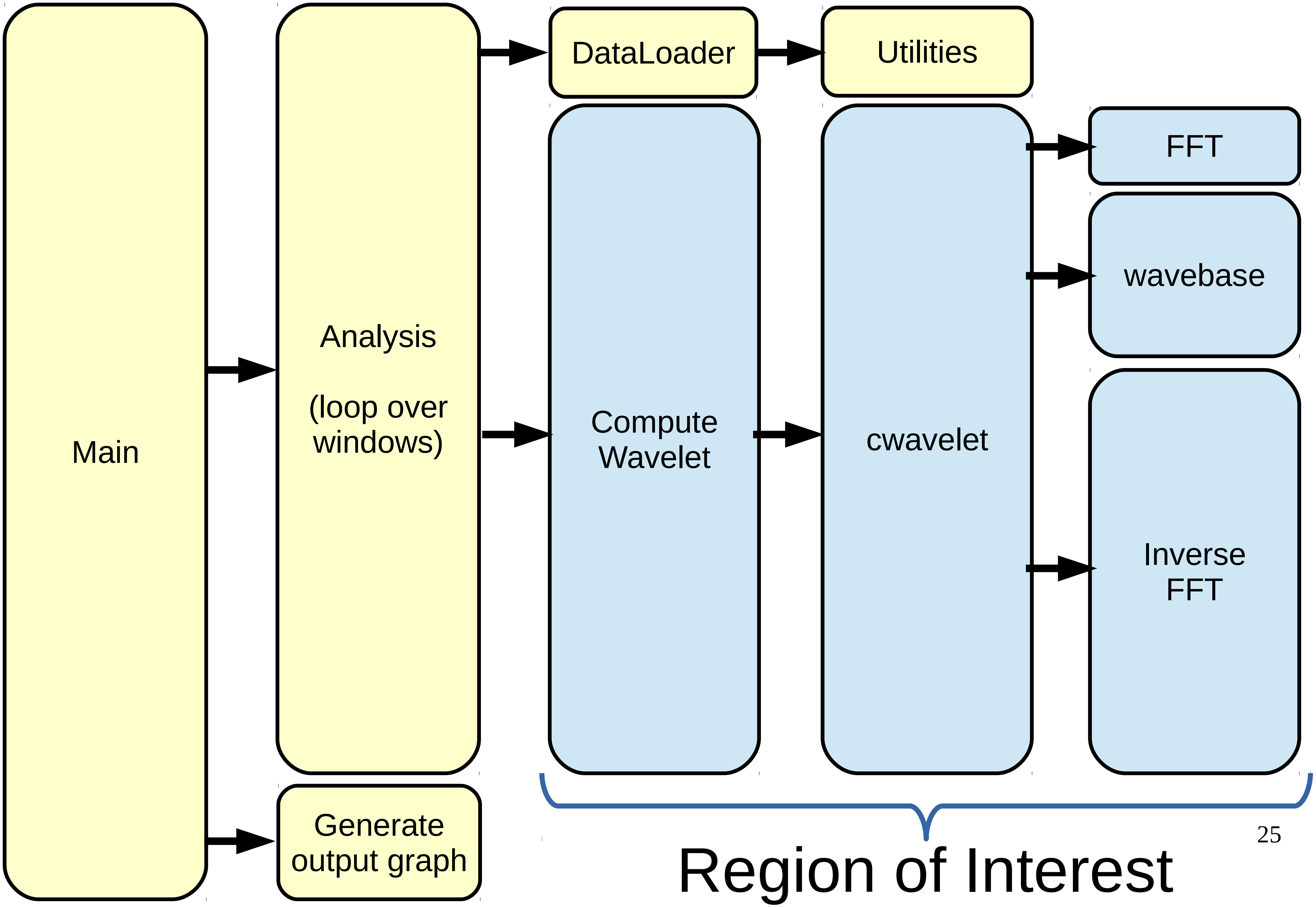
cwavelet

FFT

wavebase

Inverse FFT

Call Graph


Main
Analysis

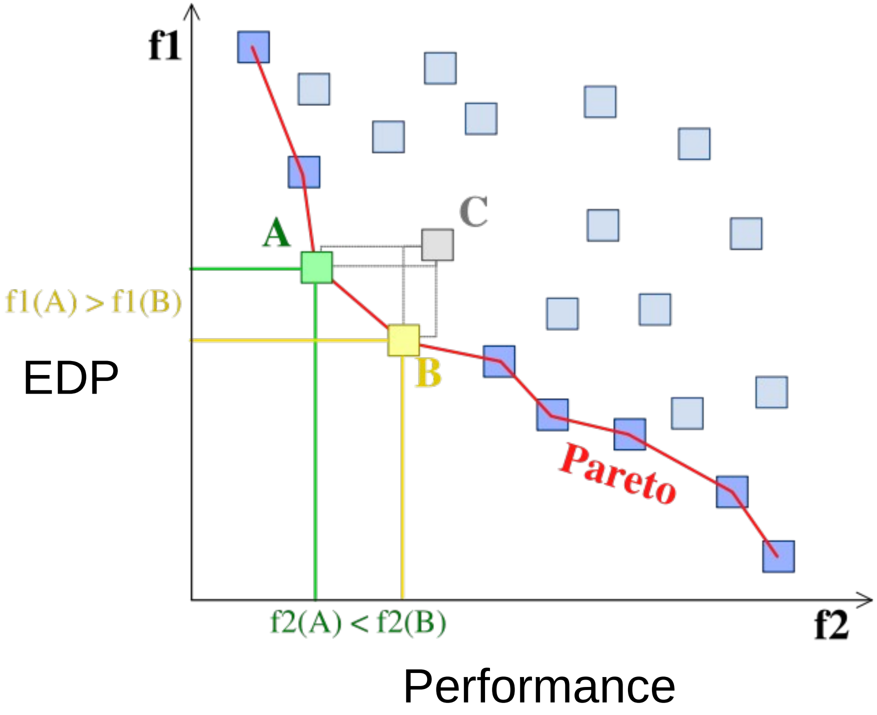
(loop over windows)
Generate output graph
DataLoader
Utilities
Compute Wavelet
cwavelet
FFT
wavebase
Inverse FFT
Region of Interest

Questions about the application?

Single Core Assignment

Single Core Assignment

- Find the most suited super scalar architectures for the early seizure detection algorithm in terms of performance (CPI) and energy-delay-product (CPI*Energy/Cycle)*CPI.

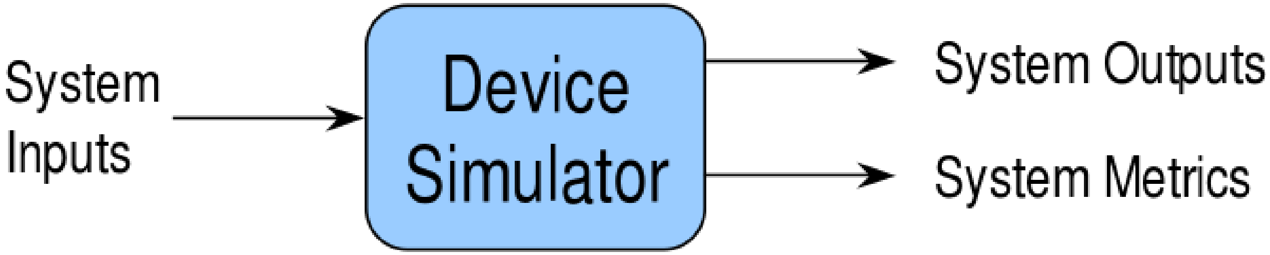


Tools of the Trade

- Simple Scalar
  - Super scalar simulator
  - Designed for fast DSE at architecture level
  - Used in many published papers
- Wattch
  - Architecture level power estimation
  - Complementary to simple scalar

A Computer Architecture Simulator Primer

- What is an architectural simulator?
  - Tool that reproduces the behavior of a computing device



System Inputs

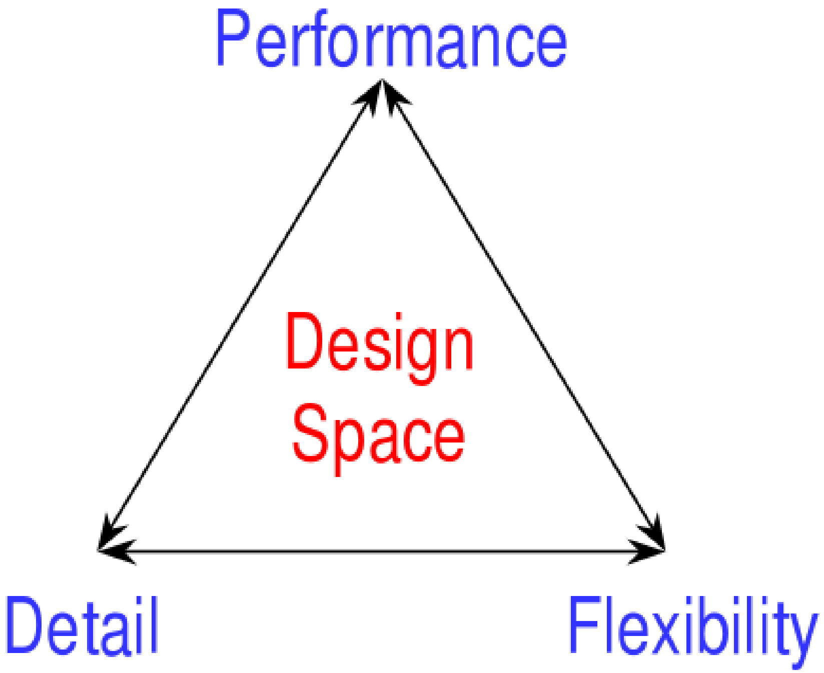
Device Simulator

System Outputs

System Metrics

- Why use a simulator?
  - Leverage faster, more flexible S/W development cycle
    - Permits more design space exploration
    - Facilitates validation before H/W becomes available
    - Level of abstraction can be throttled to design task
    - Possible to increase/improve system instrumentation

The Zen of Hardware Model Design



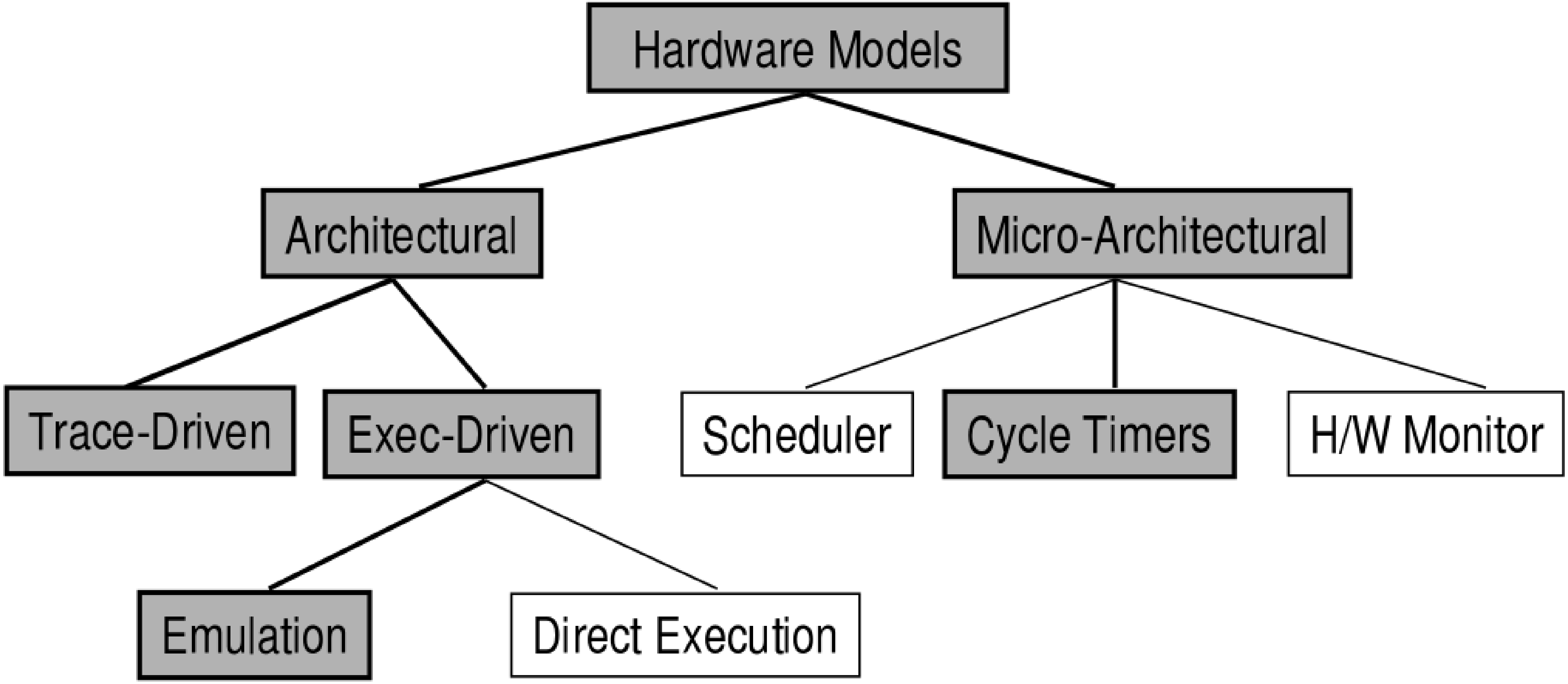
Performance: speeds design cycle

Flexibility: maximizes design scope

Detail: minimizes risk

- Infrastructure goals will drive which aspects are optimized
- SimpleScalar favors performance and flexibility

A Taxonomy of Hardware Modeling Tools

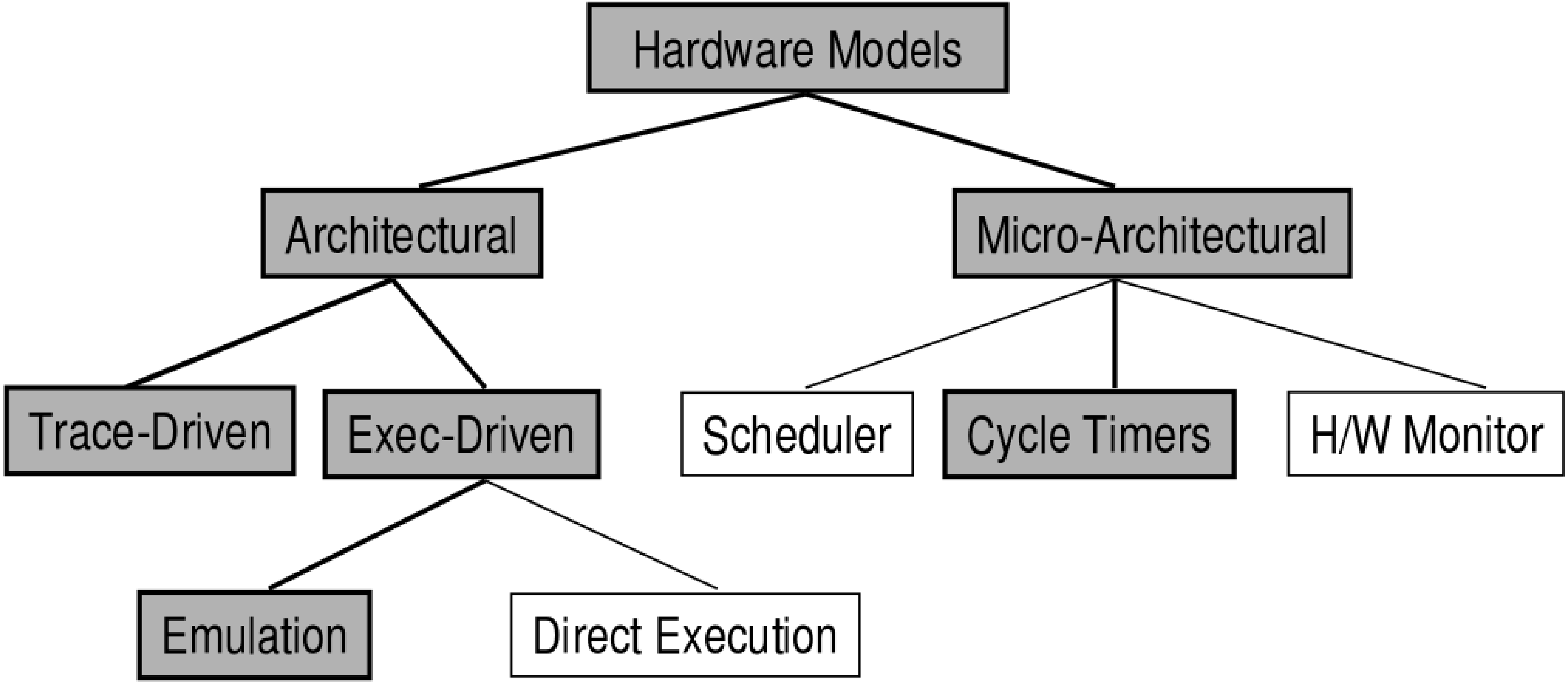


- Shaded tools are included in the SimpleScalar tool set

Functional vs. performance simulators

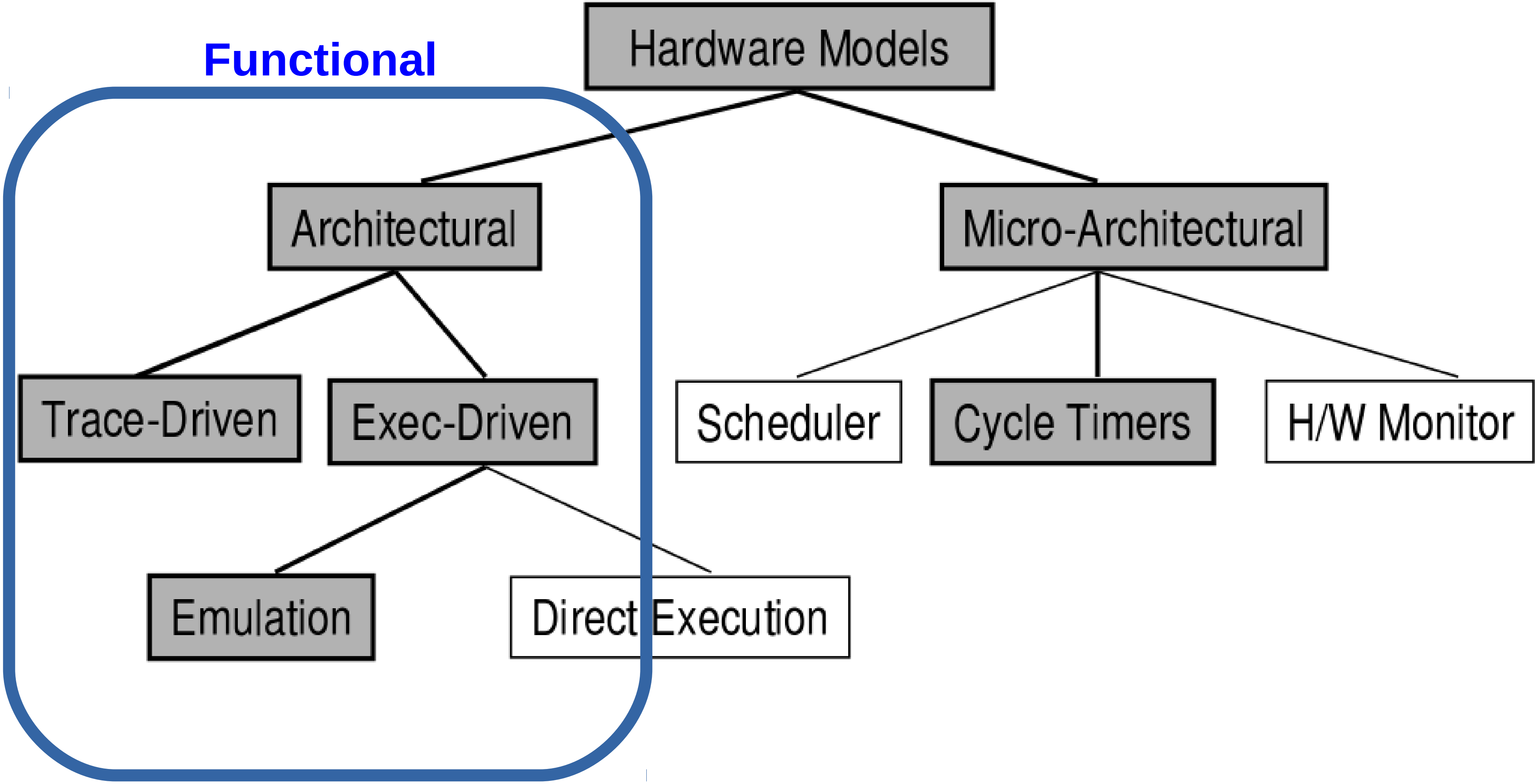
- Functional simulators implement the architecture
  - Perform the actual execution
  - Implement what programmers see
- Performance (or timing) simulators implement the microarch.
  - Model system resources/internals
  - Measure time
  - Implement what programmers do not see

A Taxonomy of Hardware Modeling Tools



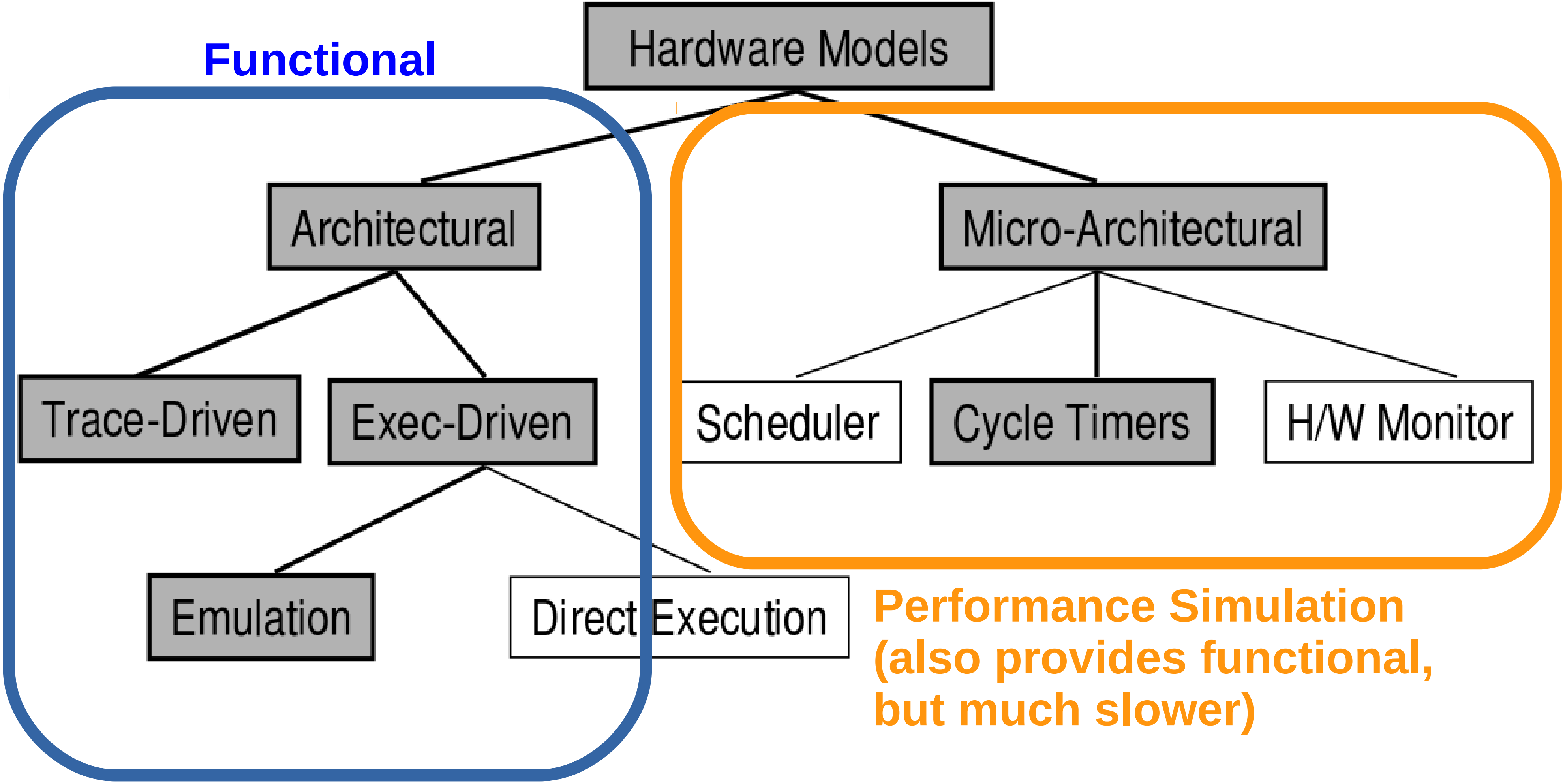
- Shaded tools are included in the SimpleScalar tool set

A Taxonomy of Hardware Modeling Tools



- Shaded tools are included in the SimpleScalar tool set

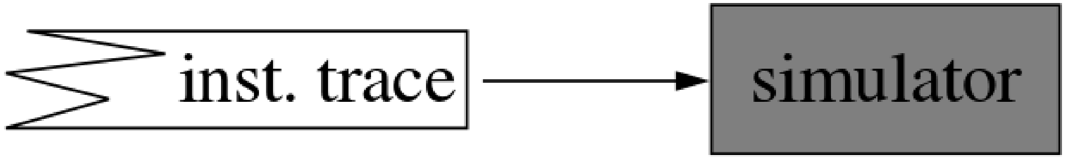
A Taxonomy of Hardware Modeling Tools



- Shaded tools are included in the SimpleScalar tool set

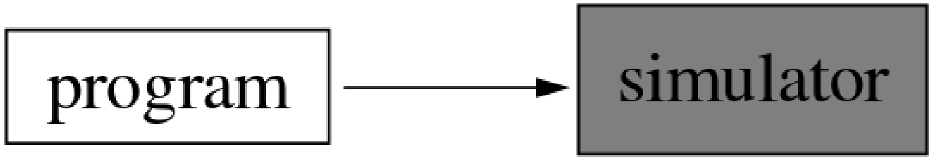
Execution- vs. Trace-driven Simulation

- trace-based simulation:



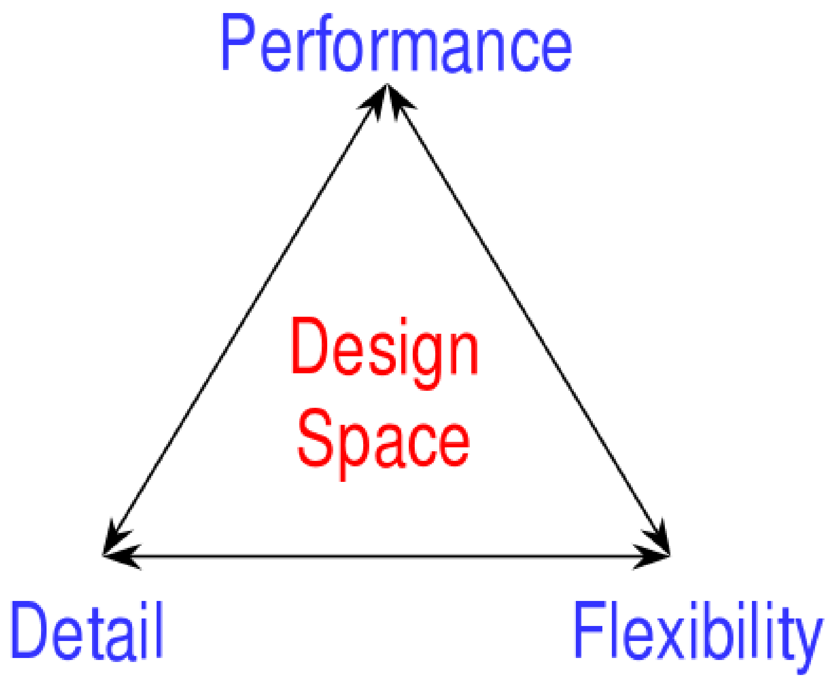
  - reads a “trace” of insts saved from previous execution
  - easiest to implement, no functional component needed
  - no feedback into trace (e.g. mis-speculation)

- execution-driven simulation:



  - simulator “runs” the program, generating stream dynamically
  - more difficult to implement, many advantages
  - direct execution: instrumented program runs on host

The Zen of Hardware Model Design



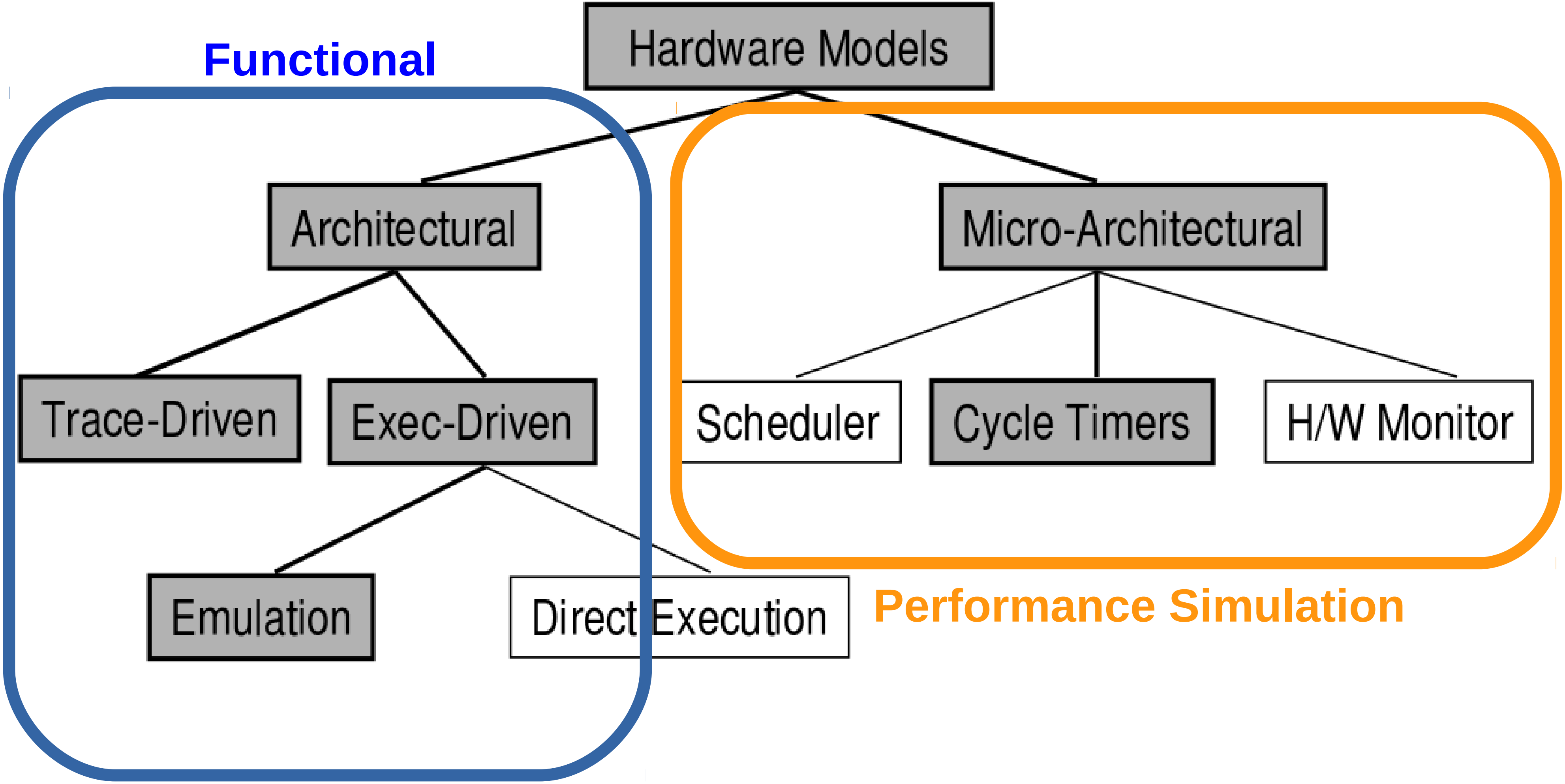
Performance: speeds design cycle

Flexibility: maximizes design scope

Detail: minimizes risk

- Infrastructure goals will drive which aspects are optimized
- SimpleScalar favors performance and flexibility

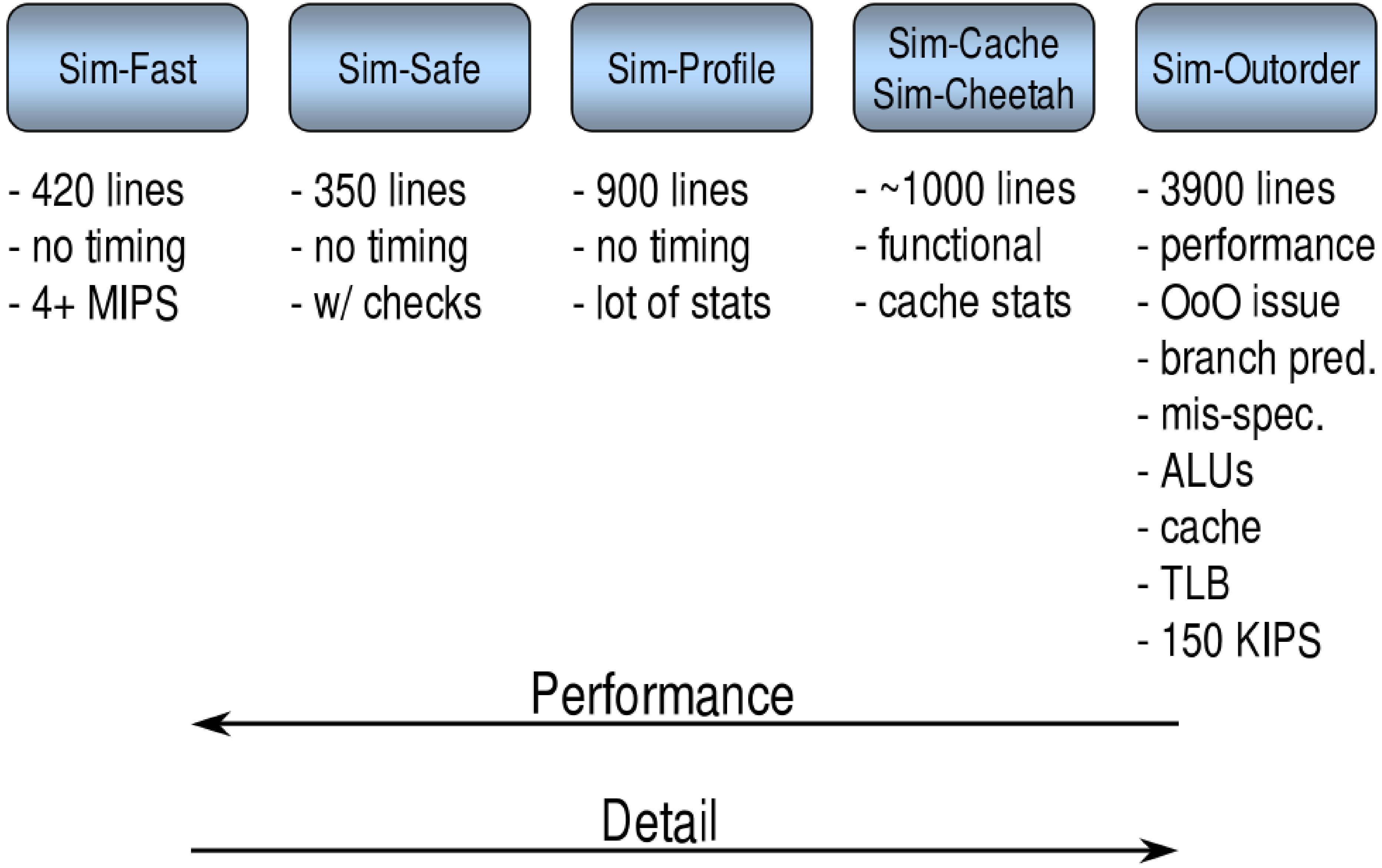
A Taxonomy of Hardware Modeling Tools



- Shaded tools are included in the SimpleScalar tool set

Fast, but low details **Slow, but high details**

Standard Models



Sim-Fast

- 420 lines
- no timing
- 4+ MIPS

Sim-Safe

- 350 lines
- no timing
- w/ checks

Sim-Profile

- 900 lines
- no timing
- lot of stats

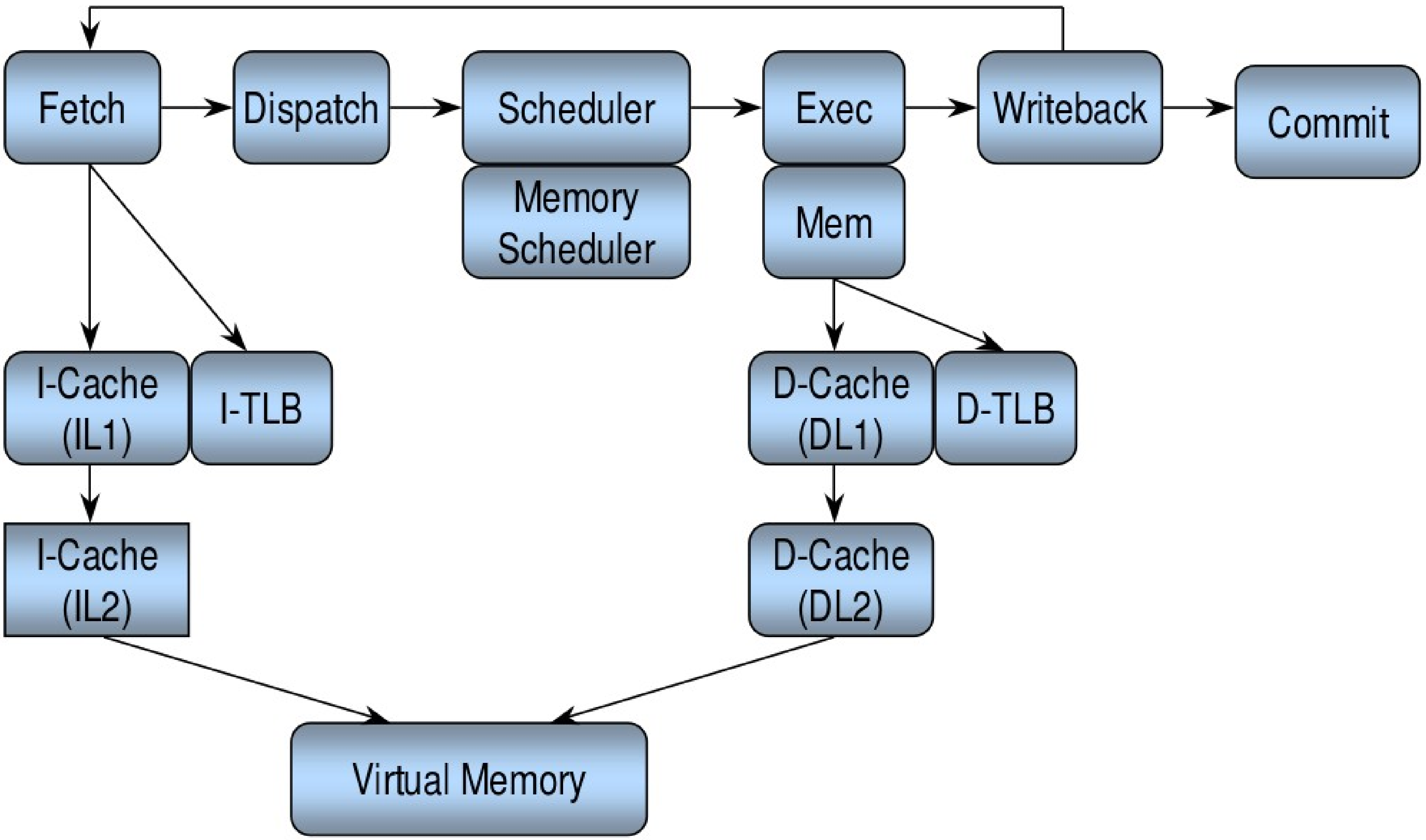
Sim-Cache
Sim-Cheetah

- ~1000 lines
- functional
- cache stats

Sim-Outorder

- 3900 lines
- performance
- OoO issue
- branch pred.
- mis-spec.
- ALUs
- cache
- TLB
- 150 KIPS

Out-of-Order Issue Simulator



How to use?

- To get started, follow the steps on the website:

  http://www.es.ele.tue.nl/~mwijtvliet/5MD00_SC/

- Once you can run the example, start exploring the design space!
- You can configure the processor in many ways, take a look at the tunable parameters at:

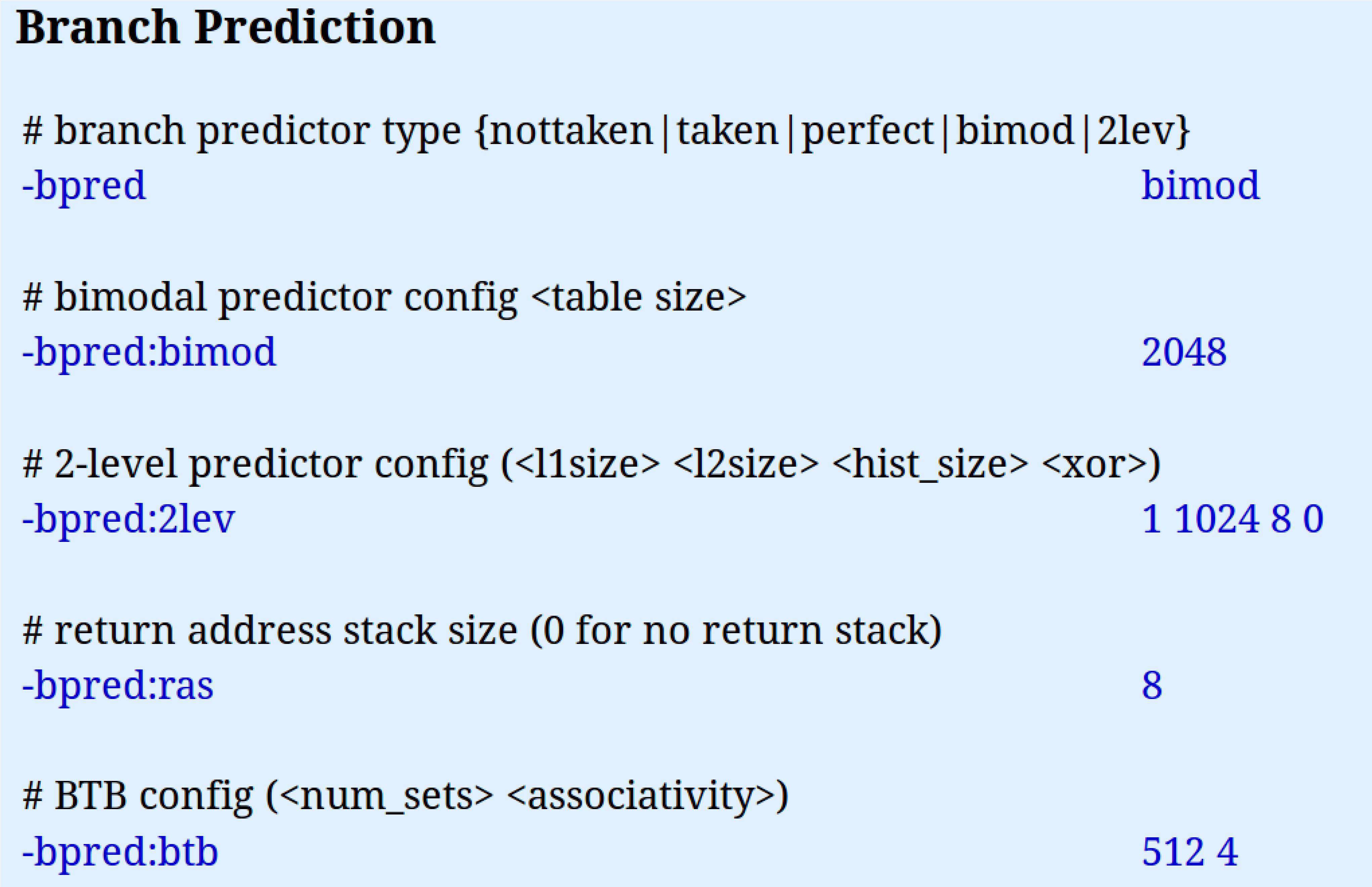
  http://www.es.ele.tue.nl/~mwijtvliet/5MD00_SC/?page=parameters

  https://courses.cs.washington.edu/courses/cse471/06sp/hw/simpleScalar3.0guide.pdf

Make sure you understand all the parameters before tuning

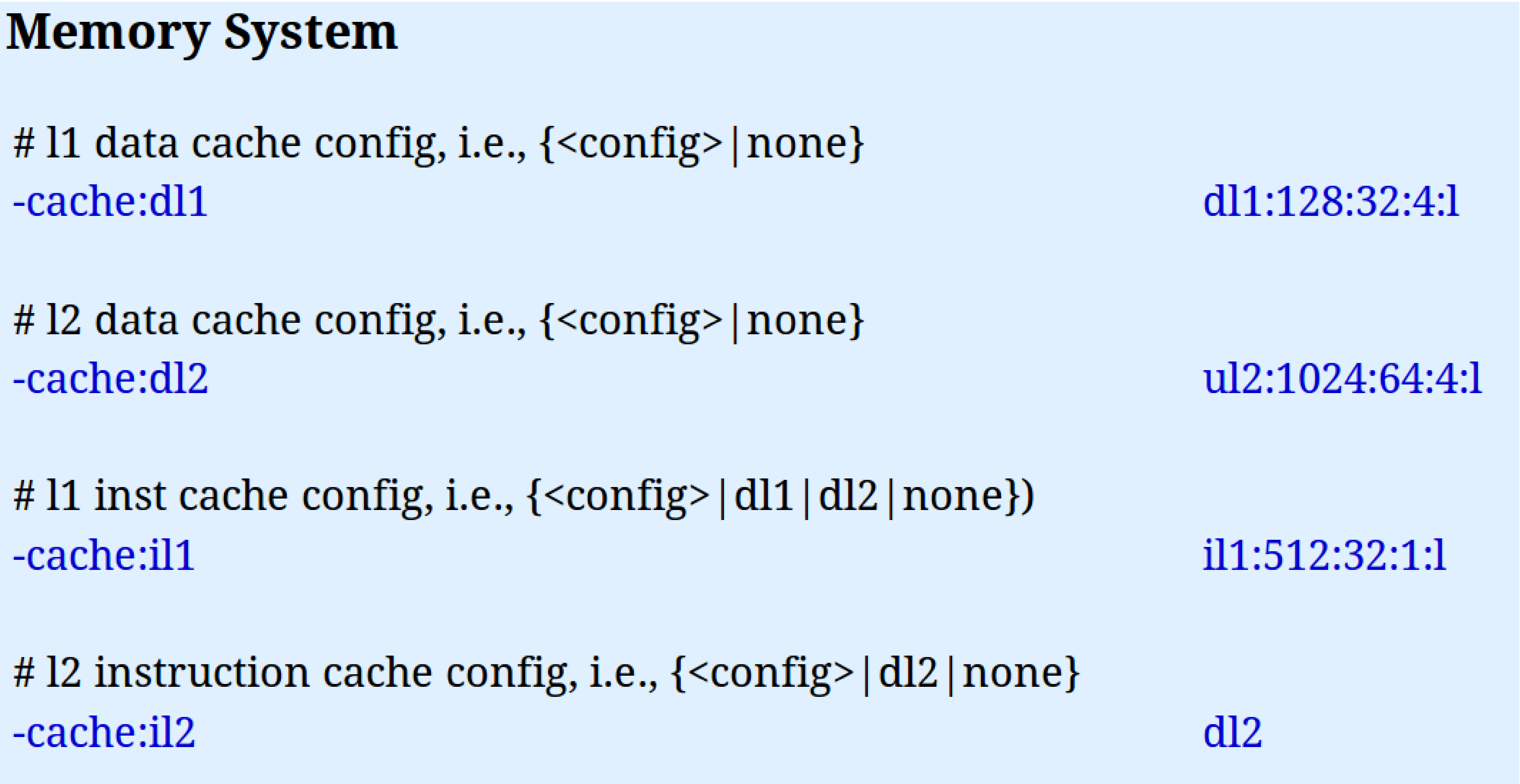
- If there is a parameter you don't understand, check in the cheat-sheet what it stands for. If you do not know the technique/term, google it!
- There is no point in tuning parameters if you do not know what they do.

Tunable Parameters



Branch Prediction

```
# branch predictor type {nottaken|taken|perfect|bimod|2lev}
-bpred                                                  bimod

# bimodal predictor config <table size>
-bpred:bimod                                            2048

# 2-level predictor config (<l1size> <l2size> <hist_size> <xor>)
-bpred:2lev                                             1 1024 8 0

# return address stack size (0 for no return stack)
-bpred:ras                                              8

# BTB config (<num_sets> <associativity>)
-bpred:btb                                              512 4
```



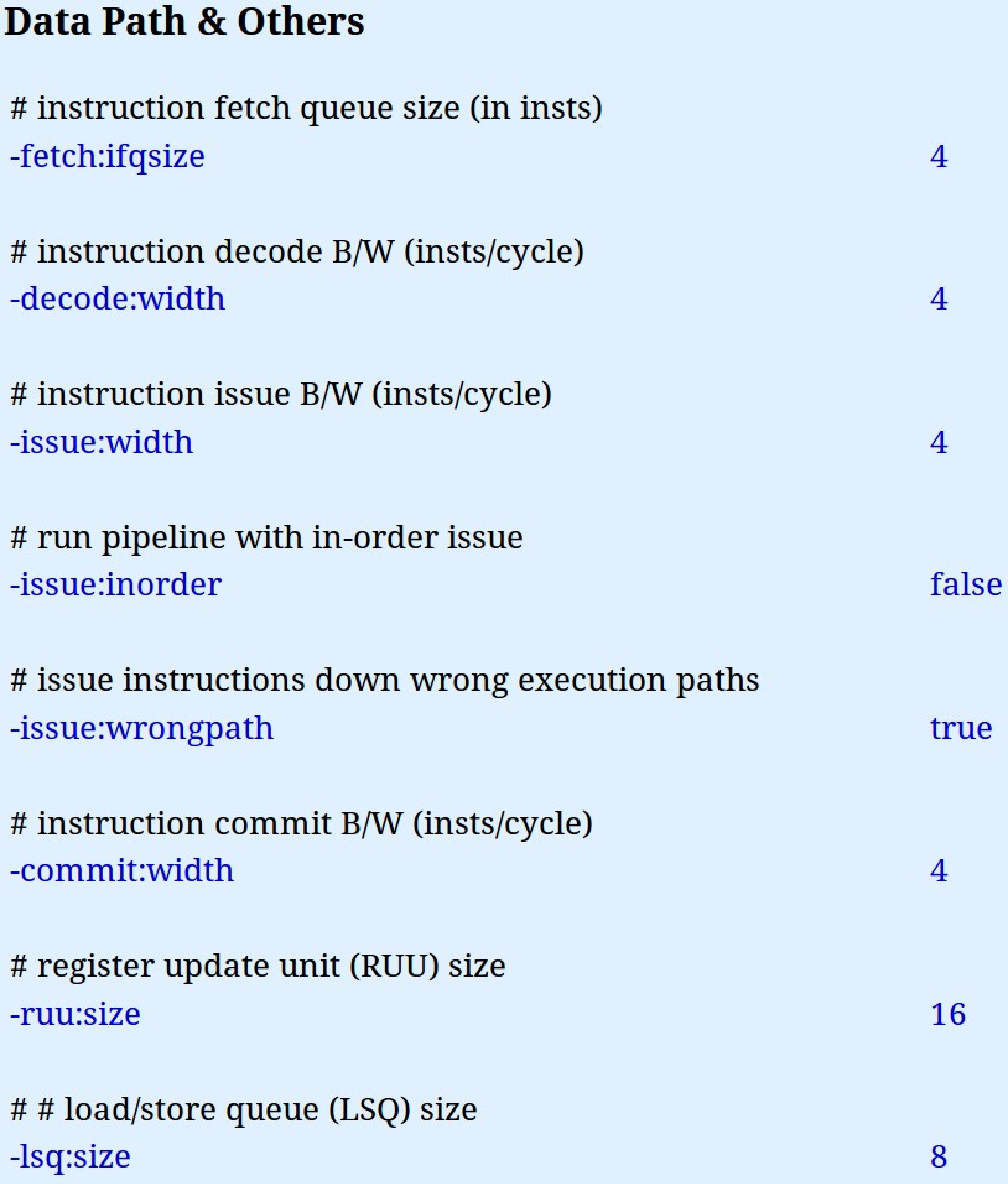
Memory System

```
# l1 data cache config, i.e., {<config>|none}
-cache:dl1                                                    dl1:128:32:4:l

# l2 data cache config, i.e., {<config>|none}
-cache:dl2                                                    ul2:1024:64:4:l

# l1 inst cache config, i.e., {<config>|dl1|dl2|none})
-cache:il1                                                    il1:512:32:1:l

# l2 instruction cache config, i.e., {<config>|dl2|none}
-cache:il2                                                    dl2
```

```
<name>:<nsets>:<bsize>:<assoc>:<repl>
        <name>    name of the cache being defined
        <nsets>   number of sets in the cache
        <bsize>   block size of the cache
        <assoc>   associativity of the cache
        <repl>    block replacement strategy, 'l'-LRU, 'f'-FIFO, 'r'-random
```

Function Units

```
# total number of integer ALUs available
-res:ialu                                                   4

# total number of integer multiplier/dividers available
-res:imult                                                  1

# total number of floating point ALUs available
-res:fpalu                                                  4

# total number of floating point multiplier/dividers available
-res:fpmult                                                 1
```



Data Path & Others

```
# instruction fetch queue size (in insts)
-fetch:ifqsize                                            4

# instruction decode B/W (insts/cycle)
-decode:width                                             4

# instruction issue B/W (insts/cycle)
-issue:width                                              4

# run pipeline with in-order issue
-issue:inorder                                            false

# issue instructions down wrong execution paths
-issue:wrongpath                                          true

# instruction commit B/W (insts/cycle)
-commit:width                                             4

# register update unit (RUU) size
-ruu:size                                                 16

# # load/store queue (LSQ) size
-lsq:size                                                 8
```

Approach

- In your report we want to see why you chose certain configurations.
  - Observe the performance
  - Identify bottlenecks as good as possible
  - Try to improve the bottlenecks and document in your report:
    - What you observed
    - What you tried and why you thought it would help
    - The effect of the tuning (did it work? If not, why?)

Approach

- In your report we want to see why you chose certain configurations.
  - Observe the performance
  - Identify bottlenecks as good
  - Try to improve the bottlene your report:
    - What you observed
    - What you tried and why you thought it would help
    - The effect of the tuning (did it work? If not, why?)

For example sim-profile

Sim-Profile: Program Profiling Simulator

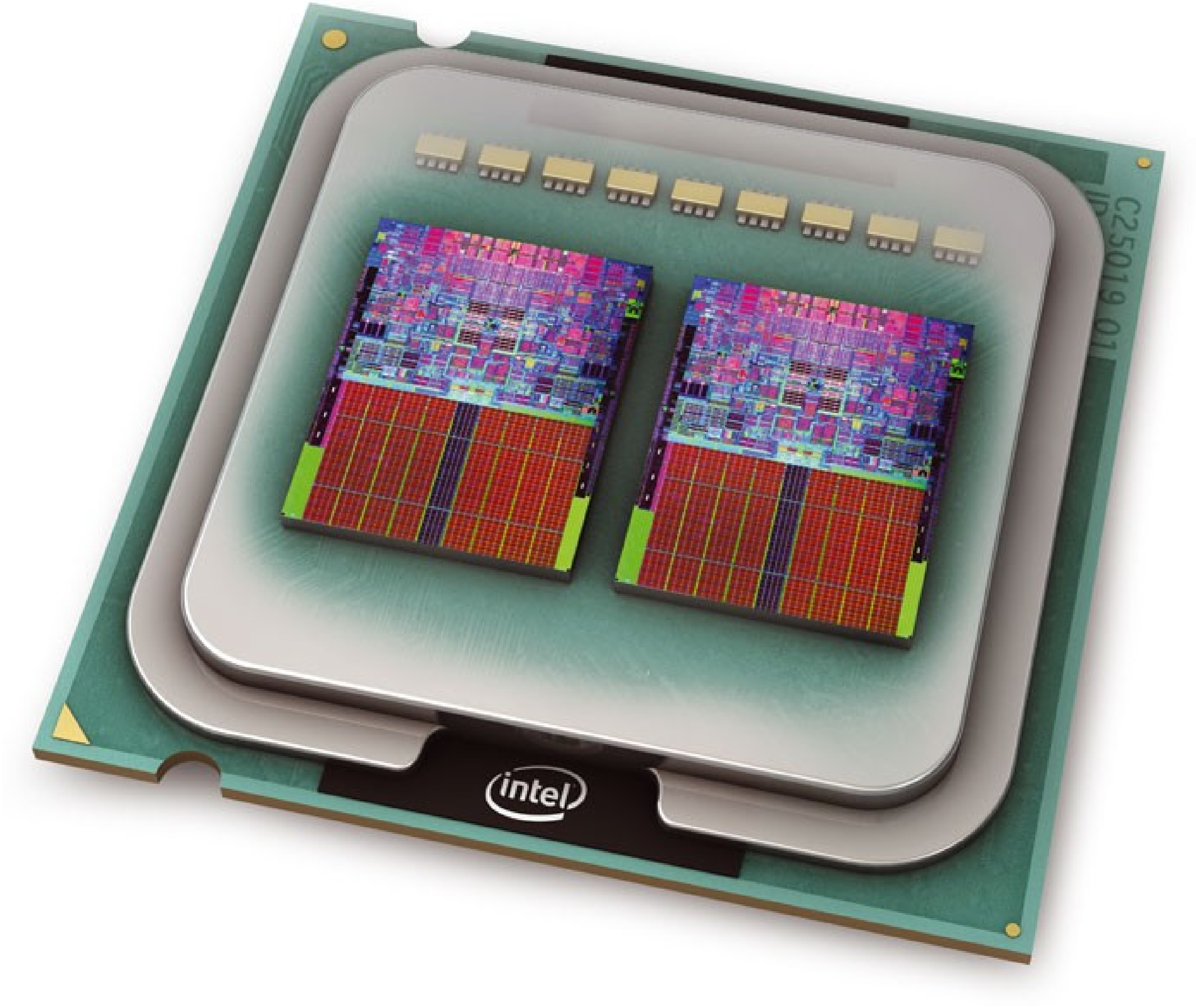
- Generates program profiles, by symbol and by address
- Extra options

`-iclass`	- instruction class profiling (e.g., ALU, branch)
`-iprof`	- instruction profiling (e.g., bnez, addi, etc...)
`-brprof`	- branch class profiling (e.g., direct, calls, cond)
`-amprof`	- address mode profiling (e.g., displaced, R+R)
`-segprof`	- load/store segment profiling (e.g., data, heap)
`-tsymprof`	- execution profile by text symbol (i.e., funcs)
`-dsymprof`	- reference profile by data segment symbol
`-taddrprof`	- execution profile by text address
`-all`	- enable all of the above options
`-pcstat <stat>`	- record statistic `<stat>` by text address

- NOTE: "`-taddrprof`" == "`-pcstat sim_num_insn`"

Questions regarding the single core assignment?

Multi Core Assignment



Multi Core Assignment

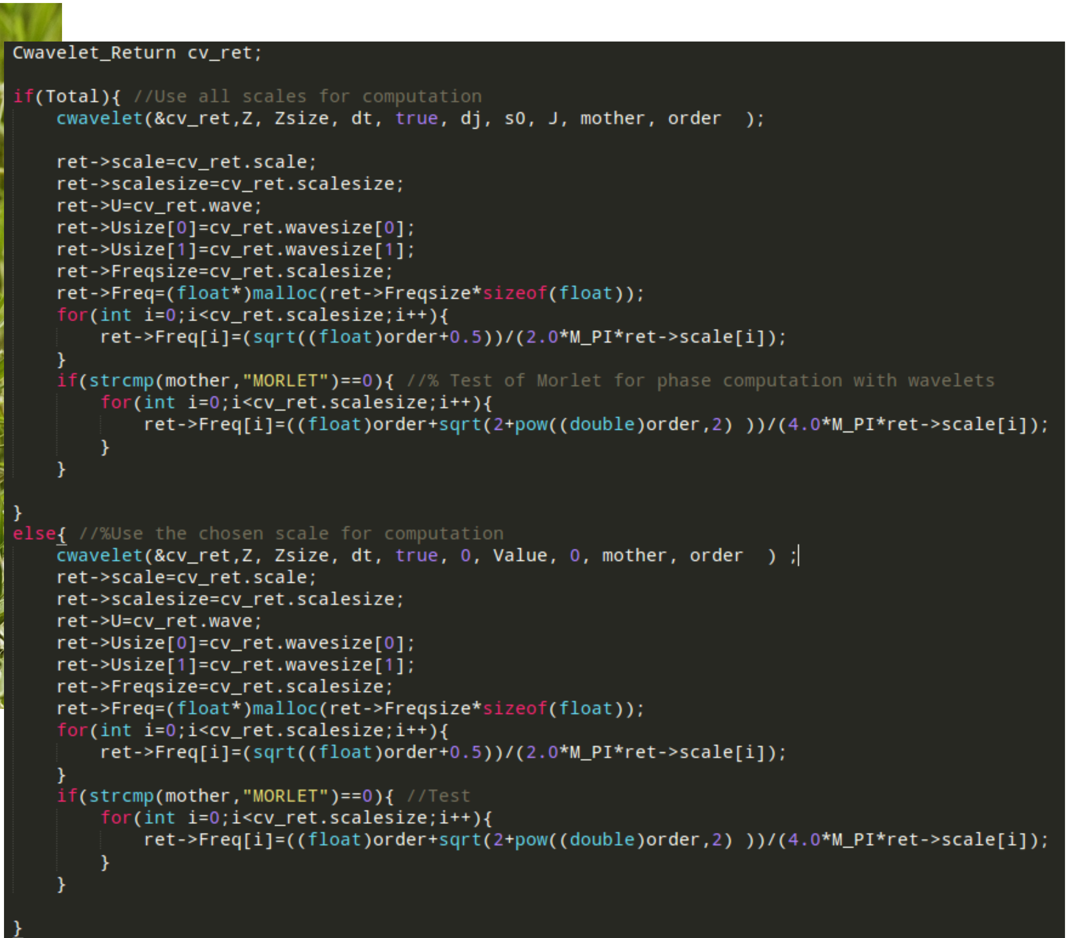
- Map the early seizure detection onto a multi core x86 platform
- Minimize Energy-Delay-Area-Product (EDAP)
  - Parallelize the code
  - Tune the multi core platform (Memory, interconnect, number of cores)
  - Optimize the given c-code (e.g. loop transformations to increase locality)
  - Any other technique you can think of!

Tools of the Trade

- Sniper Sim

Tools of the Trade

- Sniper Sim

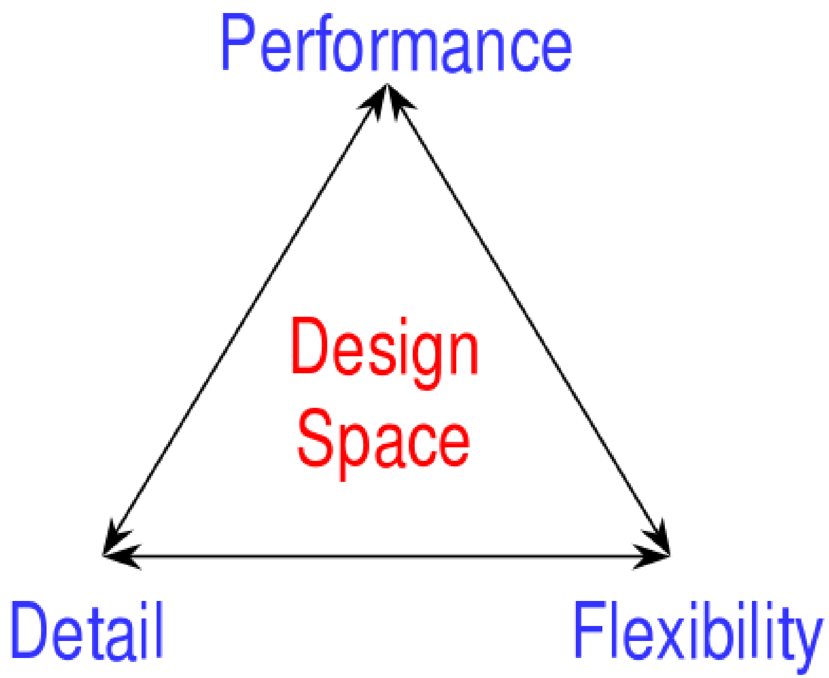


```
Cwavelet_Return cv_ret;

if(Total){ //Use all scales for computation
    cwavelet(&cv_ret,Z, Zsize, dt, true, dj, s0, J, mother, order  );

    ret->scale=cv_ret.scale;
    ret->scalesize=cv_ret.scalesize;
    ret->U=cv_ret.wave;
    ret->Usize[0]=cv_ret.wavesize[0];
    ret->Usize[1]=cv_ret.wavesize[1];
    ret->Freqsize=cv_ret.scalesize;
    ret->Freq=(float*)malloc(ret->Freqsize*sizeof(float));
    for(int i=0;i<cv_ret.scalesize;i++){
        ret->Freq[i]=(sqrt((float)order+0.5))/(2.0*M_PI*ret->scale[i]);
    }
    if(strcmp(mother,"MORLET")==0){ //% Test of Morlet for phase computation with wavelets
        for(int i=0;i<cv_ret.scalesize;i++){
            ret->Freq[i]=((float)order+sqrt(2+pow((double)order,2) ))/(4.0*M_PI*ret->scale[i]);
        }
    }

}
else{ //%Use the chosen scale for computation
    cwavelet(&cv_ret,Z, Zsize, dt, true, 0, Value, 0, mother, order  ) ;
    ret->scale=cv_ret.scale;
    ret->scalesize=cv_ret.scalesize;
    ret->U=cv_ret.wave;
    ret->Usize[0]=cv_ret.wavesize[0];
    ret->Usize[1]=cv_ret.wavesize[1];
    ret->Freqsize=cv_ret.scalesize;
    ret->Freq=(float*)malloc(ret->Freqsize*sizeof(float));
    for(int i=0;i<cv_ret.scalesize;i++){
        ret->Freq[i]=(sqrt((float)order+0.5))/(2.0*M_PI*ret->scale[i]);
    }
    if(strcmp(mother,"MORLET")==0){ //Test
        for(int i=0;i<cv_ret.scalesize;i++){
            ret->Freq[i]=((float)order+sqrt(2+pow((double)order,2) ))/(4.0*M_PI*ret->scale[i]);
        }
    }

}
```

Tools of the Trade

- Sniper Sim
  - Tool to analyze performance of multicore systems
  - Cores themselves are modeled on a functional level
  - Interconnect, memory and number of cores (i.e. the system level) can be configured and has performance models
- McPat
  - Integrated Power, Area and Timing modeling framework
  - Integrates with Sniper Sim toolflow
- OpenMP
  - Framework to parallelize source for multicore platforms

The Zen of Hardware Model Design



Performance: speeds design cycle

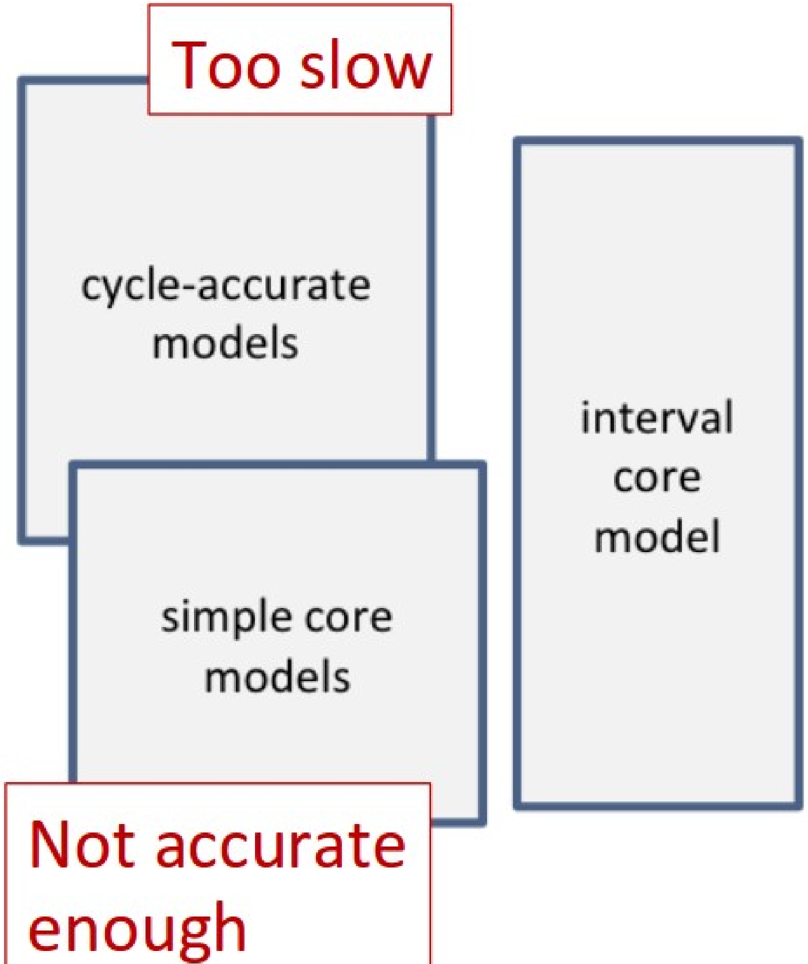
Flexibility: maximizes design scope

Detail: minimizes risk

- Infrastructure goals will drive which aspects are optimized
- SimpleScalar favors performance and flexibility

Needed Detail Depends on Focus

Component	Single-event time scale	Required sim time
RTL	single clock cycle	millions of cycles
OOO execution	↓	↓
Core memory ops		
L1 cache access		
LLC access		
Off-socket	microseconds	seconds



Too slow

cycle-accurate models

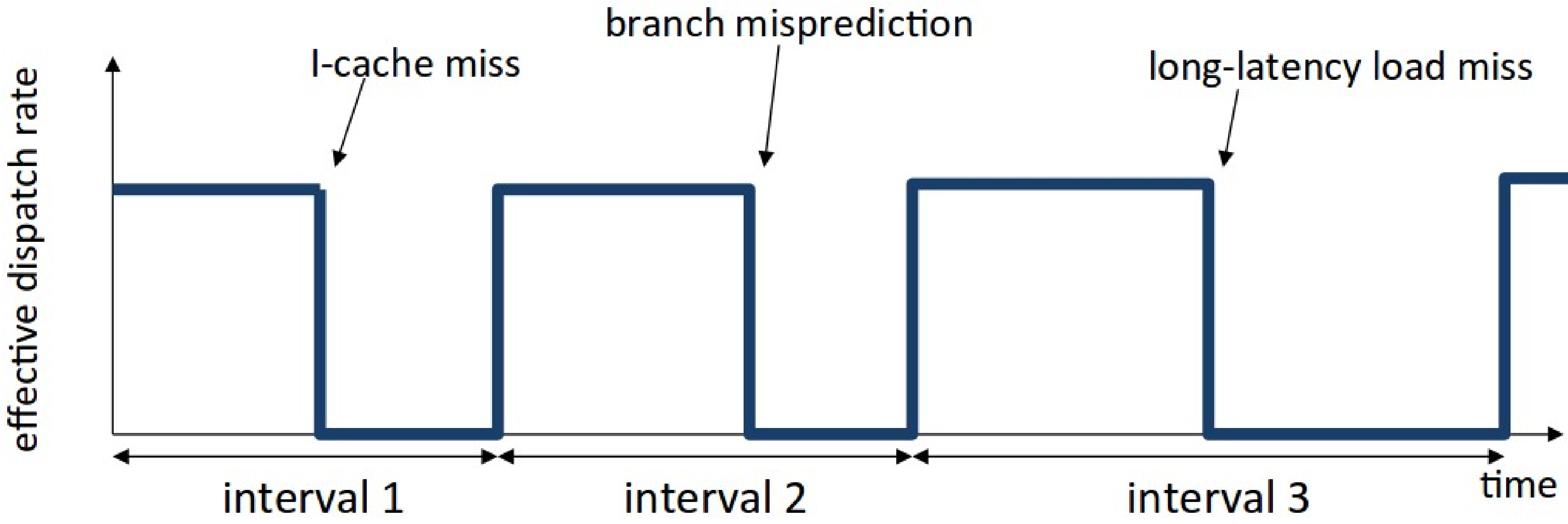
simple core models

interval core model

Not accurate enough

INTERVAL SIMULATION

- Out-of-order core performance model with in-order simulation speed

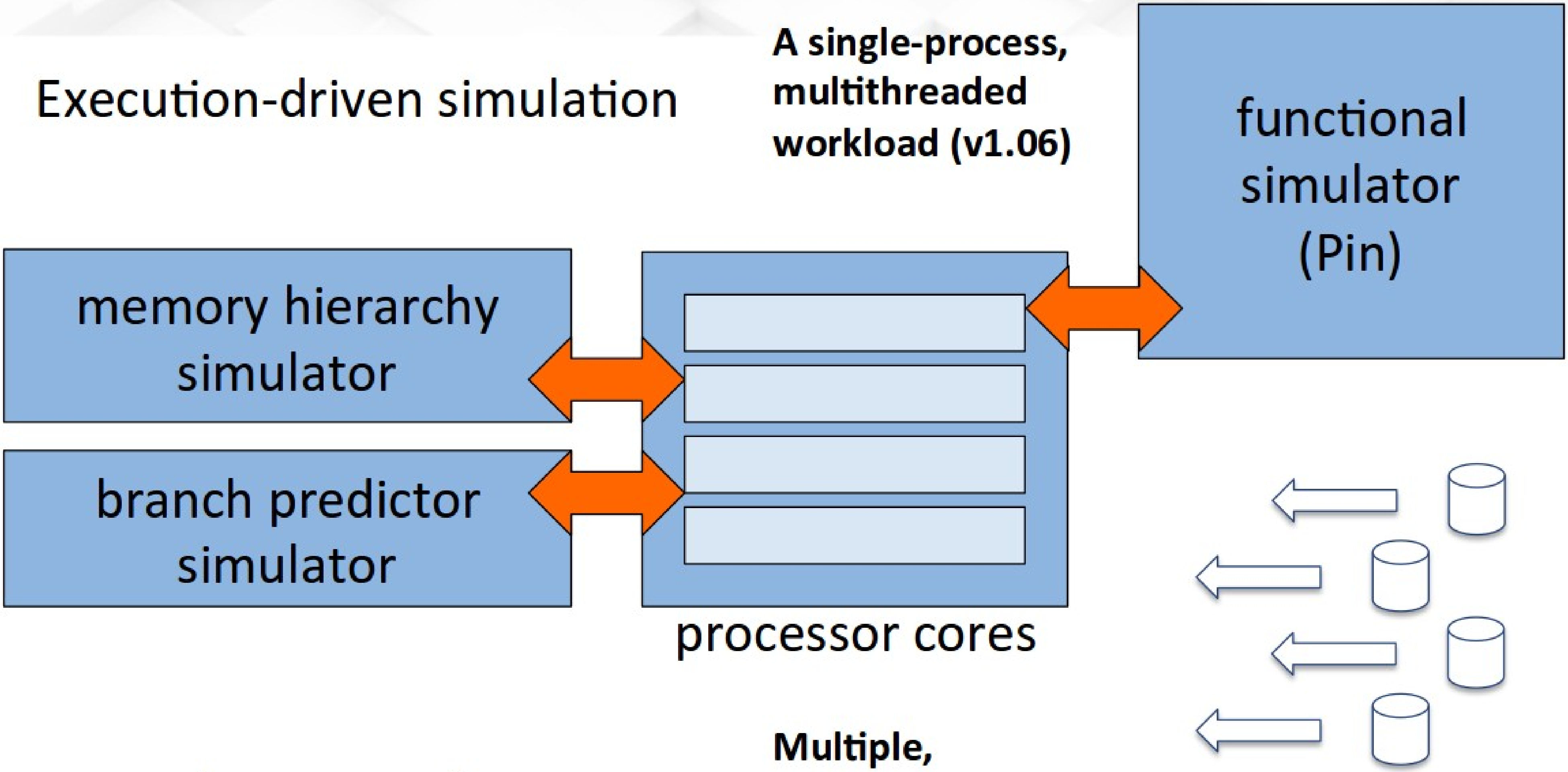


D. Genbrugge et al., HPCA'10
S. Eyerman et al., ACM TOCS, May 2009
T. Karkhanis and J. E. Smith, ISCA'04, ISCA'07

Simulation in Sniper

Execution-driven simulation

A single-process, multithreaded workload (v1.06)



functional simulator (Pin)

memory hierarchy simulator

branch predictor simulator

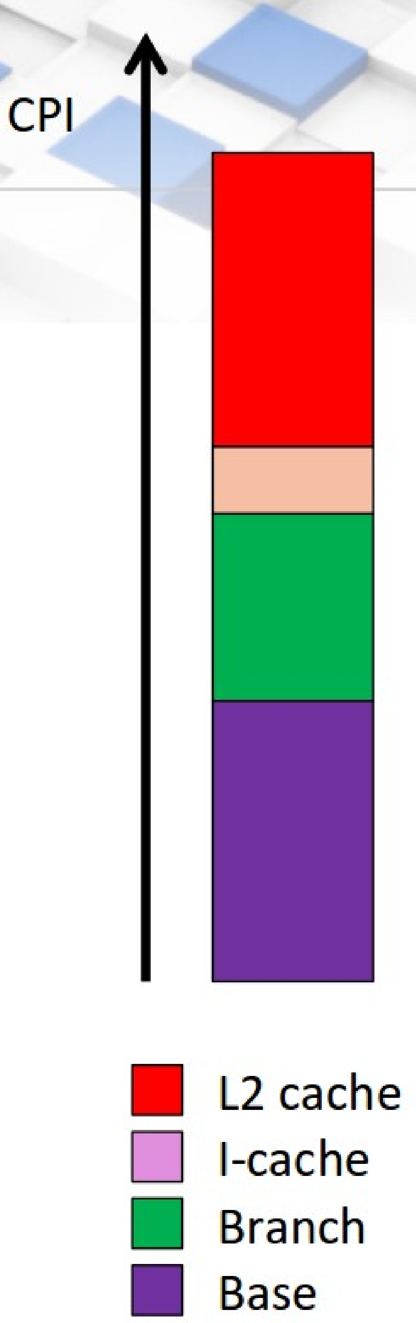
processor cores

Trace-driven simulation

Multiple, single-threaded workloads (v2.0)

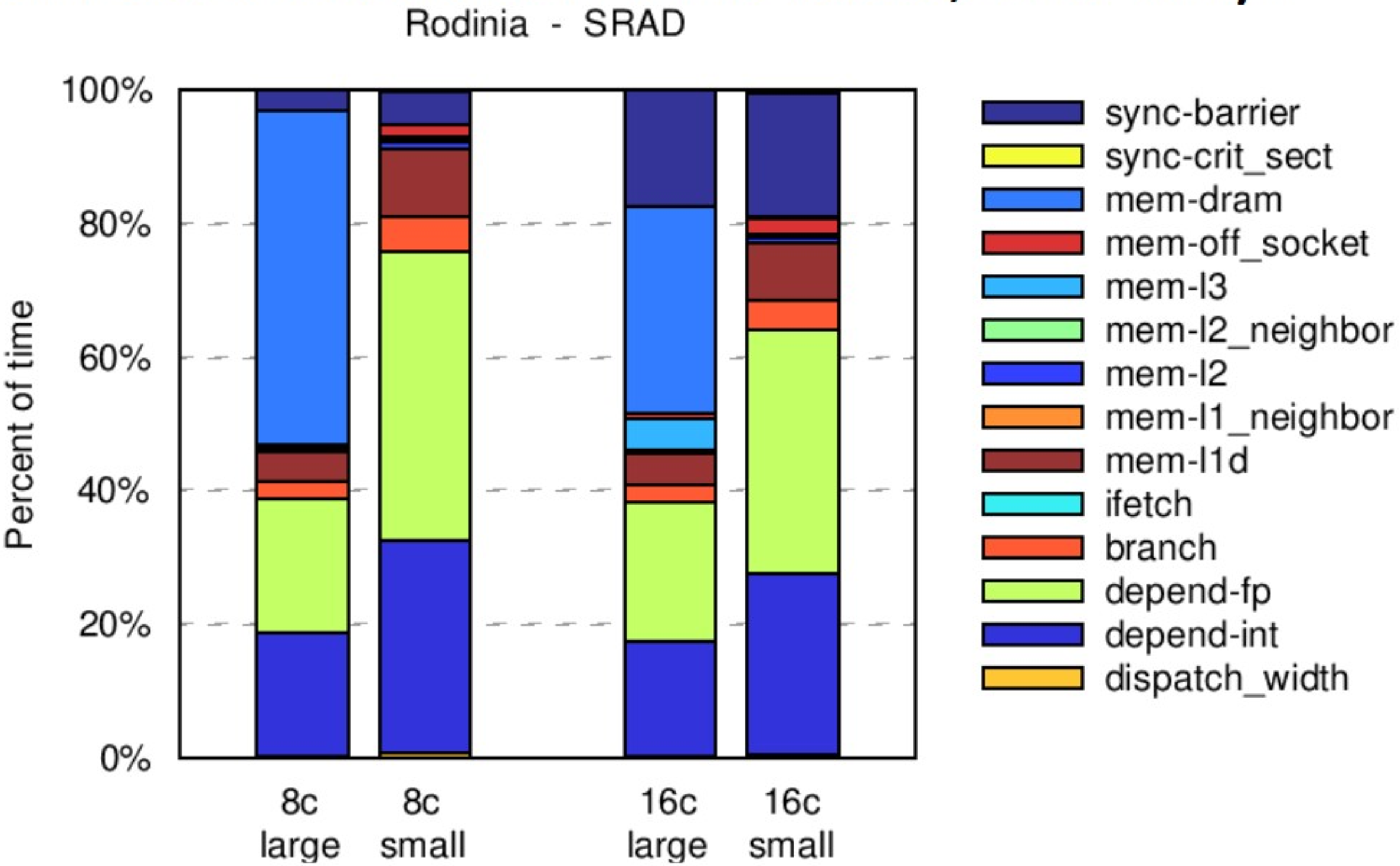
Cycle stacks

- Where did my cycles go?
- CPI stack: cycles per instruction, broken up in components
- Normalize by either
  - Number of instructions (CPI stack)
  - Execution time (time stack)
- Different from miss rates as cycle stacks directly quantify the effect on performance

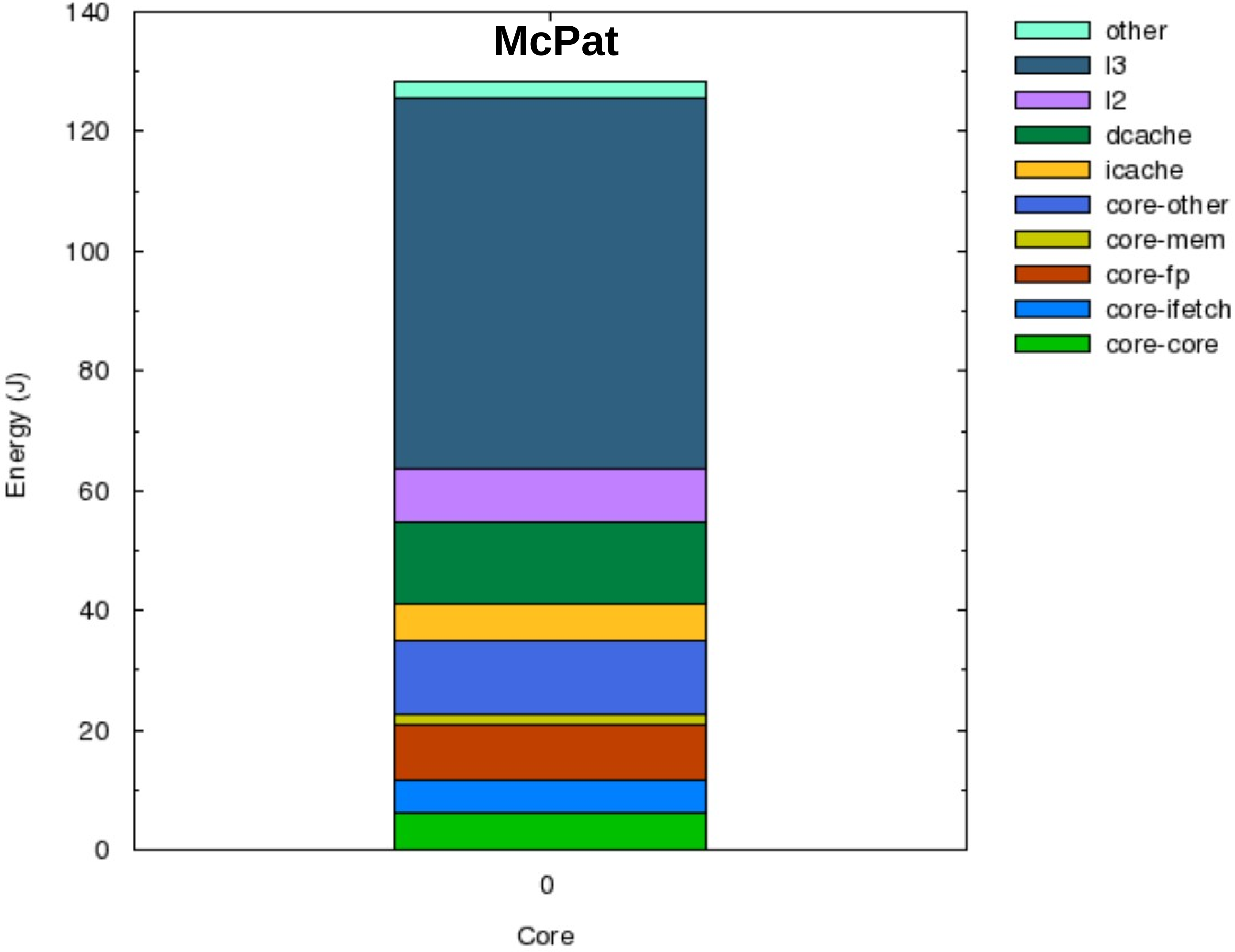


CYCLE STACKS AND SCALING BEHAVIOR

- Scaling to more cores, larger input set size
- How does execution time scale, and why?



McPat


140
120
100
80
60
40
20
0
Energy (J)
0
Core
other
l3
l2
dcache
icache
core-other
core-mem
core-fp
core-ifetch
core-core

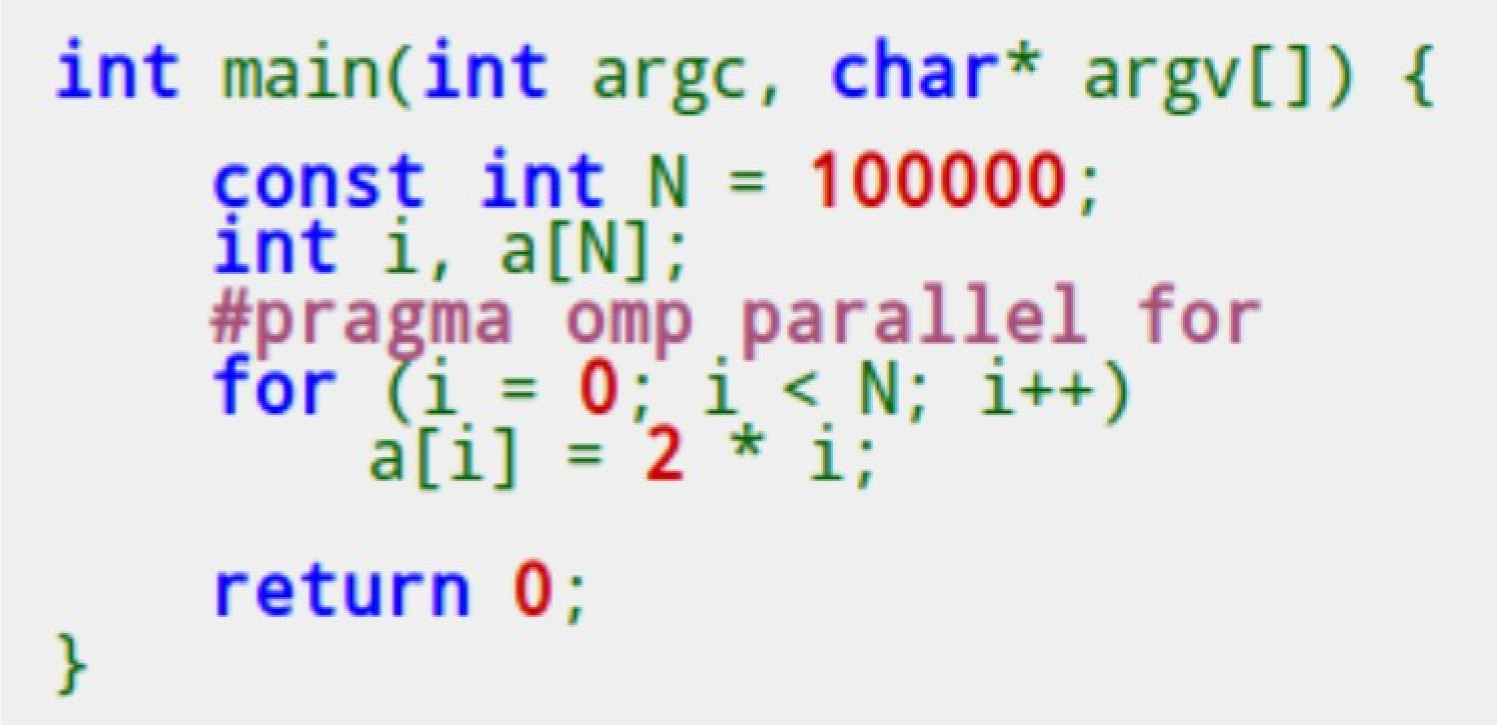
How to Use

- Again, check the website to get started: http://www.es.ele.tue.nl/~mwijtvliet/5MD00_MC/?page=preparation
- N.B. Download the application source again, since the makefile differs from the previous assignment!
- **N.B. Extract the application source in the correct directory!! Sniper Sim uses relative paths in it's Makefiles!! (see guidelines page)**
- For the tunable parameters, check the sniper-manual (in particular chapters 5 & 6).
- The website also explains how to extract the energy, delay and area metrics from the tools

Again, make sure you understand what you tune!

- Read the sniper sim manual to understand the configuration options. In general the options are a bit more high level, and probably easier to understand.
- Procedure (for report):
  - Profile the application, identify bottlenecks
  - Try to improve the mapping/system
    - Document what you tried and what you were expecting
  - Show the results. Did your solution work? If not, try to explain why.
- Try to obtain the minimum EDAP!

OpenMP



```c
int main(int argc, char* argv[]) {

    const int N = 100000;
    int i, a[N];
    #pragma omp parallel for
    for (i = 0; i < N; i++)
        a[i] = 2 * i;

    return 0;
}
```

OpenMP

- N.B. OpenMP does not check (and certainly not prove) whether the specified parallelization is correct (or beneficial). When a piece of code is annotated, the compiler simply assumes that it is OK to parallelize it.

Beware of race conditions!

Finally

- **Please use the forums to ask your questions!!**
- Use a topic title that captures your problem as accurately as possible (i.e. not 'code not working').
- Before you ask, check if someone else had a similar issue. Perhaps the answer to your question is already there.
- Remember Google is your friend, it can typically provide answers much faster than the TAs
- If you want to talk to a TA, send us an email to make an appointment.

Slides in this presentation were shamelessly copied from:

- http://snipersim.org/documents/2012-06-09%20Sniper%20ISCA%20Tutorial.pdf
- https://courses.cs.washington.edu/courses/cse471/06sp/hw/simpleScalar3.0guide.pdf
- http://www.simplescalar.com/docs/simple_tutorial_v2.pdf
- http://www.simplescalar.com/docs/simple_tutorial_v4.pdf