Timing Analysis of Embedded Software for Families of Microarchitectures

Jan Reineke, UC Berkeley

Edward A. Lee, UC Berkeley

Representing Distributed Sense and Control Systems (DSCS) theme of MuSyC

With thanks to:
Hiren Patel
Alberto Sangiovanni-Vincentelli

Clark Kerr Campus, UC Berkeley

November 17, 2011

This work has received funding from NSF #0720882 and MURI #FA9550-06-0312.

The Challenge: Microarchitecture Selection

```
// Perform the convolution.
for (int i=0; i<10; i++) {
  x[i] = a[i]*b[j-i];
  // Notify listeners.
  notify(x[i]);
}
```

Em

Select a microarchitecture that
a) satisfies all timing requirements, and
b) minimizes cost/size/energy.

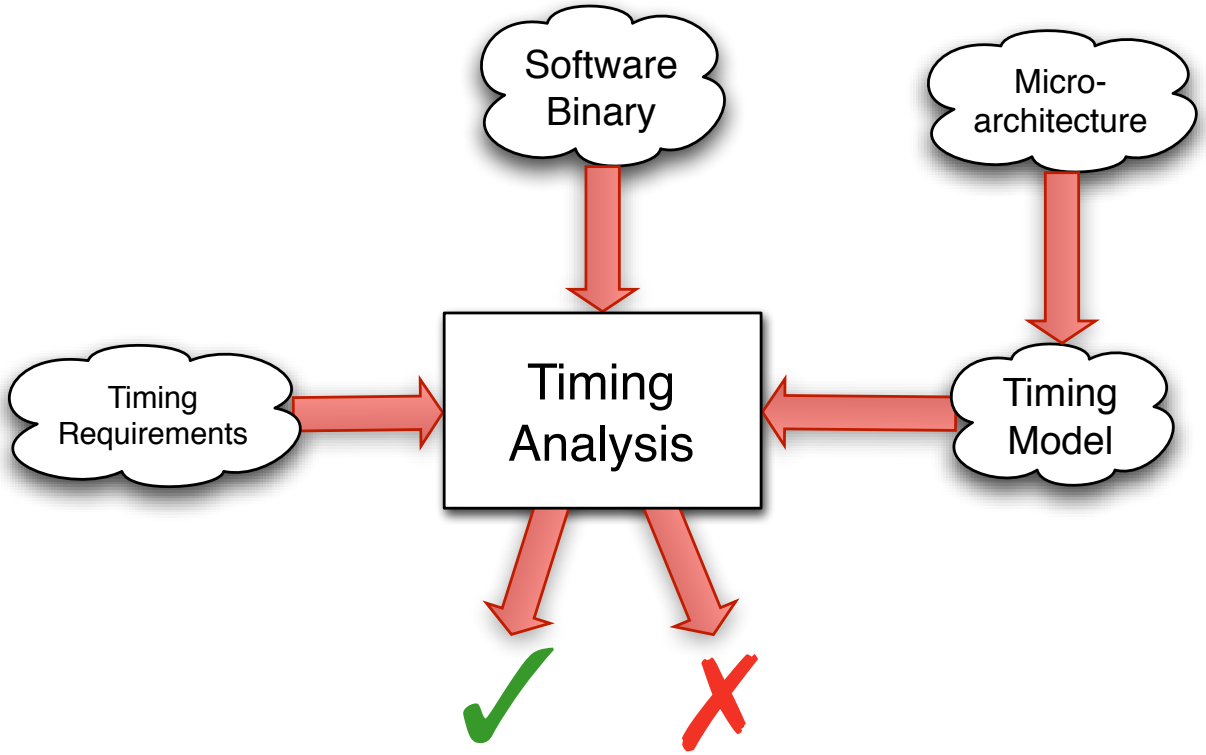
Timing Requirements

Choices:

- *Processor frequency*
- *Sizes and latencies of local memories*
- *…ating-point unit*
- *…*

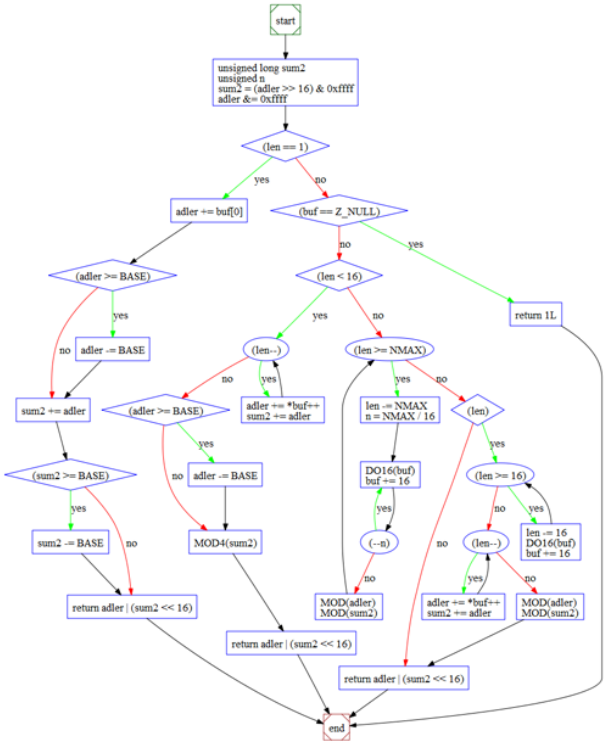
Family of Microarchitectures = Platform

Timing Analysis Today



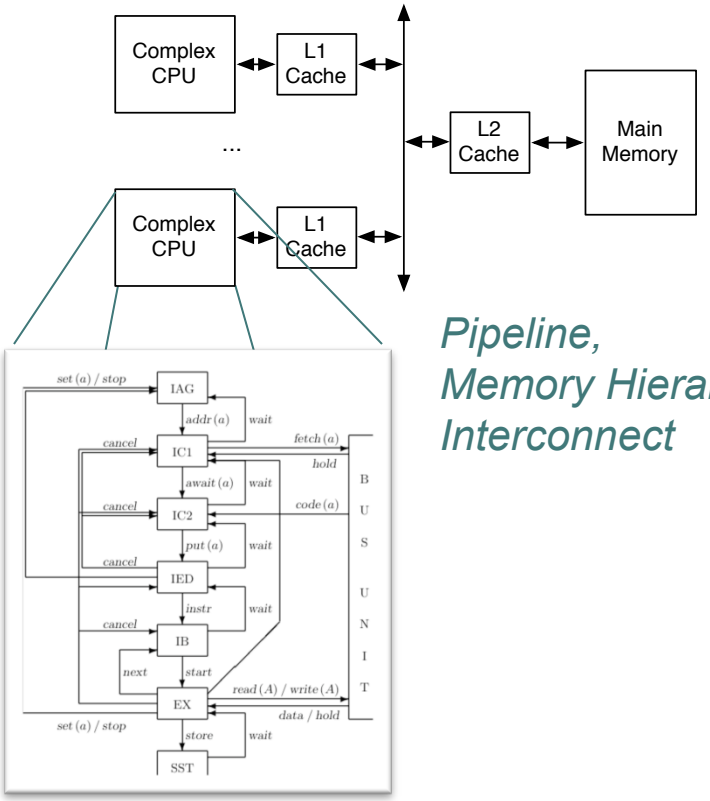
What does the execution time of a program depend on?

Input-dependent control flow



+

Microarchitectural State



Pipeline, Memory Hierarchy, Interconnect

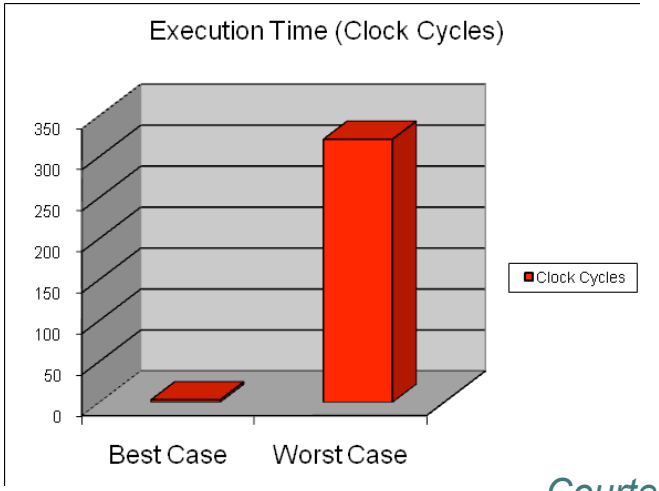
Reineke et al., Berkeley 4

Example of Influence of Microarchitectural State

x=a+b; →

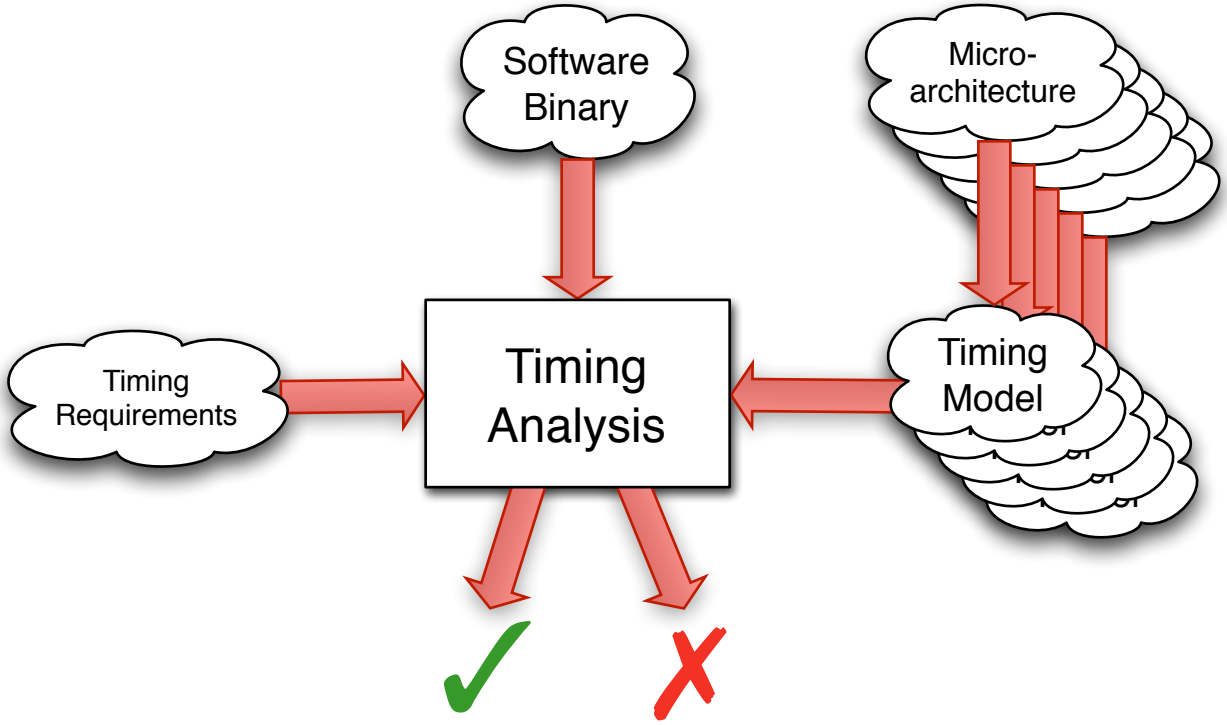
```
LOAD   r2, _a
LOAD   r1, _b
ADD    r3,r2,r1
```

Motorola PowerPC 755



Courtesy of Reinhard Wilhelm.

Architecture Selection: How it might be done today

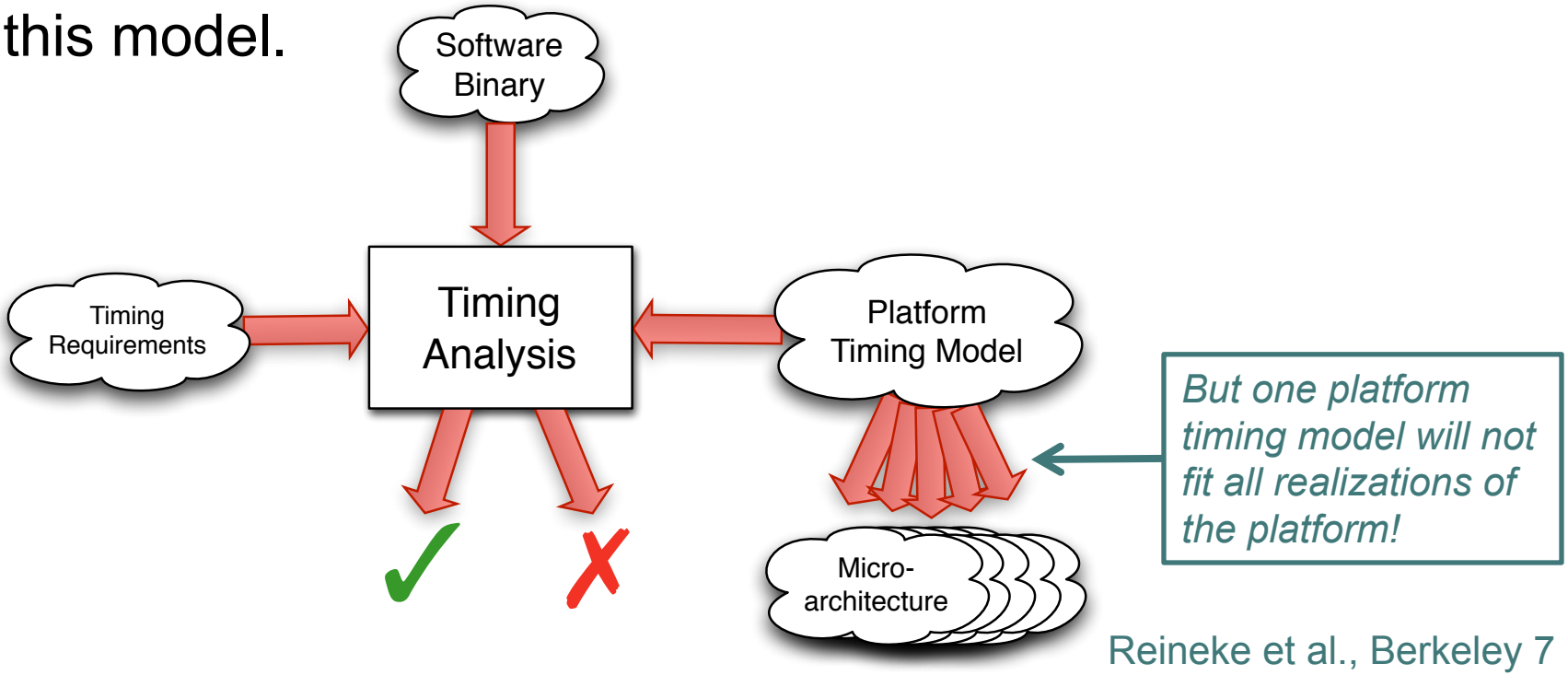


1. For every microarchitecture ➔ a different timing model
   time-consuming, costly, error-prone, and still often inaccurate
2. Space of microarchitectures and their configurations is huge

How it will be done tomorrow: Start out with the Timing Model

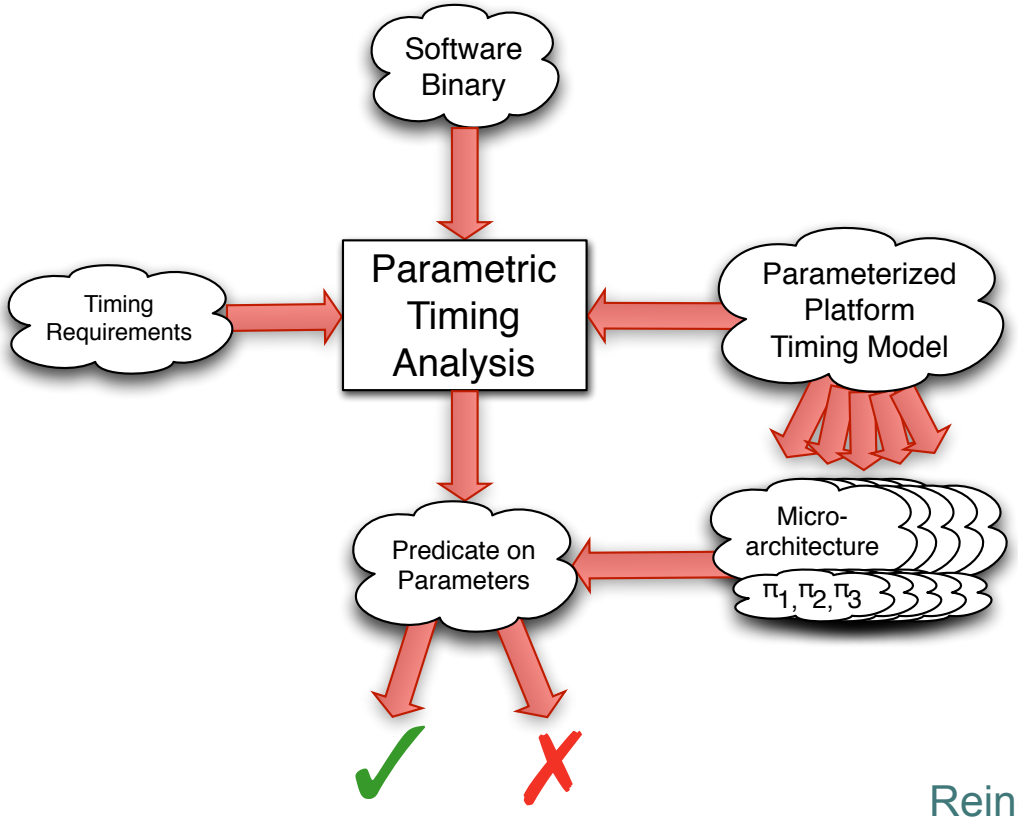
Reverse process:

1. Design platform timing model for precise and efficient timing analysis.
2. Develop family of microarchitectures conforming to this model.



Support Wide Range of Realizations: Parameterize the Timing Model

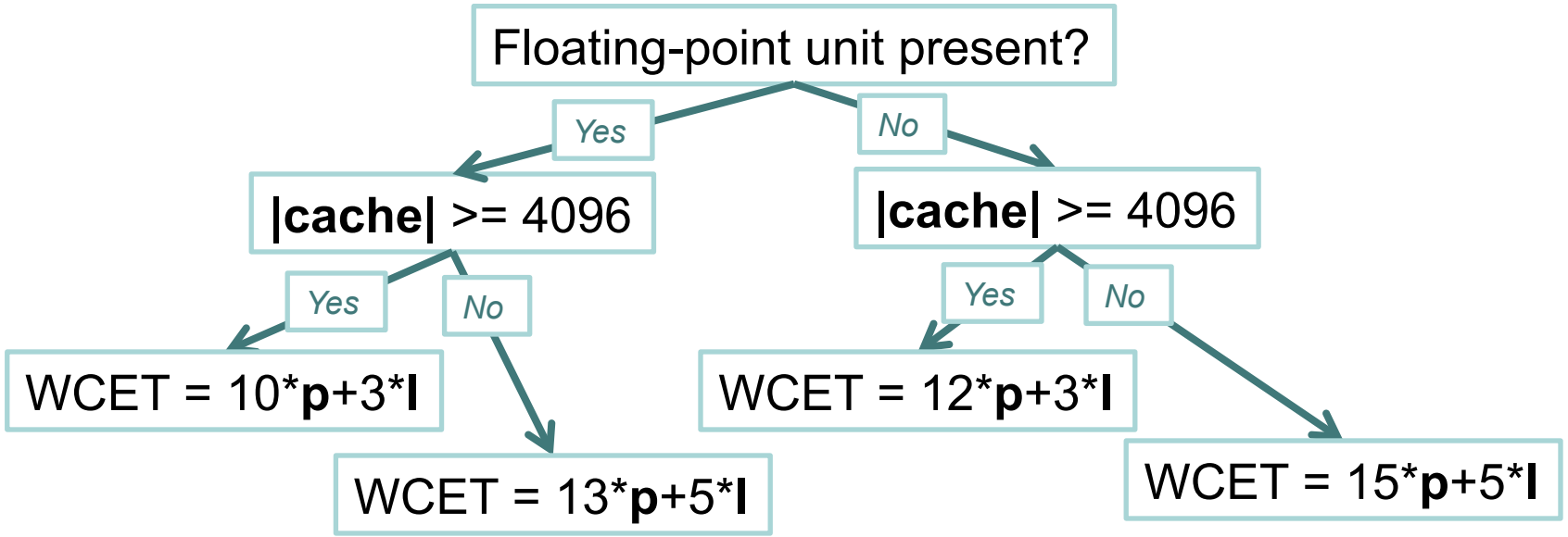
1. Parameterize the model to enable wide range of realizations.
2. Parametric timing analysis: symbolic computation of *all microarchitectures* that satisfy the timing requirements.



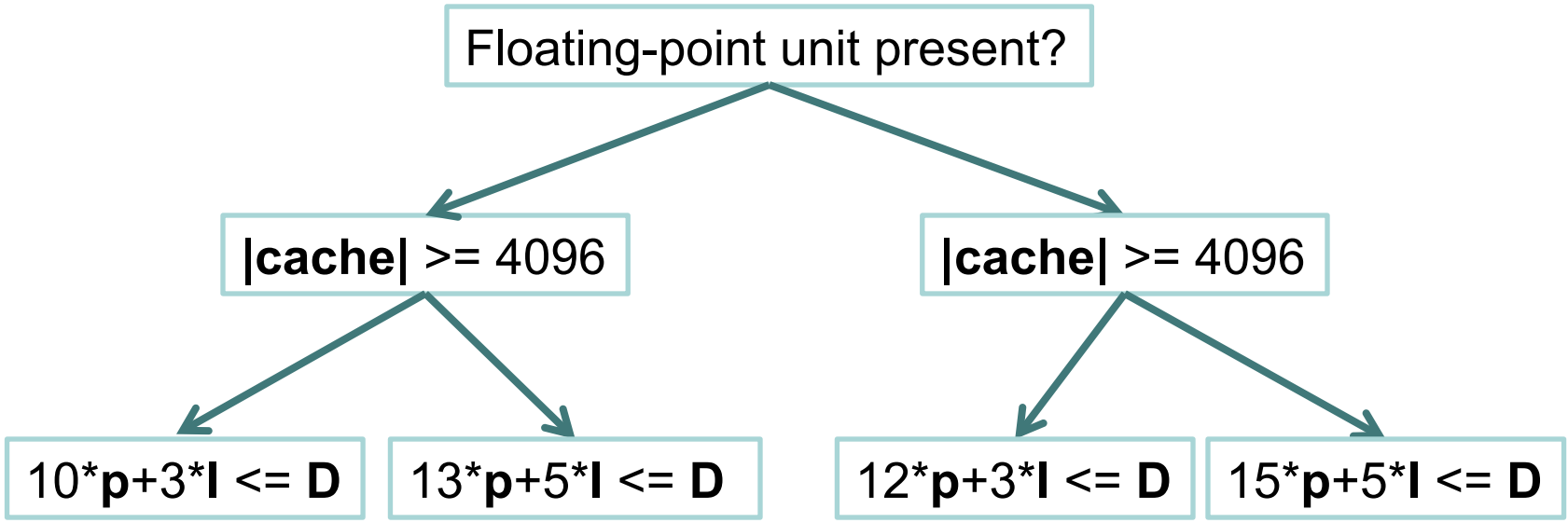
Example of Parameters and Analysis Results: WCET in Terms of Parameters

Assume four parameters:

1. Presence/absence of floating-point unit
2. Size of cache **|cache|** in bytes
3. Processor period **p**=1/frequency in nanoseconds
4. Latency of main memory **l** in nanoseconds



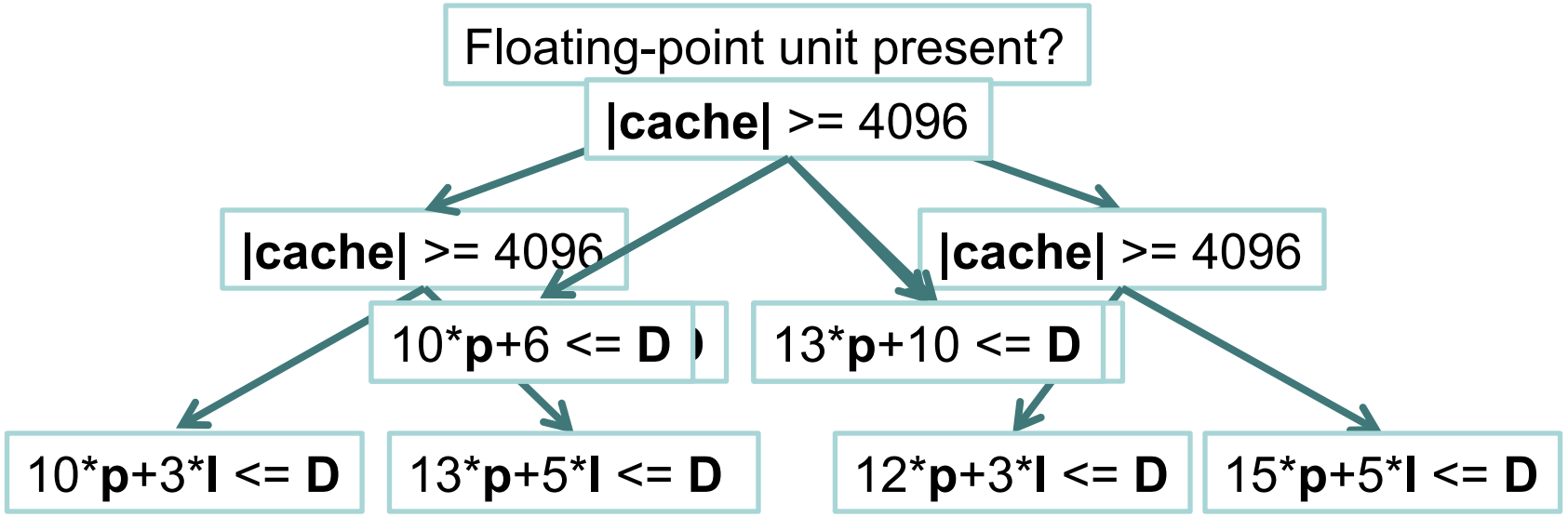
Example of Analysis Results: Assume Deadline **D**



This enables us to determine the cheapest/most-energy efficient/… microarchitecture satisfying the constraints!

Related Challenge: Configuration at Runtime

Microarchitecture selected at design time eliminates some of the options, but not necessarily all:



Given a model of the power consumption of the processor, we can derive energy optimal cache and processor configurations in terms of the deadline.

Related Challenge: Configuration at Runtime

Microarchitecture selection is based on worst-case assumptions at **design time**.

At **runtime** we may have more knowledge:

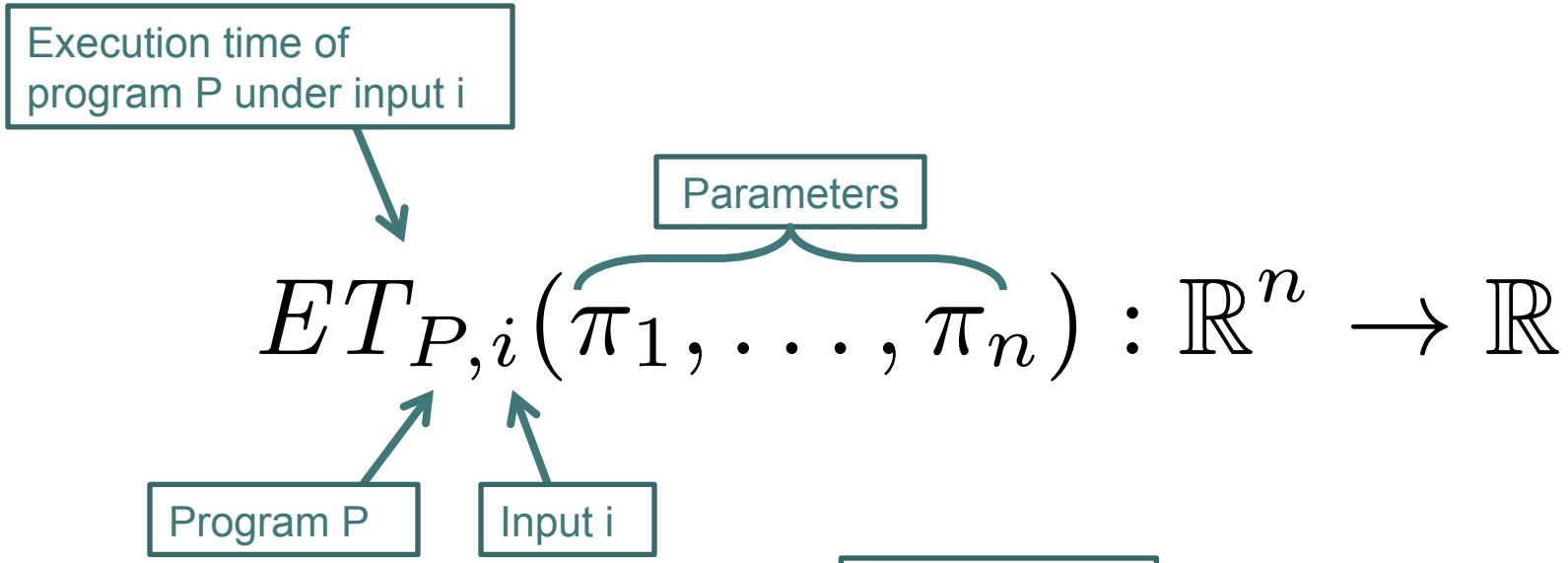
We can exploit this knowledge to reconfigure a microarchitecture at runtime:

- *Frequency/voltage scaling*
- *Distribution of shared resources:*
  - *Shared cache/scratchpad*
  - *Functional units*

Plant

at design time.

Parameterized Timing Models: Formalization



Necessary and Desirable Properties of Parameterized Timing Models

Necessary:
Execution time should be monotone in parameters.

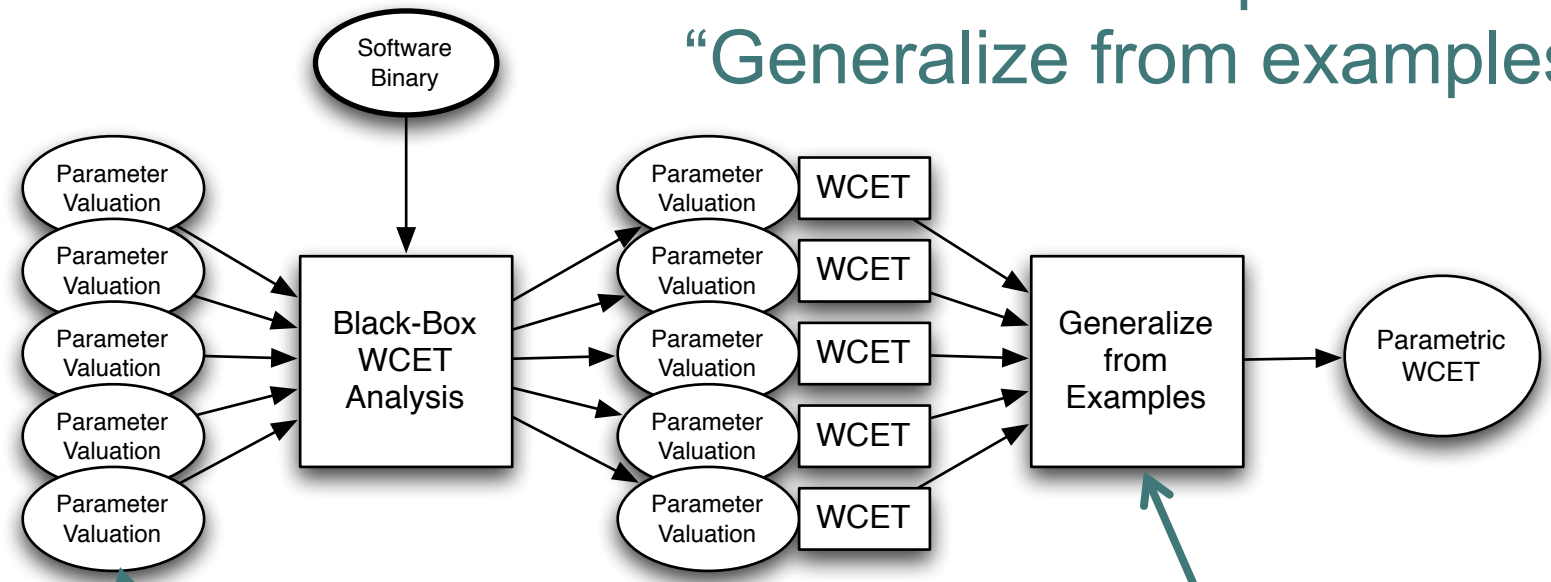
"a higher frequency always yields a shorter execution time"
"a smaller cache always yields a longer execution time"

Desirable for efficiency:
Execution time should depend linearly on parameters.

"doubling the processor frequency will decrease execution time by a factor of two"

Parametric Timing Analysis: Black-Box Approach

"Reduce to known problem" and "Generalize from examples"



We know how to select representative parameter valuations.

Exploits properties of parameterized timing models to generalize from samples to parametric formula.

Context: Precision-Timed (PRET) Machines

http://chess.eecs.berkeley.edu/pret/

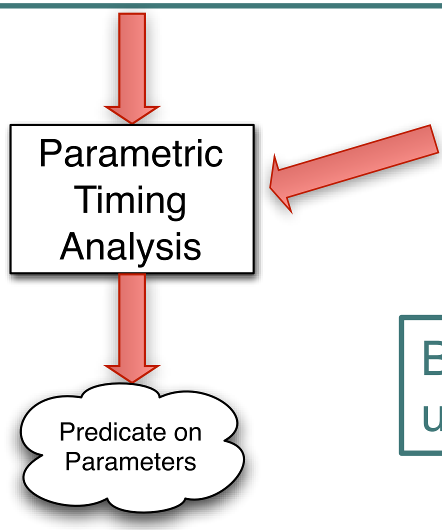
Modern microarchitectures are optimized for average-case performance at the expense of timing predictability and repeatability.

The goal of the PRET project is to introduce timing predictability and repeatability without sacrificing performance.

Instruction Set Architecture:
Precise control over timing through deadline instructions.

***Microarchitecture**:*
The Precision-Timed ARM adheres to a simple timing model without sacrificing performance:

- *Thread-interleaved pipeline*
- *Scratchpad memories in place of caches*
- *Predictable DRAM controller*



Black-box WCET analysis can be realized using *GameTime* (Sanjit Seshia).

Conclusions and Future Work

- Today, timing models originate from previously developed microarchitectures.
- Tomorrow, timing models will be carefully designed to enable precise and efficient parametric timing analysis, and microarchitectures will follow.
- Ongoing work: Proof-of-concept implementation for a parameterized PRET timing model.

Raffaello Sanzio da Urbino – The Athens School