A Retargetable Software Timing Analyzer Using Architecture Description Language

Xianfeng Li: **Peking Univ. (PKU)**

Abhik Roychoudhury: National Univ. of Singapore (NUS)

Tulika Mitra: National Univ. of Singapore (NUS)

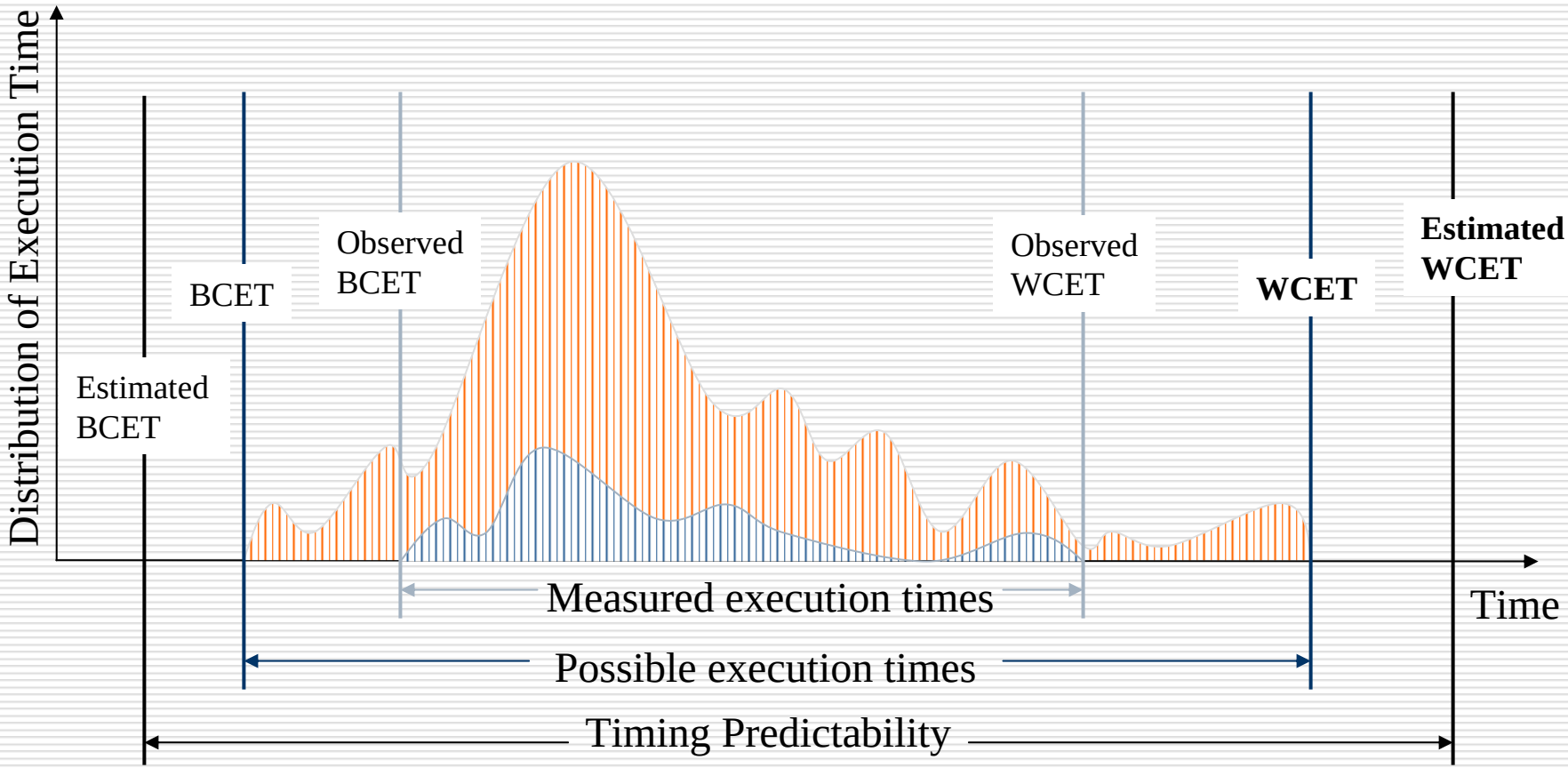
Prabhat Mishra: Univ. of Florida (UFL)

Xu Cheng: Peking Univ. (PKU)

Timing Analysis

- Problem statement
  - Given a program and an architecture, estimate the maximum/minimum execution time of the program for all possible inputs
- Worst Case Execution Time (WCET)
  - Essential information for real-time scheduling

Worst-Case Execution Time (WCET)



Timing Analysis

- Problem statement
  - Given a program and an architecture, estimate the maximum/minimum execution time of the program for all possible inputs
- Worst Case Execution Time (WCET)
  - Essential information for real-time scheduling
- Static WCET Analysis
  - Derive a conservative estimate on WCET
  - Program path + instruction timing

Microarchitecture Modeling

- Microarchitecture innovations
  - For average case performance
  - Poor predictability – bad for real-time
- Microarchitecture modeling started 15+ years ago
  - human understanding
  - custom modeling algorithms
  - handcrafted code
- A retargetable framework is needed
  - Input:
    - Program
    - Processor spec. (Architecture Description Language)
  - Output: WCET

WCET Analysis Framework

- Integer Linear Programming (ILP) based analysis

maximize

$$T_{prog} = \sum N_B * C_B$$

where

$B \in BasicBlocks(prog)$

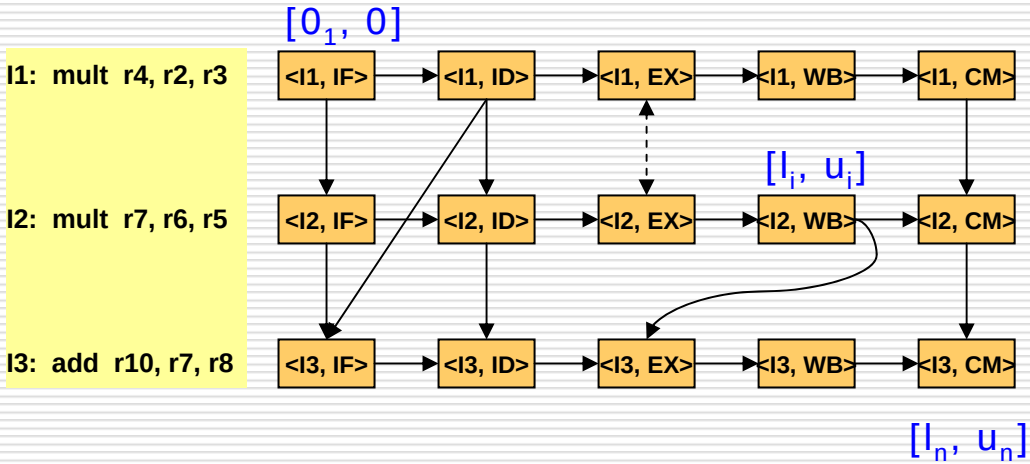
N_B: execution count of B

C_B: WCET of B ← **affected by hardware**

subject to Control Flow Graph (CFG) constraints

Basic Block Timing Analysis (C_B)

- Execution graph:
  - Nodes: <instr, stage>
  - Relations:
    - dependences
    - contentions
    - parallelisms



- Time interval based
  - Not simulation:
    all cases are covered
  - No state space explosion:
    avoid enumeration on
    single time points

Algorithm outline:

```
Initialization(G);
repeat {
    for each node v in G {
        earliest_time(v);
        latest_time(v);
    }
} until (a fixed point is reached)
```

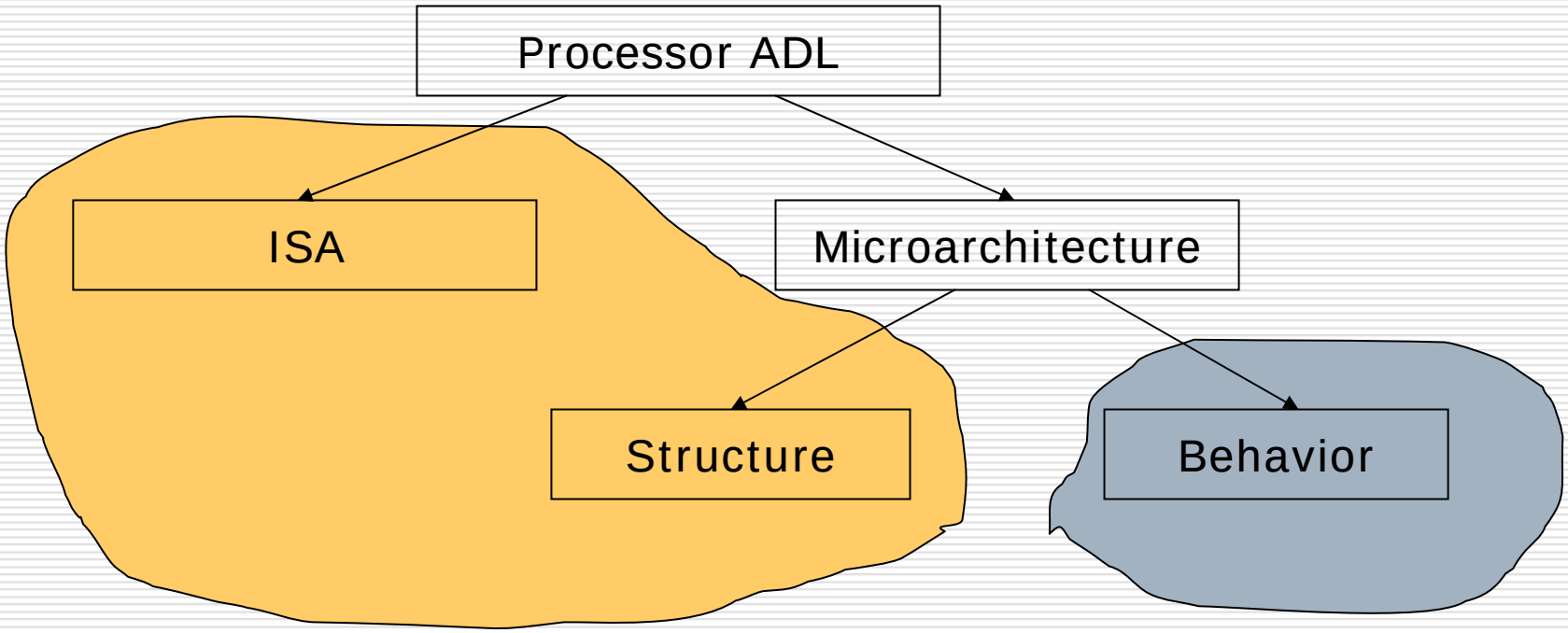
Modeling Out-of-Order Processors for WCET Analysis
Li et al., Real-Time Systems Journal, Nov 2006

Architecture Description Language (ADL)

- Formal description of processor architecture
  - ISA + Microarchitecture
- Existing ADLs:
  - HMDES, LISA, MADL, UPFAST, EXPRESSION...
- Investigated retargeting issues
  - Compilation, simulation, synthesis, validation

EXPRESSION ADL

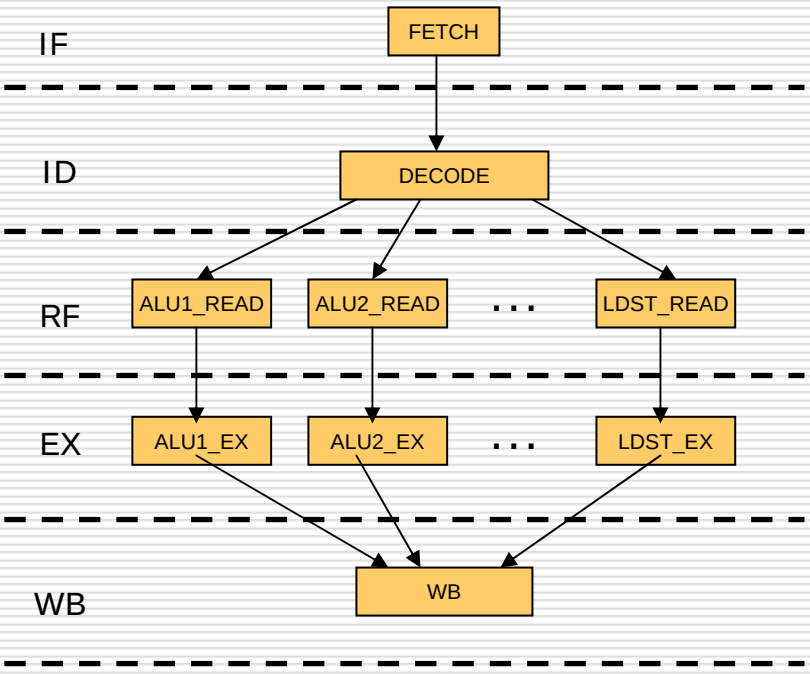
- Developed by Prof. Dutt's group at UC-Irvine



described in main ADL file

described in separate C++ files

An Example ADL (acesMIPS)




```
(PIPELINE_SECTION
  (PIPELINE  FETCH  DECODE  READ_EXECUTE  WB)
  (READ_EXECUTE (ALTERNATE  read_ex0 ...  read_ex4)
  (read_ex0(PIPELINE  ALU1_READ  ALU1_EX))
  ...
  (read_ex4(PIPELINE  LDST_READ  LDST_EX))
```

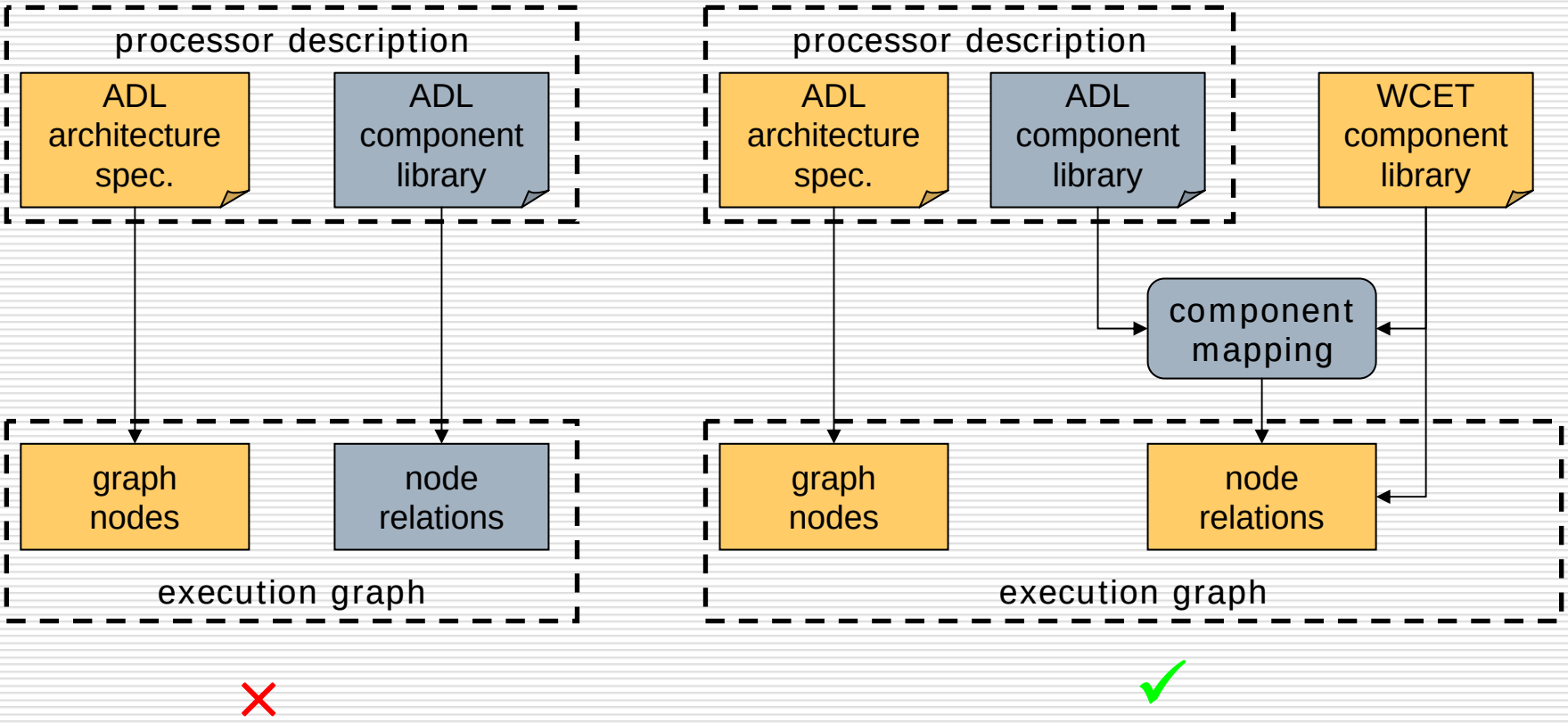
```
(ARCHITECTURE_SECTION
  (SUBTYPE  UNIT  FetchUnit  DecodeUnit  OpReadUnit ...)
  ...
  (DecodeUnit DECODE
    (CAPACITY  12)
    (OPCODES all)
    ...
```

ADL component library →

```
// DerivedUnit.h
class DecodeUnit : public SimpleUnit
{
    private:
        int  instBufsize;
        ...
}
```

From ADL To Execution Graph

- Execution graph: nodes + relations



WCET Component Library

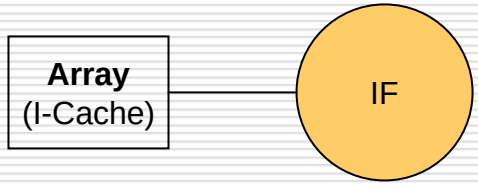
- Models characterizing WCET properties of components

- Goals:
  - Abstract, architecture independent
  - ADL component to WCET component mapping is easy

- Observations:
  - Pipeline path: a series of processing elements interfaced by storage elements
  - Behaviors of PEs are often dictated by SEs
  - Limited classes of SEs based on their timing behavior

WCET Component Library (cont)

- Array model
  - access latency

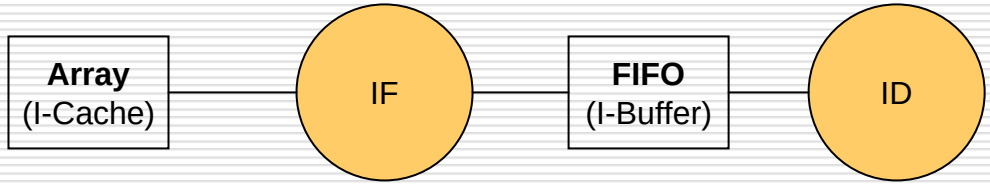


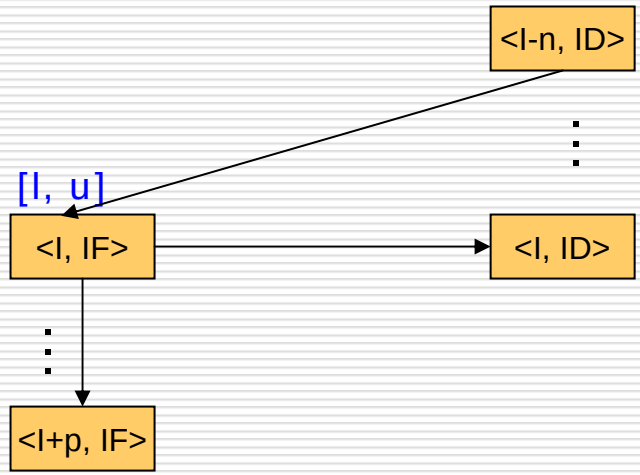
[l, u]

<l, IF>

WCET Component Library (cont)

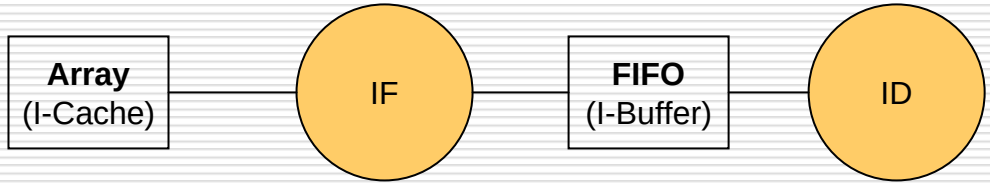
- Array model
  - access latency
- FIFO model
  - producer-consumer
  - size limit
  - parallelism limit





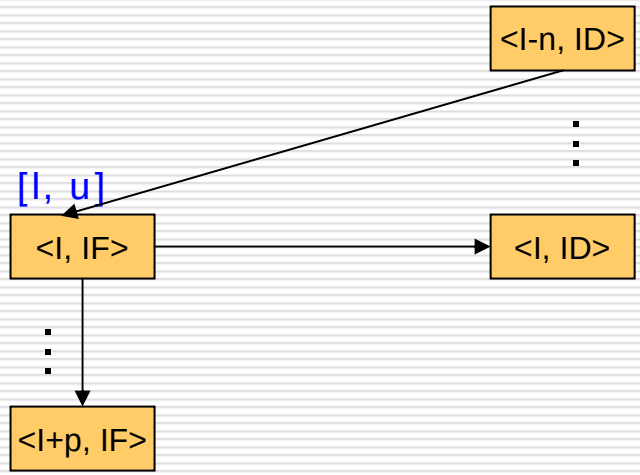
WCET Component Library (cont)

- Array model
  - access latency



- FIFO model
  - producer-consumer
  - size limit
  - parallelism limit

- Pool model
  - No statically decided access order
  - No general timing characterization
  - Some can be abstracted with limited retargetability (Out-of-order issue queue)



Case Study: acesMIPS

- A MIPS R4000-like processor (by EXPRESSION group)
- Retargeting effort: ~2 weeks
  - ADL-WCET component mapping
  - Handcrafted code
  - Simulation, estimation, and result analysis
- One-time effort: ~2 months
  - WCET component library
  - execution graph construction and WCET analysis
  - EXPRESSION-WCET interface

Effort on writing acesMIPS ADL description is not accounted

ADL Component	WCET Component	
	Input side	Output side
FetchUnit	Array	FIFO
DecodeUnit	FIFO	FIFO
OpReadUnit	Array/FIFO	FIFO
WBUnit	FIFO	Array

Benchmark	Simulation	Estimation	Ratio
fdct	10310	12310	1.19
fft	2937925	3880257	1.32
isort	149057	175724	1.18
matmul	57431	72150	1.26
matsum	442556	482570	1.09

Related Work

- A retargetable technique for predicting execution time of code segments
  Harmon, Baker, Whalley [Real-Time Systems, 1994]

- Retargatable static software timing analysis
  Chen, Malik, August [*ISSS* 2001]

- Efficient analysis of pipeline models for WCET computation
  Wilhelm [*WCET Workshop*, 2005]

Related Work [RTS 1994]

- Basic idea: micro-analysis
  - Instr => primitive operations
  - Timing analysis guided by pattern-driven timing rules
- Pros
  - Primitive ops are architecture-independent
- Cons
  - Timing rules are architecture-dependent
    => based on understanding of an architecture's timing model
  - Based on simpler architectures
    => not powerful enough for modern processor

Related Work [ISSS 2001]

- Basic idea: MESCAL and MADL
  - MESCAL => compiler and simulator
  - Obtain WCET of a code fragment by simulation
- Pros
  - Avoid most handcrafted code
- Cons
  - Simulation does not guarantee WCET
  - For VLIW with restrictions on pipeline complexities
- Comparison with our work
  - Retargetable simulation vs retargetable static analysis

Related Work [WCET 2005]

- Basic idea: HDL spec
  - Define pipeline analysis as computations on FSMs
  - With the help on Binary Decision Diagram (BDD)
- Pros
  - Automated analysis
- Cons
  - State explosion problem
- Comparison with our work
  - Based on higher level of abstraction
  - A compact pipeline model based on execution graph

Summary

- The major barrier of WCET analysis is retargetability
- Retargetable WCET analysis framework
  - Processor ADL specification as input
  - Execution graph based basic block timing analysis
  - WCET component library
  - ADL to execution graph
- Case study (acesMIPS)