AROMA: A Highly Accurate Microcomponent-based Approach for Embedded Processor Power Analysis

Zih-Ci Huang*, Chi-Kang Chen*§, Ren-Song Tsay*

*** National Tsing-Hua University, Taiwan**

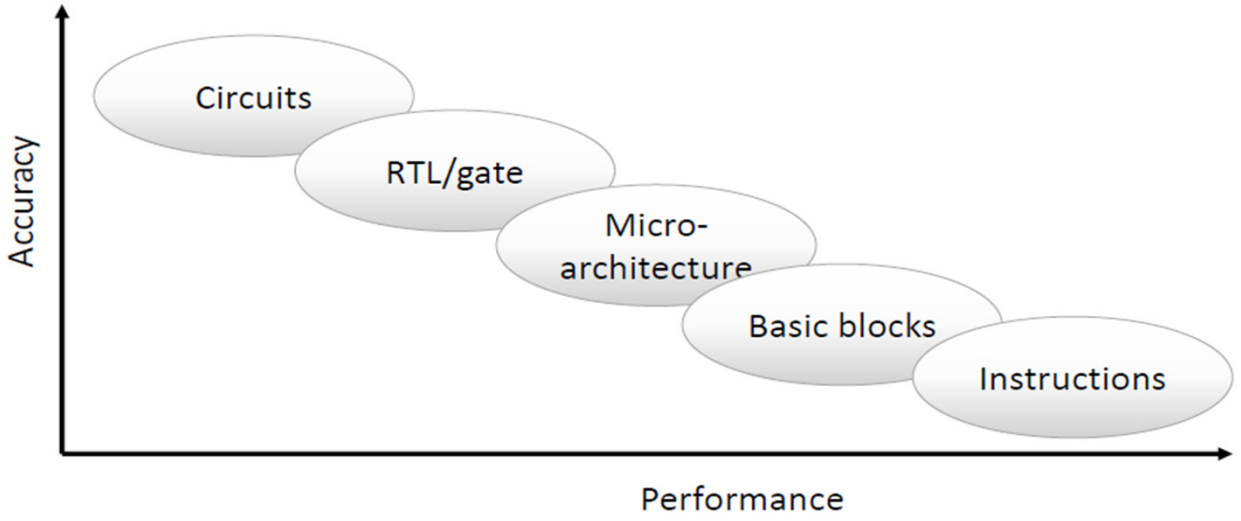
§ ITRI, Taiwan

Outline

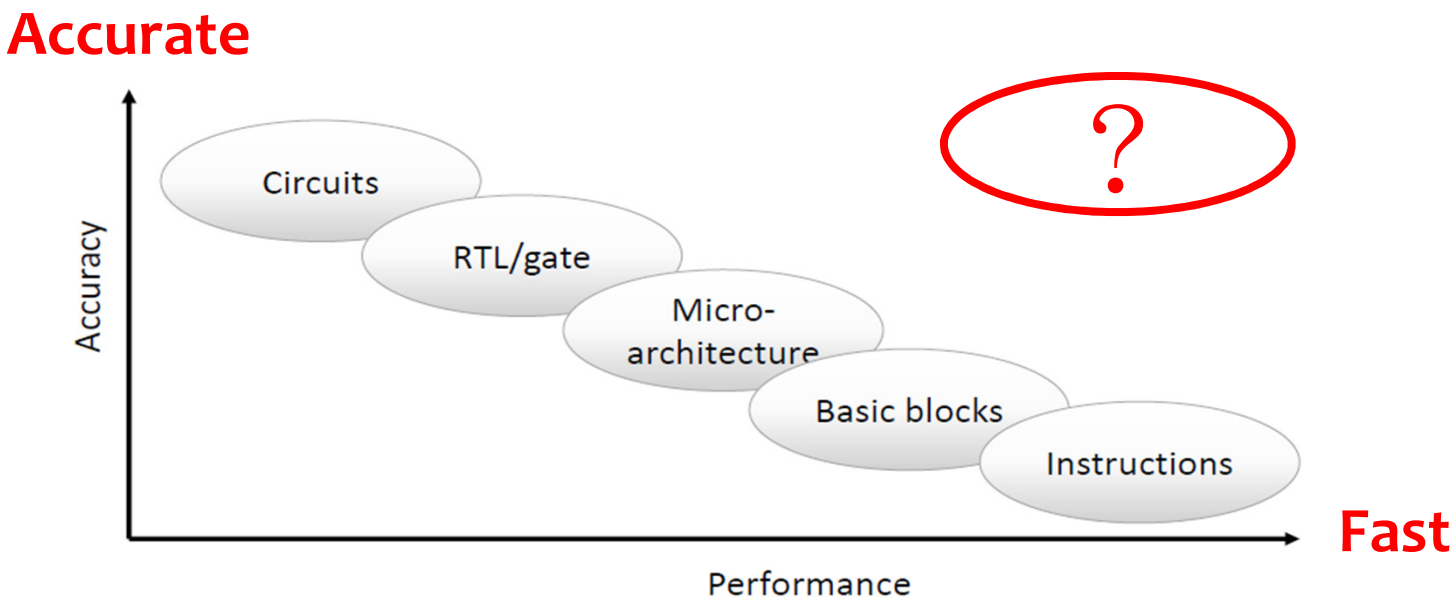
* Existing Power Estimation Issues
* Related Work
* AROMA
* Experimental Results
* Conclusions

Issues of Existing Approaches

* Either slow or inaccurate

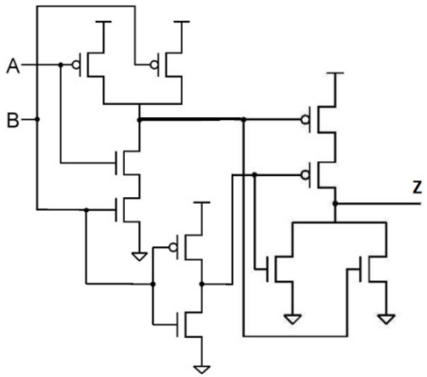


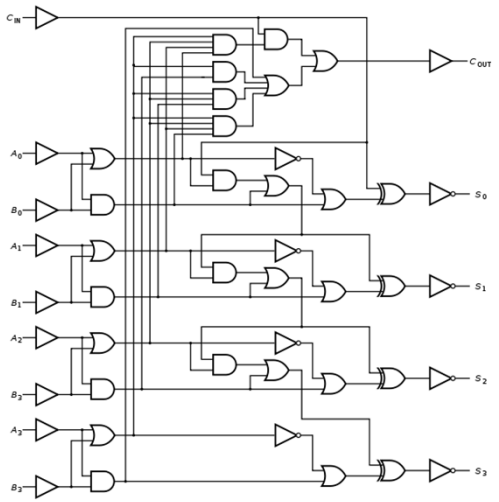
Can It Be Both Fast and Accurate ?



Related Work – Circuit

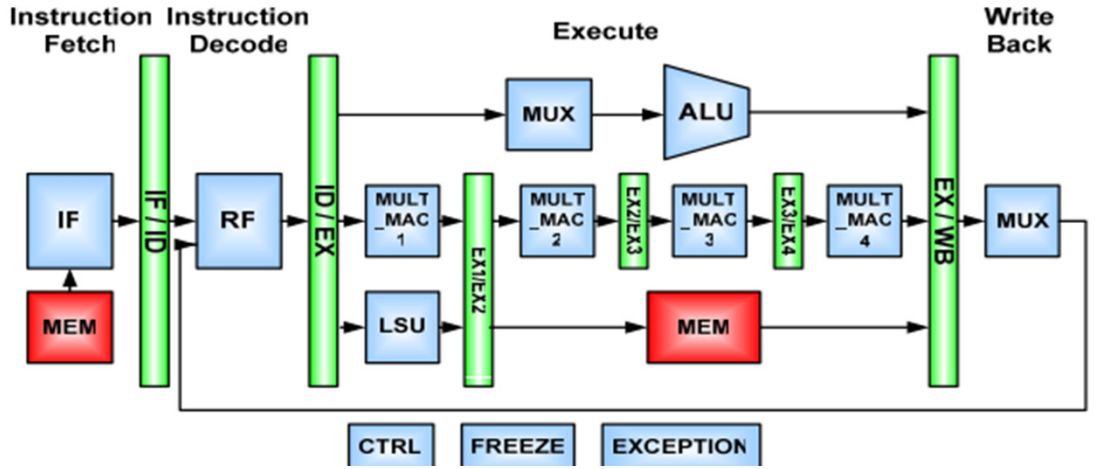
* Circuit-level / Gate-level / RTL power analysis
  * Very Accurate (golden result)
  * Extremely Slow (only good for simple/small designs)





Related Work - ALPA

* ALPA (architecture level power analysis) 2000 ~ 2008
  * Reasonably accurate using capacitive power models
  * Limited performance based on cycle-accurate simulator



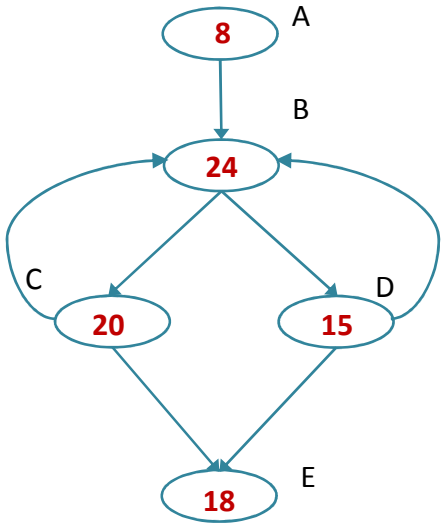
Related Work - ILPA

* ILPA (instruction level power analysis) 1994 ~
  * Pre-characterize power for each instruction
  * High Performance
    * Power = ∑ (Inst. Count)*(Power/Inst.)

Number	Instruction	Base Cost (mA)	Cycles
1	NOP	275.7	1
2	MOV DX,BX	302.4	1
3	MOV DX,[BX]	428.3	1
4	MOV DX,[BX][DI]	409.0	2
5	MOV [BX],DX	521.7	1
6	MOV [BX][DI],DX	451.7	2
7	ADD DX,BX	313.6	1
8	ADD DX,[BX]	400.1	2

Related Work – BB-based

* BB-based (basic-block based power analysis) 2011 ~ 2013
  * Pipeline effect can be captured
  * Higher performance

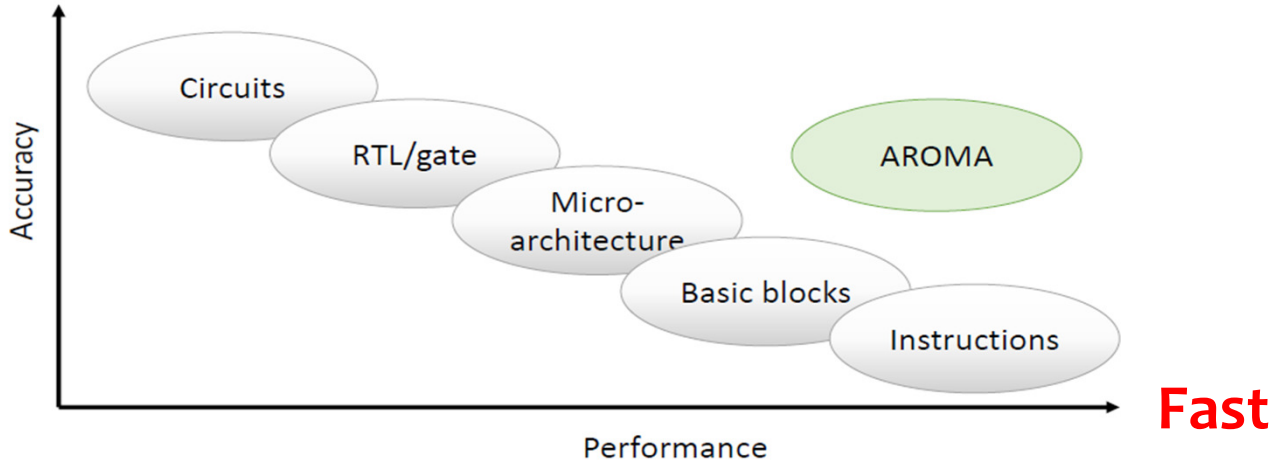


time	IF	ID	EXE	MEM	WB
1	i1				
2	i2	i1			
3	i3	i2	i1		
4	i4	i3	i2	i1	
5	i5	i4	i3	i2	i1
6	I5	I4	i3	*	i2
7	I5	I4	i3	*	*
8		i5	i4	i3	*
9			i5	i4	i3
10				i5	i4
11					i5

AROMA – Our Proposal

* Pre-characterized instruction-pair-based µ-component
* Instruction-µcomponent processor timing simulator

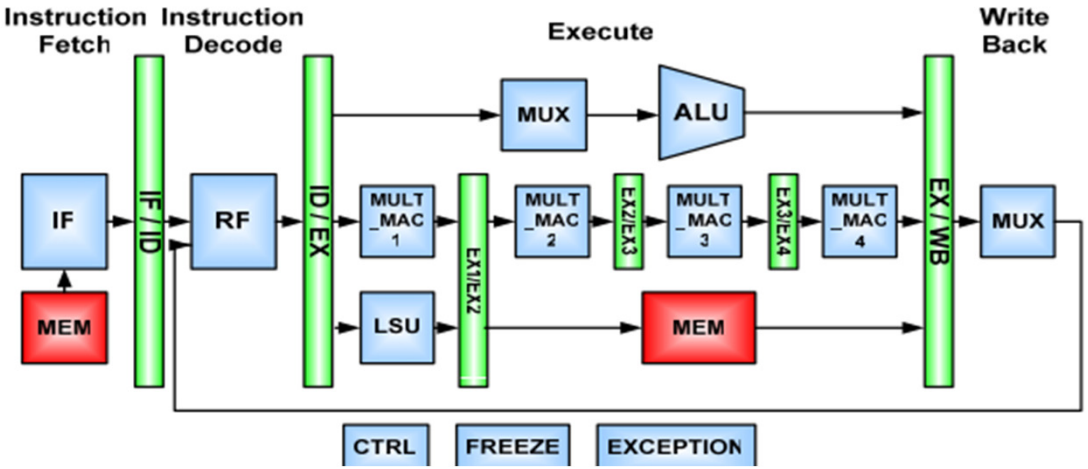
Accurate



Fast

µ-component-based Methodology

* Instruction follows a regular flow in pipeline
* Each instruction uses specific µ-components

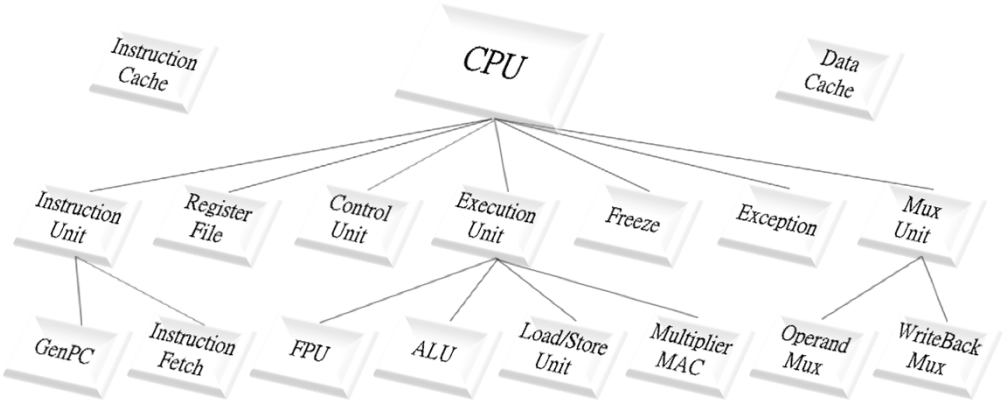


Instruction {µ-component}

add {Reg File, Ctrl, ALU, Mux}
mul {Reg File, Ctrl, Mult, Mux}

µ-component Identification

* µ-component input causes switching which consumes power
* Minimize variations of switching effect influence (inaccuracy)



Input Instruction

add {Reg File, Ctrl, ALU, Mux}
mul {Reg File, Ctrl, Mult, Mux}

Initial µ-component set

µc = {Reg File, Ctrl, ALU, Mux, Mult}

µ-component Identification

Initial µ-component set

µc = {Reg File, Ctrl, ALU, Mux, Mult}

After add inst.

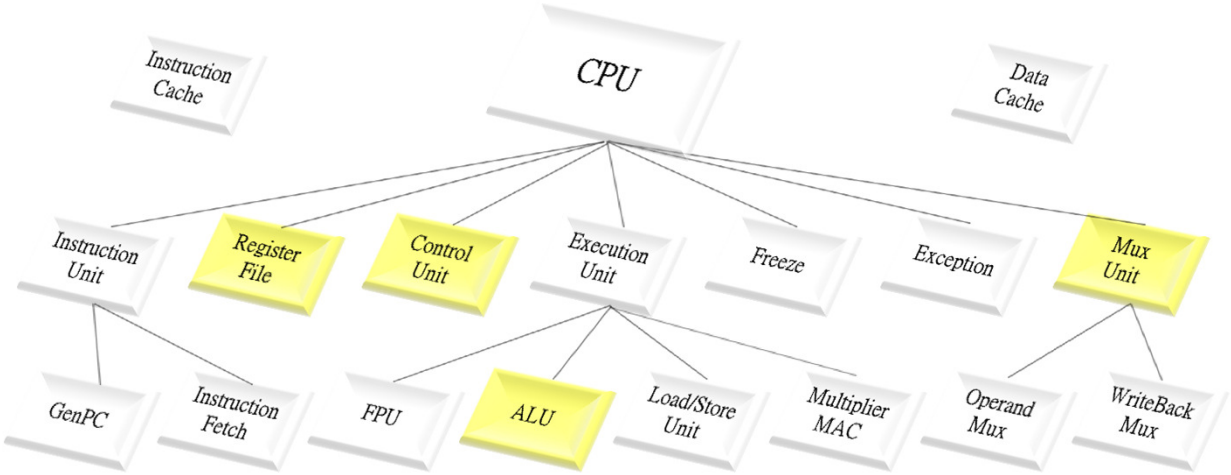
µc1 = {Reg File, Ctrl, ALU, Mux}

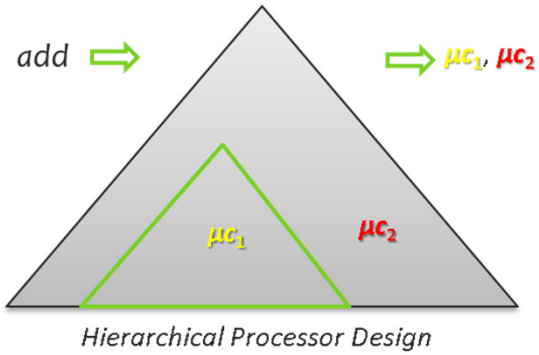
µc2 = {Mult} (The rest)

Input Instruction

add {Reg File, Ctrl, ALU, Mux}

mul {Reg File, Ctrl, Mult, Mux}





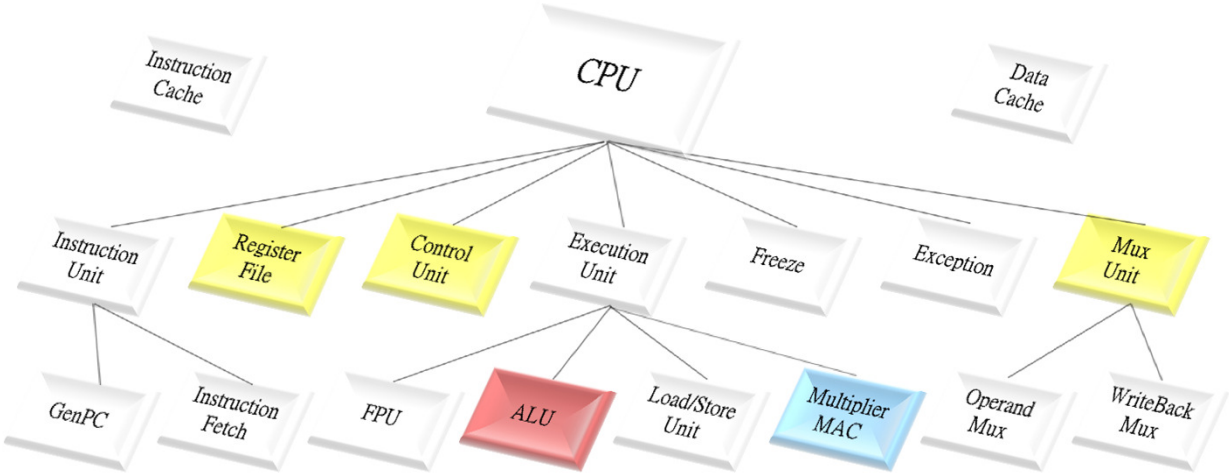
µ-component Identification

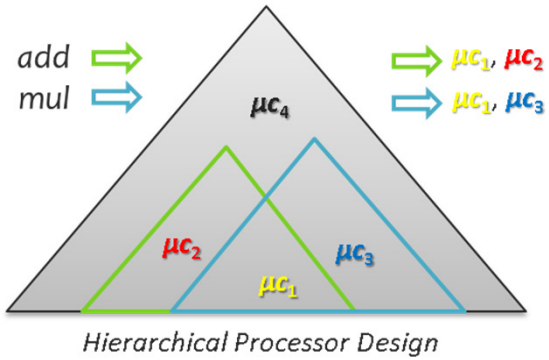
After mul inst.

µc1 = {Reg File, Ctrl, Mux}
µc2 = {ALU}
µc3 = {Mult}
µc4 = {} (The rest)

Input Instruction

add {Reg File, Ctrl, ALU, Mux}
mul {Reg File, Ctrl, Mult, Mux}





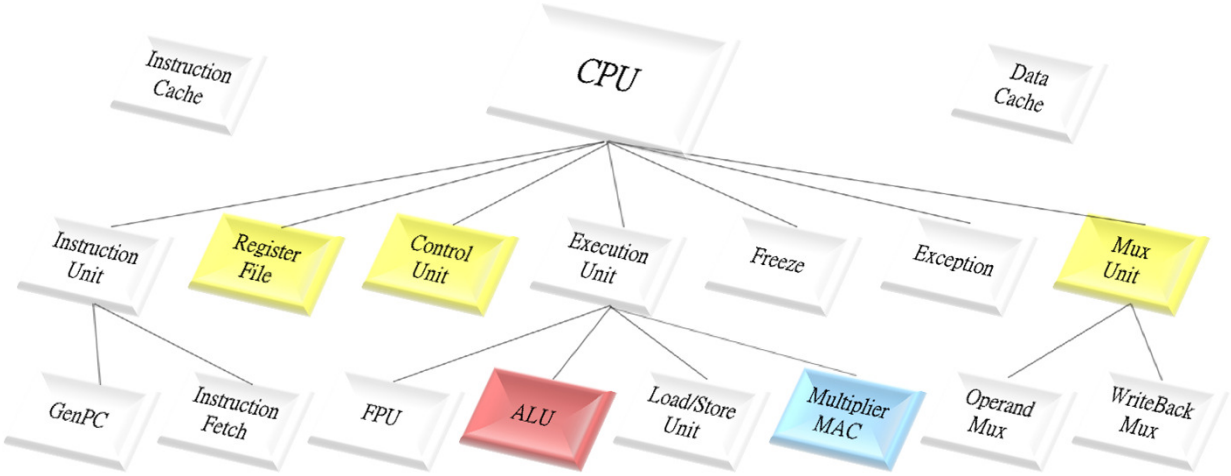
Hierarchical Processor Design

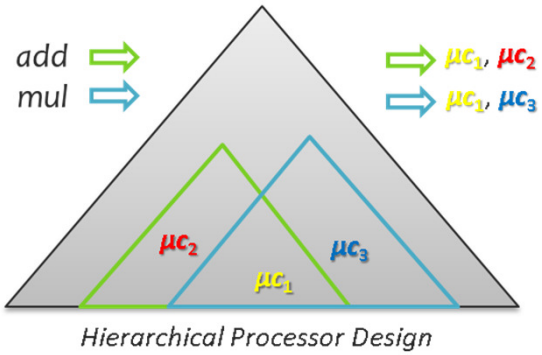
µ-component Identification

Output

µc1 = {Reg File, Ctrl, Mux}
µc2 = {ALU}
µc3 = {Mult}

add = {µc1, µc2 }
mul = {µc1, µc3 }

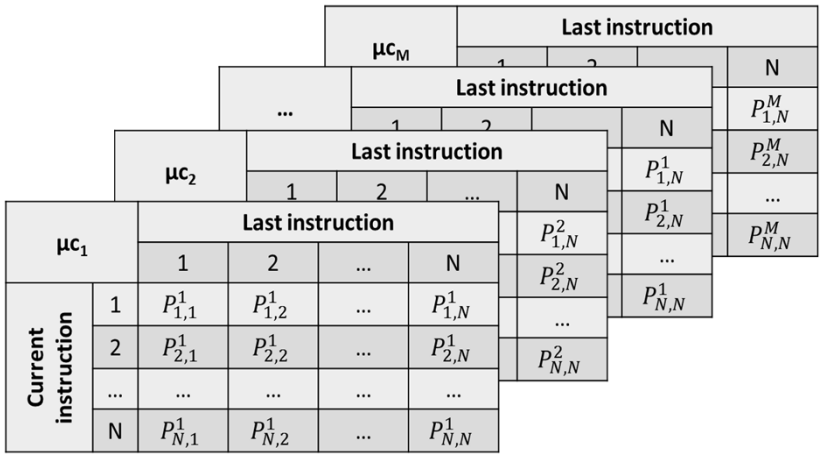




Hierarchical Processor Design

Power Table Construction

* Instruction → Control signals on each μ- component
* Consecutive instruction → Control signal transitions

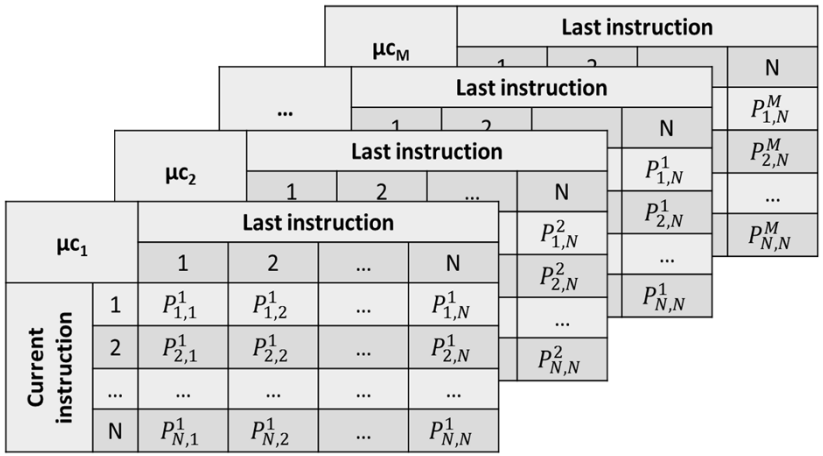


M: μ-component number

N: Instruction number

Power Table Construction

* Enumerate all potential consecutive instruction-pairs

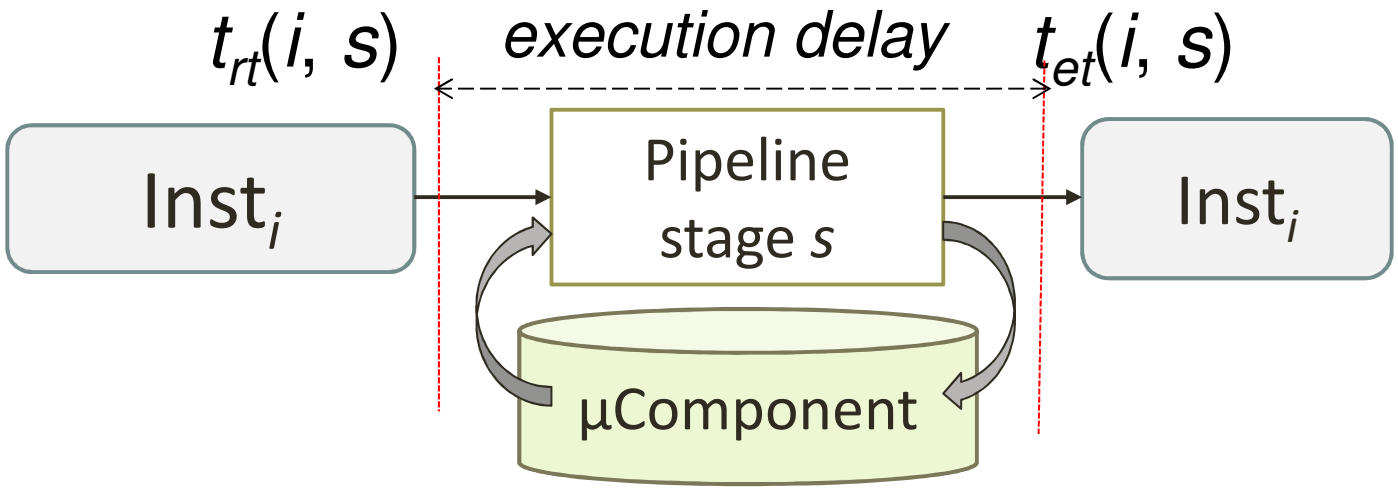


M: μ-component number

N: Instruction number

μ-component based Power Analysis

* Key idea is to know the timing of μ-component usage

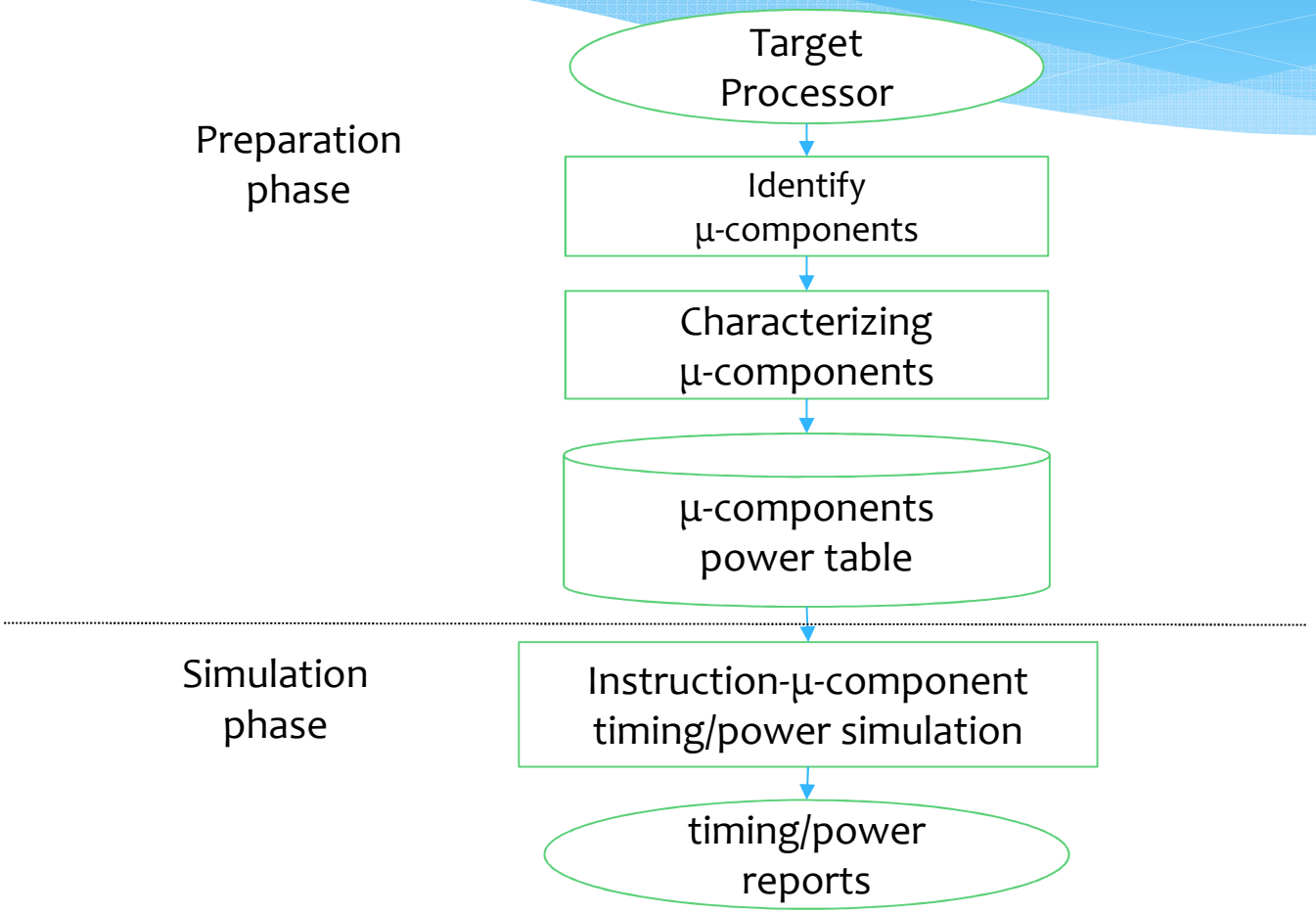


μ-component based Power Analysis

* Instruction execution → μ-component usage status
* Illustrative example:
  * add-mul → add-NOP + NOP mul

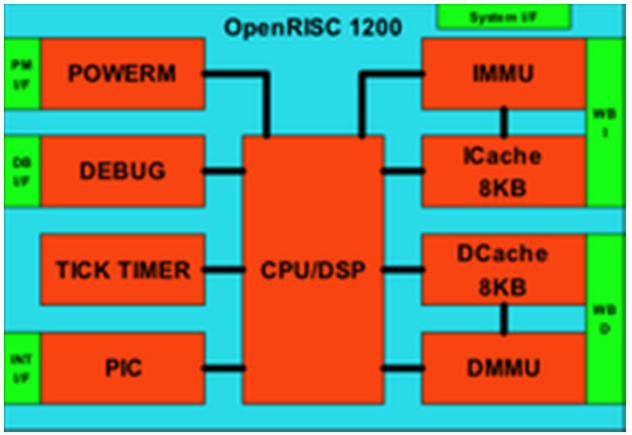
	IF	ID	EXE	WB	r1	r2	r3	r4	r5
add r3, r2, r1	1	2	3	4	-	-	2-4	-	-
mul r4, r3, r2	2-4	5	6-8	9	-	-	-	5-9	-
mul r5, r4, r3	5-9	10	11-12	13	-	-	-	-	10-13

µ-component-based Methodology



Experimental Environment

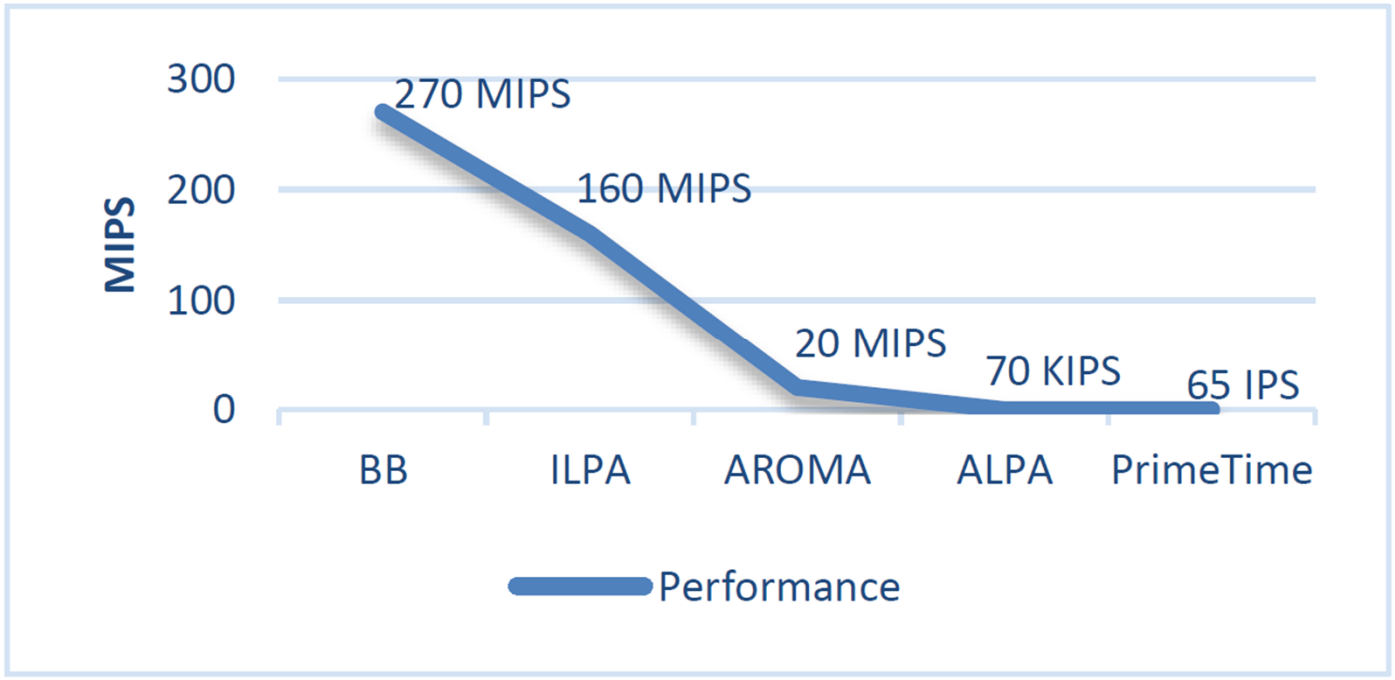
* Target processor
  * OpenRISC OR1200 processor (32-bit, 4-stages)
  * 0.18 um 6LM process
* Host machine
  * Intel Xeon 3.4 GHz dual-core
  * 2GB ram



Experimental Environment

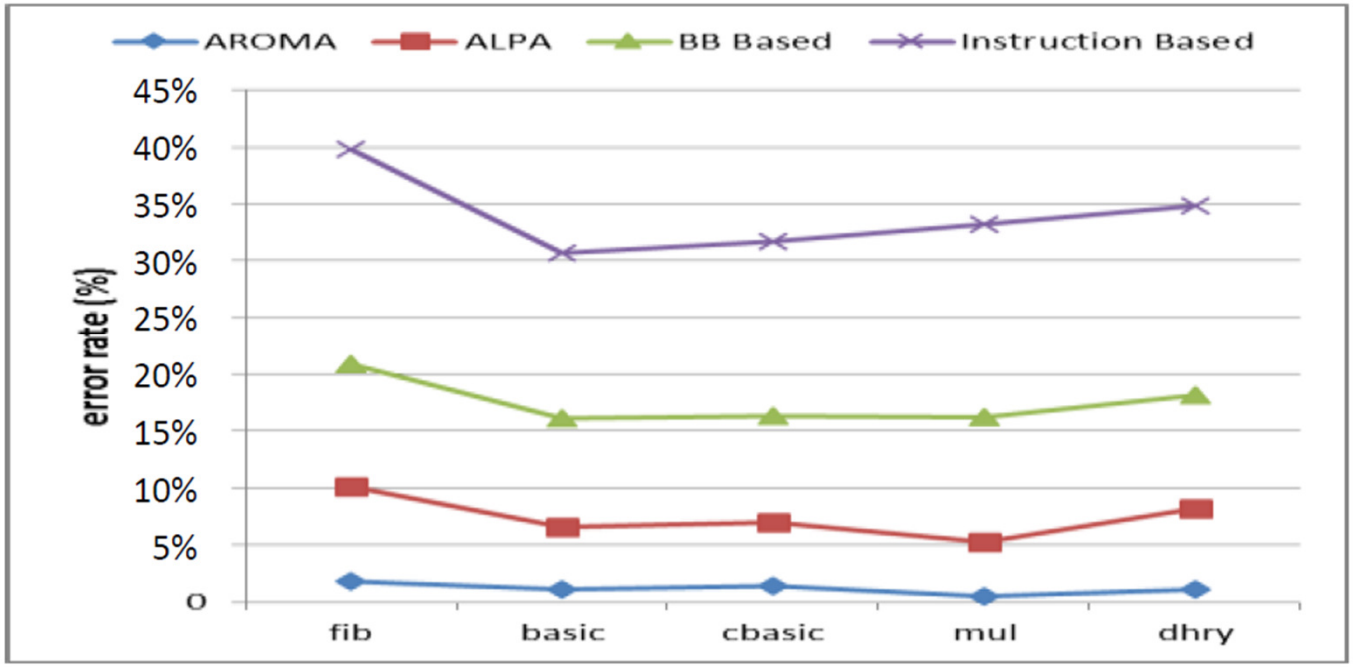
* Synthesizer
  * Design Compiler
* Power Characterization (Power Table Generation)
  * PrimeTime PX
* Benchmarks
  * Fibonacci series
  * OR1200 RISC Processor Benchmark:
    * basic, cbasic, mul, dhry

Performance Comparison



BB: Basic-block based Power Annotation
ILPA: Instruction Level Power Analysis
ALPA: Architecture Level Power Analysis

Accuracy Comparison



Conclusions

* Innovative power estimation approach
  * µ-component based
  * Capture CMOS switching activities
* Contribution of this work
  * Fast and very accurate
* Future Work
  * Extend the idea to bus, memory and others

Thanks

Q & A