Energy Minimization of Pipeline Processor Using a Low Voltage Pipelined Cache

Vincent J. Mooney III, Krishna Palem, Jun Cheol Park, and Kyu-won Choi

Georgia Institute of Technology

{mooney, palem, jcpark, kwchoi}@ece.gatech.edu

Outline

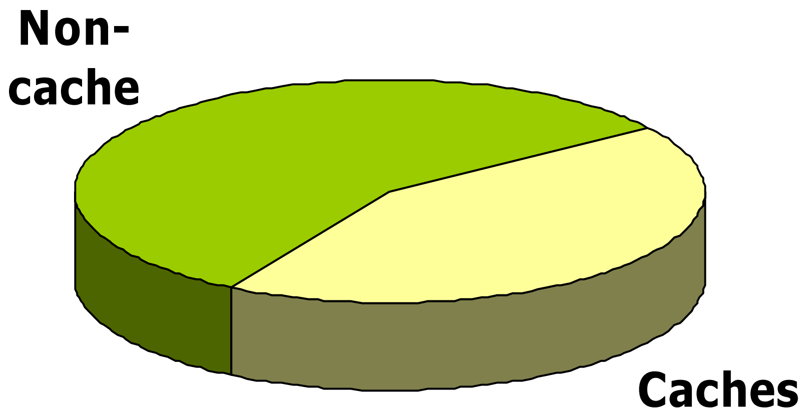
- Introduction
- Motivation and previous work
- Approach
- Methodology
- Results
- Conclusion and future work

Introduction

- Power/energy is a top most bottle neck in embedded systems
- Mobile devices require longer usage time
- Trade-off between performance and power
- Reducing power/energy without performance loss

Motivation & previous work

- A cache is a power hungry component of a system
- Caches consume 42% of a Strong ARM 110 processor*



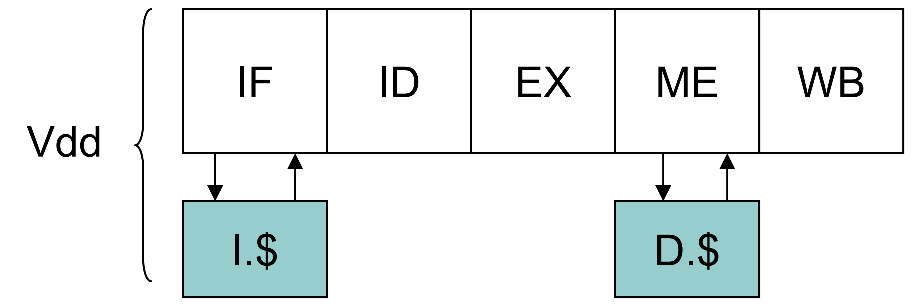
*J. Montanaro and et. al., “A 160-mhz, 32-b, 0.5-w cmos risc microprocessor,” *IEEE Journal of Solid-State Circuits*, 31(11):1703–1714, 1996.

Motivation & previous work

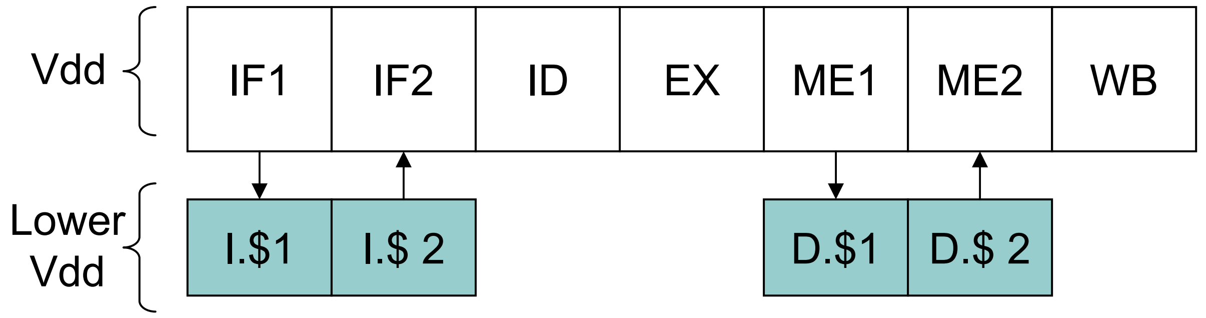
- Intel XScale processor supports multiple frequencies and voltages
  - L. T. Clarl and et. al., “An embedded 32-b microprocessor core for lowpower and high-performance applications,” IEEE Journal of Solid-State Circuits, 36(11):1599–1608, November 2001.
- High voltage supply for critical paths and low voltage supply for non-critical paths
  - V. Moshnyaga and H. Tsuji, “Cache energy resuction by dual voltage supply,” In Proc. Int. Symp. Circuit and System, pages 922–925, 2001.
- Pipelining a cache to achieve lower cycle time
  - T. Chappell, B. Chappell, S. Schuster, J. Allan, S. Klepner, R. Joshi, and R. Franch, “A 2-ns cycle, 3.8-ns access 512-kb cmos ecl sram with a fully pipelined architecture,” IEEE Journal of Solid-State Circuits, 26(11):1577–1585, 1991.

Approach

Case1. Non-pipelined caches with the same voltages as the processor



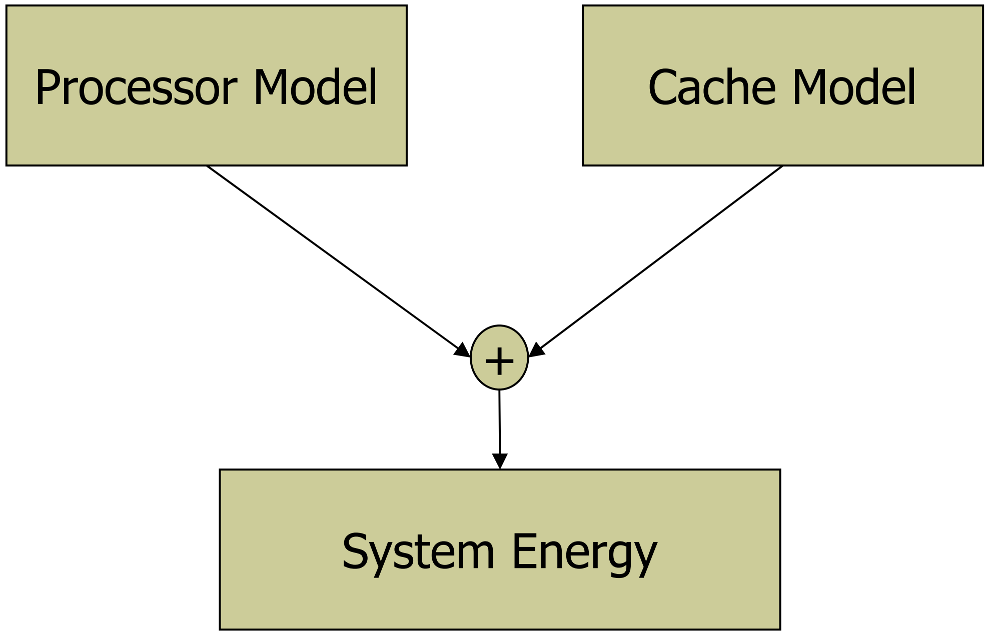
Case2. Caches pipelined with lower supply voltage and same cycle time with case1



Approach (Cont.)

- Case 2 uses same cycle time as case 1: ideally same execution time
- Case 2 saves power using lower supply voltage
- Two bottle necks
  - Branch penalty: branch misprediction adds overhead for pipelined instruction cache
  - Load use penalty: a load instruction immediately followed by dependent instruction adds overheads for pipelined data cache

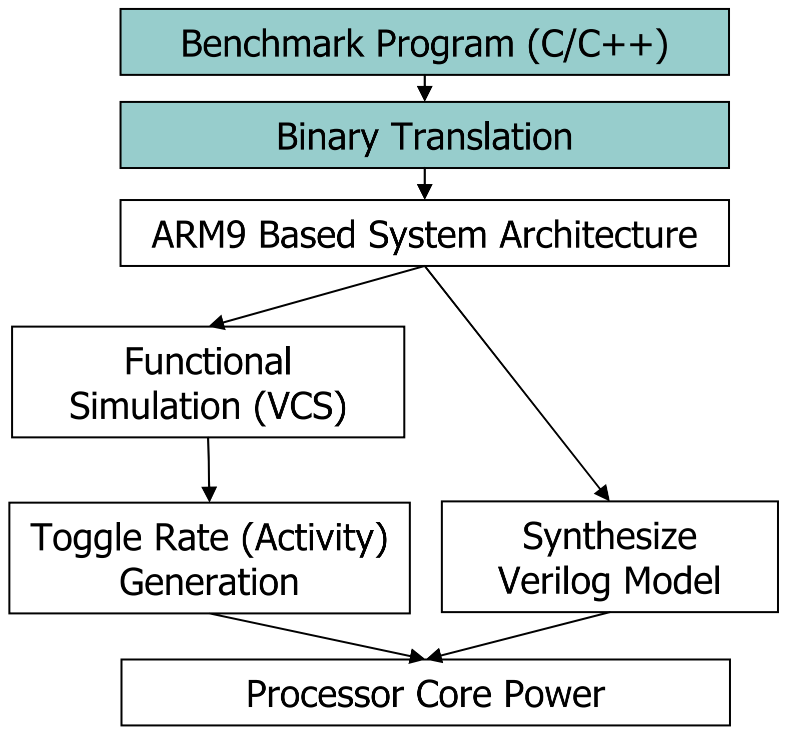
Methodology



Processor Model

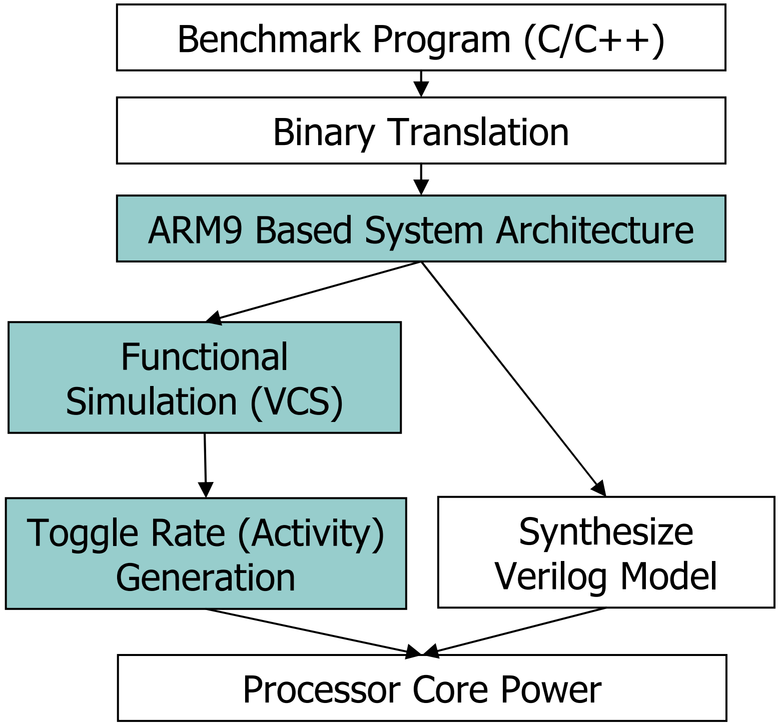
- MARS
  - A cycle-accurate Verilog model of a 5-stage RISC processor from U. Mich.
  - Capable of running ARM instruction
  - Non-pipelined caches
  - BTFN (backward taken forward non-taken) branch prediction
- MARS with 7-stage pipeline
  - 128 entry BTB (branch target buffer) with 2-bit counter
  - 2-stage IF (instruction fetch), 2-stage ME (memory access)

Processor Model (Cont.)


Benchmark Program (C/C++)
Binary Translation
ARM9 Based System Architecture
Functional Simulation (VCS)
Toggle Rate (Activity) Generation
Synthesize Verilog Model
Processor Core Power

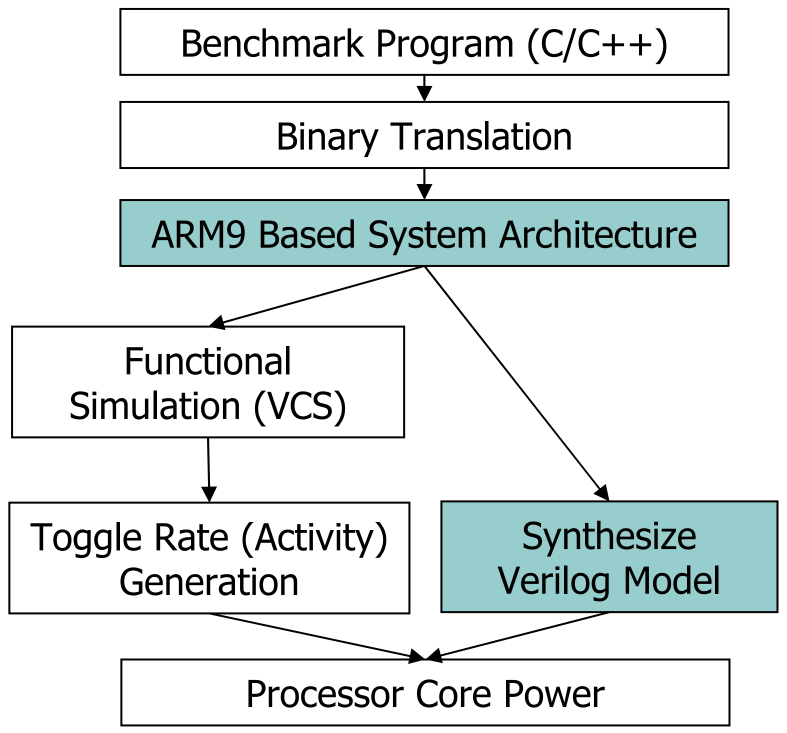
- Compile benchmarks using ARM-gcc compiler and generate hex ARM instructions called VHX

Processor Model (Cont.)


Benchmark Program (C/C++)
Binary Translation
ARM9 Based System Architecture
Functional Simulation (VCS)
Toggle Rate (Activity) Generation
Synthesize Verilog Model
Processor Core Power

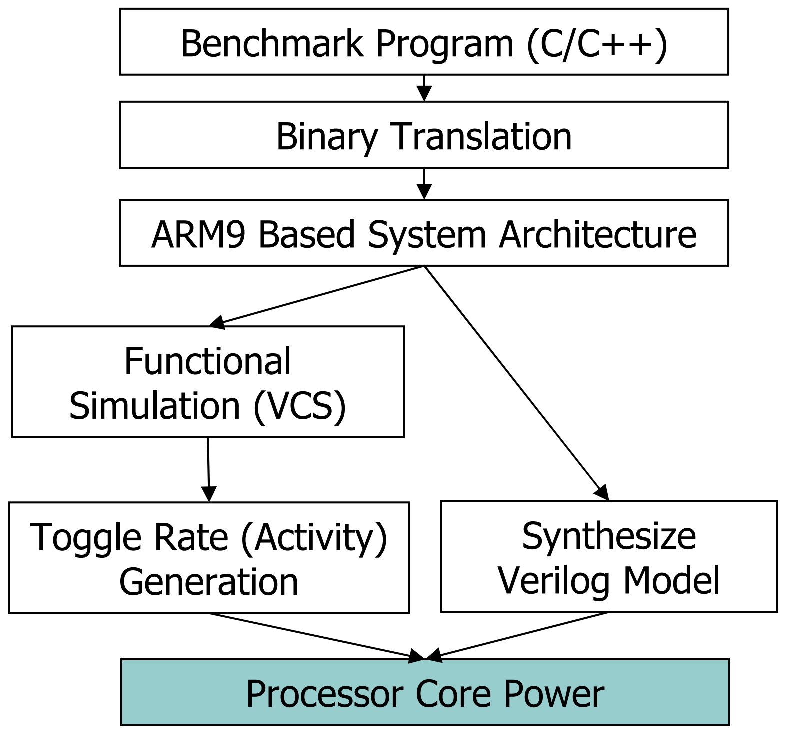
- Functional simulation using Synopsys VCS
- Collect toggle rate of internal logic signals using Synopsys VCS simulation

Processor Model (Cont.)


Benchmark Program (C/C++)
Binary Translation
ARM9 Based System Architecture
Functional Simulation (VCS)
Toggle Rate (Activity) Generation
Synthesize Verilog Model
Processor Core Power

- Synthesize Verilog model using TSMC .25μ library

Processor Model (Cont.)


Benchmark Program (C/C++)
Binary Translation
ARM9 Based System Architecture
Functional Simulation (VCS)
Toggle Rate (Activity) Generation
Synthesize Verilog Model
Processor Core Power

- Estimate power using Synopsys Power Compiler

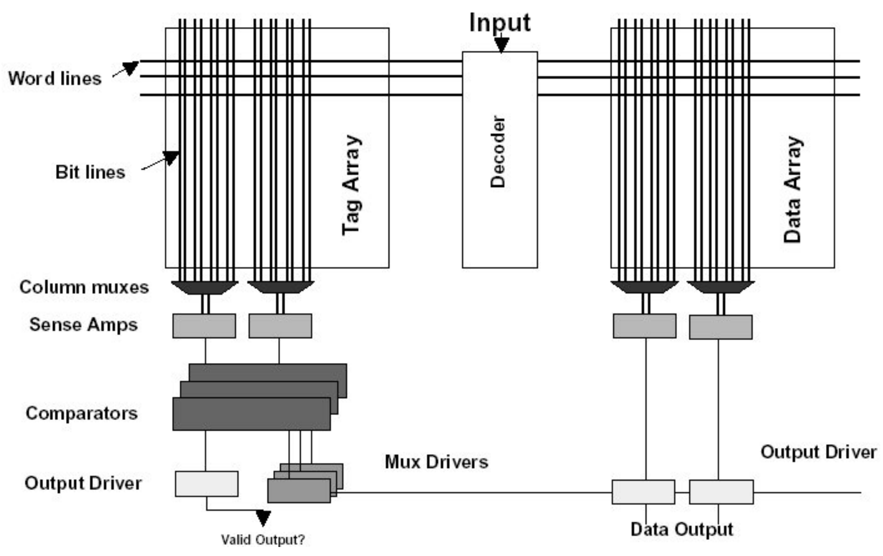
Cache model

- CACTI 2.0*
  - An integrated cache access time, cycle time, and power model
  - Time and power estimation of each component
  - RC based more detailed delay model used for technology scaling (i.e. supply voltage, threshold voltage)*

*G. Reinman and N. Jouppi, Cacti version 2.0, http://www.research.digital.com/wrl/people/jouppi/CACTI.html.
**N.Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison Wesley, Santa Clara, California, 1992.

Cache model (Cont.)

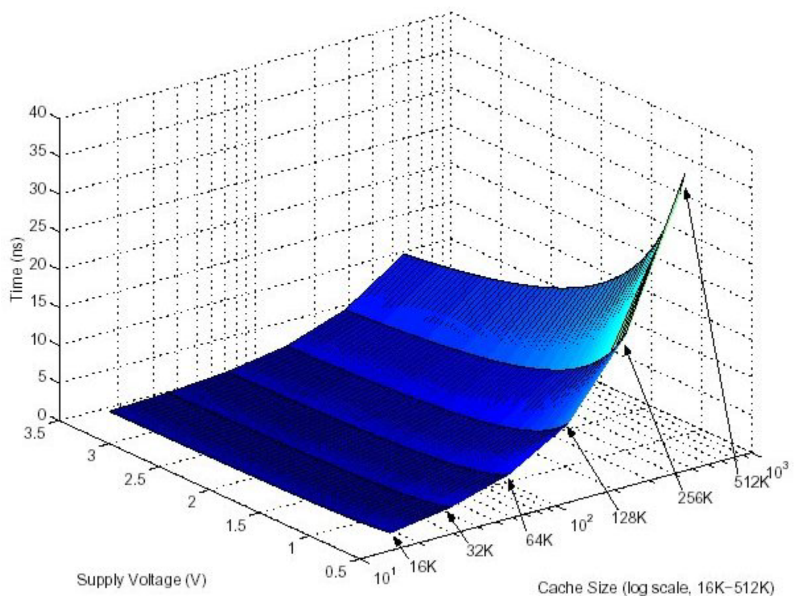
CACTI 2.0 cache model



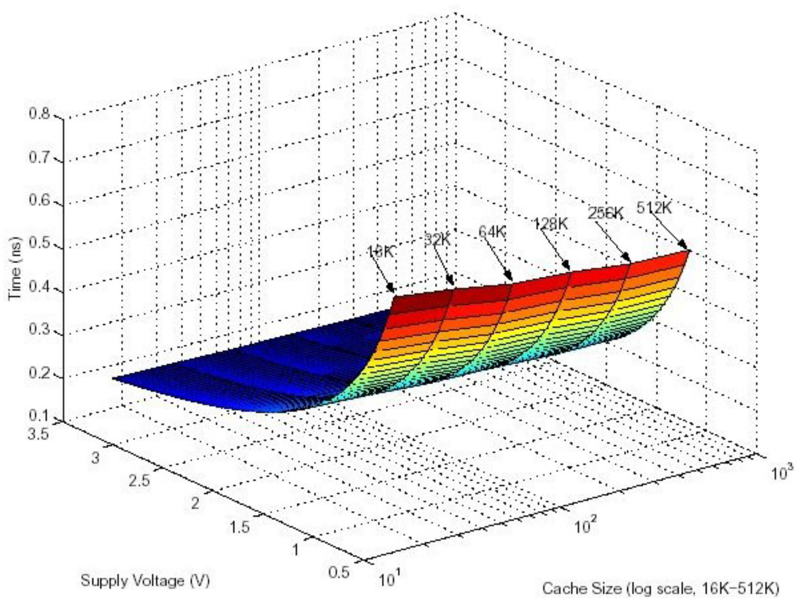
- The cache circuit is split into two parts for pipelining
  - Pipeline stage 1: decoder, tag array, data array
  - Pipeline stage 2: mux, sense-amplifier, comparator
- Timing order of the circuit-level critical path considered
- Direct mapped and 32B block size
- 16KB, 32KB, 64KB, 128KB, 256KB, 512KB cache size simulated

Cache model (Cont.)

Delay for Pipeline 1
(Cache Size: 16K-512K, Block Size: 32, Direct Mapped)



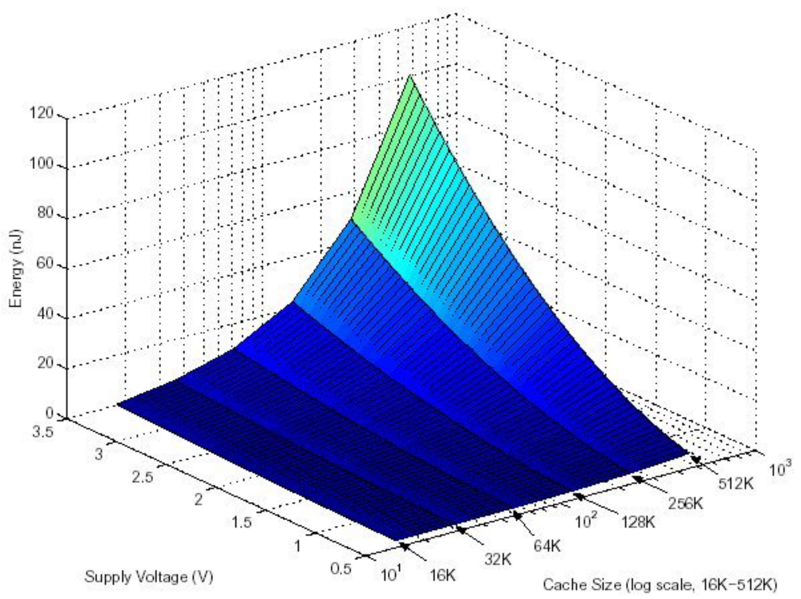
Delay for Pipeline 2
(Cache Size: 16K-512K, Block Size: 32, Direct Mapped)



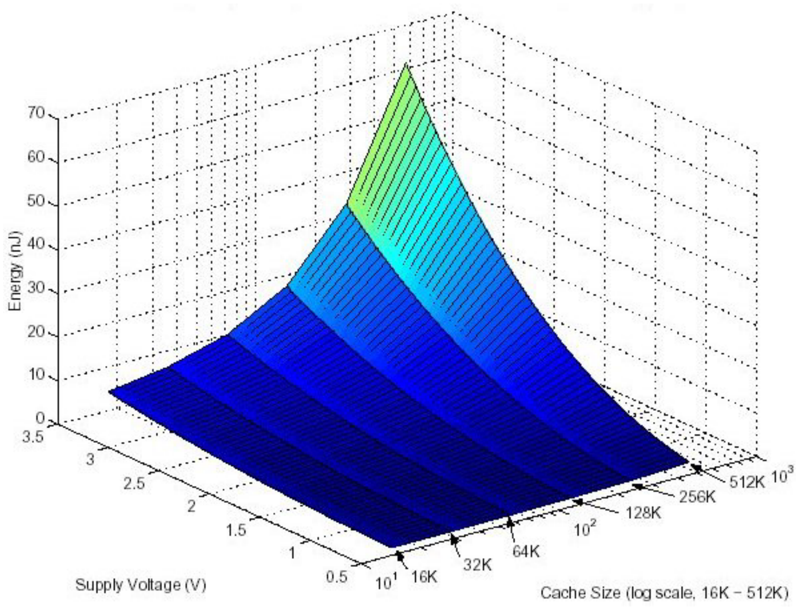
- Delay is increased according to the supply voltage
- Delay of the pipeline stage 1 is also dependent on the cache size

Cache model (Cont.)

Energy for Pipeline 1
(Cache Size: 16K-512K, Block Size: 32, Direct Mapped)



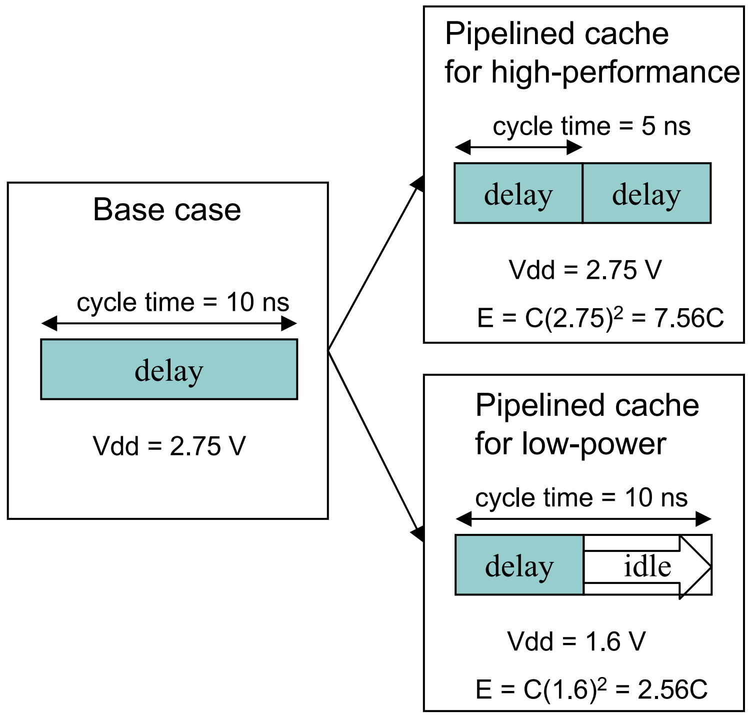
Energy for Pipeline 2
(Cache Size: 16K-512K, Block Size: 32, Direct Mapped)



- Energy is dependent on the cache size and the supply voltage

Description	Parameters
Cache size	16KB, 32KB, 64KB, 128KB, 256KB, 512KB
Block size	32bytes
Associativity	directed mapped
Number of sets	512, 1K, 2K, 4K, 8K, 16K
Number of segments per word line (data)	1
Number of segments per bit line (data)	1
Number of segments per word line (tag)	1
Number of segments per bit line (tag)	1
Number of rows in a subarray	512, 1K, 2K, 4K, 8K, 16K
Number of columns in a subarray	256

Optimization of energy and delay



Energy savings = (7.56 – 2.56)C/7.56*100 = 66%

- Pipelined cache for high-performance
  - Reduced cycle time with same supply voltage
- Pipelined cache for low-power
  - Reduced supply voltage without changing cycle time

Optimization of energy and delay (Cont.)

- Optimized supply voltage for cache

Voltage optimization procedure for pipelined cache

```
Input: Vdd Range, delay_base
Output: Power optimal Vdd
Vdd Range ← [2.75V – 0.6V]
Vdd(0) = Max(Vdd Range);
For i steps do
        Calculate delay_stage1(Vdd(i));
        Calculate delay_stage2(Vdd(i));
        If Max[delay_stage1{Vdd(i)}, delay_stage2{Vdd(i)}] < dealy_base
                Vdd_optimal = Vdd(i);
        endIf
        Decrease Vdd(i);
endFor
```

Optimization of energy and delay (Cont.)

- Pipelined cache saves maximum 69.60% energy saving

Energy/delay for a pipelined cache

	Base case			Pipelined cache				
Cache(KB)	Vdd(V)	Delay(nS)	Energy(nJ)	Delay1(nS)	Delay2(nS)	Vdd(V)	Energy(nJ)	% saving
16	2.75	0.648	5.689	0.438	0.210	1.6	1.729	69.60
32	2.75	1.021	9.019	0.814	0.206	2	4.534	49.73
64	2.75	1.741	15.357	1.540	0.201	2.3	10.450	31.95
128	2.75	3.190	27.942	2.991	0.199	2.5	22.767	18.52
256	2.75	6.254	54.605	6.060	0.195	2.65	50.442	7.62
512	2.75	12.224	105.477	12.030	0.194	2.7	101.422	3.84

Results

- Execution time increased 15.35% due to the branch misprediction penalty and load use penalty
  - More accurate branch prediction scheme required
  - Dynamic instruction scheduling such as out-of-order execution or static instruction scheduling such as compiler optimization required

Execution Time (ICache=16KB, DCache=16KB)

			Base case		Pipelined cache processor		
Benchmark	Misprediction	Load use	E.T(ns)	Core Power(mW)	E.T(ns)	Core Power(mW)	E.T.% Increment
sort_int	177	201	26595	1002	31465	1008	18.31
matmul	604	512	90485	1114	105293	1121	16.36
arith	105	151	43765	1079	47987	1086	9.65
factorial	4	1002	192345	981	221196	987	15.00
fib	125	178	40635	1057	47719	1063	17.43
Average							15.35

Results (Cont.)

- Average 24.85% power saving
- Processor core power does not change much for 5-stage and 7-stage
- Variation of total processor power is mainly dependent on cache power

Power distribution (ICache=16KB, DCache=16KB)

	Base case (mW)				Pipelined cache (mW)				
Benchmark	Core Power	I. Cache	D. Cache	Total	Core Power	I. Cache	D. Cache	Total	% Reduction
sort_int	1002	411	98	1511	1008	120	25	1154	23.67
matmul	1114	450	142	1706	1121	134	37	1292	24.27
arith	1079	488	66	1634	1086	154	18	1258	22.96
factorial	981	475	118	1574	987	143	31	1161	26.24
fib	1057	513	149	1719	1063	151	39	1253	27.09
Average									24.85

Results (Cont.)

- Average 13.33% energy saving
- The increment of execution time degrades the energy reduction
- To maximize the advantage of pipelined cache, a precise branch prediction scheme and instruction scheduler (load use) required

Energy distribution (ICache=16KB, DCache=16KB)

	Base case (nJ)				Pipelined cache (nJ)				
Benchmark	Core Energy	I. Cache	D. Cache	Total	Core Energy	I. Cache	D. Cache	Total	% Reduction
sort_int	26660	10929	2606	40195	31715	3789	794	36298	9.70
matmul	100830	40723	12823	154377	118034	14118	3898	136050	11.87
arith	47238	21363	2896	71496	52105	7406	880	60392	15.53
factorial	188628	91328	22791	302747	218220	31662	6928	256810	15.17
fib	42953	20845	6053	69851	50743	7227	1840	59810	14.38
Average									13.33

Conclusion and future work

- Pipelined cache with lower supply voltage explored
- Maximum 69.6% cache energy saving
- 24.85% power and 13.33% energy saved
- The savings of the power are masked by the execution time increment
- Branch prediction and load use penalty must be considered to maximize energy saving

Thank you.