Sleepy stack: a New Approach to Low Power VLSI Logic and Memory

Ph.D. Dissertation Defense
by
Jun Cheol Park
Advisor: Vincent J. Mooney III

School of Electrical and Computer Engineering
Georgia Institute of Technology

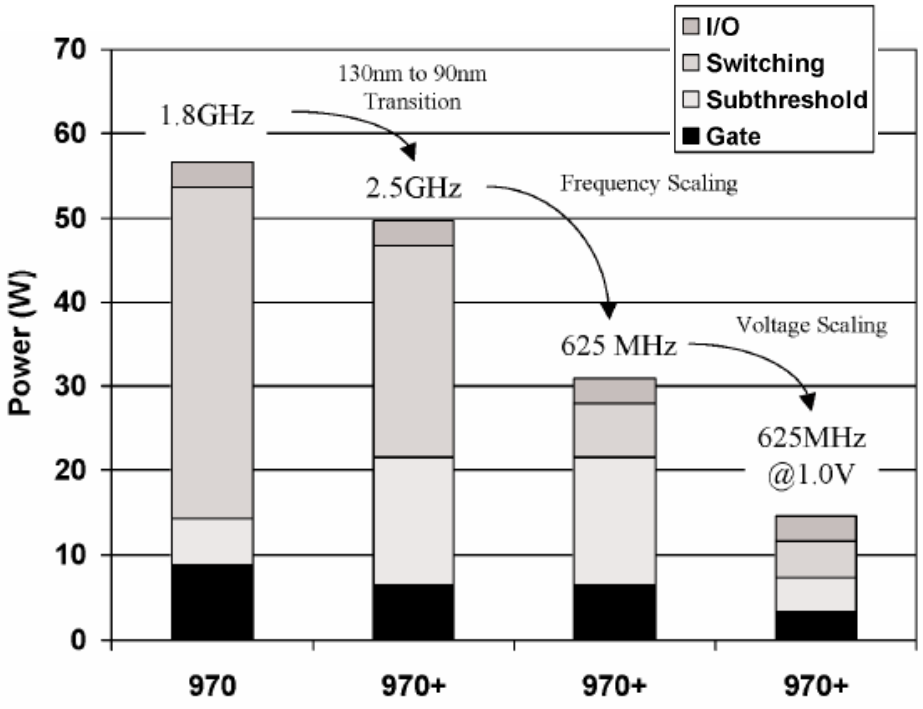
June 2005

Outline

- Introduction
- Related Work
- Sleepy stack
- Sleepy stack logic circuits
- Sleepy stack SRAM
- Low-power pipelined cache (LPPC)
- Sleepy stack pipelined SRAM
- Conclusion

Power consumption

- Power consumption of VLSI is a fundamental problem of mobile devices as well high-performance computers
  - Limited operation (battery life)
  - Heat
  - Operation cost
- Power = dynamic + static
  - Dynamic power more than 90% of total power (0.18u tech. and above)
- Dynamic power reduction:
  - Technology scaling
  - Frequency scaling
  - Voltage scaling



IBM PowerPC 970*

*N. Rohrer et al., “PowerPC 970 in 130nm and 90nm Technologies," *IEEE International Solid-State circuits Conference,* vol 1, pp. 68-69, February 2004.

Motivation

- Ultra-low leakage power
  - A cell phone calling plan with 500min (per month 43,200min)
  - Leakage reduction technique can potentially increase battery life 34X
- State-saving
  - Prior ultra-low-leakage techniques (e.g., sleep transistor) lose logic state thus requires long wake-up time
  - Users can use a cell phone without waiting long wake-up time
    - Emergency calling situation

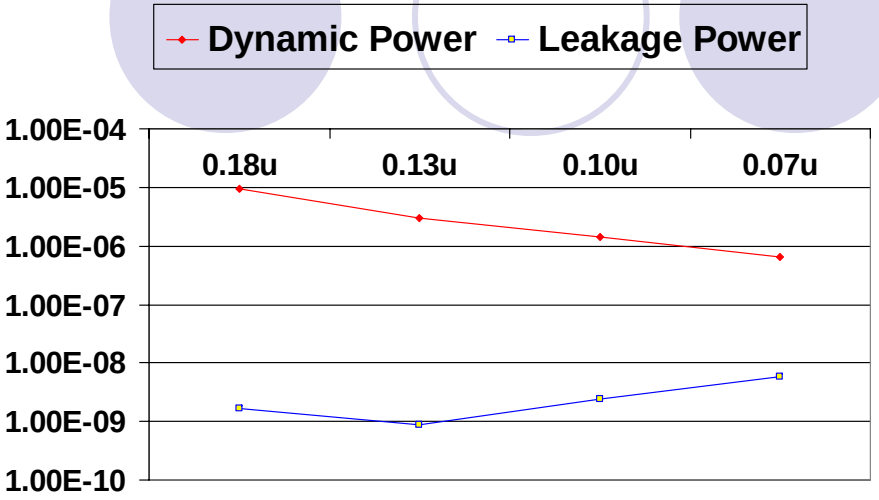
*Assume chip area ½ processor logic and ½ memory, no on-chip memory, leakage power matches to dynamic power at 0.07u tech.

	Best prior work (forced stack)			Our approach (sleepy stack)		
	Active (W)	Leakeage (W)	Energy (J) (Month)	Active (W)	Leakage (W)	Energy (J) (Month)
Proc.	1.71E-01	1.34E-01	3.49E+05	1.82E-01	7.57E-04	7.39E+03
32KB SRAM	3.83E-02	6.99E-02	1.80E+05	4.10E-02	2.74E-03	8.26E+03
Total	2.09E-01	2.04E-01	5.30E+05	2.23E-01	3.50E-03	1.56E+04

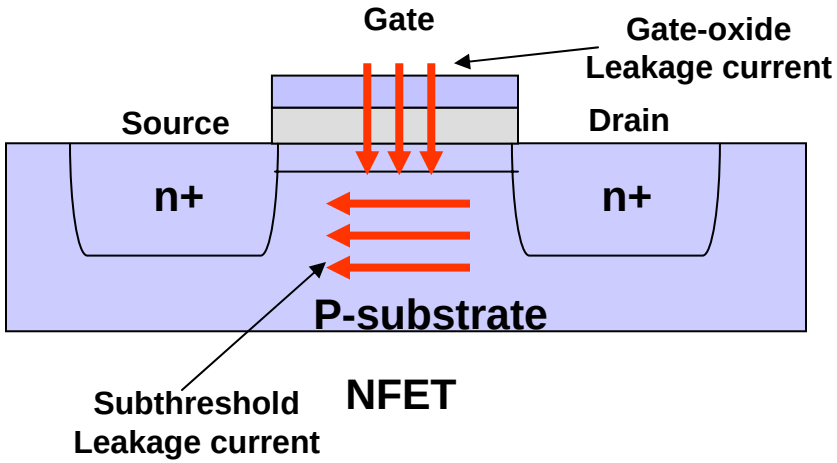
Power consumption scenario for 0.07u tech, processor

Leakage power

- Leakage power became important as the feature size shrinks
- Subthreshold leakage
  - Scaling down of V_{th}: Leakage increases exponentially as V_{th} decreases
  - Short-channel effect: channel controlled by drain
  - Our research focus
- Gate-oxide leakage
  - Gate tunneling due to thin oxide
  - High-k dielectric could be a solution



Experimental result 4-bit adder*



*Berkeley Predictive Technology Model (BPTM). [Online]. Available http://www-device.eecs.berkeley.edu/~ptm.

Contribution

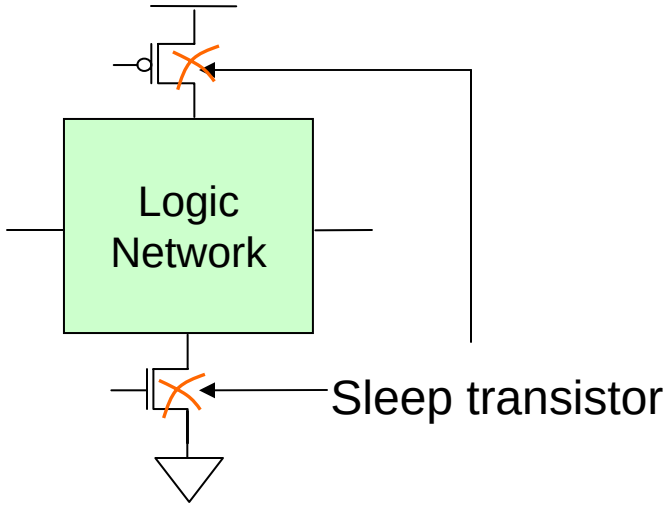
- Design of novel ultra-low leakage sleepy stack which savies state
- Design of sleepy stack SRAM cell which provides new pareto points in ultra-low leakage power consumption
- Design of low-power pipelined cache and find optimal number of pipeline stages in the given architecture
- Design of sleepy stack pipelined SRAM for low-leakage

Outline

- Introduction
- Related Work
- Sleepy stack
- Sleepy stack logic circuits
- Sleepy stack SRAM
- Low-power pipelined cache (LPPC)
- Sleepy stack pipelined SRAM
- Conclusion

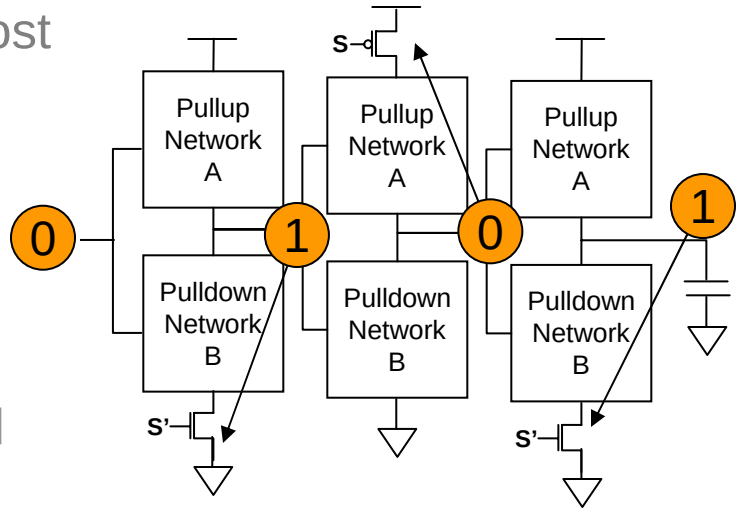
Low-leakage CMOS VLSI

- Sleep transistor
  - Multi-threshold-voltage CMOS (MTCMOS) [Mutoh95]
  - Loses state and incurs high wake-up cost
- Zigzag [Min03]
  - Non-floating by asserting a predefined input during sleep mode
  - Only retain predefined state
- Stack [Narendra01]
  - Stack effect: when two or more stacked transistors turned off together, leakage power is suppressed
  - Forcing stack
  - Cannot utilize high-V_{th} without huge delay penalties (>6.2X) (we will show for less, e.g., 2.8X)



Low-leakage CMOS VLSI

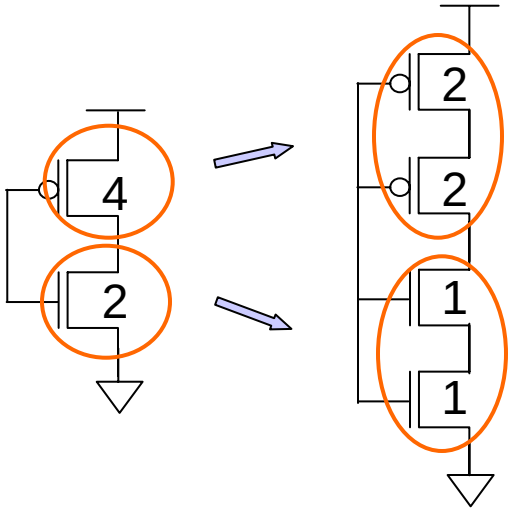
- Sleep transistor
  - Multi-threshold-voltage CMOS (MTCMOS) [Mutoh95]
  - Loses state and incurs high wake-up cost
- Zigzag [Min03]
  - Non-floating by asserting a predefined input during sleep mode
  - Only retain predefined state
- Stack [Narendra01]
  - Stack effect: when two or more stacked transistors turned off together, leakage power is suppressed
  - Forcing stack
  - Cannot utilize high-V_{th} without huge delay penalties (>6.2X) (we will show for less, e.g., 2.8X)



Zigzag

Low-leakage CMOS VLSI

- Sleep transistor
  - Multi-threshold-voltage CMOS (MTCMOS) [Mutoh95]
  - Loses state and incurs high wake-up cost
- Zigzag [Min03]
  - Non-floating by asserting a predefined input during sleep mode
  - Only retain predefined state
- Stack [Narendra01]
  - Stack effect: when two or more stacked transistors turned off together, leakage power is suppressed
  - Forcing stack
  - Cannot utilize high-V_{th} without huge delay penalties (>6.2X) (we will show for less, e.g., 2.8X)



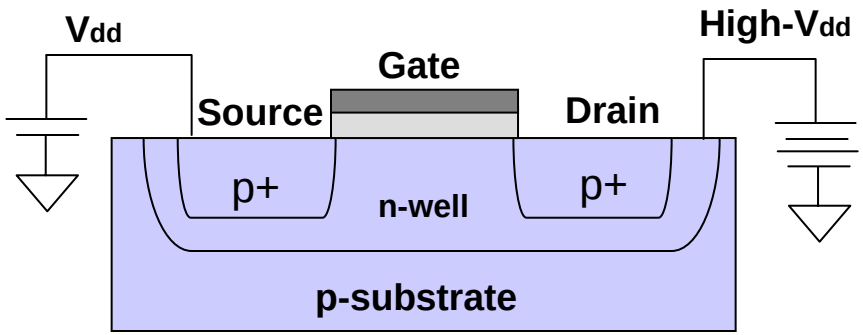
Forcing stack

Low-leakage CMOS VLSI comparison summary

- Sleepy stack approaches for logic circuits
  - Saves exact state, so does not need to restore original state (unlike the sleep transistor technique and the zigzag technique)
  - Larger leakage power saving (forced stack)

Low-leakage SRAM

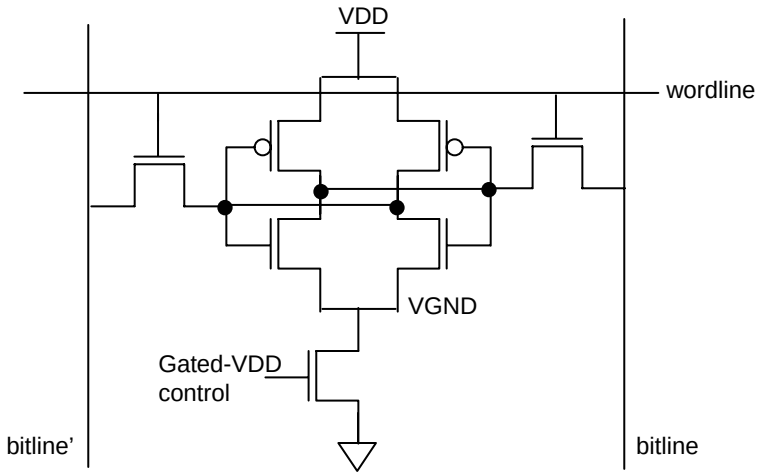
- Auto-Backgate-Controlled Multi Threshold CMOS (ABC-MTCMOS) [Nii98]
  - Reverse source-body bias during sleep mode
  - Slow transition and large dynamic power to charge n-wells
- Gated-Vdd [Powell00](Prof. K. Roy)
  - Isolate SRAM cells using sleep transistor
  - Loses state during sleep mode
- Drowsy cache [Flautner02]
  - Scaling Vdd dynamically
  - Smaller leakage reduction (<86%) (we will show 3 orders magnitude reduction)



ABC-MTCMOS

Low-leakage SRAM

- Auto-Backgate-Controlled Multi Threshold CMOS (ABC-MTCMOS) [Nii98]
  - Reverse source-body bias during sleep mode
  - Slow transition and large dynamic power to charge n-wells
- Gated-Vdd [Powell00](Prof. K. Roy)
  - Isolate SRAM cells using sleep transistor
  - Loses state during sleep mode
- Drowsy cache [Flautner02]
  - Scaling Vdd dynamically
  - Smaller leakage reduction (<86%) (we will show 3 orders magnitude reduction)

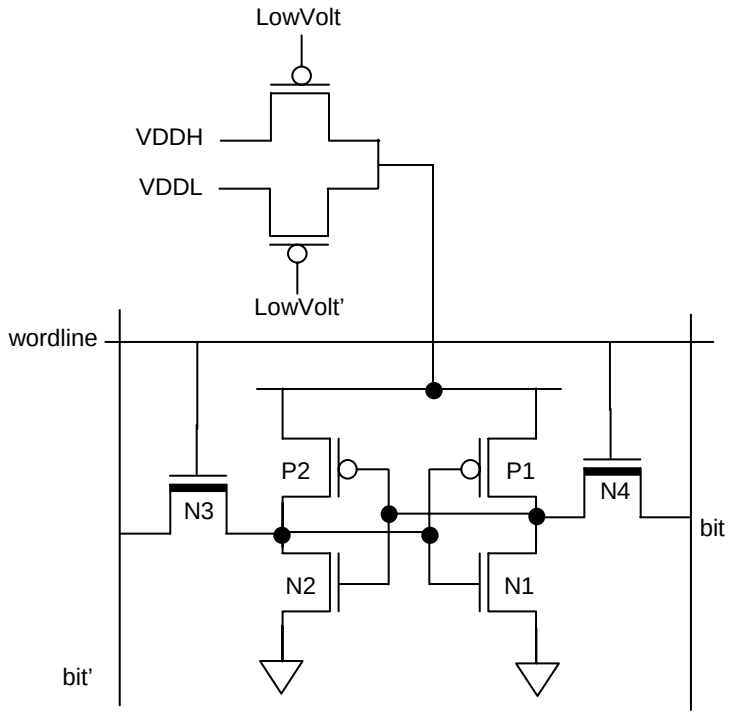


Gated-V_{DD}

*Intel introduces 65-nm sleep transistor SRAM from Intel.com , “65-nm process technology extends the benefit of Moore’s law”

Low-leakage SRAM

- Auto-Backgate-Controlled Multi Threshold CMOS (ABC-MTCMOS) [Nii98]
  - Reverse source-body bias during sleep mode
  - Slow transition and large dynamic power to charge n-wells
- Gated-V_{dd} [Powell00](Prof. K. Roy)
  - Isolate SRAM cells using sleep transistor
  - Loses state during sleep mode
- Drowsy cache [Flautner02]
  - Scaling V_{dd} dynamically
  - Smaller leakage reduction (<86%) (we will show 3 orders magnitude reduction)



Drowsy cache

Low-power pipelined cache

- Breaking down a cache into multiple segment cache can be pipelined to increase performance [Chappell91][Agarwal03]
  - No power reduction addressed
- Pipelining caches to offset performance degradation and pipelined cache [Gunadi04]
  - Saving power by enabling necessary subbank
  - Only dynamic power is addresses (0.18u technology)

Low-leakage SRAM and low power cache comparison

- Sleepy stack SRAM cell
  - No need to charge n-well (ABC-MTCMOS)
  - State-saving (gated-V_{dd})
  - Larger leakage power savings (drowsy cache)
- No prior work found that uses a pipelined cache to reduce dynamic power by scaling V_{dd} or reducing static power

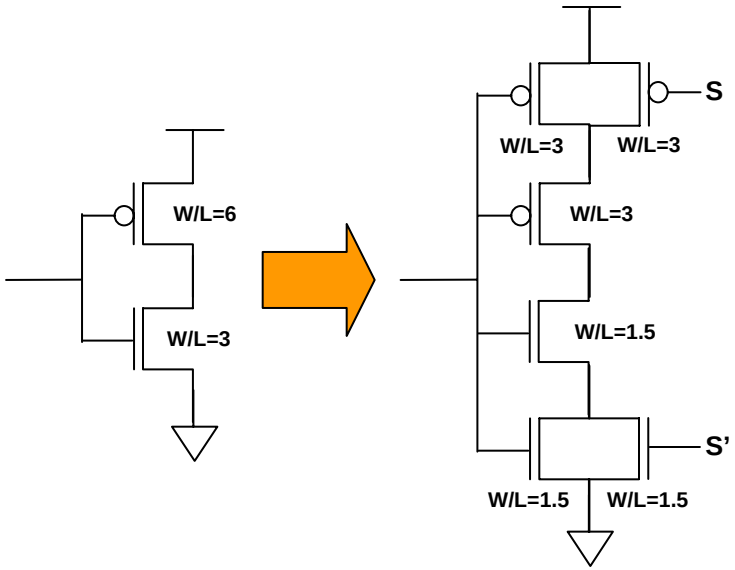
Outline

- Introduction
- Related Work
- Sleepy stack
- Sleepy stack logic circuits
- Sleepy stack SRAM
- Low-power pipelined cache (LPPC)
- Sleepy stack pipelined SRAM
- Conclusion

Introduction of sleepy stack

- New state-saving ultra low-leakage technique
- Combination of the sleep transistor and forced stack technique
- Applicable to generic VLSI structures as well as SRAM
- Target application requires long standby with fast response, e.g., cell phone

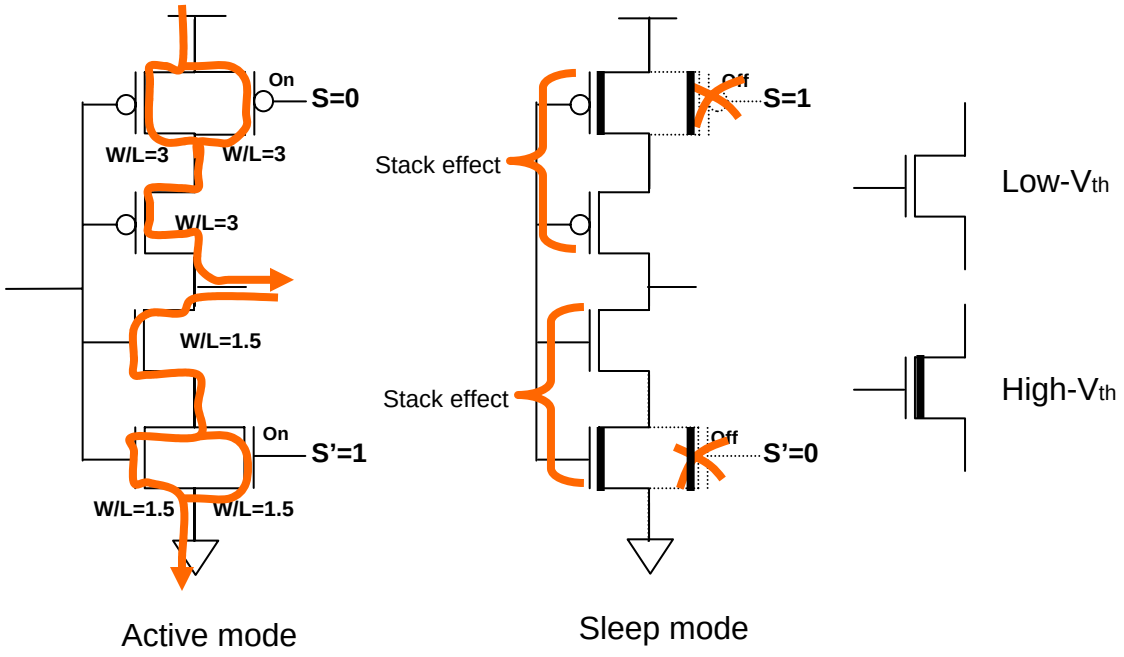
Sleepy stack structure



Conventional CMOS inverter Sleepy stack inverter

- First, Break down a transistor similar to the forced stack technique
- Then add sleep transistors

Sleepy stack operation



- During active mode, sleep transistors are on,
  then reduced resistance increases current while reducing delay
- During sleep mode, sleep transistors are off,
  stacked transistors suppress leakage current while saving state
- Can apply high-V_{th}, which is not used in the forced stack technique due to the dramatic delay increase (>6.2X)

Outline

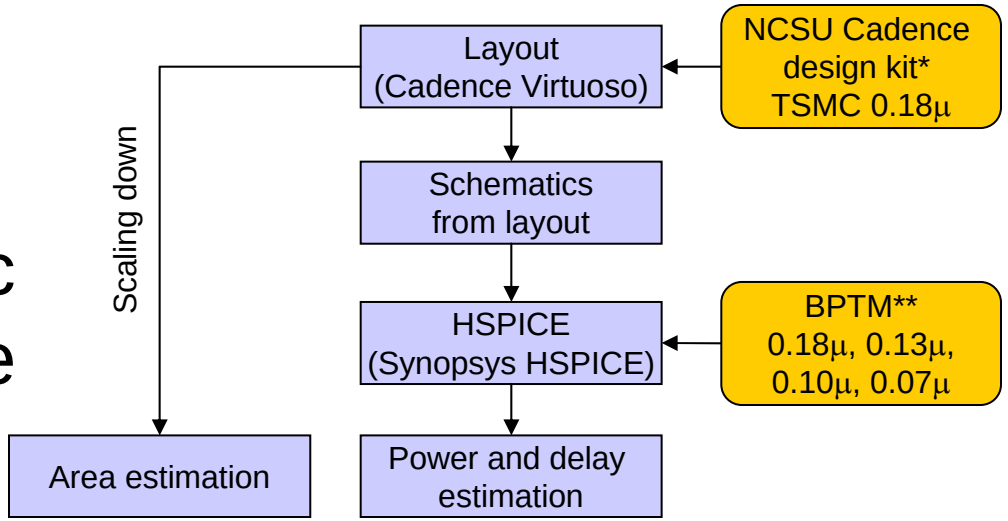
- Introduction
- Related Work
- Sleepy stack
- Sleepy stack logic circuits
- Sleepy stack SRAM
- Low-power pipelined cache (LPPC)
- Sleepy stack pipelined SRAM
- Conclusion

Experimental methodology

- Five techniques are compared
  - base case, forced stack, sleep*, zigzag* and sleepy stack* (*single- and dual-V_{th} applied)
- Three benchmark circuits are used ($C_L=3C_{inv}$)
  - A chain of 4 inverters
    - 4 inverter chain
    - $C_{in}=3C_{inv}$, $C_L=3C_{inv}$
  - 4:1 multiplexer
    - 5 stages
    - standard cell gates
    - $C_{in}=11C_{inv}$, $C_L=3C_{inv}$
  - 4-bit adder
    - 4 full adders
    - complex gate
    - $C_{in}=18.5C_{inv}$, $C_L=3C_{inv}$

Experimental methodology

- Area estimated by scaling down 0.18μ layout
- Dynamic power, static power and worst-case delay of each benchmark circuit measured



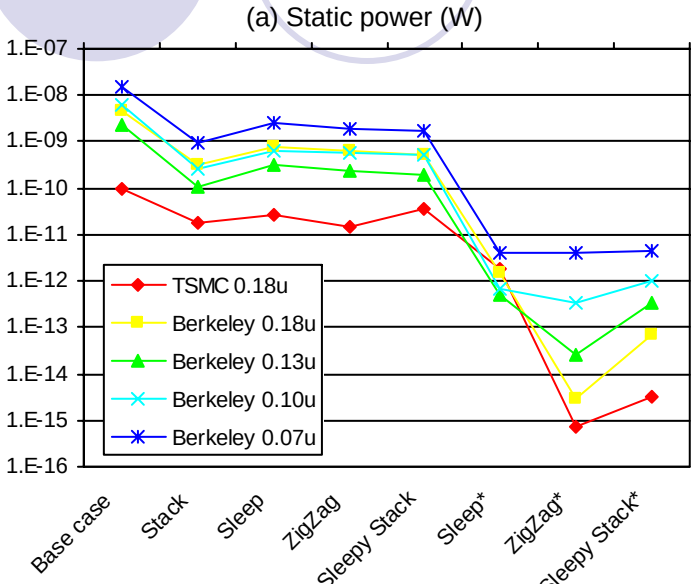
Tech.	0.07μ	0.10μ	0.13μ	0.18μ
V_{DD}	0.8V	1.0V	1.3V	1.8V

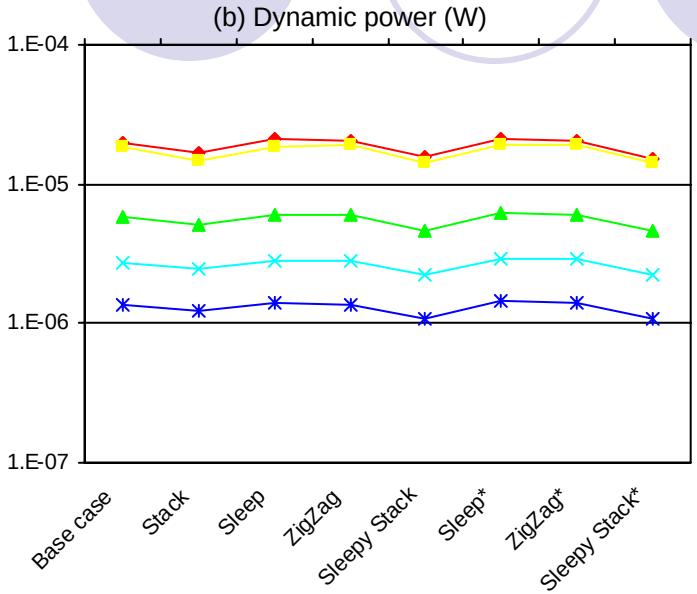
*NC State University Cadence Tool Information. [Online]. Available http://www.cadence.ncsu.edu.

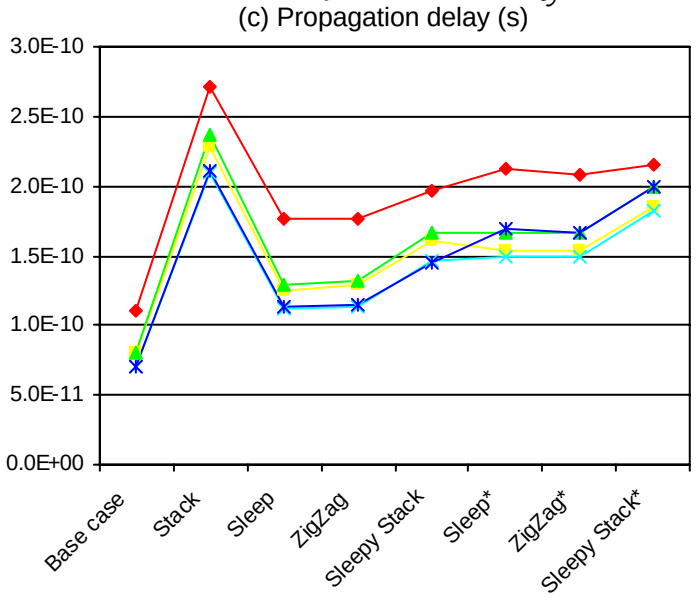
**Berkeley Predictive Technology Model (BPTM). [Online]. Available http://www-device.eecs.berkeley.edu/~ptm.

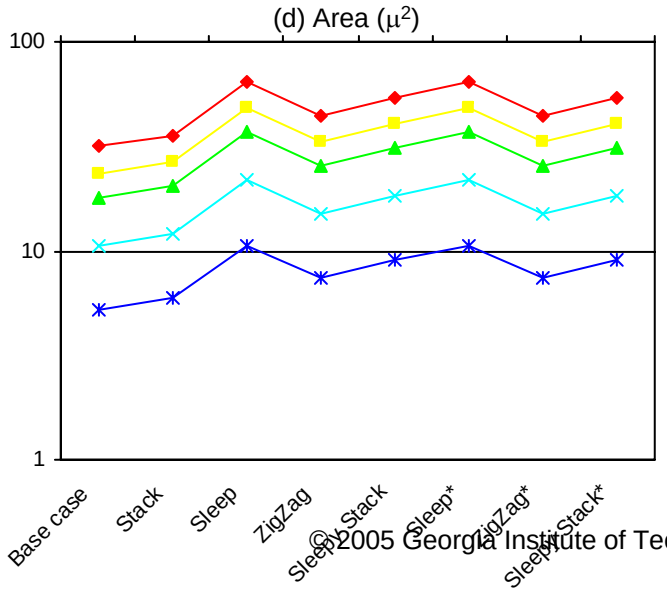
A chain of 4 inverters

* Dual-V_{th} applied (0.2V and 0.4V)









A chain of 4 inverters

- Compared mainly to forced stack (best prior leakage technique while saving state)
- Compared to forced stack, sleepy stack with dual-V_{th} achieves 215X reduction in leakage power with 6% decrease in delay
- Sleepy stack is 73% and 51% larger than base case and forced stack, respectively

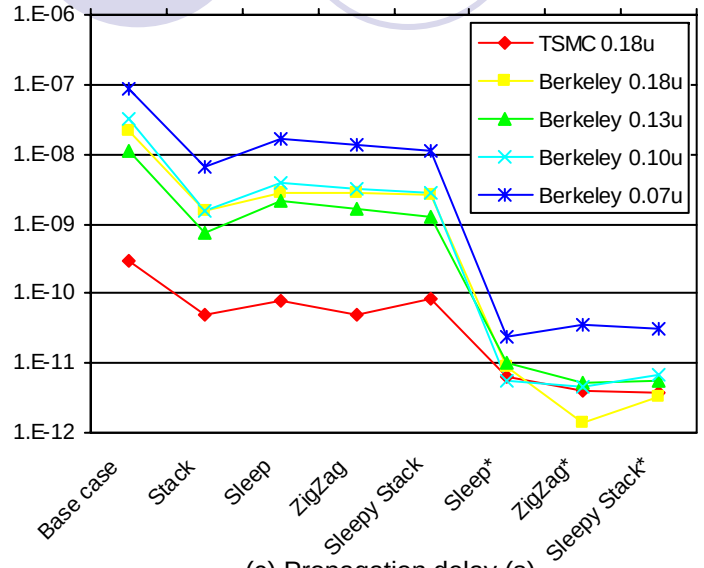
0.07u tech.

A chain of 4 inverters	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (µ2)
Base case	7.05E-11	1.57E-08	1.34E-06	5.23
Stack	2.11E-10	9.81E-10	1.25E-06	5.97
Sleep	1.13E-10	2.45E-09	1.39E-06	10.67
ZigZag	1.15E-10	1.96E-09	1.34E-06	7.39
Sleepy Stack	1.45E-10	1.69E-09	1.08E-06	9.03
Sleep (dual Vth)	1.69E-10	4.12E-12	1.46E-06	10.67
ZigZag (dual Vth)	1.67E-10	4.07E-12	1.39E-06	7.39
Sleepy Stack (dual Vth)	1.99E-10	4.56E-12	1.09E-06	9.03

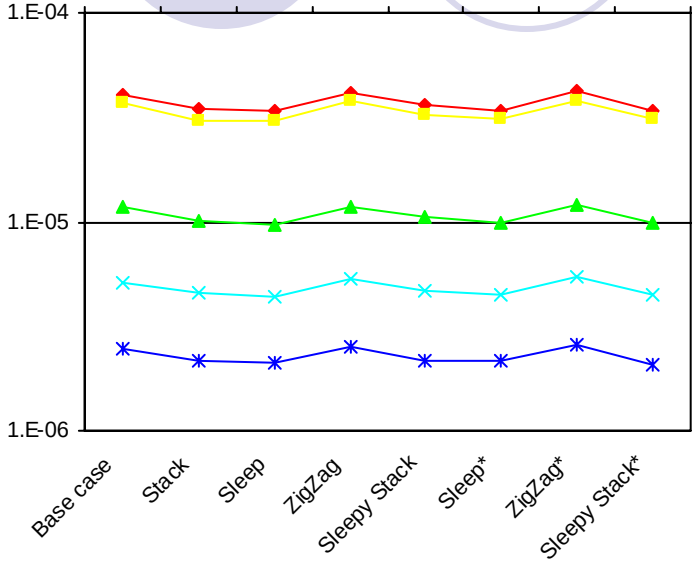
4:1 Multiplexer

* Dual-V_{th} applied (0.2V and 0.4V)

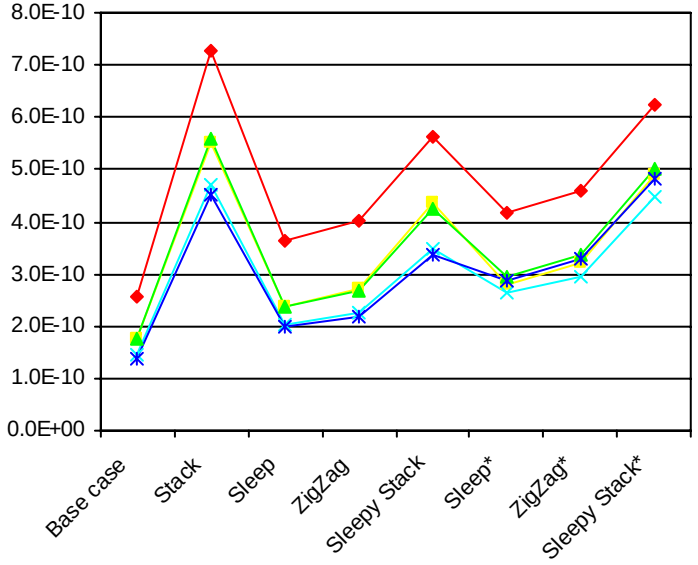
(a) Static power (W)



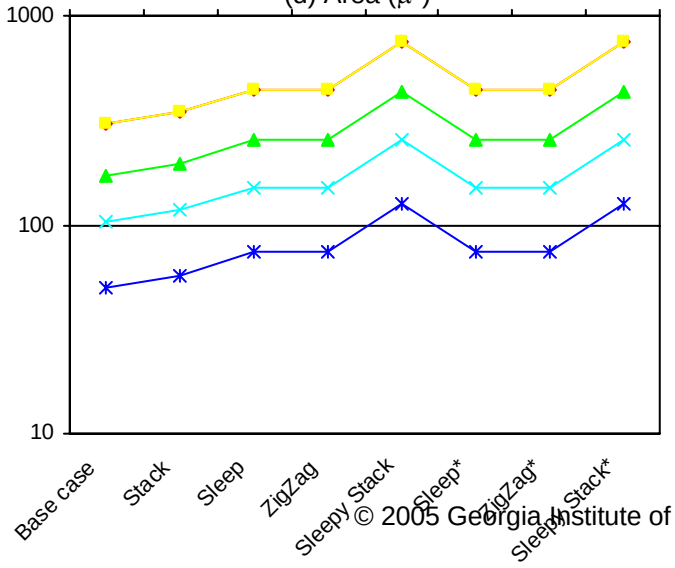
(b) Dynamic power (W)



(c) Propagation delay (s)



(d) Area (μ^2)



4:1 Multiplexer

- Compared to forced stack, sleepy stack with dual-V_{th} achieves 202X reduction in leakage power with 7% increase in delay
- Sleepy stack is 150% and 118% larger than base case and forced stack, respectively

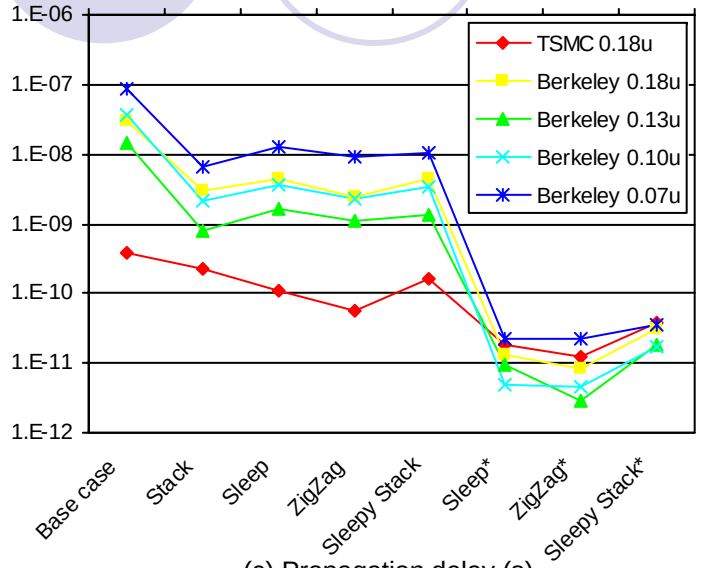
0.07u tech.

4:1 multiplexer	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (μ2)
Base case	1.39E-10	8.57E-08	2.49E-06	50.17
Stack	4.52E-10	6.46E-09	2.14E-06	57.40
Sleep	1.99E-10	1.65E-08	2.10E-06	74.11
ZigZag	2.17E-10	1.36E-08	2.54E-06	74.36
Sleepy Stack	3.35E-10	1.09E-08	2.18E-06	125.33
Sleep (dual Vth)	2.87E-10	2.41E-11	2.15E-06	74.11
ZigZag (dual Vth)	3.28E-10	3.62E-11	2.59E-06	74.36
Sleepy Stack (dual Vth)	4.84E-10	3.20E-11	2.09E-06	125.33

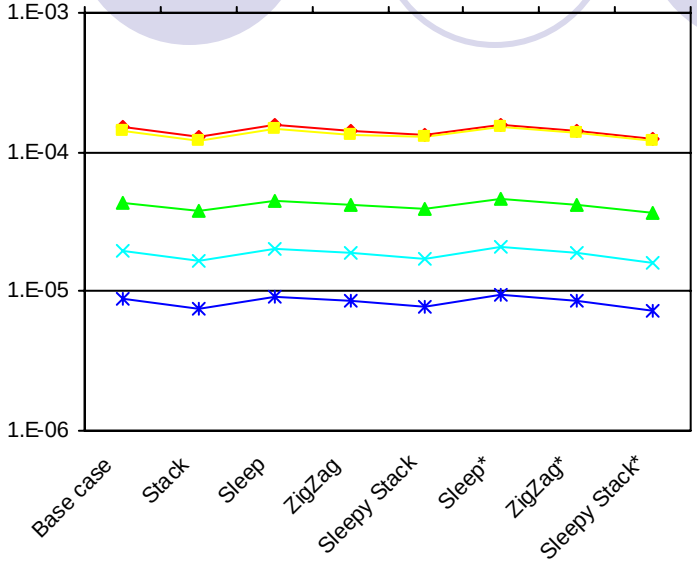
4-bit adder

* Dual-V_{th} applied (0.2V and 0.4V)

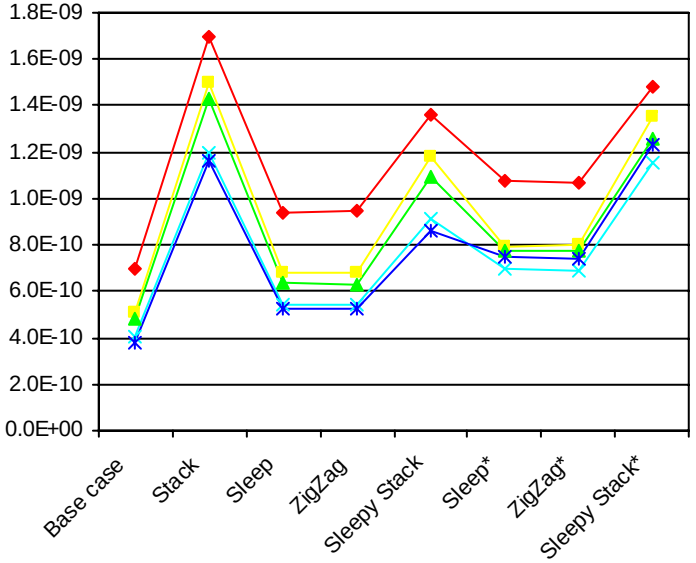
(a) Static power (W)



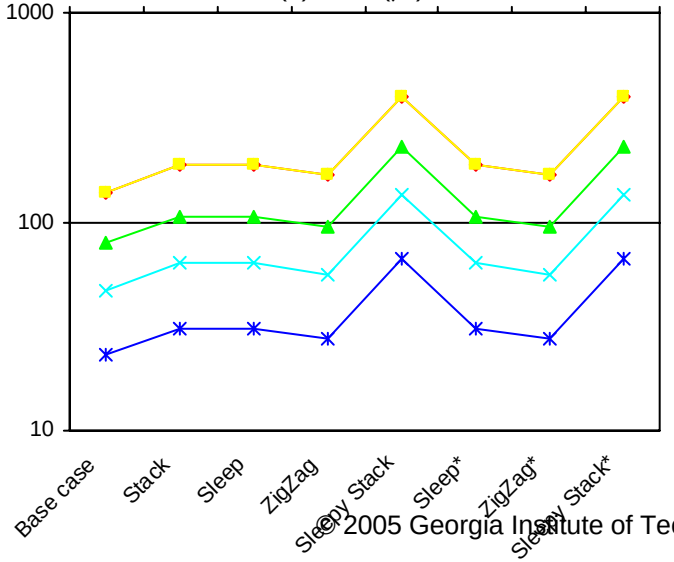
(b) Dynamic power (W)



(c) Propagation delay (s)



(d) Area (μ^2)



4-bit adder

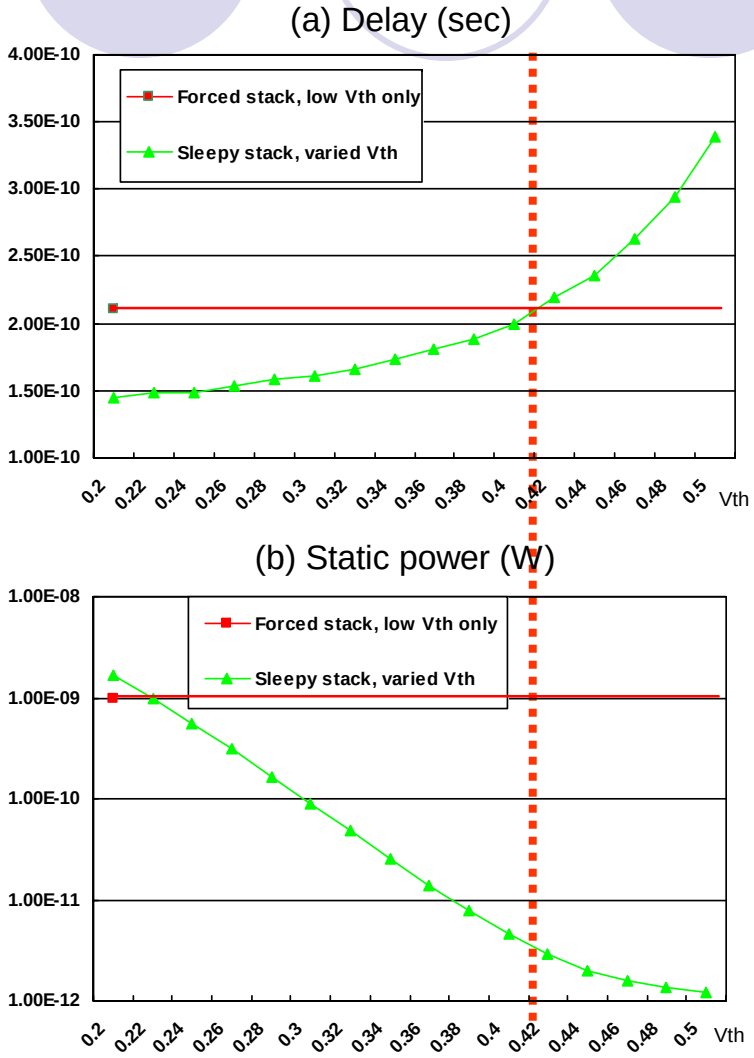
- Compared to forced stack, sleepy stack with dual-V_{th} achieves 190X reduction in leakage power with 6% increase in delay
- Sleepy stack is 187% and 113% larger than base case and forced stack, respectively

0.07u tech

4-bit adder	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (μ2)
Base case	3.76E-10	8.87E-08	8.81E-06	22.96
Stack	1.16E-09	6.77E-09	7.63E-06	30.94
Sleep	5.24E-10	1.24E-08	9.03E-06	30.94
ZigZag	5.24E-10	9.09E-09	8.44E-06	27.62
Sleepy Stack	8.65E-10	1.07E-08	7.70E-06	65.88
Sleep (dual Vth)	7.48E-10	2.23E-11	9.41E-06	30.94
ZigZag (dual Vth)	7.43E-10	2.19E-11	8.53E-06	27.62
Sleepy Stack (dual Vth)	1.23E-09	3.56E-11	7.26E-06	65.88

Sleepy stack V_{th} variation

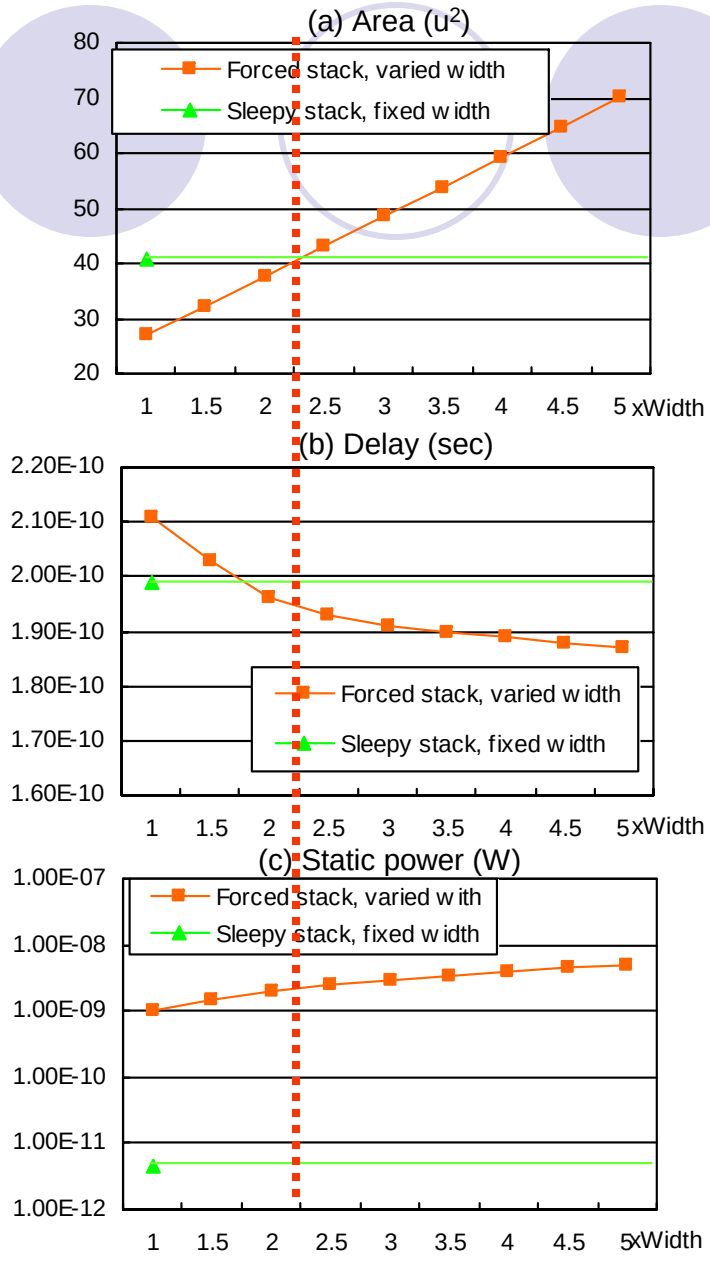
- Impact of V_{th} by comparing the sleepy stack and the forced stack
- V_{th} of the sleepy stack can be increased up to 0.4V while matching delay to the forced stack
- 215X leakage power reduction of the sleepy stack technique



Results from 4 inverters using 90nm technology

Forced stack transistor width variation

- Impact of the forced stack transistor width by comparing the sleepy stack and the forced stack using similar area
- Forced stack V_{th}=0.2V, sleepy stack (sleep and paralleled transistor) V_{th}=0.4V
- Between 2X~2.5X transistor width of the forced stack matches area with the sleepy stack
- Force stack is 1.5% faster but leakage power is 430X larger



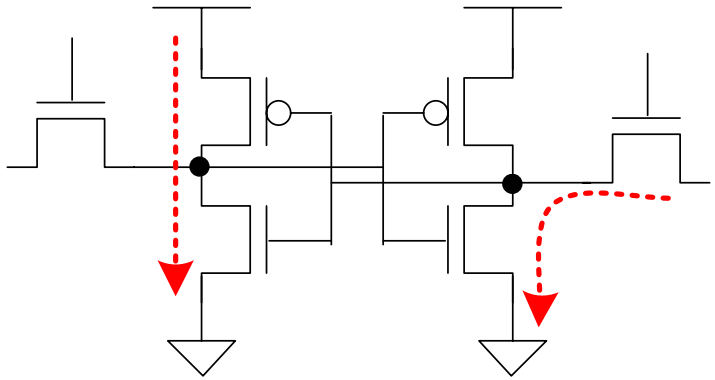
Results from 4 inverters using 0.07u technology

Outline

- Introduction
- Related Work
- Sleepy stack
- Sleepy stack logic circuits
- Sleepy stack SRAM
- Low-power pipelined cache (LPPC)
- Sleepy stack pipelined SRAM
- Conclusion

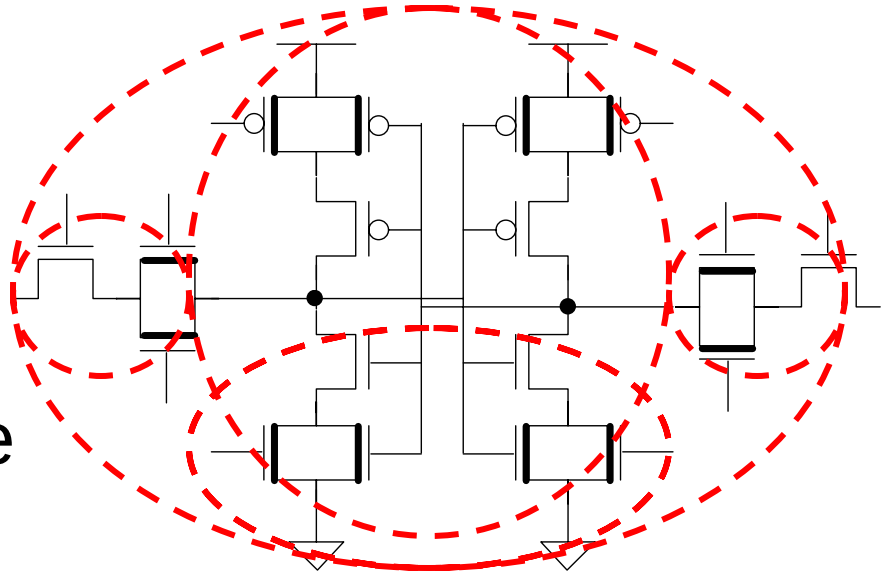
Sleepy stack SRAM cell

- Sleepy stack technique achieves ultra-low leakage power while saving state
- Apply the sleepy stack technique to SRAM cell design
  - Large leakage power saving expected in cache
  - State-saving
  - 6-T SRAM cell is based on coupled inverters
- SRAM cell leakage paths
  - Cell leakage
  - Bitline leakage



Sleepy stack SRAM cell

- Sleepy stack SRAM cell
  - PD sleepy stack
  - PD, WL sleepy stack
  - PU, PD sleepy stack
  - PU, PD, WL sleepy stack
- Area, delay and leakage power tradeoffs

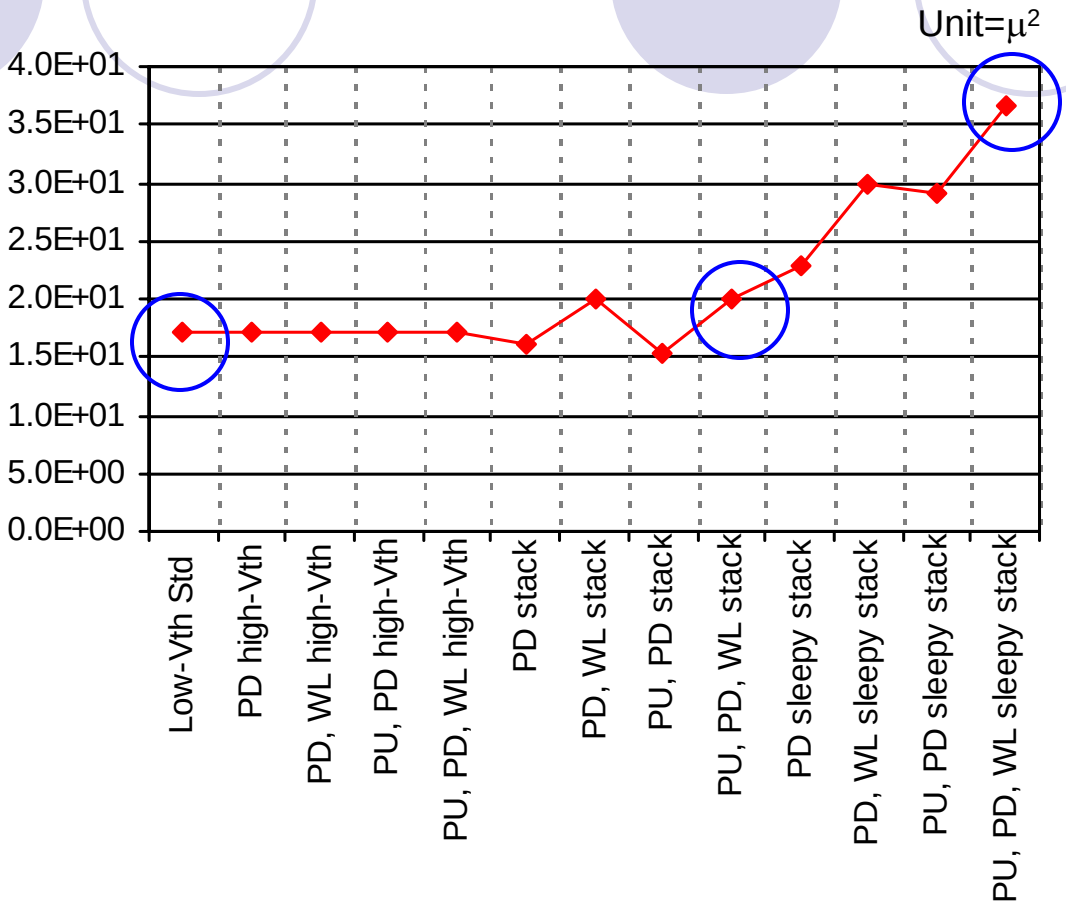


Experimental methodology

- Base case and three techniques are compared
  - High-V_{th} technique, forced stack, and sleepy stack
- 64x64 bit SRAM array designed
- Area estimated by scaling down 0.18μ layout
  - Area of 0.18u layout*(0.07u/0.18u)
- Power and read time using HSPICE targeting 0.07μ
- 1.5xV_{th} and 2.0xV_{th}
- 25^{o}C and 110^{o}C

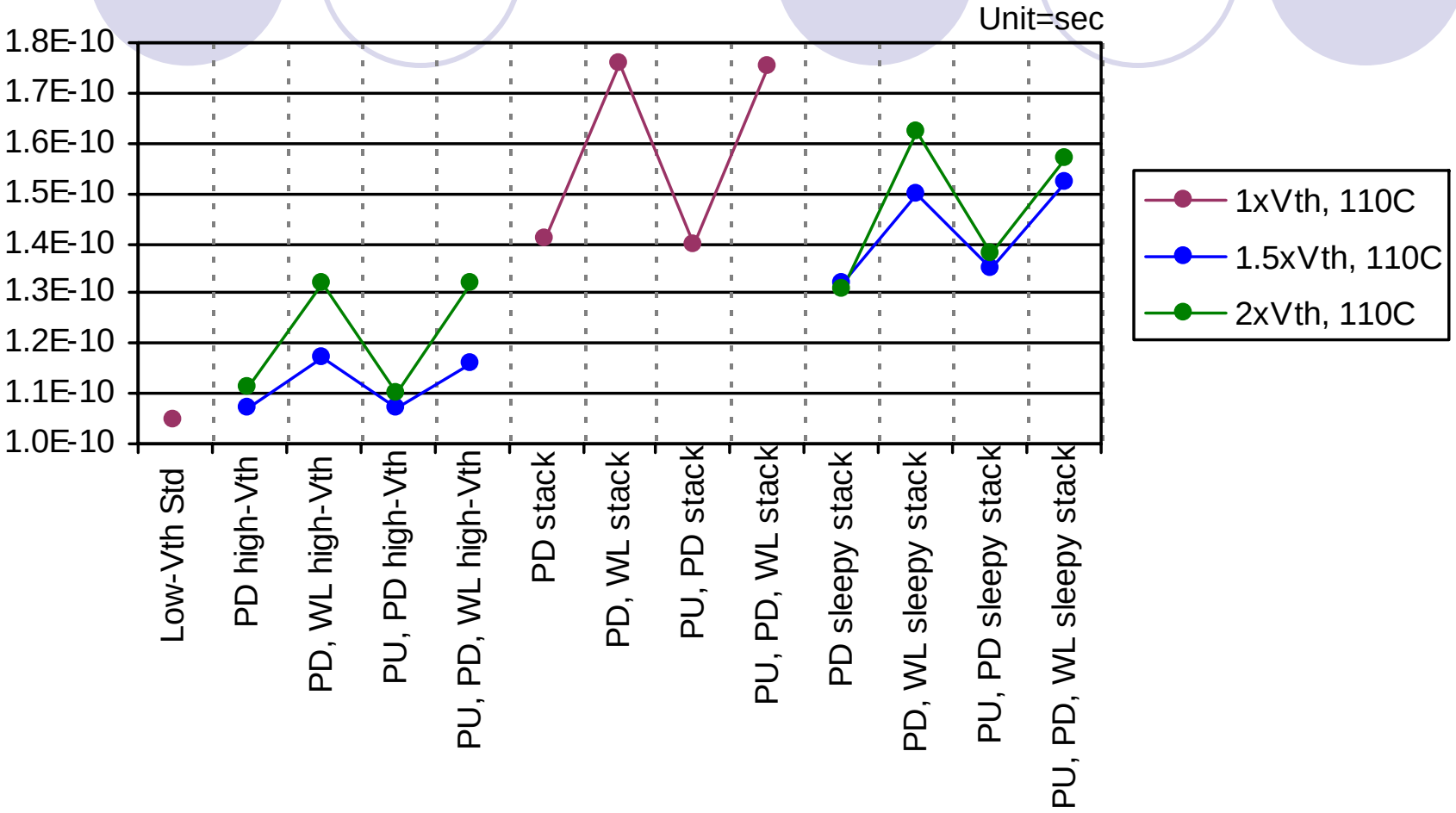
	Technique	
Case1	Low-Vth Std	Conventional 6T SRAM
Case2	PD high-Vth	High-Vth applied to PD
Case3	PD, WL high-Vth	High-Vth applied to PD, WL
Case4	PU, PD high-Vth	High-Vth applied to PU, PD
Case5	PU, PD, WL high-Vth	High-Vth applied to PU, PD, WL
Case6	PD stack	Stack applied to PD
Case7	PD, WL stack	Stack applied to PD, WL
Case8	PU, PD stack	Stack applied to PU, PD
Case9	PU, PD, WL stack	Stack applied to PU, PD, WL
Case10	PD sleepy stack	Sleepy stack applied to PD
Case11	PD, WL sleepy stack	Sleepy stack applied to PD, WL
Case12	PU, PD sleepy stack	Sleepy stack applied to PU, PD
Case13	PU, PD, WL sleepy stack	Sleepy stack applied to PU, PD, WL

Area



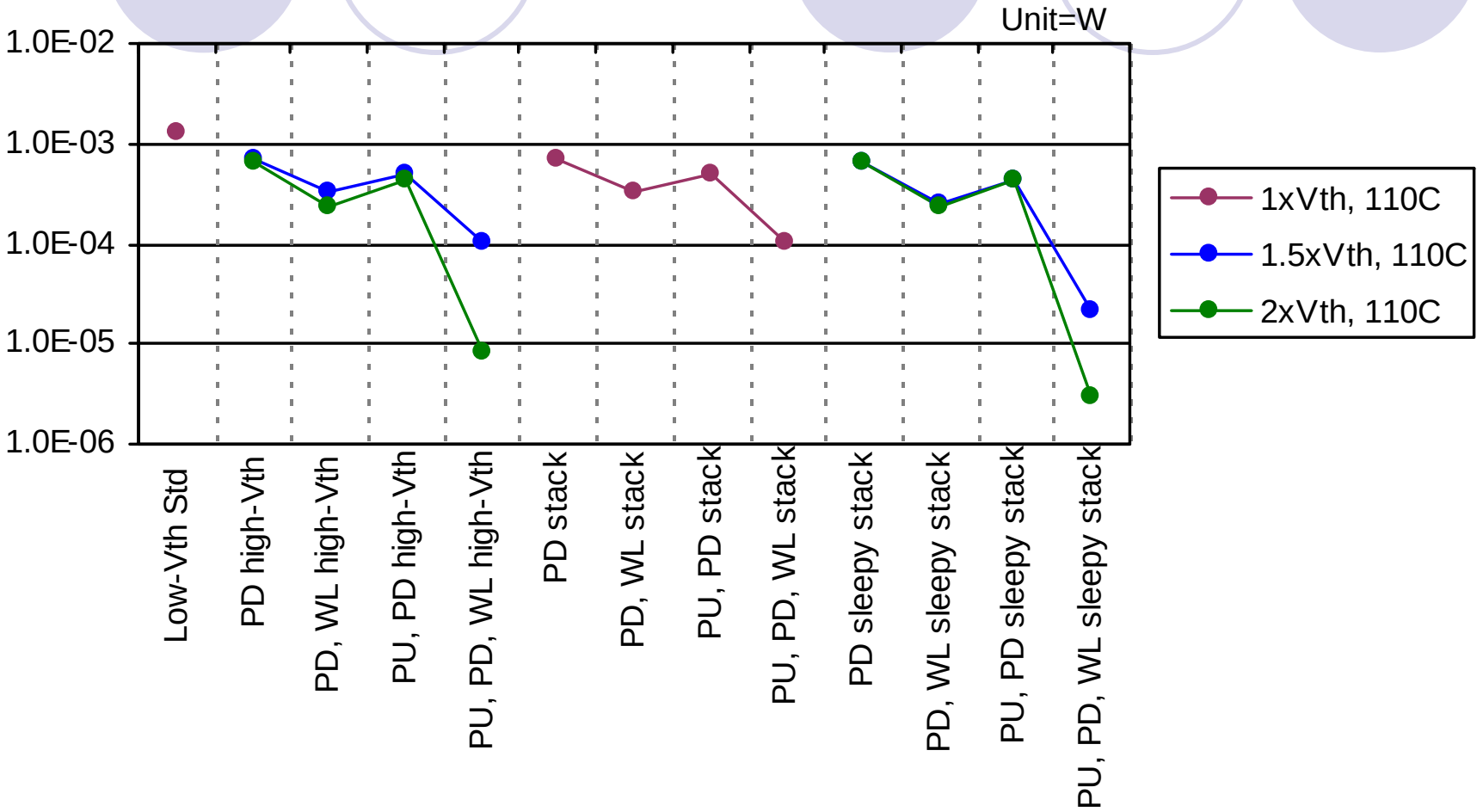
- PU, PD, WL sleepy stack is 113% and 83% larger than base case and PU, PD, WL forced stack, respectively

Cell read time



- Delay: High-V_{th} < sleepy stack < forced stack

Leakage power



- At 110°C, the worst case, leakage power: forced stack > high-V_{th} 2xV_{th} > sleepy stack 2xV_{th}

Tradeoffs

1.5xVth at 110ºC

	Technique	Leakage power (W)	Delay (sec)	Area (u^2)	Normalized leakage power	Normalized delay	Normalized area
Case1	Low-Vth Std	1.254E-03	1.05E-10	17.21	1.000	1.000	1.000
Case2	PD high-Vth	7.159E-04	1.07E-10	17.21	0.571	1.020	1.000
Case6	PD stack	7.071E-04	1.41E-10	16.22	0.564	1.345	0.942
Case10*	PD sleepy stack*	6.744E-04	1.15E-10	25.17	0.538	1.102	1.463
Case10	PD sleepy stack	6.621E-04	1.32E-10	22.91	0.528	1.263	1.331
Case4	PU, PD high-Vth	5.042E-04	1.07E-10	17.21	0.402	1.020	1.000
Case8	PU, PD stack	4.952E-04	1.40E-10	15.37	0.395	1.341	0.893
Case12*	PU, PD sleepy stack*	4.532E-04	1.15E-10	31.30	0.362	1.103	1.818
Case12	PU, PD sleepy stack	4.430E-04	1.35E-10	29.03	0.353	1.287	1.687
Case3	PD, WL high-Vth	3.203E-04	1.17E-10	17.21	0.256	1.117	1.000
Case7	PD, WL stack	3.202E-04	1.76E-10	19.96	0.255	1.682	1.159
Case11*	PD, WL sleepy stack*	2.721E-04	1.16E-10	34.40	0.217	1.111	1.998
Case11	PD, WL sleepy stack	2.451E-04	1.50E-10	29.87	0.196	1.435	1.735
Case5	PU, PD, WL high-Vth	1.074E-04	1.16E-10	17.21	0.086	1.110	1.000
Case9	PU, PD, WL stack	1.043E-04	1.75E-10	19.96	0.083	1.678	1.159
Case13*	PU, PD, WL sleepy stack*	4.308E-05	1.16E-10	41.12	0.034	1.112	2.389
Case13	PU, PD, WL sleepy stack	2.093E-05	1.52E-10	36.61	0.017	1.450	2.127

- Sleepy stack delay is matched to Case5 (“*” means delay matched to Case5=best prior work)
- Sleepy stack SRAM provides new pareto points (blue rows)
- Case13 achieves 5.13X leakage reduction (with 32% delay increase), alternatively Case13* achieves 2.49X leakage reduction compared to Case5 (while matching delay to Case5)

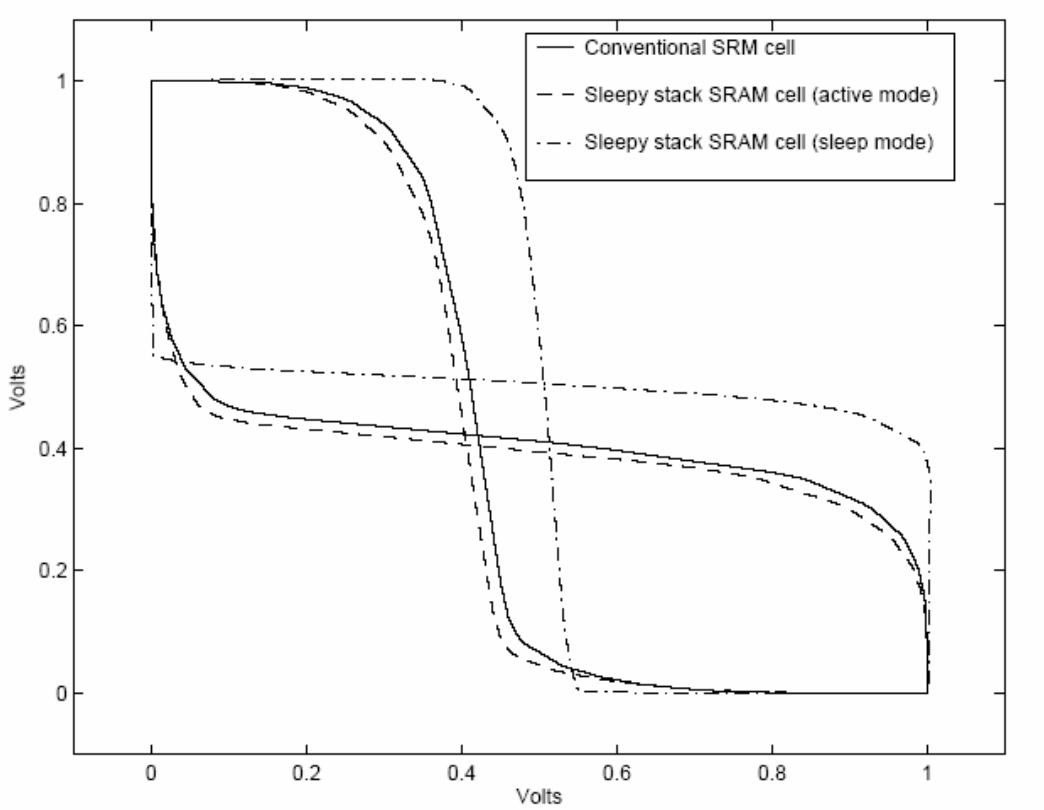
Tradeoffs

2.0xVth at 110ºC

	Technique	Static (W)	Delay (sec)	Area (u^2)	Normalized leakage	Normalized delay	Normalized area
Case1	Low-Vth Std	1.25E-03	1.05E-10	17.21	1.000	1.000	1.000
Case6	PD stack	7.07E-04	1.41E-10	16.22	0.564	1.345	0.942
Case2	PD high-Vth	6.65E-04	1.11E-10	17.21	0.530	1.061	1.000
Case10	PD sleepy stack	6.51E-04	1.31E-10	22.91	0.519	1.254	1.331
Case10*	PD sleepy stack*	6.51E-04	1.31E-10	22.91	0.519	1.254	1.331
Case8	PU, PD stack	4.95E-04	1.40E-10	15.37	0.395	1.341	0.893
Case4	PU, PD high-Vth	4.42E-04	1.10E-10	17.21	0.352	1.048	1.000
Case12*	PU, PD sleepy stack*	4.31E-04	1.33E-10	29.48	0.344	1.270	1.713
Case12	PU, PD sleepy stack	4.31E-04	1.38E-10	29.03	0.344	1.319	1.687
Case7	PD, WL stack	3.20E-04	1.76E-10	19.96	0.255	1.682	1.159
Case3	PD, WL high-Vth	2.33E-04	1.32E-10	17.21	0.186	1.262	1.000
Case11*	PD, WL sleepy stack*	2.29E-04	1.30E-10	32.28	0.183	1.239	1.876
Case11	PD, WL sleepy stack	2.28E-04	1.62E-10	29.87	0.182	1.546	1.735
Case9	PU, PD, WL stack	1.04E-04	1.75E-10	19.96	0.083	1.678	1.159
Case5	PU, PD, WL high-Vth	8.19E-06	1.32E-10	17.21	0.007	1.259	1.000
Case13*	PU, PD, WL sleepy stack*	3.62E-06	1.32E-10	38.78	0.003	1.265	2.253
Case13	PU, PD, WL sleepy stack	2.95E-06	1.57E-10	36.61	0.002	1.504	2.127

- Sleepy stack delay is matched to Case5 (“*” means delay matched to Case5=best prior work)
- Sleepy stack SRAM provides new pareto points (blue rows)
- Case13 achieves 2.77X leakage reduction (with 19% delay increase over Case5), alternatively Case13* achieves 2.26X leakage reduction compared to Case5 (while matching delay to Case5)

Static noise margin



	Technique	Static noise margin (V)	
		Active mode	Sleep mode
Case1	Low-Vth Std	0.299	N/A
Case10	PD sleepy stack	3.167	0.362
Case11	PD, WL sleepy stack	0.324	0.363
Case12	PU, PD sleepy stack	0.299	0.384
Case13	PU, PD, WL sleepy stack	0.299	0.384

- Measure noise immunity using static noise margin (SNM)
- SNM of the sleepy stack is similar or better than the base case

Outline

- Introduction
- Related Work
- Sleepy stack
- Sleepy stack logic circuits
- Sleepy stack SRAM
- Low-power pipelined cache (LPPC)
- Sleepy stack pipelined SRAM
- Conclusion

Low-power pipelined cache (LPPC)

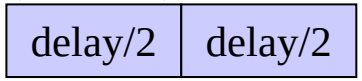
(a) Base case

cycle time = 4.3 ns

V_{DD} = 2.25V, C_L=1pF
f = 233Mhz, E.T. = 1sec
E = ½*1pF*(2.25)^2 x 233 Mhz x 1sec
= 0.589mJ

(b) Pipelined cache
for high-performance

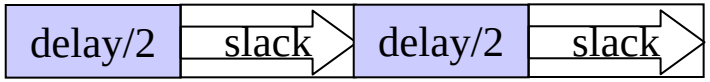
cycle time = 2.15ns



V_{DD} = 2.25 V, C_L=1pF
f = 466Mhz, E.T. = 0.5sec
E = ½*1pFX(2.25)^2 x 466 Mhz x 0.5sec
= 0.589mJ

(c) Low-power pipelined cache

cycle time = 4.3ns

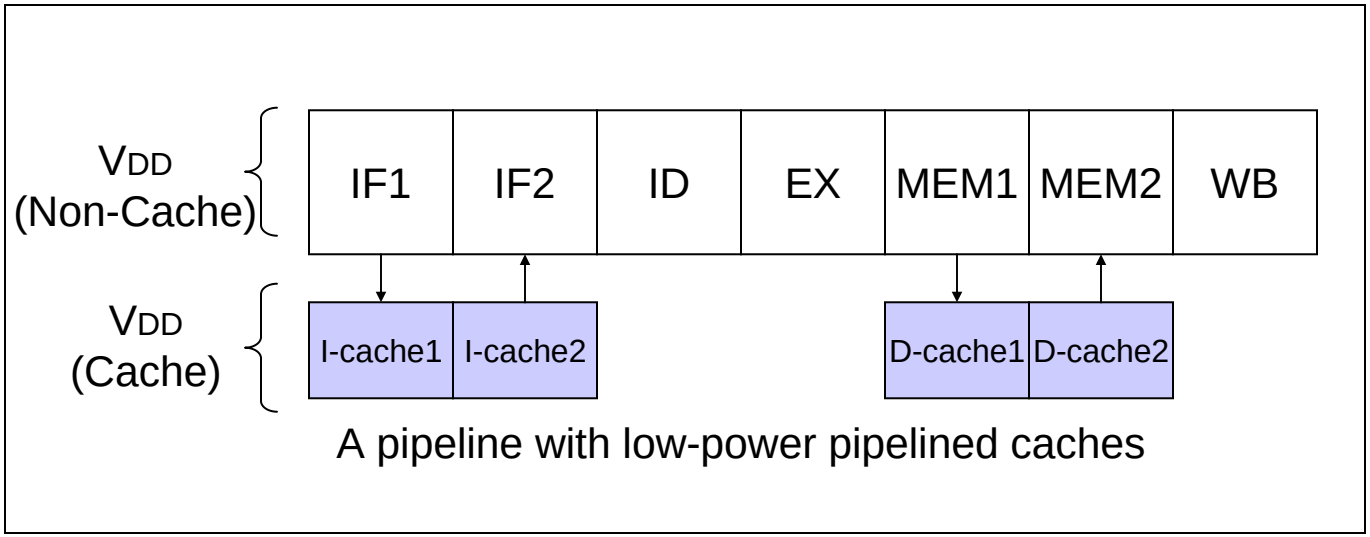


V_{DD} = 1.25 V, C_L=1pF
f = 233Mhz, E.T. = 1sec
E = ½*1pF(1.25)^2 x 233 Mhz x 1sec
= 0.128mJ

*Energy saving = 78.3%

Low-power pipelined cache (LPPC)

- Extra slack by splitting cache stages
- Generic pipelined cache increases clock frequency by reducing delay*
- Dynamic power reduction by lowering V_{dd} of caches
- Optimal pipeline depth to a given architecture



A pipeline with low-power pipelined caches

*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*, vol. 26, no. 11, pp. 1577-1585, 1991.

Pipelining techniques

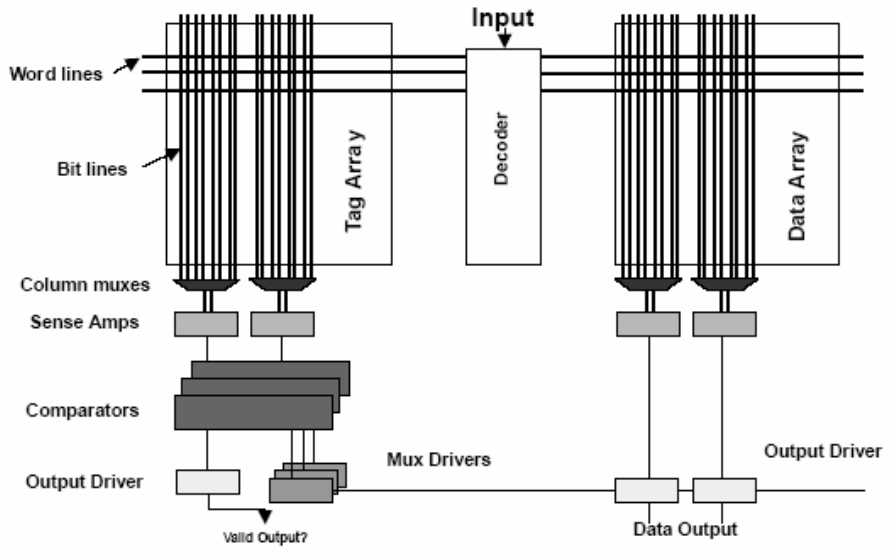
- Latched-pipelining
  - Place latches in-between stages
  - Typically used for pipelined processor
  - Easy to implement
  - When applied to the cache pipelining, the delay of each pipeline stage could be different
- Wave-pipelining
  - Use existing gates as a virtual storage element
  - Even distribution of delay is potentially possible
  - Complex timing analysis is required
  - Used for industry SRAM design
    - UltraSPARC-IV uses wave-pipelined SRAM (90nm tech)*
    - Hitachi designs 300-Mhz 4-Mbit wave-pipelined CMOS SRAM**

*UntraSPARC IV Processor Architecture Overview, February, http://www.sun.com.

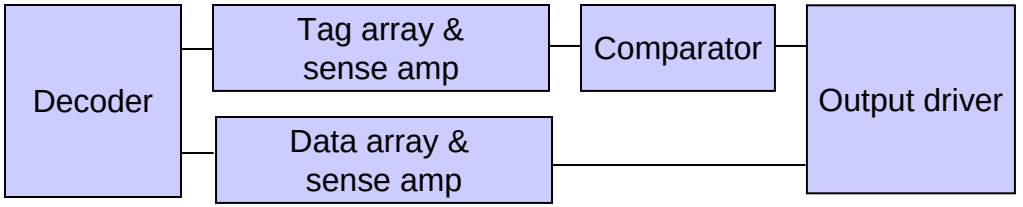
**K. Ishibashi et al., "A 300 MHz 4-Mb Wave-pipeline CMOS SRAM Using a Multi-Phase PLL," *IEEE International Solid-State Circuits Conference*, pp. 308-309, February 1995.

Cache delay model

- Modify CACTI* to measure cycle time of pipelined cache with variable V_{dd}
- Latch pipelined cache
  - Divide CACTI cache model into four segments
  - Merge adjacent segments to form 2-, 3- and 4-stage pipelined cache
- Wave pipelined cache
  - Cycle time using wave variable in CACTI



CACTI cache structure

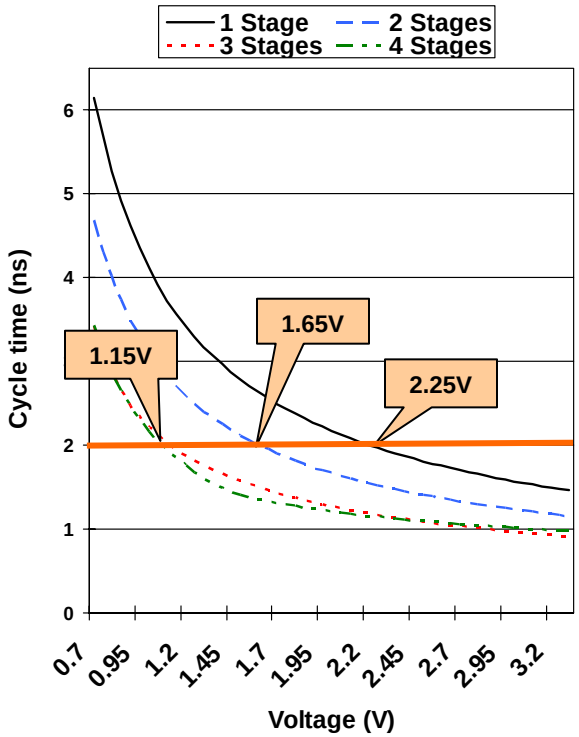


Pipeline segmentation for latch pipelined cache

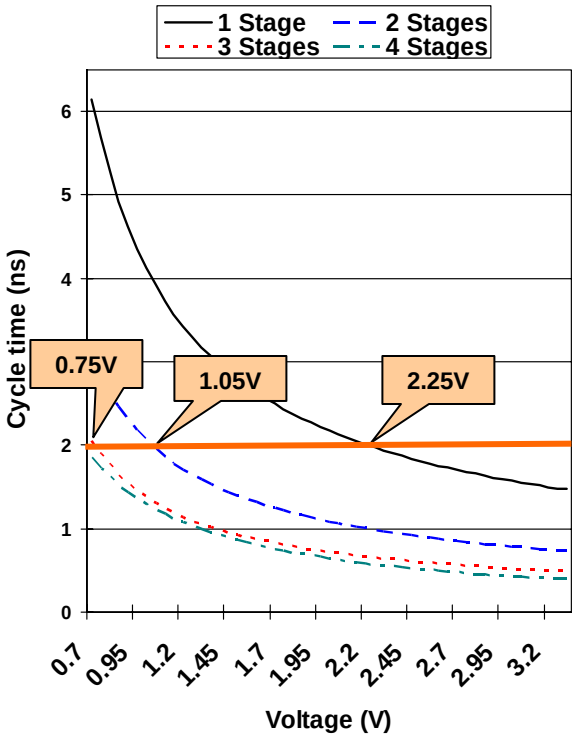
*Reinman, G. and Jouppi, N., CACTI 2.0: An integrated cache timing and power model. [Online]. Available http://www.research.compaq.com/wrl/people/jouppi/CACTI.html

Cache cycle time

- Measure cycle time while varying Vdd and pipeline depth with 0.25µ tech.
- Cycle time is maximum delay of stages



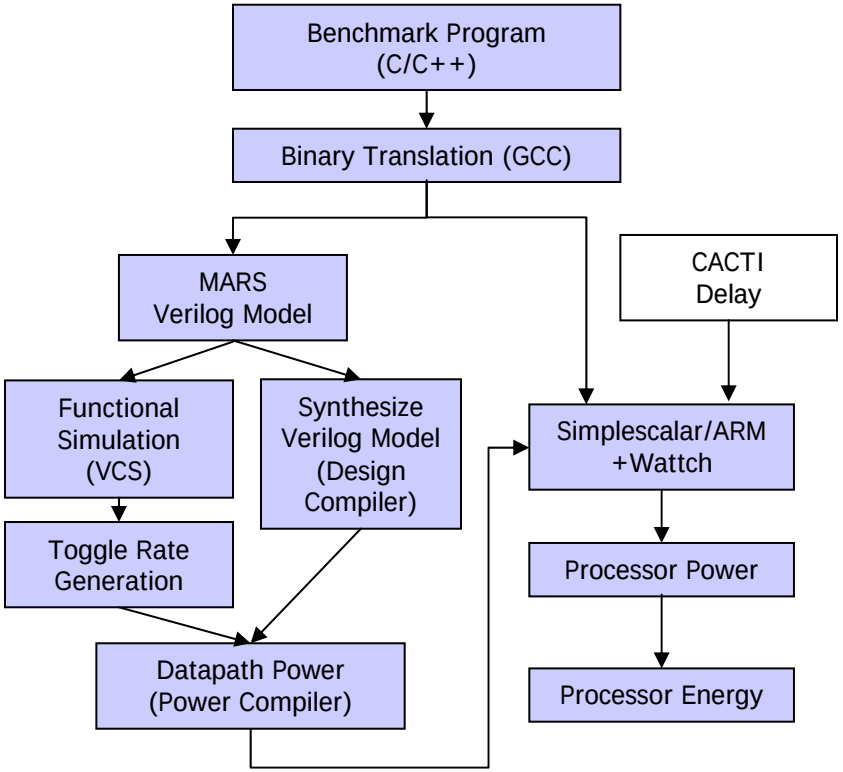
Latch pipelined cache



Wave pipelined cache

Experimental Setup

- Evaluate targeting embedded processor
- Simplescalar/ARM+Wattch* for performance and power estimation
- Modify Simplescalar/ARM+Wattch to simulate a variable stage pipelined cache processor
- Michigan ARM-like Verilog processor model (MARS**) for the power estimation of buffers between broken (non-cache) pipeline stages
- Expand MARS pipeline and measure power consumption using synthesis based power measurement method



* D. Brooks et al., "Wattch: A Framework for Architectural Level Power Analysis and Optimizations," *Proceedings of the International Symposium on Computer Architecture*, pp. 83-94, June 2000.

**The Simplescalar-Arm power modeling project. [Online]. Available http://www.eecs.umich.edu/~jringenb/power

Architecture configuration and benchmarks

- Simplescalar/ARM+Wattch configuration is modeled after Intel StrongARM
- Branch target buffer used to hide branch delay
- Compiler optimization used to hide load delay
- 7 benchmarks targeting embedded system domain

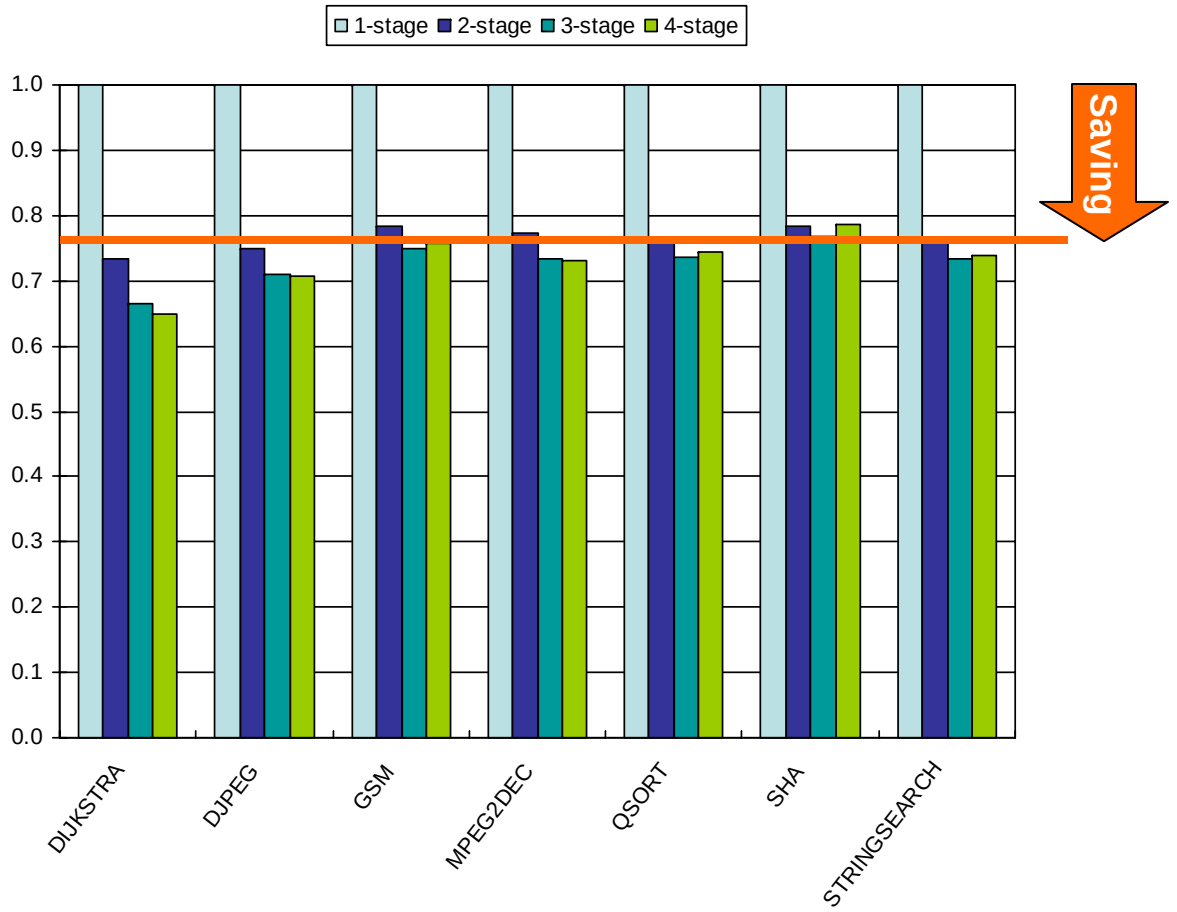
Execution type	In-order
Branch predictor	128 entry BTB
L1 I-cache	32KB 4-way
L1 D-cache	32KB 4-way
L2 cache	None
Memory bus width	4-byte
Memory latency	12 cycles
Clock speed	233Mhz
V_{dd} (Core)	2.25V
V_{dd} (Cache)	2.25V, 1.05V 0.75V

Execution cycles

Benchmark	1-stage cache	2-stage pipelined cache				3-stage pipelined cache				4-stage pipelined cache			
			Increase (%)				Increase (%)				Increase (%)		
	cycles	cycles	Total	Icache	Dcache	cycles	Total	Icache	Dcache	cycles	Total	Icache	Dcache
DIJKSTRA	100,437,638	109,881,681	9.40	1.13	8.27	127,199,801	26.65	2.38	24.26	141,488,147	30.11	5.11	25.00
DJPEG	10,734,606	11,380,700	6.02	0.34	5.68	12,243,399	14.06	0.74	13.32	13,091,913	15.41	1.84	13.57
GSM	21,522,735	21,859,916	1.57	0.46	1.11	23,078,322	7.23	1.03	6.19	24,173,369	11.08	1.59	9.49
MPEG2DEC	28,461,724	29,398,489	3.29	0.28	3.01	31,741,379	11.52	1.27	10.25	33,969,889	15.87	2.37	13.50
QSORT	90,206,190	93,280,904	3.41	1.91	1.50	97,111,275	7.65	4.10	3.55	100,787,822	10.08	6.74	3.34
SHA	17,533,248	17,600,241	0.38	0.34	0.04	18,014,403	2.74	0.74	2.00	18,413,569	4.98	1.14	3.84
STRINGSEARCH	6,356,925	6,668,413	4.90	2.13	2.77	7,048,747	10.88	4.58	6.31	7,409,552	13.40	6.94	6.46
Average			4.14	0.94	3.20		11.53	2.12	9.41		14.42	3.68	10.74

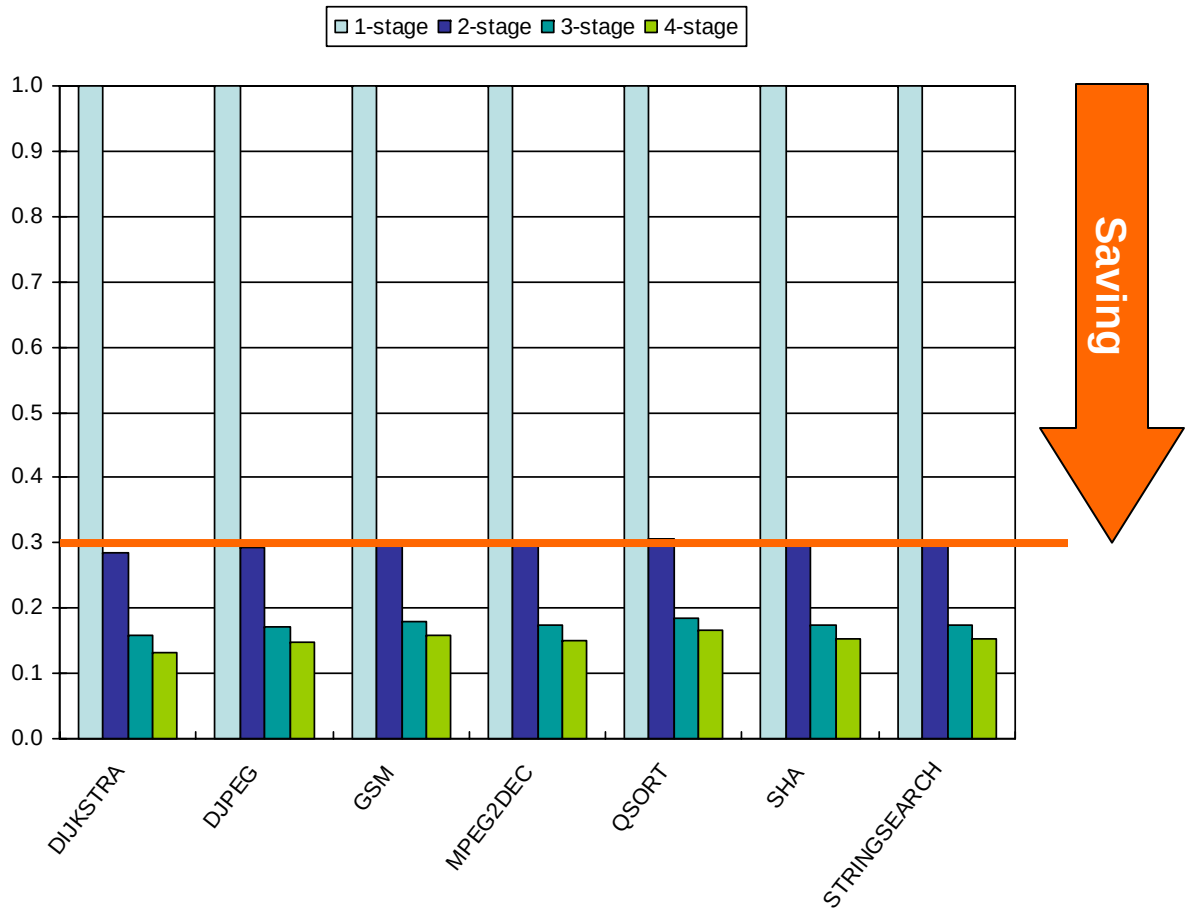
- E.T. increases as the pipelined cache deepens due to pipelining penalties (branch misprediction, load delay)
- 2-stage pipelined cache increases execution time by 4.14%

Normalized processor power consumption



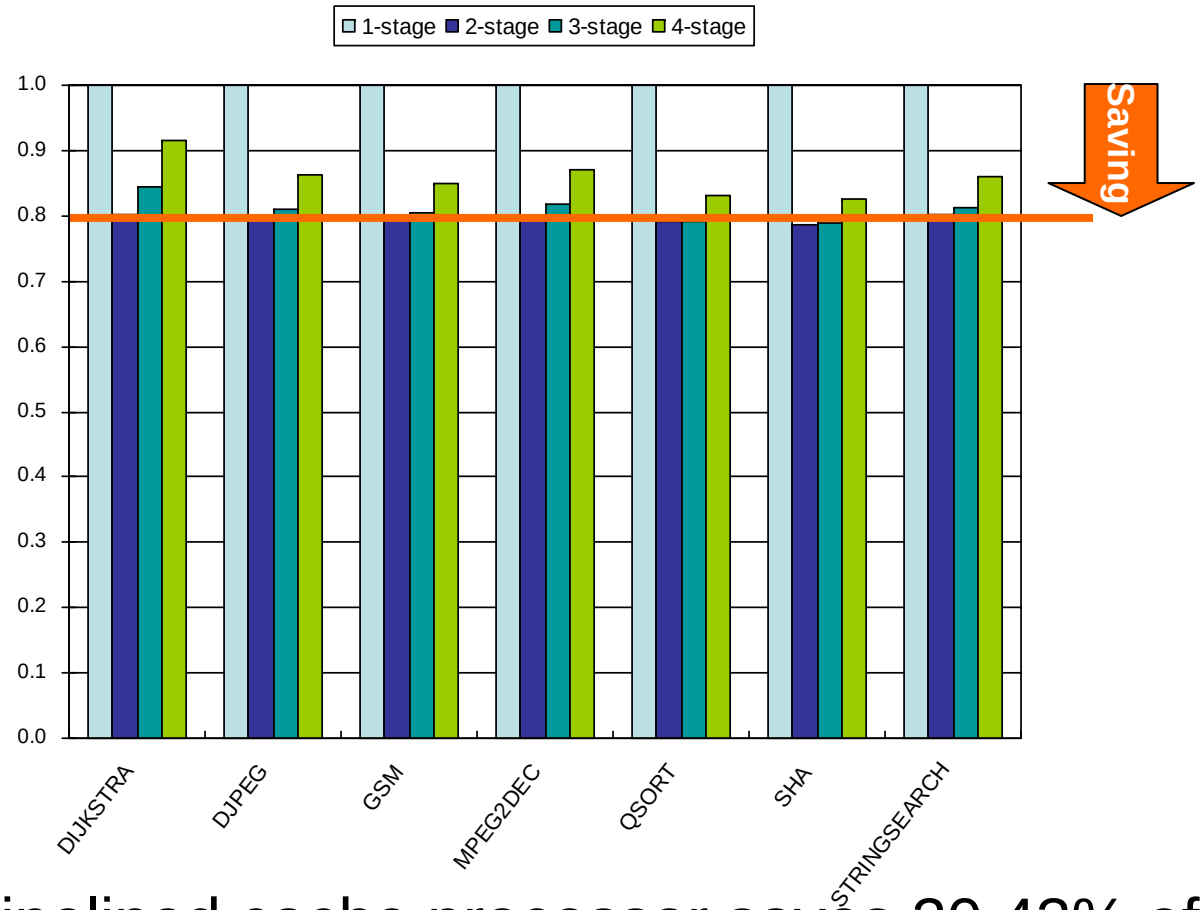
- 2-stage pipelined cache processor saves 23.55% of power

Normalized cache power consumption



- 2-stage pipelined caches save about 70% of cache power

Normalized energy consumption



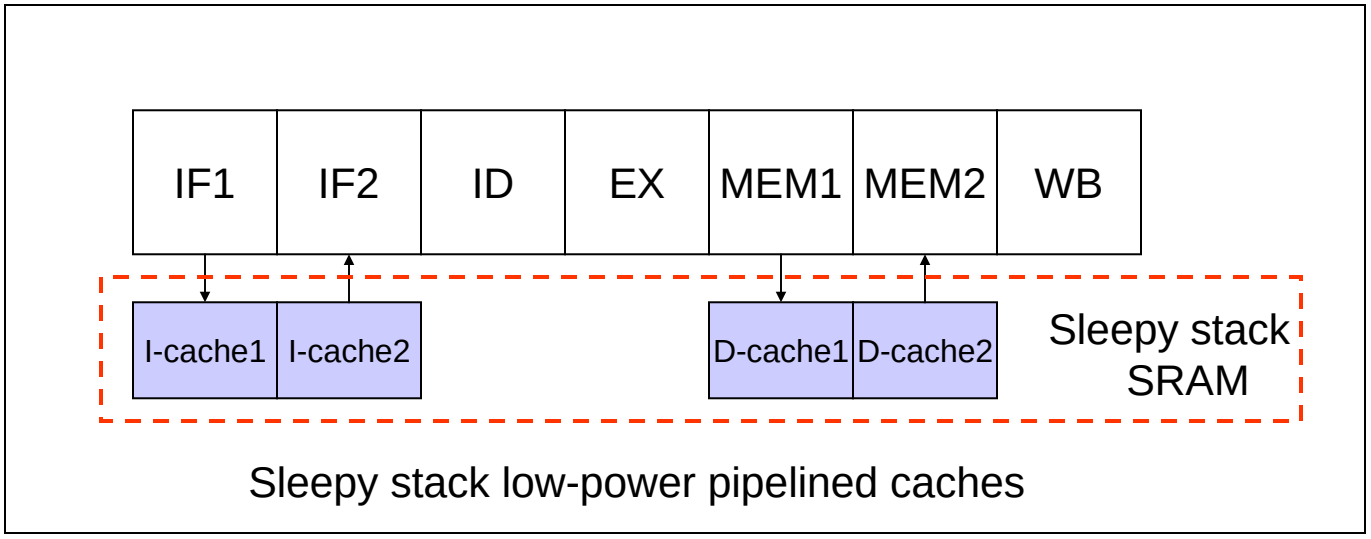
- 2-stage pipelined cache processor saves 20.43% of energy

Outline

- Introduction
- Related Work
- Sleepy stack
- Sleepy stack logic circuits
- Sleepy stack SRAM
- Low-power pipelined cache (LPPC)
- Sleepy stack pipelined SRAM
- Conclusion

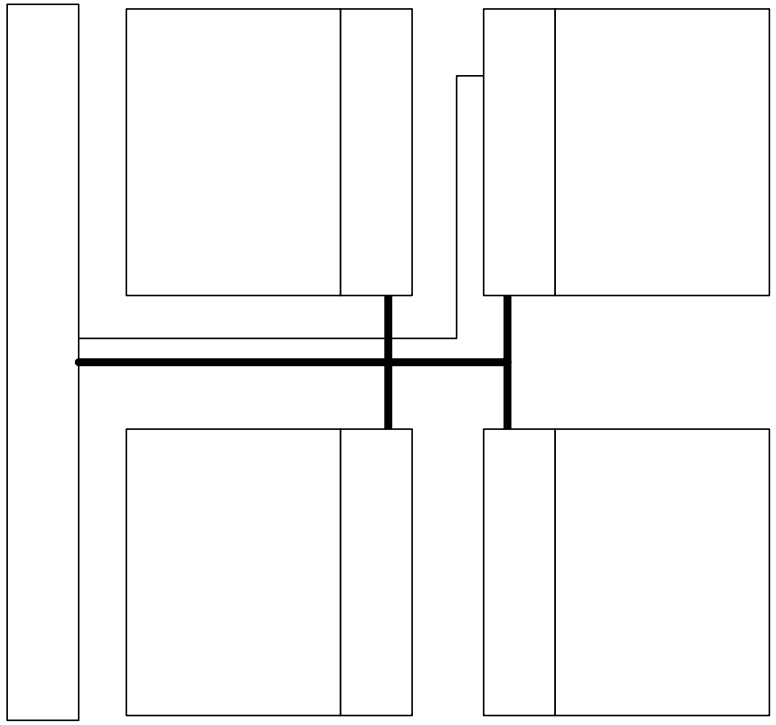
Sleepy stack pipelined SRAM

- Combine the sleepy technique and low-power pipelined cache (LPPC)
- Leakage reduction while maintaining performance
- Use 2-stage LPPC, i.e., 7-stage pipeline

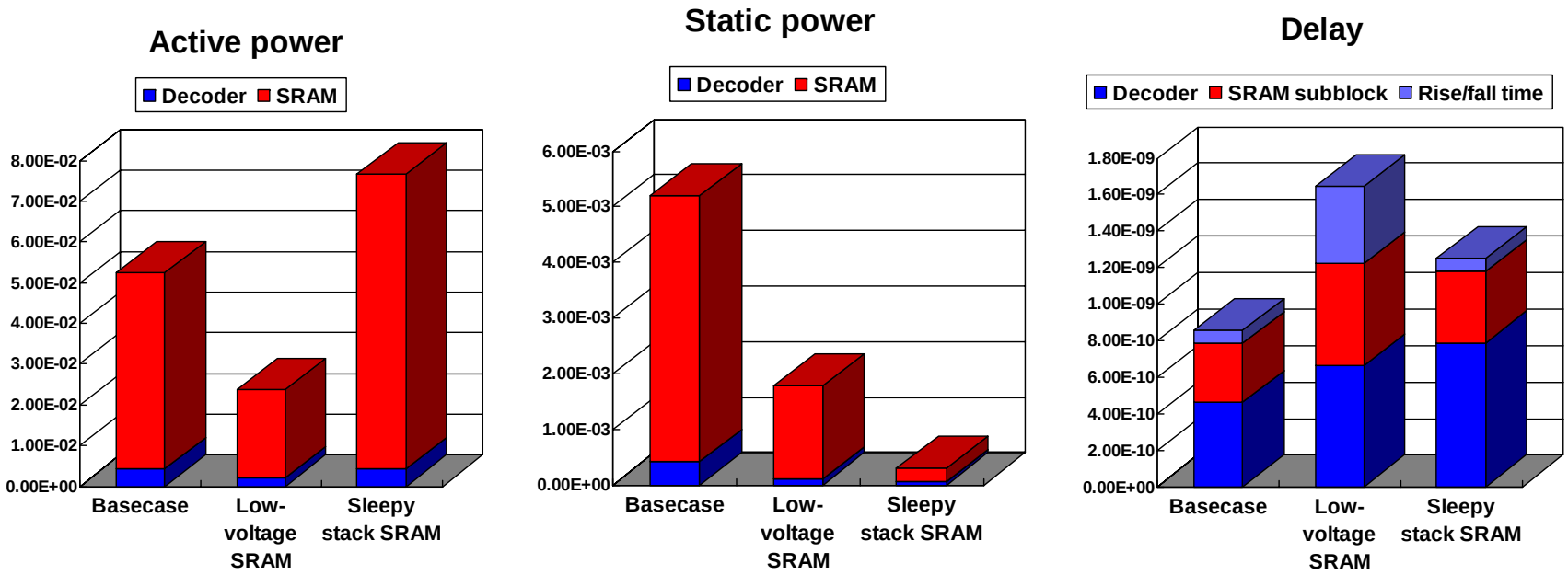


Methodology

- Model the base case 32KB SRAM with 4 subblocks targeting 0.07μ technology (V_{dd}=1.0V)
- Sleepy stack is applied to
  - SRAM cell
  - Pre-decoder and row-decoder except global wordline drivers
- Low-voltage pipelined SRAM with V_{dd}=0.7V
- Dynamic power, leakage power, and delay are measured using HSPICE
- Measured parameters are fed into Simplescaler/ARM to measure process performance

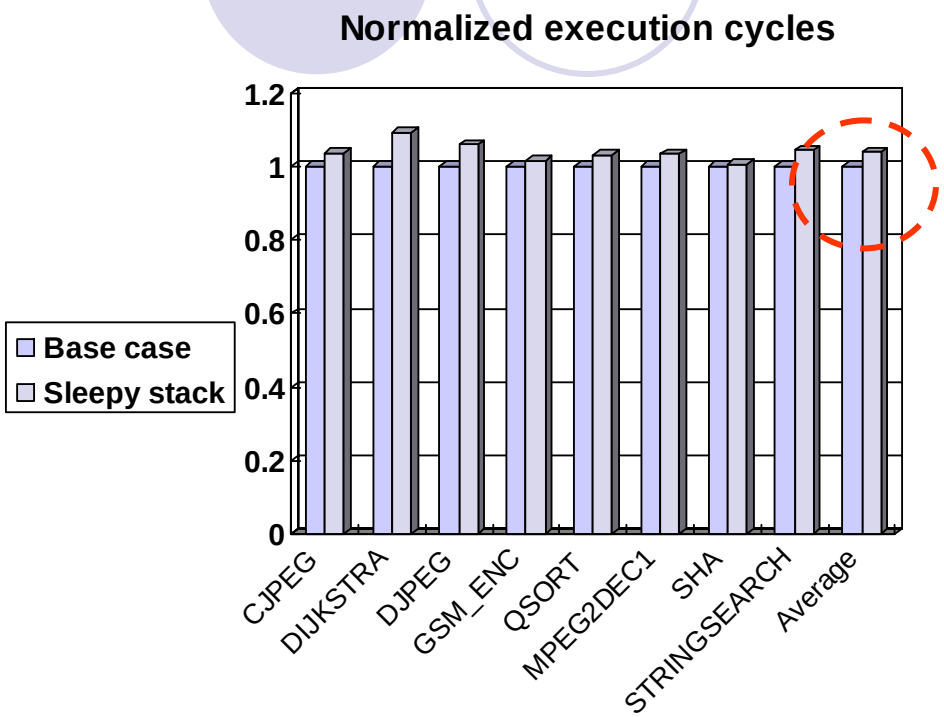


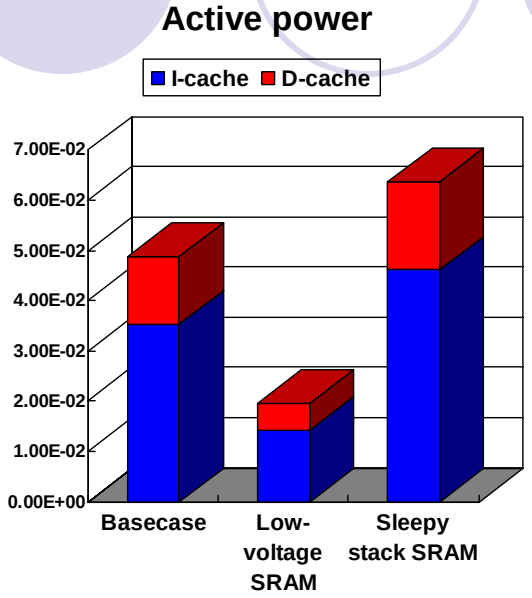
SRAM performance



- Active power increases 36% (sleepy stack) and decreases 58% (low-voltage SRAM)
- Sleepy stack SRAM achieves 17X leakage reduction (low-voltage SRAM 3X)
- Delay increases 33% (low-voltage SRAM 66%) (before pipelining)
- Estimated area overhead of sleepy stack is less that 2X

Processor performance





- Average 4% execution cycle increase with same cycle time (33% of delay increase before pipelining)
- Active power of sleepy stack pipelined SRAM increase 31% (low-voltage SRAM active power decreases 60%)
- When sleep mode is 3 times longer than active mode, the sleepy stack pipelined cache is effective to save energy

Conclusion and contribution

- Sleepy stack structure achieves dramatic leakage power reduction (4-inverters, 215X over forced stack) while saving state with some delay and area overhead
- Sleepy stack SRAM cell provides new pareto points in ultra-low leakage power consumption (2.77X over high-V_{th} with 19% delay increase or 2.26X without delay increase)
- Low-power pipelined cache reduces cache power by lowering cache supply voltage (2-stage pipelined cache 20% of energy with 4% delay increase)
- Sleepy stack pipelined SRAM achieves 17X leakage reduction with small execution cycle (4%) increase and less than 2X estimate area increase

Publications

[1] **J. C. Park**, V. J. Mooney and P. Pfeiffenberger, “Sleepy Stack Reduction in Leakage Power,” *Proceedings of the International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS’04)*, pp. 148-158, September 2004.

[2] P. Pfeiffenberger, **J. C. Park** and V. J. Mooney, “Some Layouts Using the Sleepy Stack Approach,” Technical Report GIT-CC-04-05, Georgia Institute of Technology, June 2004, [Online] Available http://www.cc.gatech.edu/tech_reports/index.04.html.

[3] A. Balasundaram, A. Pereira, **J. C. Park** and V. J. Mooney, “Golay and Wavelet Error Control Codes in VLSI,” *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC'04)*, pp. 563-564, January 2004.

[4] A. Balasundaram, A. Pereira, **J. C. Park** and V. J. Mooney, “Golay and Wavelet Error Control Codes in VLSI,” Technical Report GIT-CC-03-33, Georgia Institute of Technology, December 2003, [Online] Available http://www.cc.gatech.edu/tech_reports/index.03.html

[5] **J. C. Park**, V. J. Mooney and S. K. Srinivasan, “Combining Data Remapping and Voltage/Frequency Scaling of Second Level Memory for Energy Reduction in Embedded Systems,” *Microelectronics Journal*, 34(11), pp. 1019-1024, November 2003. Kluwer Academic/Plenum Publishers, pp. 211-224, May 2002.

Publications

[6] **J. C. Park**, V. J. Mooney, K. Palem and K. W. Choi, "Energy Minimization of a Pipelined Processor using a Low Voltage Pipelined Cache," *Conference Record of the 36th Asilomar Conference on Signals, Systems and Computers (ASILOMAR'02)*, pp. 67-73, November 2002.

[7] K. Puttaswamy, K. W. Choi, **J. C. Park**, V. J. Mooney, A. Chatterjee and P. Ellervee, "System Level Power-Performance Trade-Offs in Embedded Systems Using Voltage and Frequency Scaling of Off-Chip Buses and Memory," *Proceedings of the International Symposium on System Synthesis (ISSS'02)*, pp. 225-230, October 2002.

[8] S. K. Srinivasan, **J. C. Park** and V. J. Mooney, "Combining Data Remapping and Voltage/Frequency Scaling of Second Level Memory for Energy Reduction in Embedded Systems," *Proceedings of the International Workshop on Embedded System Codesign (ESCODES'02)*, pp. 57-62, September 2002.

[9] K. Puttaswamy, L. N. Chakrapani, K. W. Choi, Y. S. Dhillon, U. Diril, P. Korkmaz, K. K. Lee, **J. C. Park**, A. Chatterjee, P. Ellervee, V. J. Mooney, K. Palem and W. F. Wong, "Power-Performance Trade-Offs in Second Level Memory Used by an ARM-Like RISC Architecture," in the book Power Aware Computing, edited by Rami Melhem, University of Pittsburgh, PA, USA and Robert Graybill, DARPA/ITO, Arlington, VA, USA, published by Kluwer Academic/Plenum Publishers, pp. 211-224, May 2002.

[10] **J. C. Park** and V. J. Mooney, "Pareto Points in SRAM Design Using the Sleepy Stack Approach," *IFIP International Conference on Very Large Scale Integration* (IFIP VLSI-SOC'05), 2005.

