ISCA 2002

Dynamic Fine-Grain Leakage Reduction Using Leakage-Biased Bitlines

Seongmoo Heo, Kenneth Barr,
Mark Hampton, and Krste Asanović
Computer Architecture Group, MIT LCS

Leakage Power

- **Growing impact of leakage power**
  - **Increase of leakage power due to scaling of transistor lengths and threshold voltages**
  - **Power budget limits use of fast leaky transistors**
- **Challenge:**
  - **How to maintain performance scaling in face of increasing leakage power?**

Leakage Reduction Techniques

Static: Design-time Selection of Slow Transistors (SSST) for non-critical paths

- Replace fast transistors with slow ones on non-critical paths
- Tradeoff between delay and leakage power

Dynamic: Run-time Deactivation of Fast Transistors (DDFT) for critical paths

- DDFT switches critical path transistors between inactive and active modes

Observation:

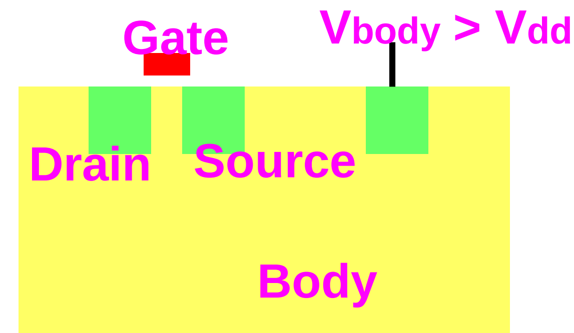
Critical paths dominate leakage after applying SSST techniques

Example: PowerPC 750

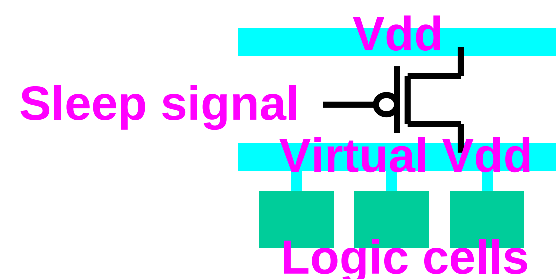
- **5% of transistor width is low Vt, but these account for *>50%* of total leakage.**

⇒DDFT could give large leakage savings

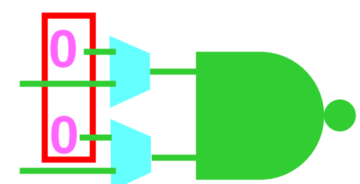
Existing DDFT Circuit Techniques



- **Body Biasing**
  - V_t increase by reverse-biased body effect
  - Large transition time and wakeup latency due to well cap and resistance



- **Power Gating**
  - Sleep transistor between supply and virtual supply lines
  - Increased delay due to sleep transistor



- **Sleep Vector**
  - Input vector which minimizes leakage
  - Increased delay due to mux and active energy due to spurious toggles after applying sleep vector

Fine-Grain DDFT Techniques

- Have to turn off small pieces of an active processor for short periods of time
  - Difficult to turn off large pieces for long periods

  → Fine-grain DDFT techniques
- Requirements of Fine-grain DDFT techniques
  - Circuits with low active delay penalty, low energy moving in and out of sleep, and fast wakeup time
  - Micro-architectural scheduling to keep the sleep time as long and often as possible
- Compare to coarse-grain DDFT techniques
  - O.S. puts whole processor to sleep for a long time $\Rightarrow$ doesn't save power when running code
  - Low steady-state leakage only concern.

Highlights of This Work

- We introduce metrics for comparing fine-grain dynamic deactivation techniques
  - Steady-stage leakage, Transition time, Fixed transition energy, Breakeven time
- We present a new circuit-level leakage reduction technique, Leakage-Biased Bitlines (LBB)
  - Low deactivation energy and fast wakeup
- We save leakage power of I-Cache and Multiported regfile by LBB
  - I-cache: idle subbank deactivation
  - Multiported regfile: idle read ports and dead register deactivation

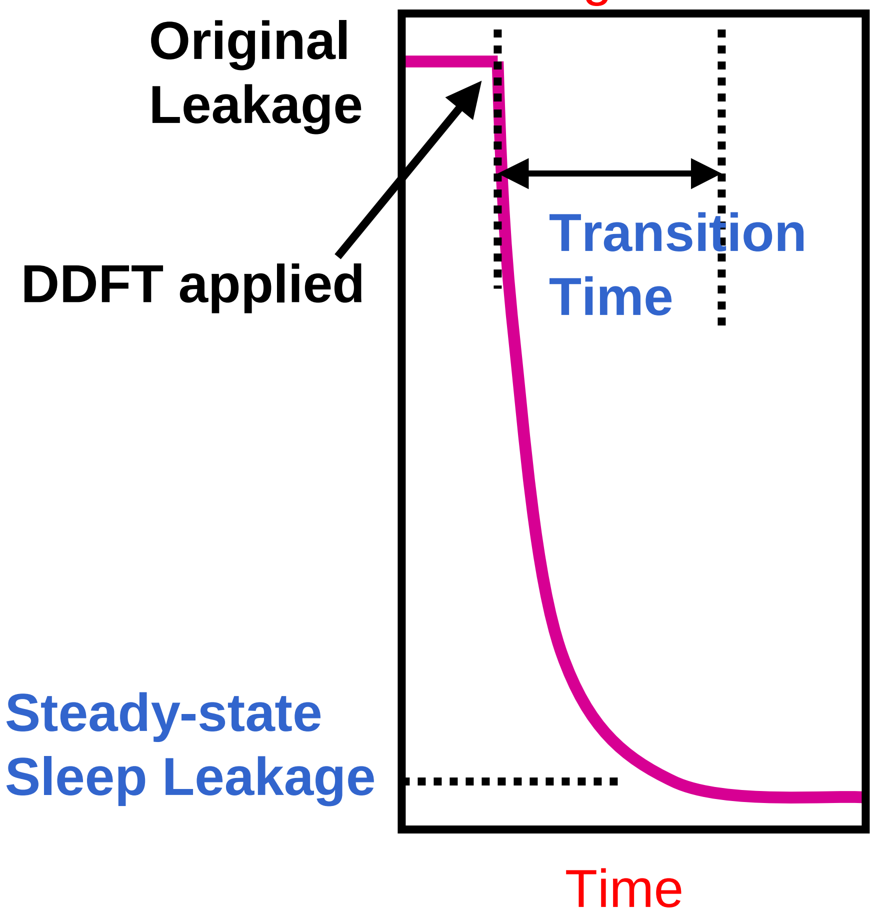
Outline

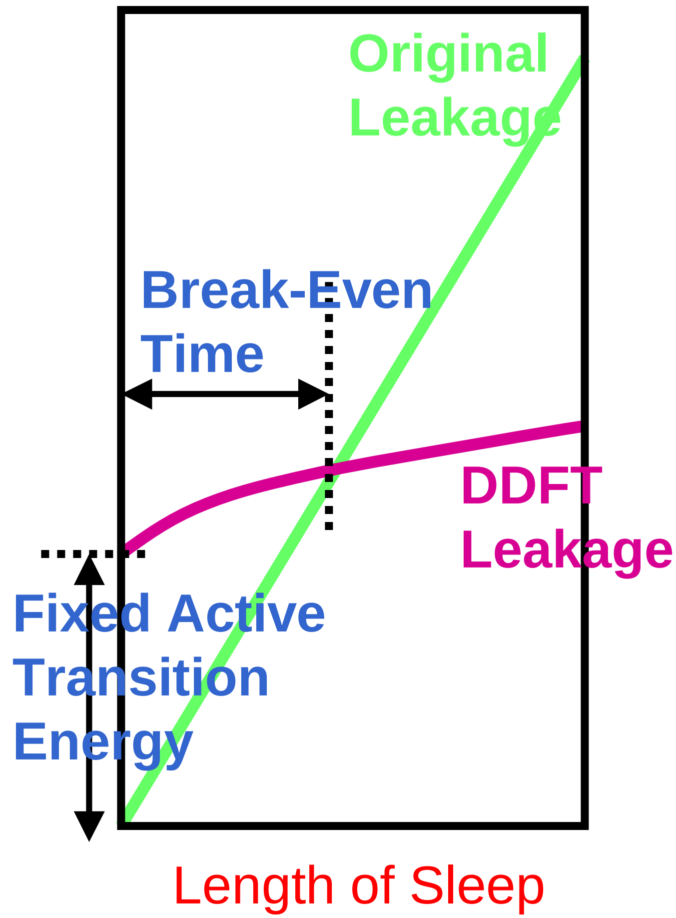
1. Methodology and DDFT Metrics
2. Cache Leakage Saving
   - Idle subbank deactivation
3. Multiported Regfile Leakage Saving
   - Dead reg deactivation (Horizontal)
   - Idle read port deactivation (Vertical)
4. Conclusion

Methodology

- **Process Technology**
  - 180nm DVT process modeled after 0.18um TSMC LVT and MVT processes
  - Scaled to 130, 100, and 70nm processes based on SIA roadmap
  - Optimistic/pessimistic leakage prediction: 2x/4x increase of leakage current density (nA/um)
- **Evaluation with SimpleScalar**
  - Modified to model unified physical register file
  - 4 issue, 100 integer physical regs, 16KB/4-Way/32-B block I-Cache and D-Cache, Unified L-2 Cache
  - SPECint95 refs
- **Energy measurements**
  - Hspice simulation for 180nm process and scaled to other processes accordingly

Metrics for Fine-Grain DDFT Techniques





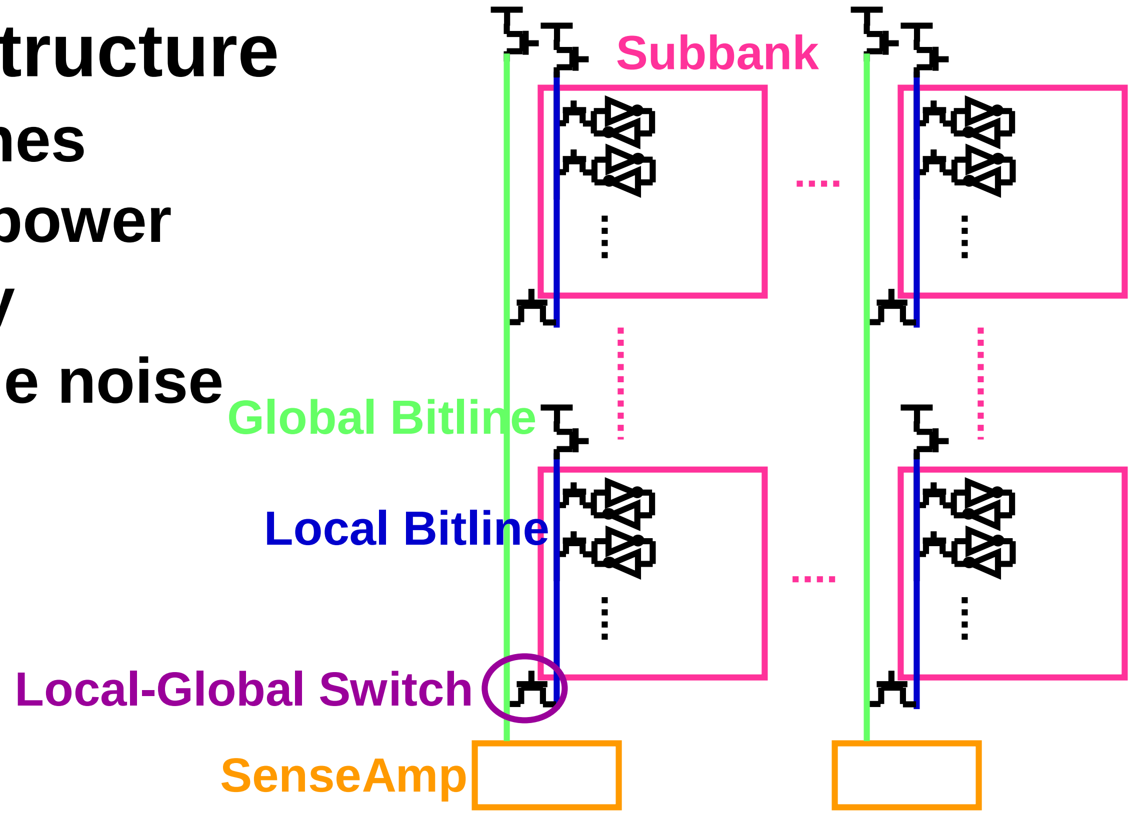
- Wakeup Latency
- Active delay and power

L1 Cache and Multiported Regfile

- **Good targets for Fine-grain DDFT techniques**
  - **Timing-critical**
    - **Contrast: L2 cache is a better target for SSST (long channel or HVT transistors)**
  - **Large leakage current**
    - **Cache: Large number of fast transistors**
    - **Multiported Regfile: Ever increasing number of registers and ports**
      - ***Alpha 21464 register file is 5x larger than 64KB data cache***

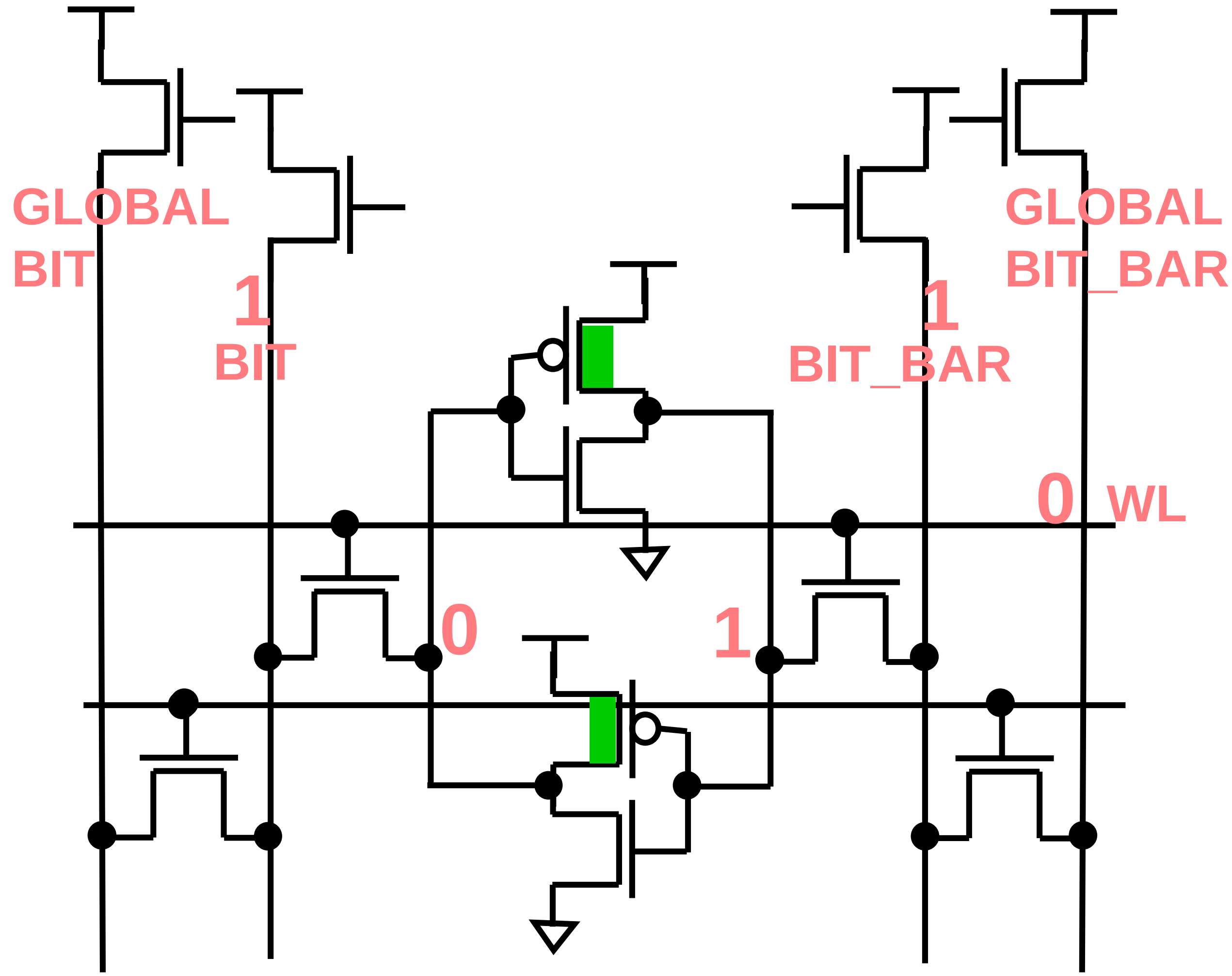
LBB for Caches

- Modern cache structure
  : Hierarchical Bitlines
  - To save active power
  - To reduce delay
  - To reduce bitline noise



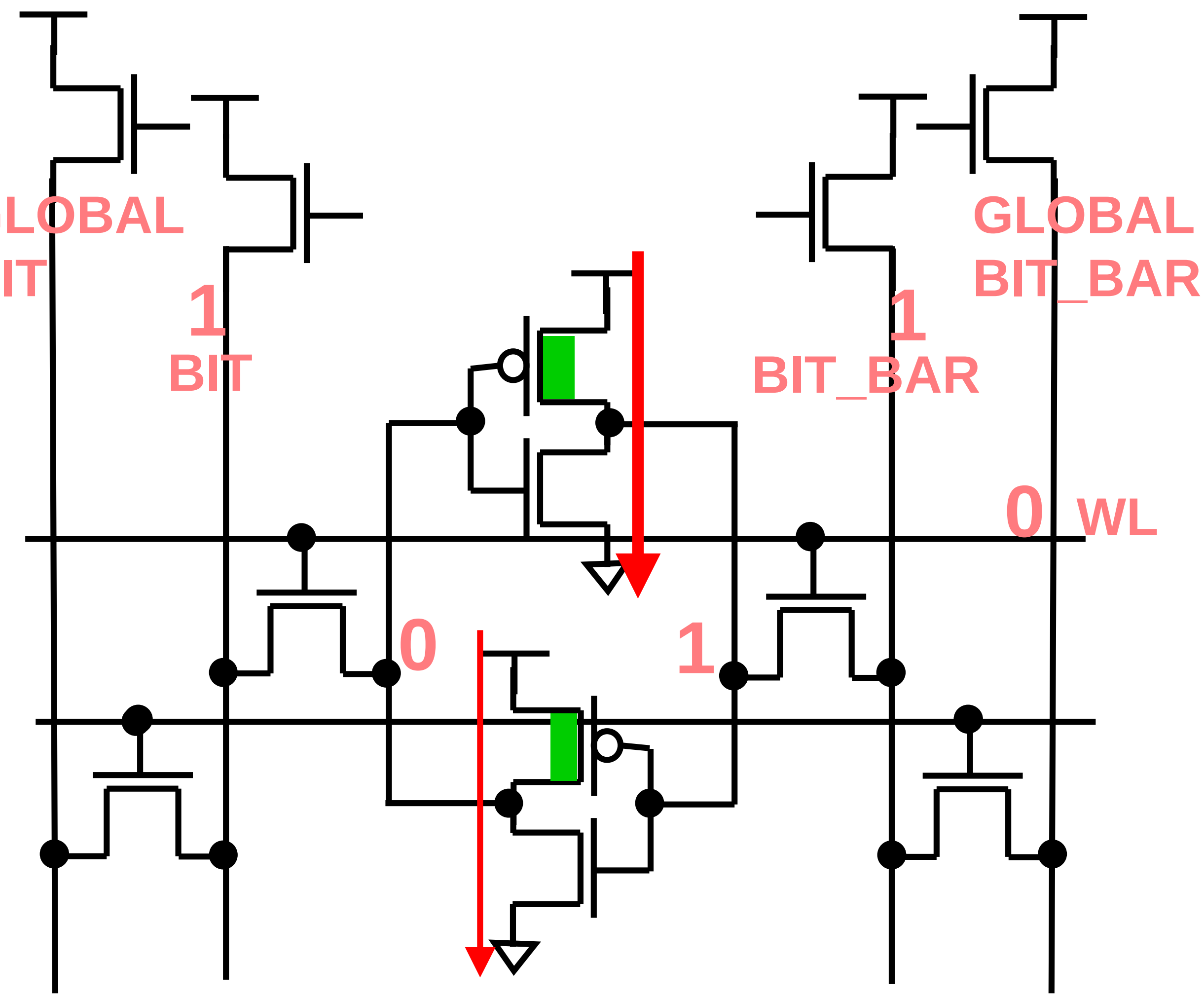
- Local bitlines (32-bit cells) disconnected from senseamp by local-global switch.
- LBB for Caches: If a subbank is not in use, turn off precharge transistors and delay precharging.

Cache: Dual Vt SRAM cell

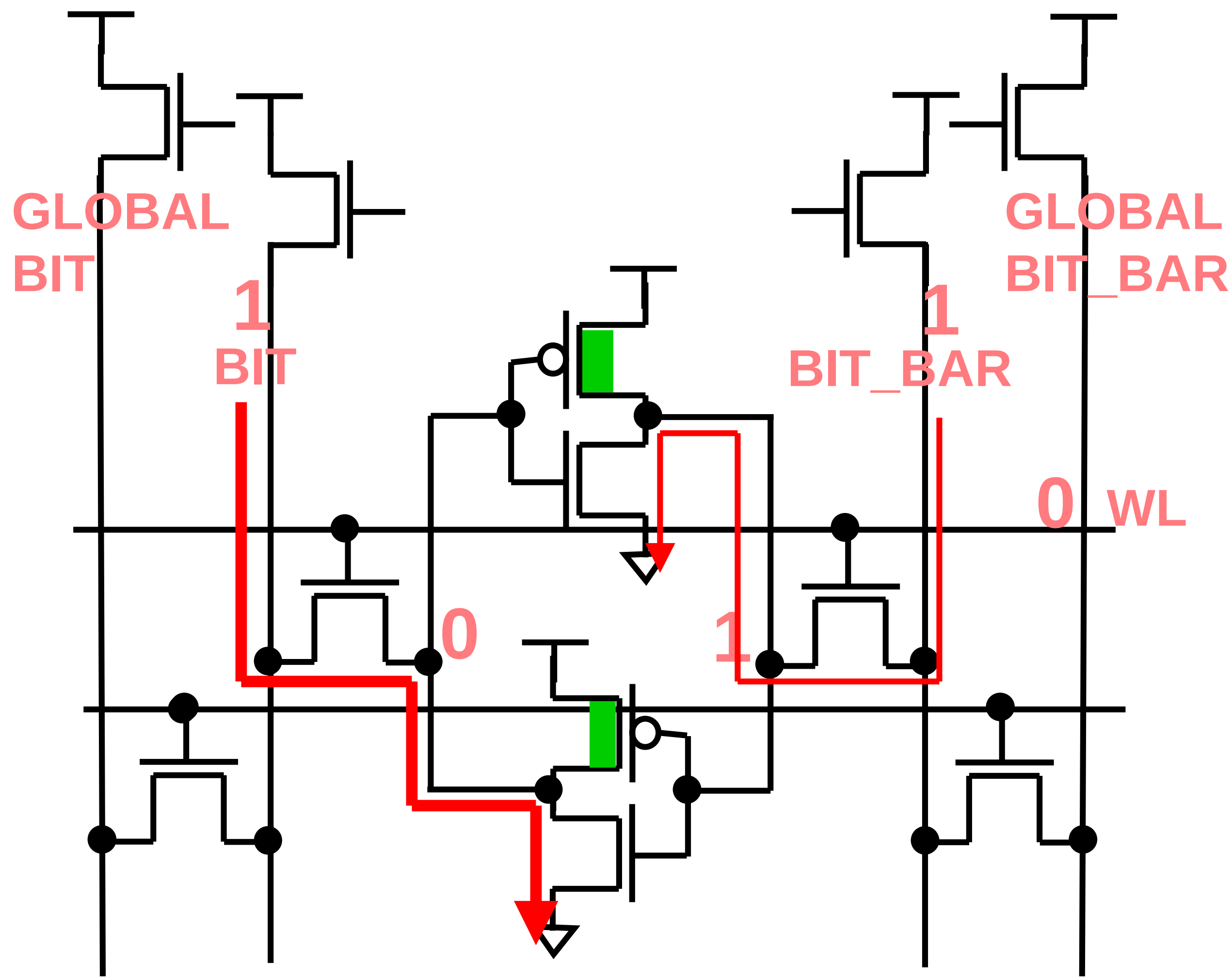


HVT transistors: green-colored

Cache: Dual Vt SRAM cell

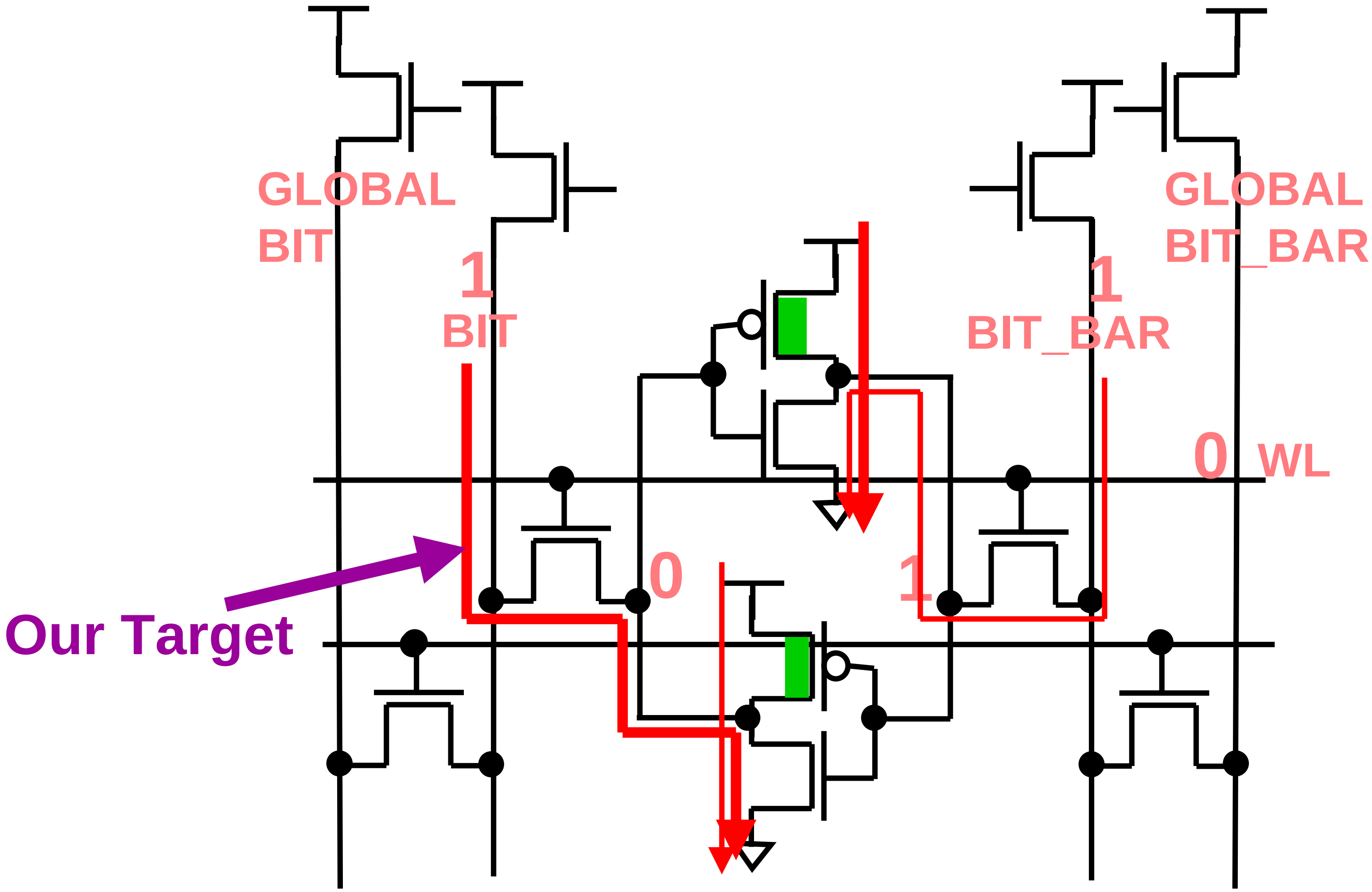


Cache: Dual Vt SRAM cell

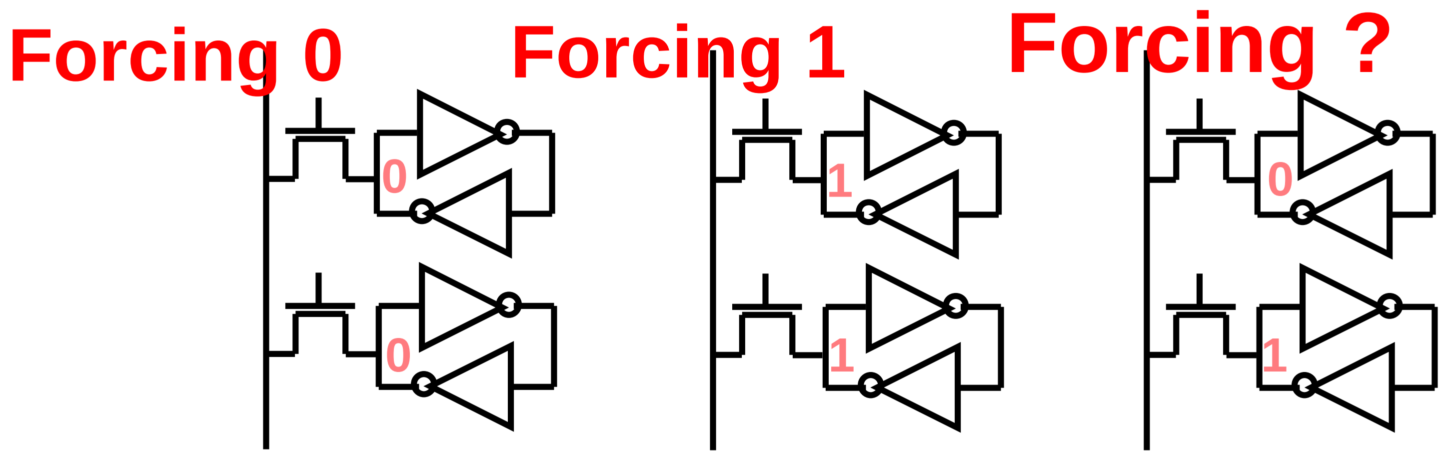


Bitline leakage depends on the stored value

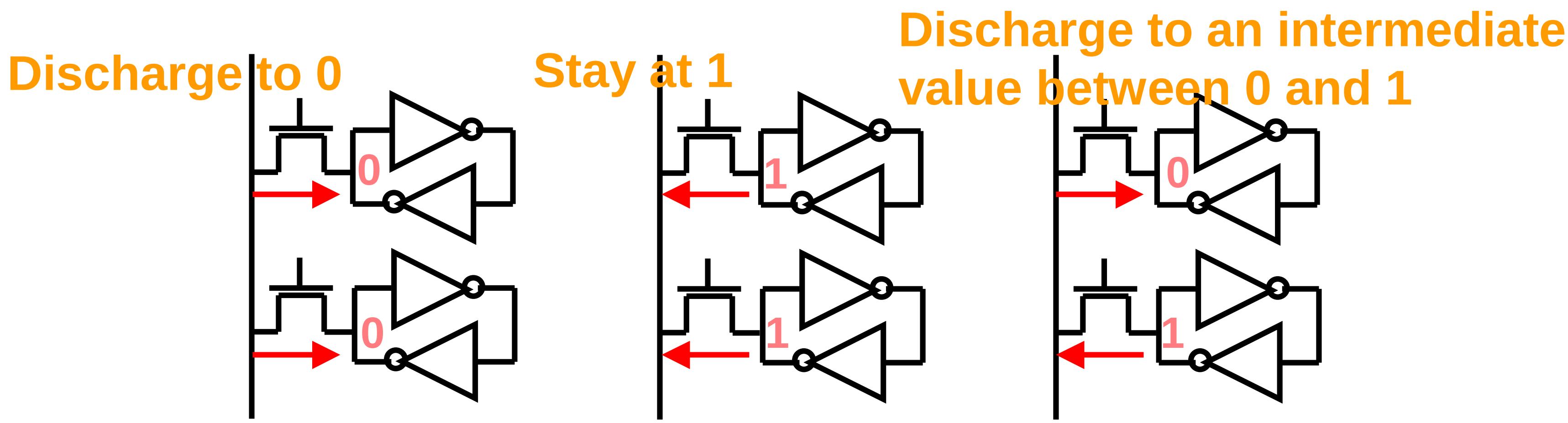
Cache: Dual Vt SRAM cell



Bitline leakage depends on the stored value


Forcing 0
0
0
Forcing 1
1
1
Forcing ?
0
1

Leakage-Biased Bitlines (LBB)

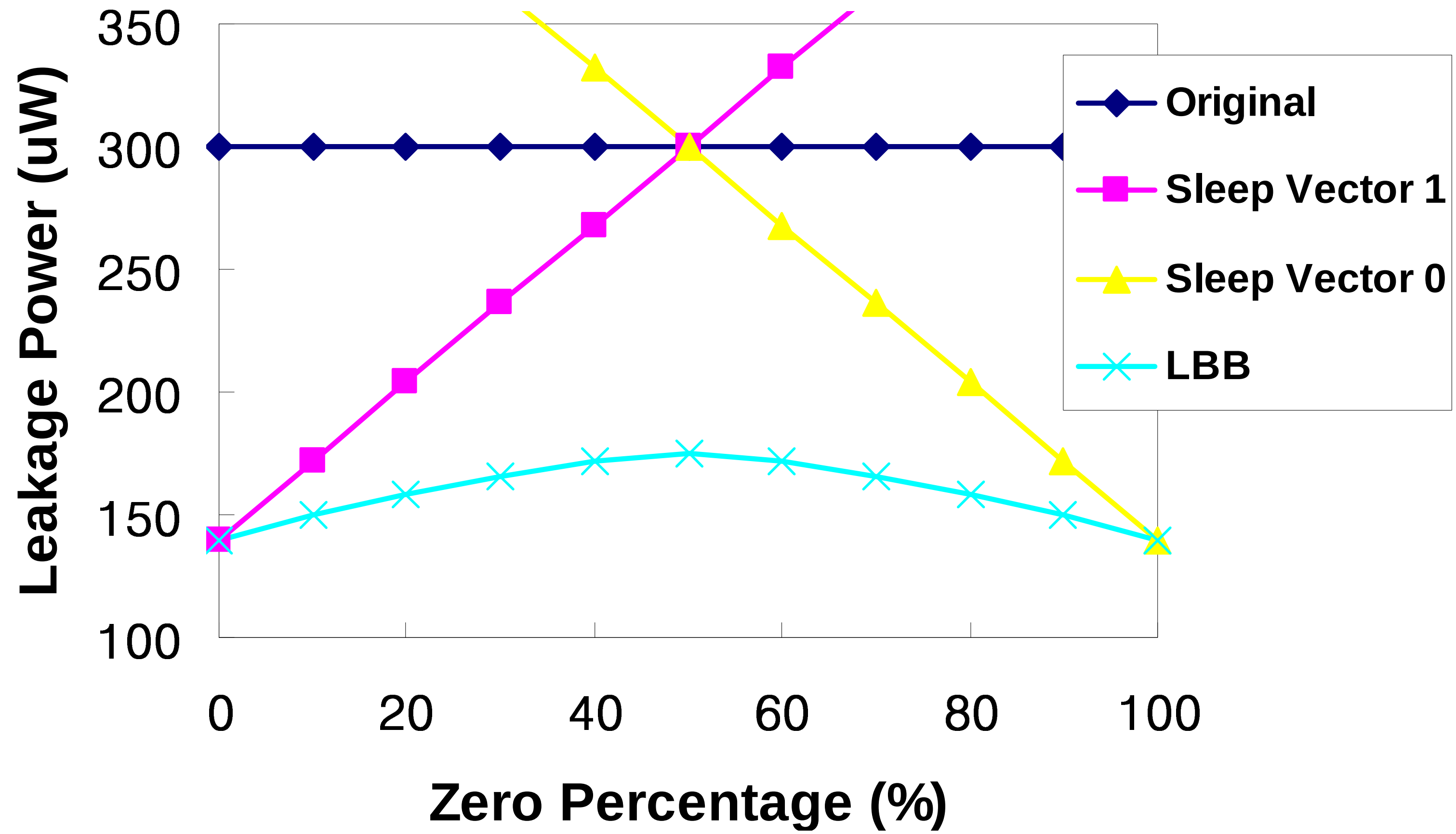


- **LBB lets bitlines float by turning off the local HVT NMOS precharge transistors**
  - **No static current draw because local bitline isolated**
  - ***LBB uses leakage itself to bias bitlines to the voltage which minimizes leakage!***
- **A good fine-grain dynamic technique**
  - **Minimal transition energy:**
    - **Same number of precharges (delayed precharge)**
  - **Minimal transition time:**
    - **Wakeup latency is only that of precharge phase**

LBB versus Sleep Vector

- **LBB finds the minimal leakage state.**
  - **Always better than sleep vectors**

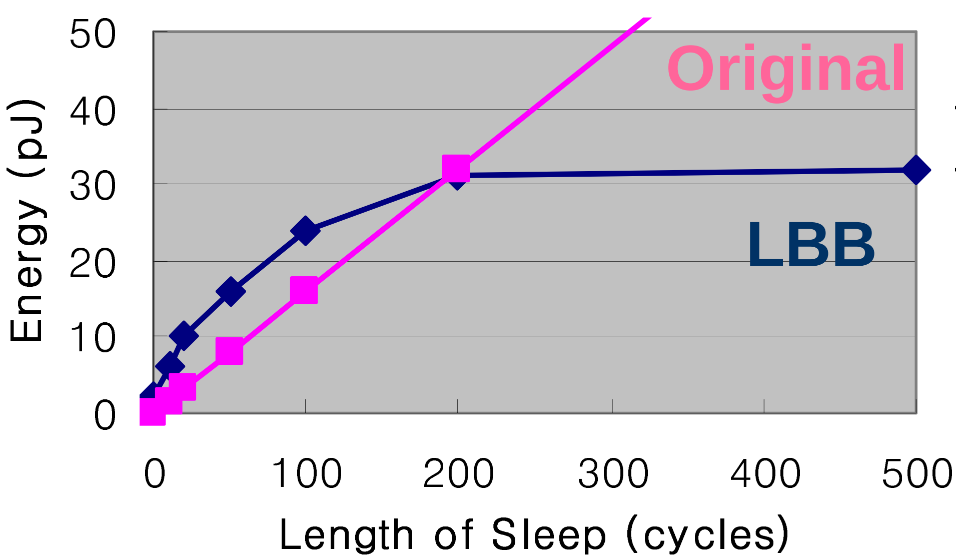
Leakage Power of 32x16B SRAM subbank



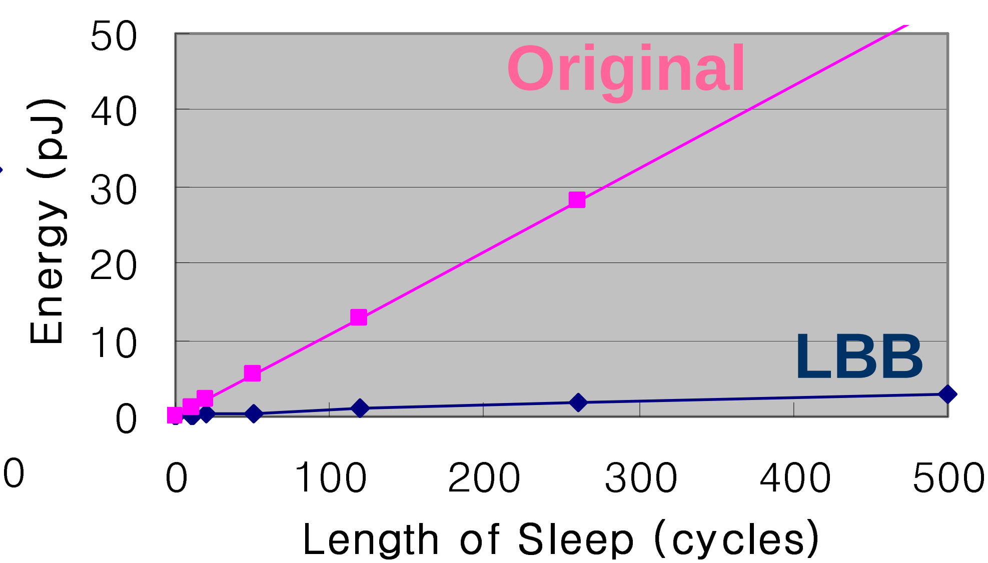
Cumulative Leakage Energy

32-row x 32B SRAM subbank
(optimistic leakage current used. 75% zero assumed)

180nm



70nm



Dynamic energy cost: Need to replace the lost charge

- **-LBB curve increases fast in the beginning**

Decrease of Breakeven time

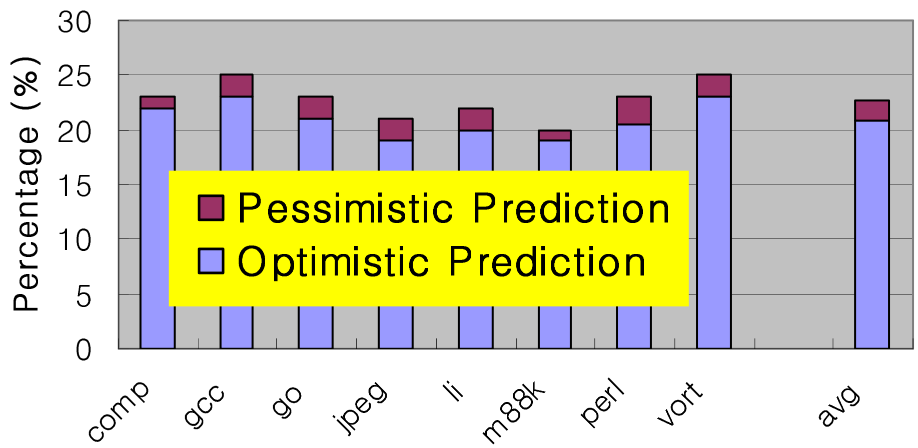
- **-180nm: 200 cycles, 70nm: less than a cycle**
- **-Active energy scales down faster than leakage energy**

Performance Issues for LBB Caches

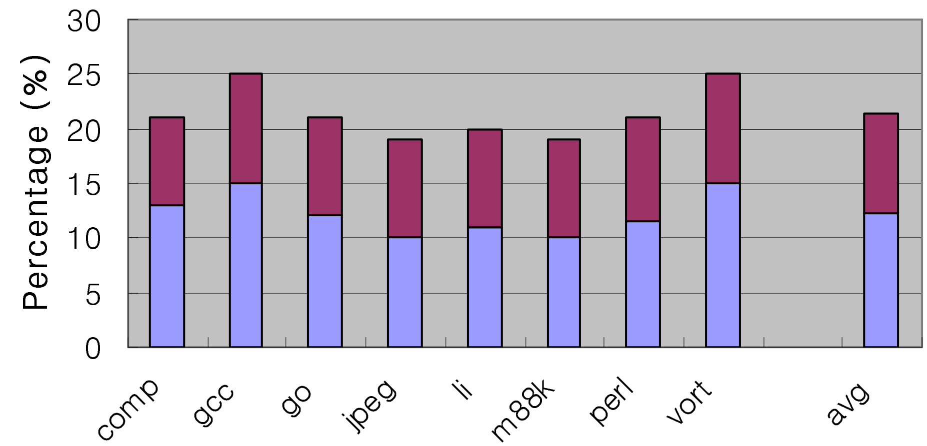
- **Subbank must be precharged before use**
  - Case 1 (best): subbank decode and precharge happen before more complex word-line decode, therefore no penalty.
  - Case 2 (worst): add additional pipeline stage for precharge
    - One cycle increase in branch misprediction penalty
  - Focus on I-Cache because any latency increase can be partly hidden by branch prediction

I-Cache Subbank Deactivation

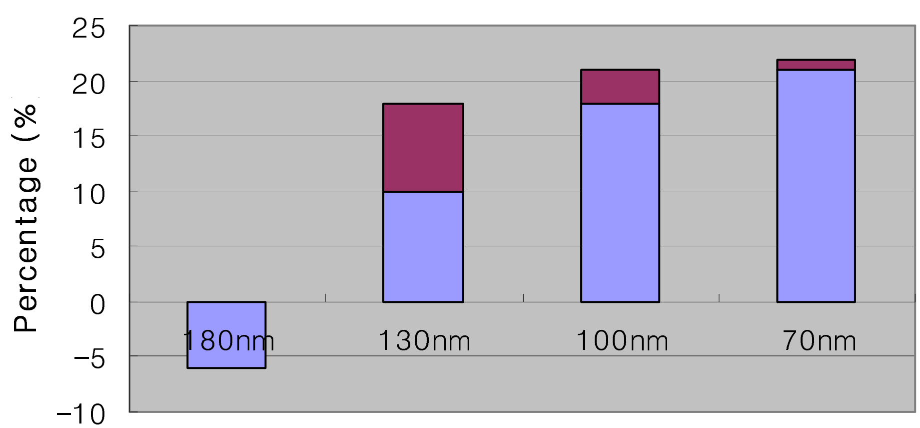
Leakage energy saving at 70nm process



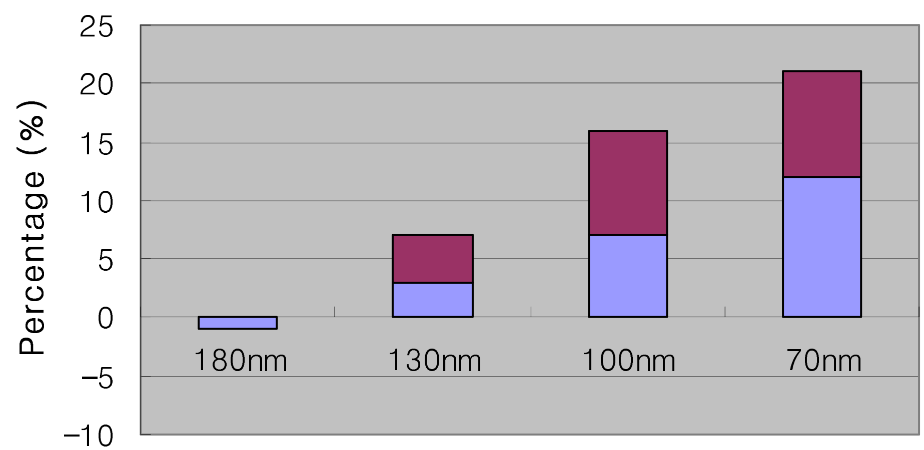
Total energy saving at 70nm process



Leakage energy saving across processes



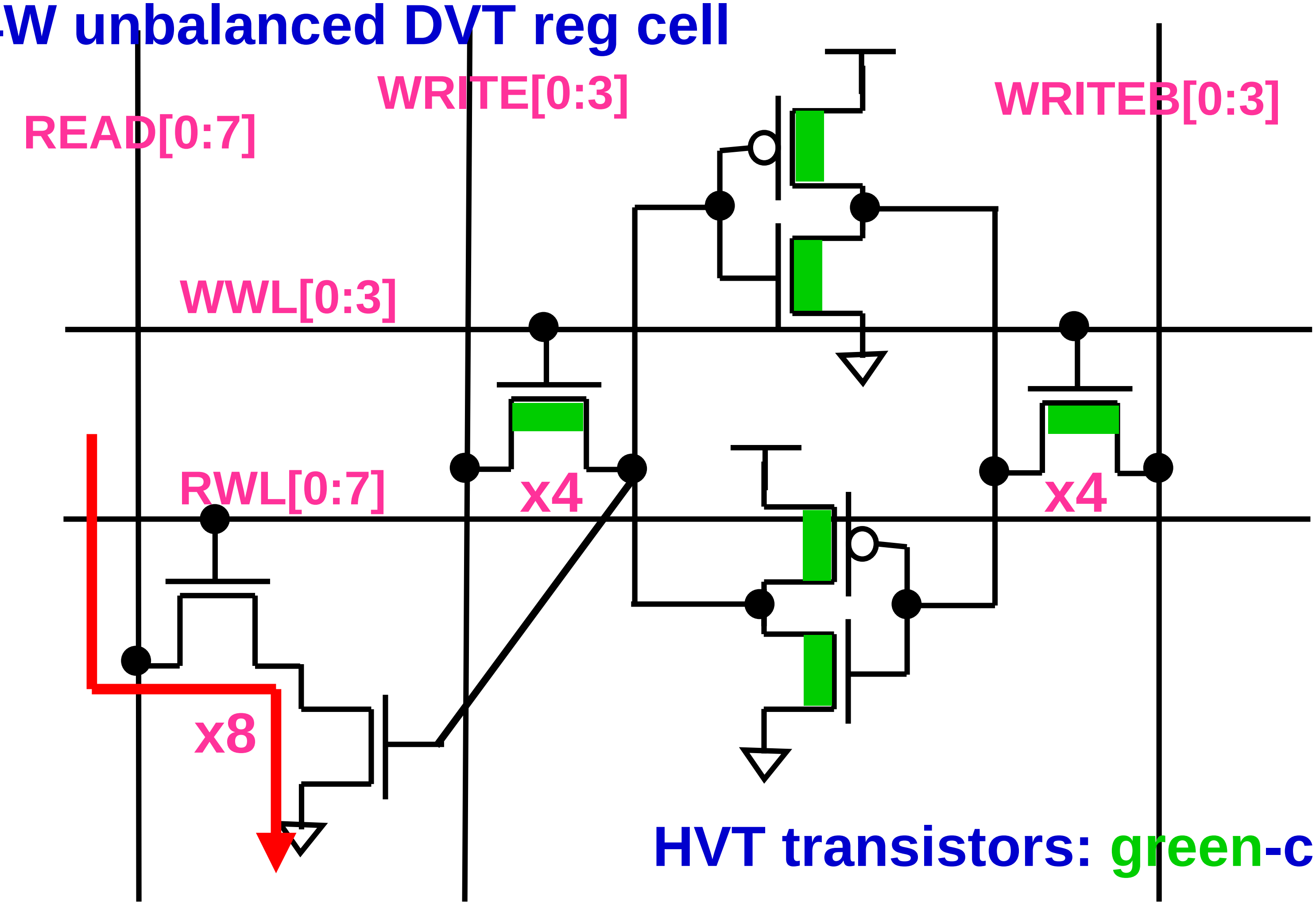
Total energy saving across processes



Case 2 (worst) assumption (adding additional pipeline stage)
→ 2.5% IPC decrease on average

Multiported Regfile Cell

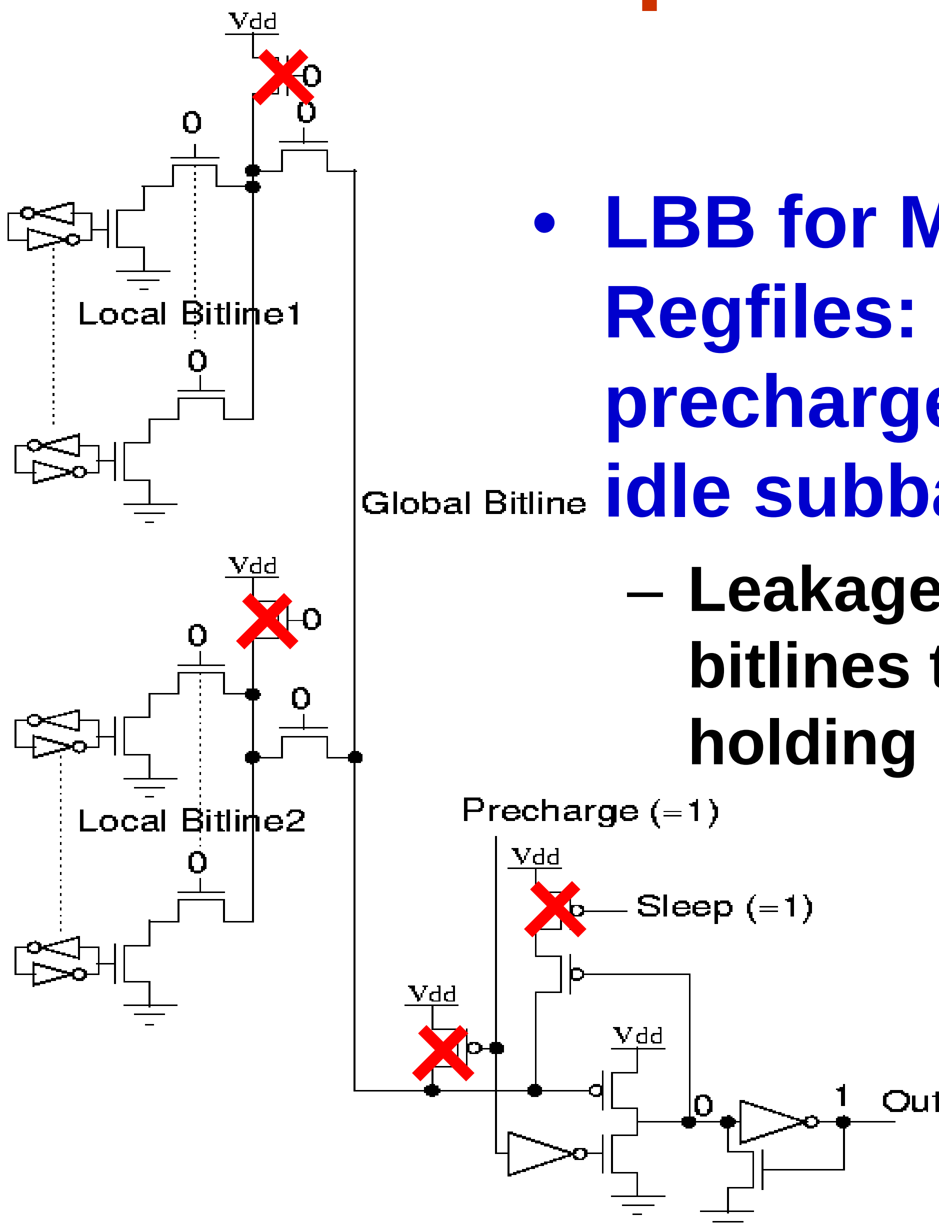
8R, 4W unbalanced DVT reg cell



HVT transistors: green-colored

- Simplified but active/leakage power-aware baseline

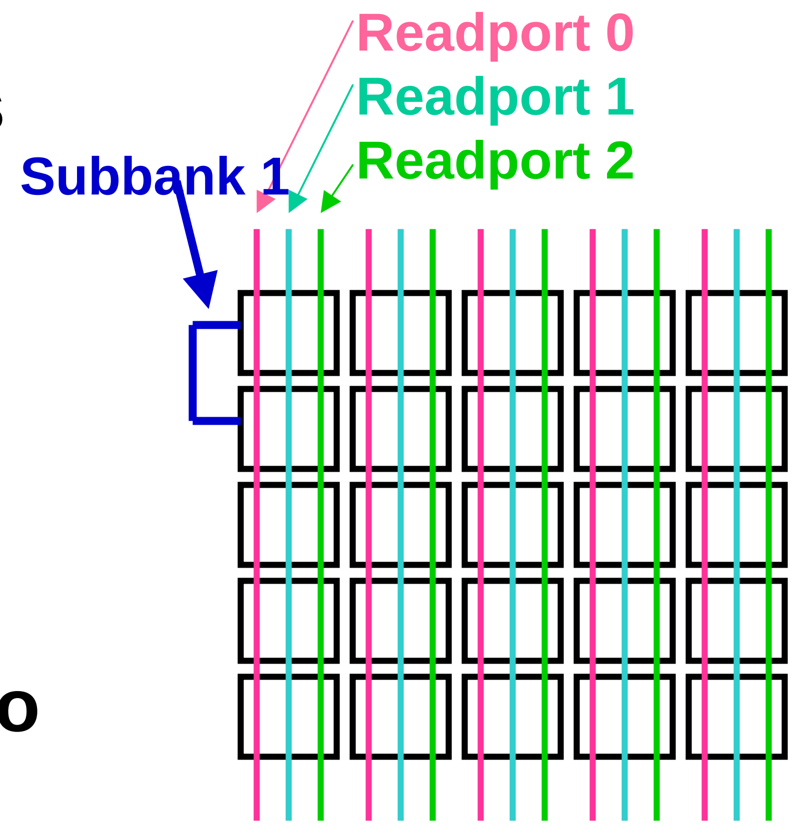
LBB for Multiported Regfiles



- **LBB for Multiported Regfiles: Turn off the precharge transistor on idle subbank read ports**
  - **Leakage current discharges bitlines to 0 if any bits are holding 1.**

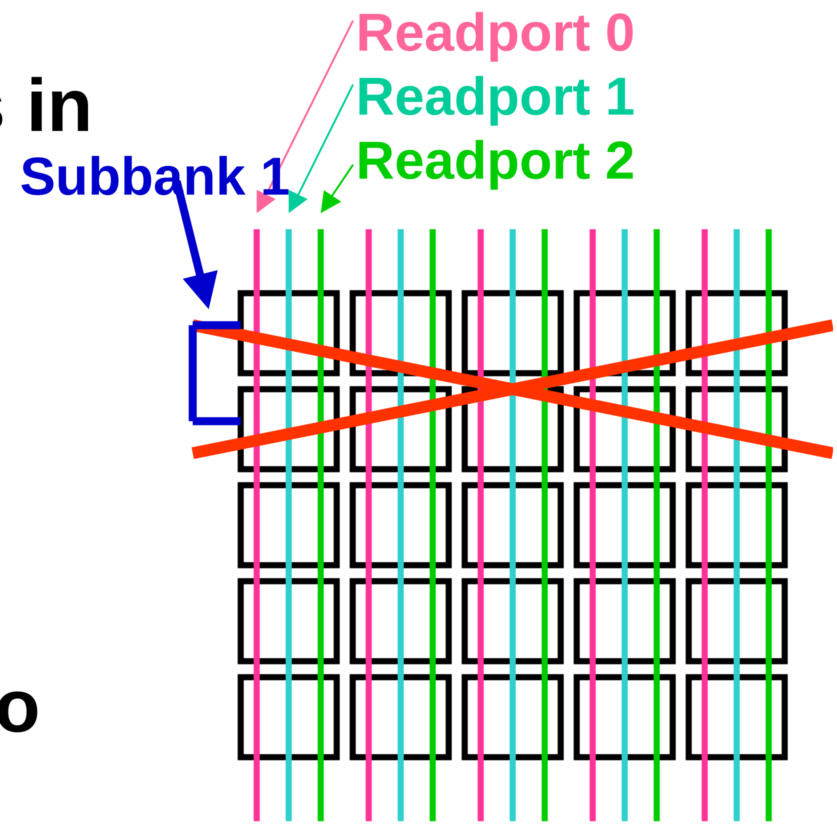
Dead Register Deactivation

- *Horizontal technique*
- Dead registers = Registers in free list
- If all registers in a subbank are dead, all read ports in the subbank are turned off by LBB
- No performance penalty since there is ample time to re-precharge between allocation and write.



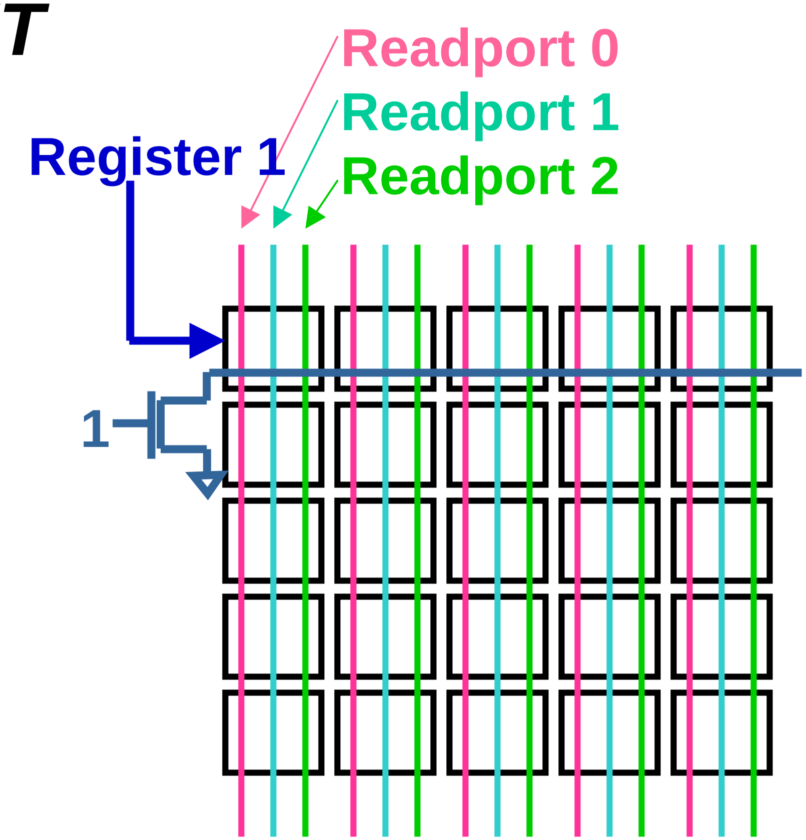
Dead Register Deactivation

- *Horizontal technique*
- Dead registers = Registers in free list
- If all registers in a subbank are dead, all read ports in the subbank are turned off by LBB
- No performance penalty since there is ample time to re-precharge between allocation and write.



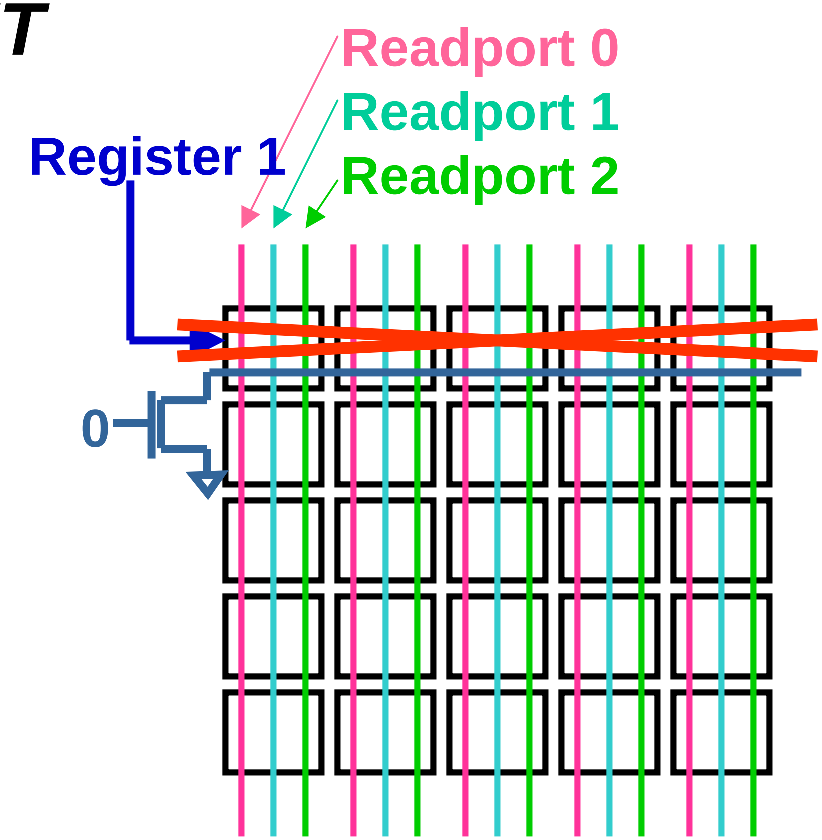
NMOS Sleep Transistor (NST)

- ***Alternative horizontal DDFT***
- **To turn off dead registers using NMOS sleep transistors (NST)**
- **Advantage: registers can be turned off individually**
- **Disadvantage: increased read access time**
  - **Set delay penalty to 5% (tradeoff between delay and leakage)**



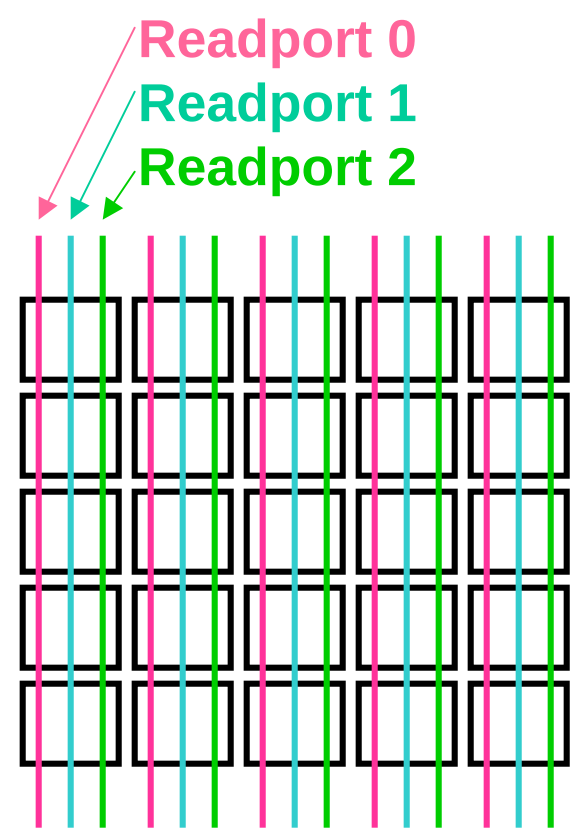
NMOS Sleep Transistor (NST)

- *Alternative horizontal DDFT*
- To turn off dead registers using NMOS sleep transistors (NST)
- Advantage: registers can be turned off individually
- Disadvantage: increased read access time
  - Set delay penalty to 5% (tradeoff between delay and leakage)



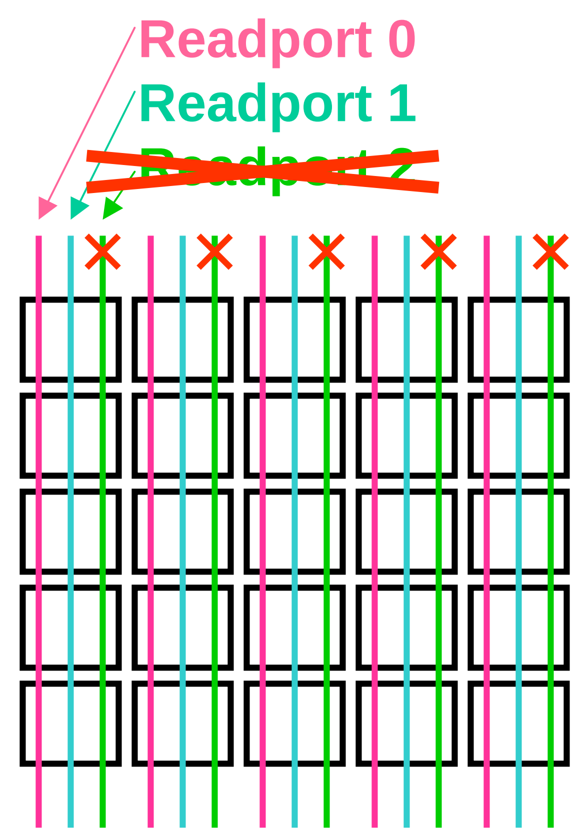
Idle Readport Deactivation

- *Vertical technique*
- Idle read ports when fewer than max # of instructions are issued in a superscalar machine
- Idle read ports deactivated by LBB
- No performance penalty since it is known whether a read port is needed before it is known which register will be accessed in the pipeline.



Idle Readport Deactivation

- *Vertical technique*
- Idle read ports when fewer than max # of instructions are issued in a superscalar machine
- Idle read ports deactivated by LBB
- No performance penalty since it is known whether a read port is needed before it is known which register will be accessed in the pipeline.

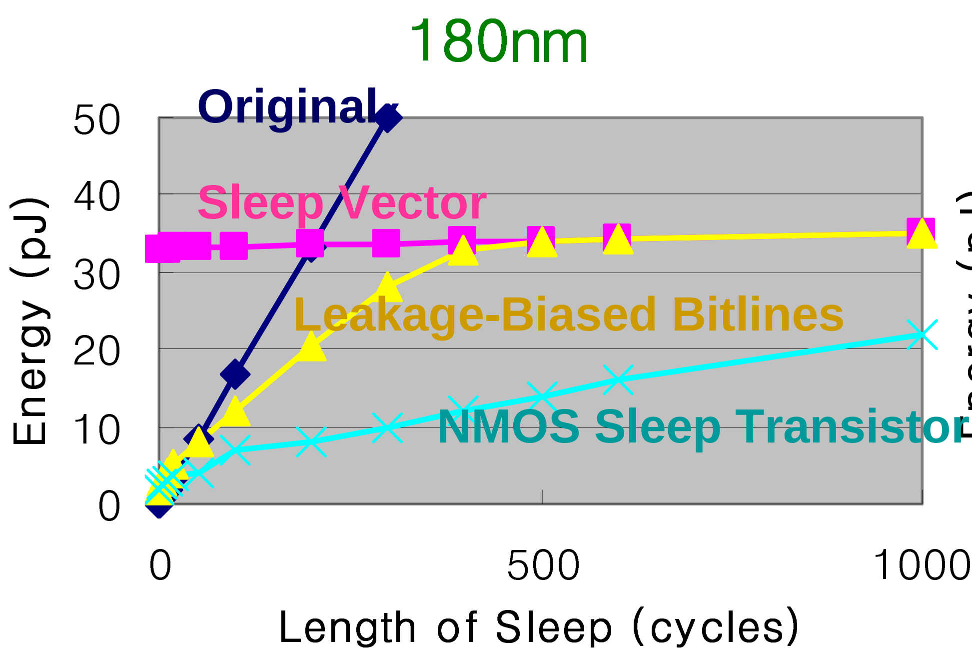


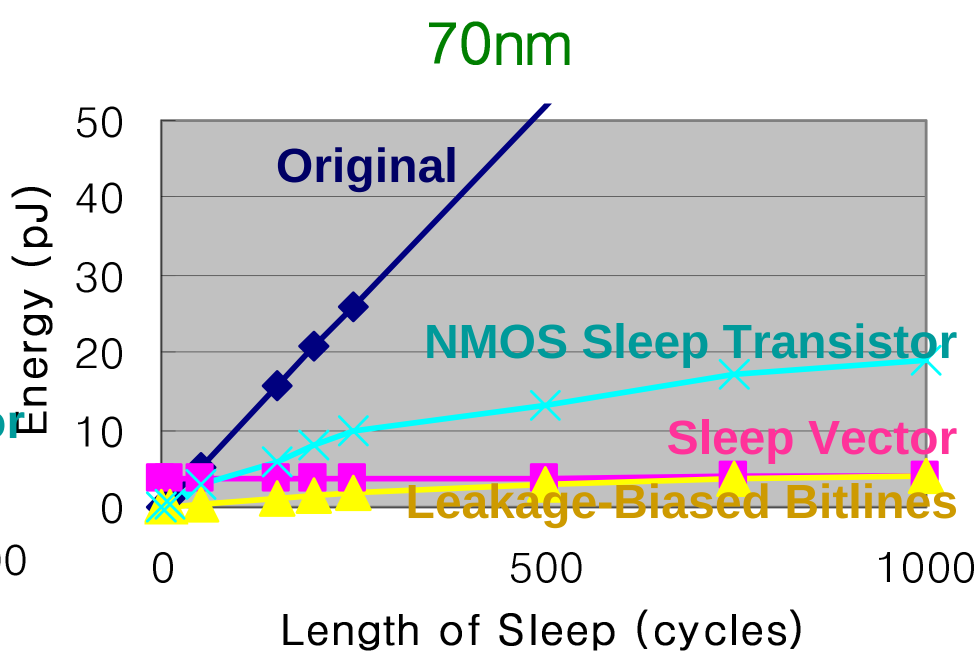
Comparison of DDFTs

32 x 32-b Regfile subbank

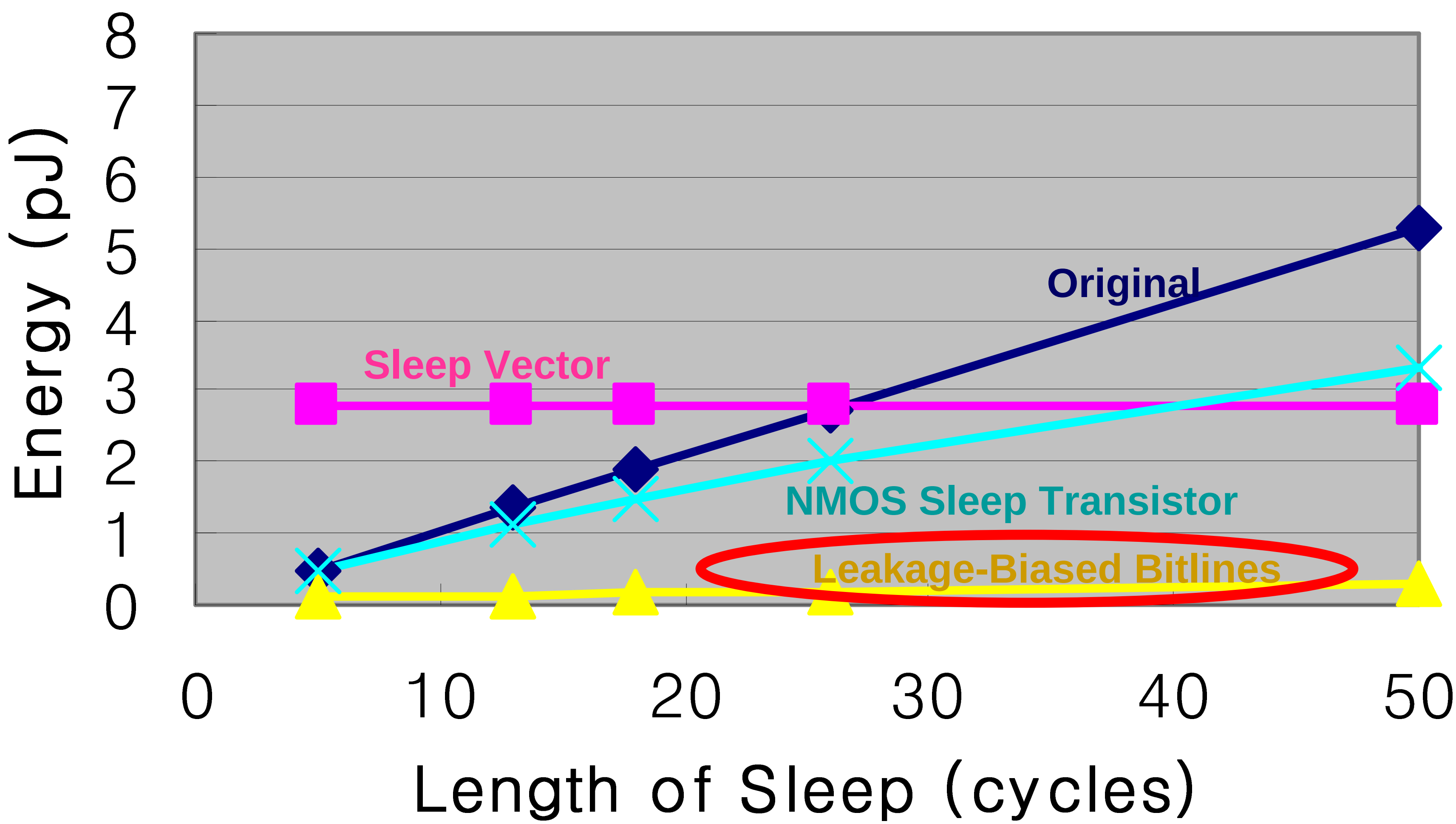
(75% zero assumed. Optimistic leakage current used.)

Process Tech. (nm)	180	130	100	70
Original (uW)	177.9	214.1	263.6	276.7
SV steady-state (uW)	2.0	2.4	3.0	3.1
LBB steady-state (uW)	2.0	2.4	3.0	3.1
NST steady-state (uW)	1.8	2.2	2.7	2.9





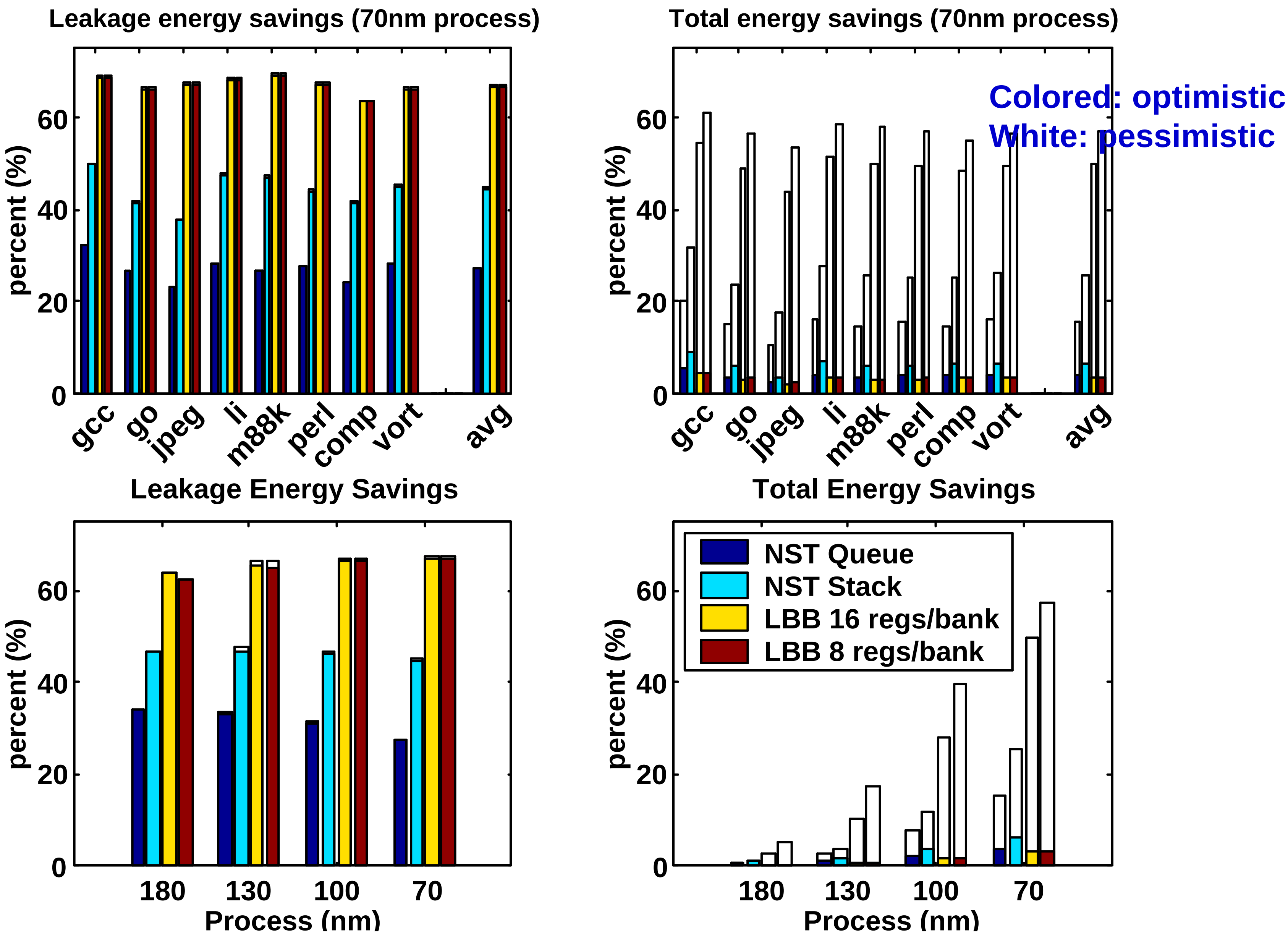
Comparison of DDFTs
Blowup: 70nm



Dead Register/Subbank Deactivation Policies

- Free list policies for NST (NMOS Sleep Transistor): queue and stack
  - queue: conventional
  - stack: keeps some regs dead for longer
    - 2.4-10% greater savings than queue at 70nm
    - Benefit increases as feature sizes shrink
- Subbank allocation policy for LBB: stack
  - Allocate a new subbank only when the previous bank is empty of dead registers

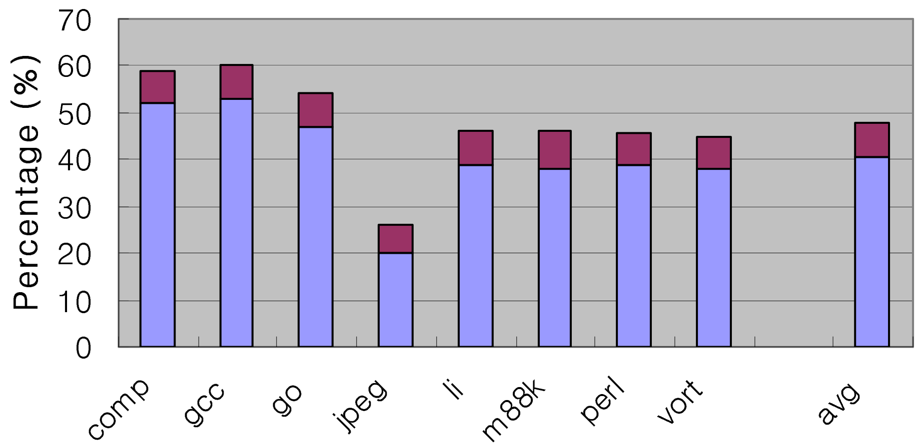
Dead Reg Deactivation (Horizontal)



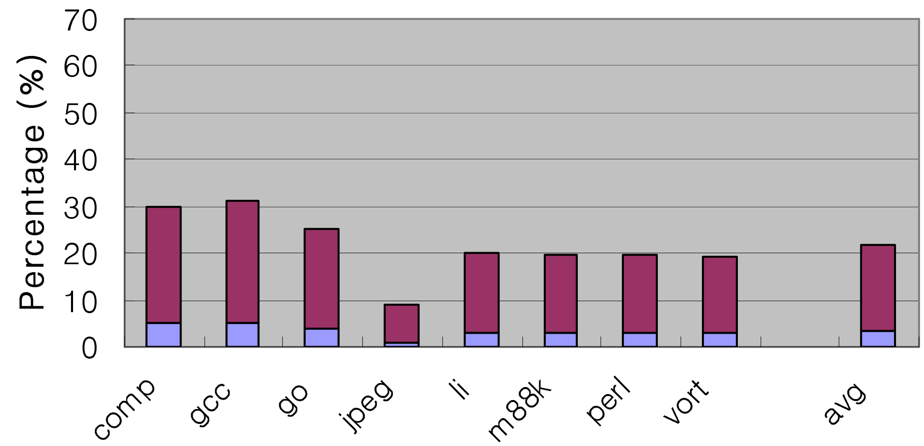
NST stack better than NST queue, LBB stack better than either NST

Read Port Deactivation (Vertical)

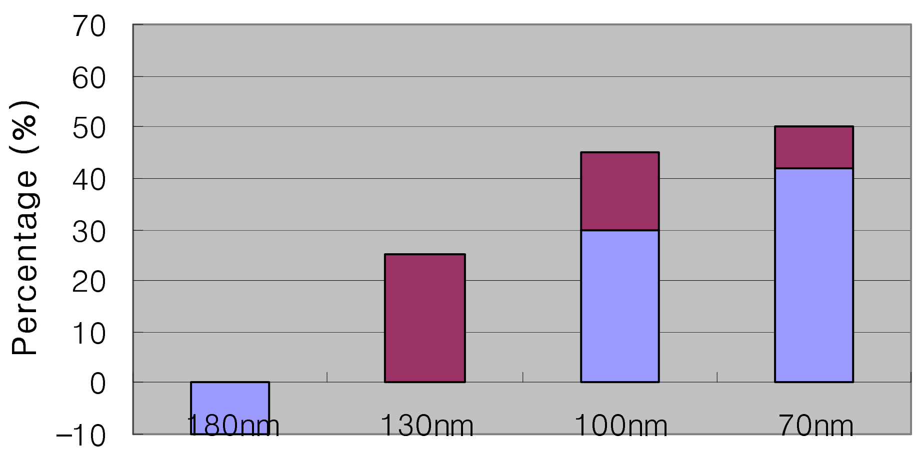
Leakage energy saving at 70nm process



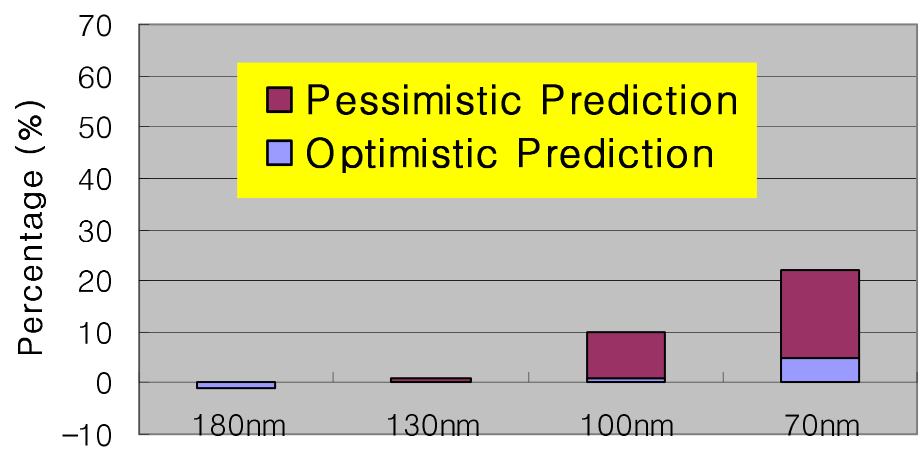
Total energy saving at 70nm process



Leakage energy saving across processes



Total energy saving across processes



- **More energy saving for wider issue processors**
- **Readport deactivation can be combined with dead subbank deactivation.**

Conclusion

- **Most leakage power is in critical paths**
  - **Dynamic leakage reduction (DDFT) desired**
- **LBB allows Fine-grain dynamic leakage reduction with zero or minimal performance penalty.**
  - **0% performance penalty for multiported regfiles**
- **Sleep time can be improved by changing micro-architectural scheduling policies.**
  - **Stack better than queue for free list policy**
- **Follow on work:**
  - **Leakage-biased domino logic to save leakage power in critical ALUs [*VLSI Symposium 2002*]**

Acknowledgments

- Thanks to Christopher Batten, Ronny Krashinsky, Rajesh Kumar, and anonymous reviewers

- Funded by DARPA PAC/C award F30602-00-2-0562, NSF CAREER award CCR-0093354, and a donation from Infineon Technologies.

DDFT Examples

	Body Biasing	Power Gating	Sleep Vector
Steady-state leakage power	Less than 5% (depends on V_{body})	Less than 5% (depends on sleep transistor)	Less than 50% (depends on the circuit)
Transition time, Wakeup latency	0.1~100us	Less than a cycle	Less than a cycle
Transition energy ,Breakeven time	Well cap switching energy	Sleep transistor gate cap switching energy	Active energy consumed due to spurious toggling after sleep vector
Delay Impact	No	Yes. Due to sleep transistor	Yes. Due to mux
Etc		Area for sleep transistor and virtual supplies	Finding sleep vector is hard