



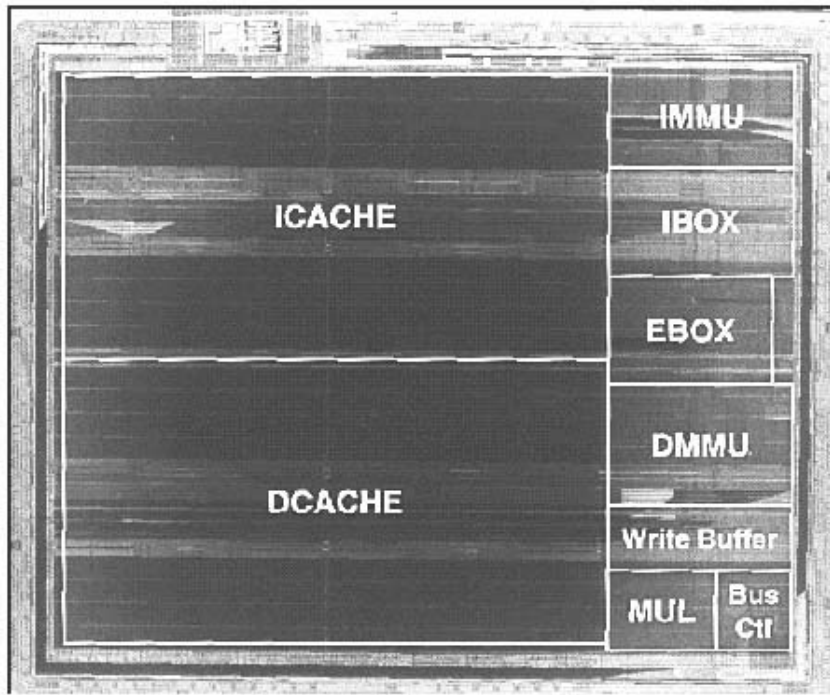
# Scheduling Reusable Instructions for Power Reduction

J. S. Hu, [N. Vijaykrishnan](#), S. Kim, M. Kandemir,  
and M. J. Irwin

Microsystems Design Lab

The Pennsylvania State University

# Power: StrongARM SA-110



Die of DEC StrongARM

## ■ Power dissipation

➔ ICache	27%
➔ IBox	18%
□ EBox	8%
□ IMMU	9%
□ DCache	16%
□ DMMU	8%
□ Clock	10%
□ Write Buffer	2%
□ Bus Ctrl	2%
□ PLL	< 1%

# Related Work

- Stage-skip pipeline
  - A small decoded instruction buffer [1][2]
- Loop caches
  - Dynamic/preloaded/hybrid loop caches [3][4][5]
- Filter cache
  - Filter dcache [6], decode filter cache [7]

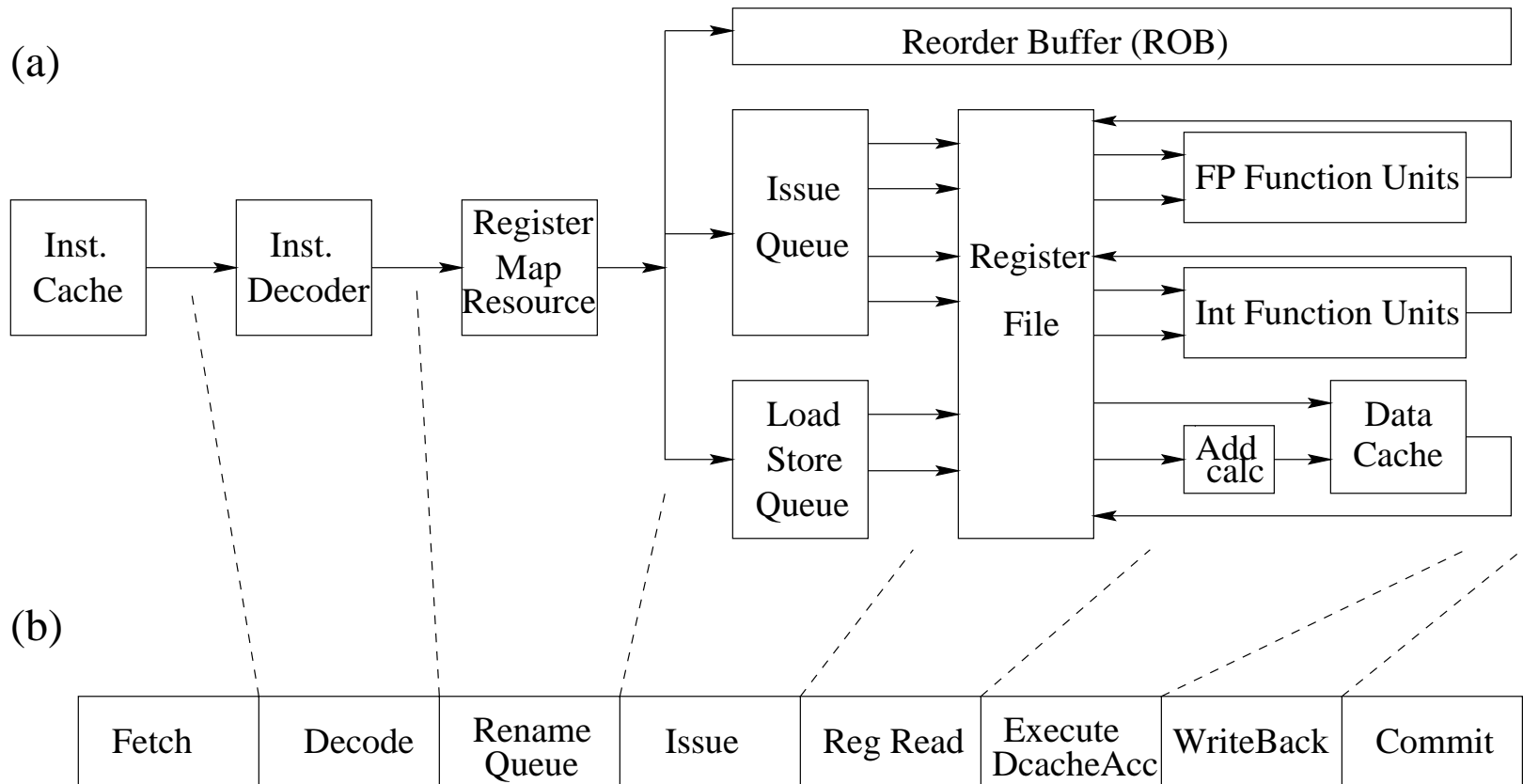
# Related Work

- [1] M. Hiraki et al. Stage-skip pipeline: A low power processor architecture using a decoded instruction buffer. In *Proc. International Symposium on Low Power Electronics and Design*, 1996.
- [2] R. S. Bajwa et al. Instruction buffering to reduce power in processors for signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(4):417–424, December 1997.
- [3] L. H. Lee, B. Moyer, and J. Arends. Instruction fetch energy reduction using loop caches for embedded applications with small tight loops. In *Proc. International Symposium on Low Power Electronics and Design*, 1999.
- [4] T. Anderson and S. Agarwala. Effective hardware-based two-way loop cache for high performance low power processors. In *IEEE Int'l Conf. on Computer Design*, 2000.
- [5] A. Gordon-Ross, S. Cotterell, and F. Vahid. Exploiting fixed programs in embedded systems: A loop cache example. *IEEE Computer Architecture Letters*, 2002.
- [6] J. Kin et al. The filter cache: An energy efficient memory structure. In *Proc. International Symposium on Microarchitecture*, 1997.
- [7] W. Tang, R. Gupta, and A. Nicolau. Power savings in embedded processors through decode filter cache. In *Proc. Design and Test in Europe Conference*, 2002.

# Our Proposed Approach

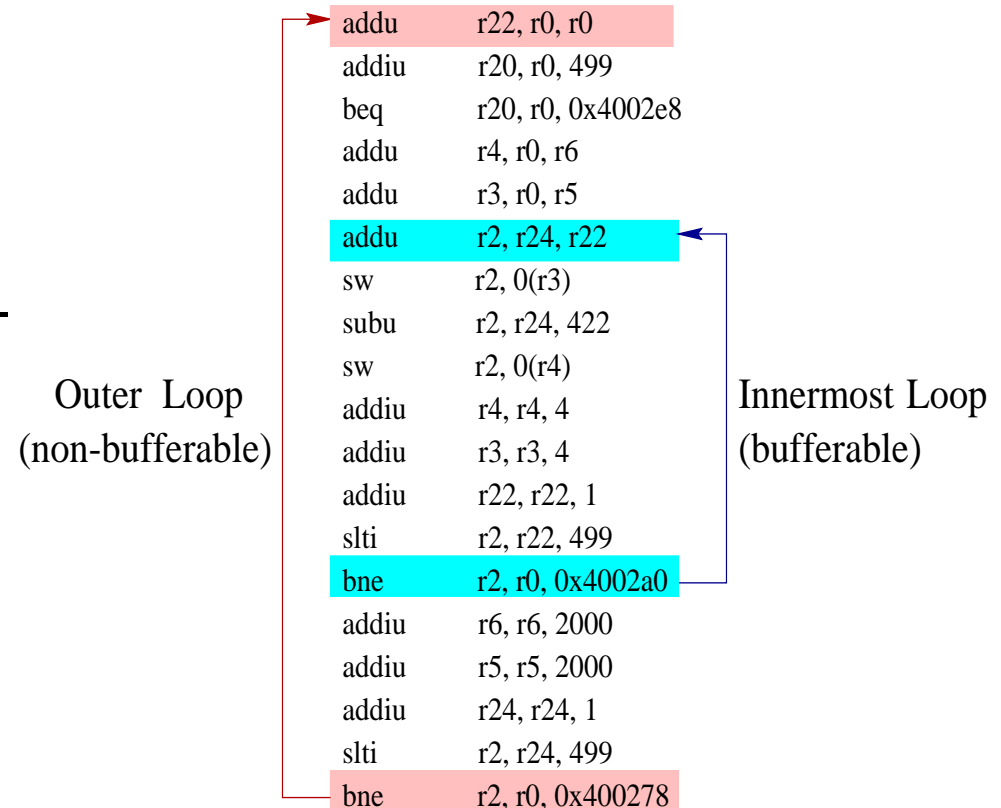
- Scheduling reusable loop instructions within the issue queue
  - No need of an additional instruction buffer
  - Utilize the existing issue queue resources
  - Be able to gate the front-end of pipeline
  - Automatically unroll loops in the issue queue
  - No ISA modification

# Embedded Processor based on MIPS Core

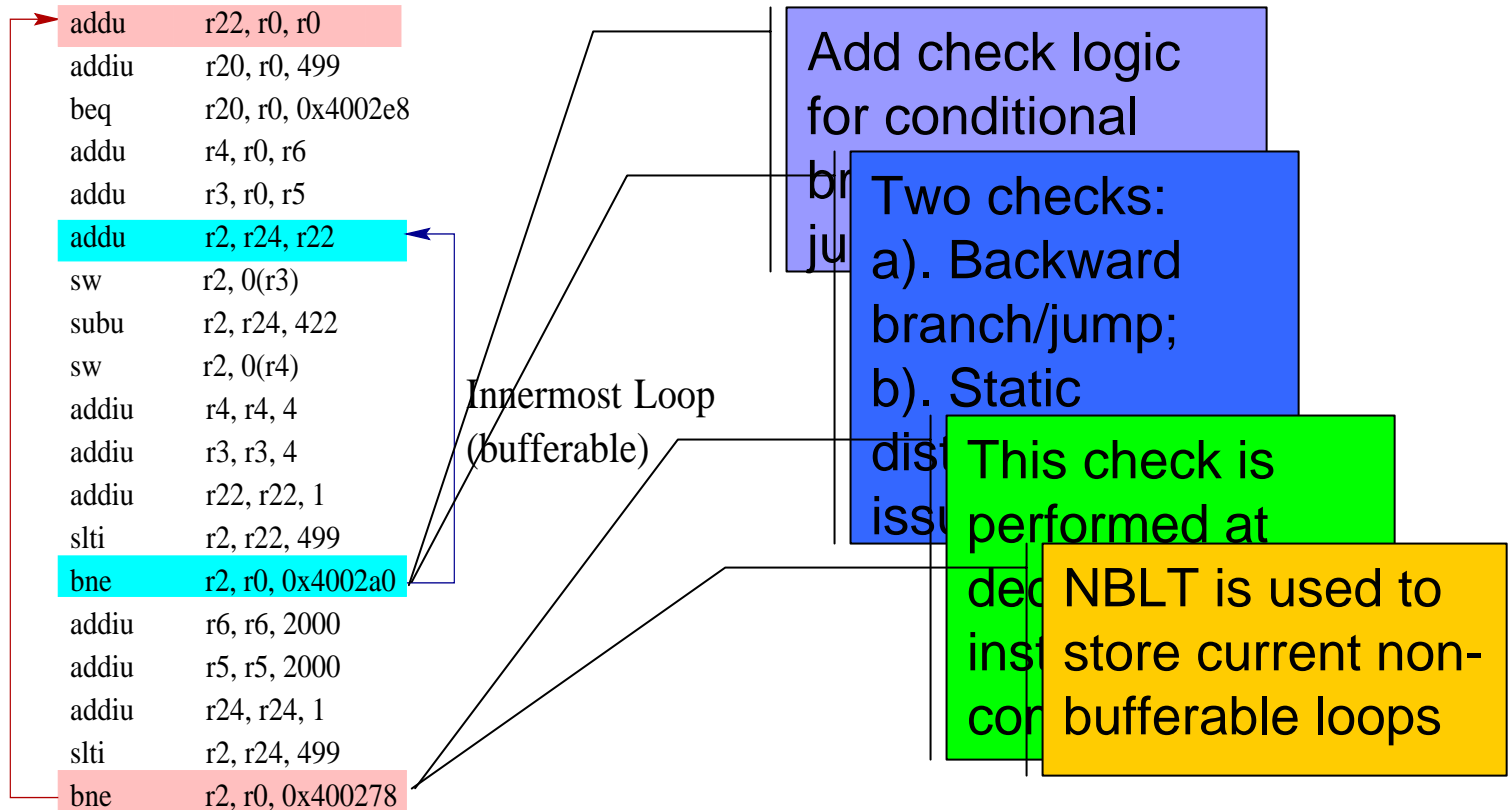


# Schedule Reusable Instructions

- Array-intensive embedded applications
- Utilizing issue queue
- Reusable instructions – innermost loops
- Self-steaming issue queue
- Gate front-end of the datapath

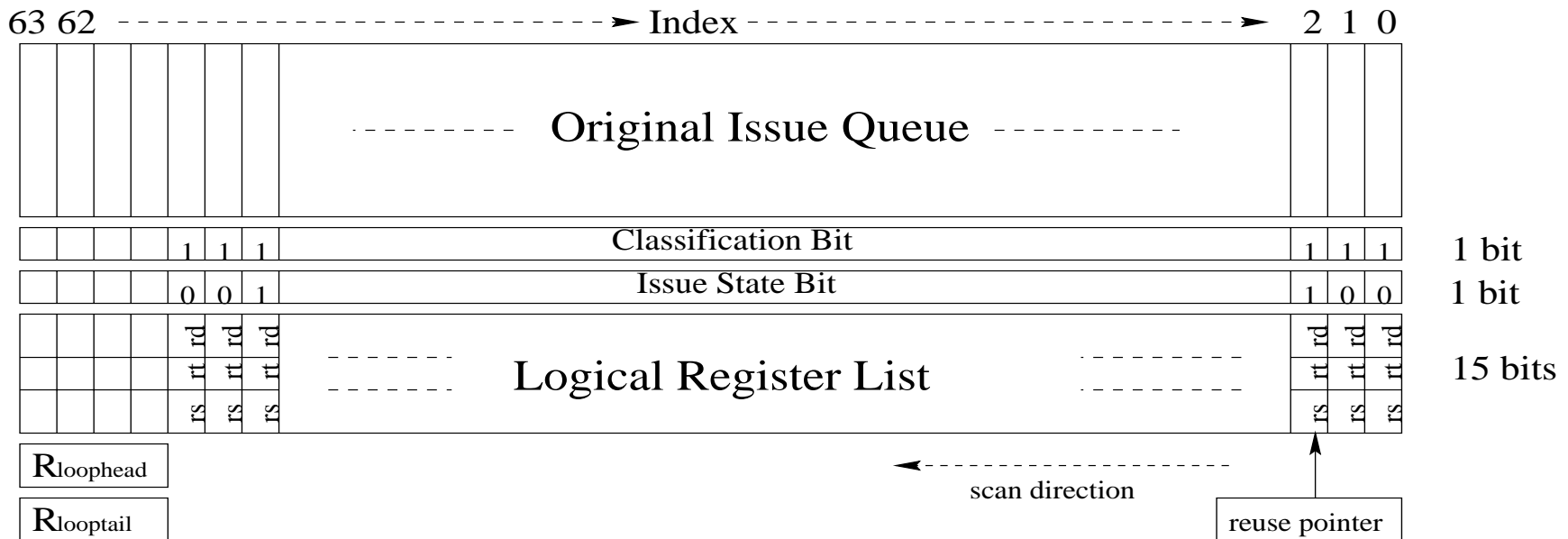


# Loop Detection

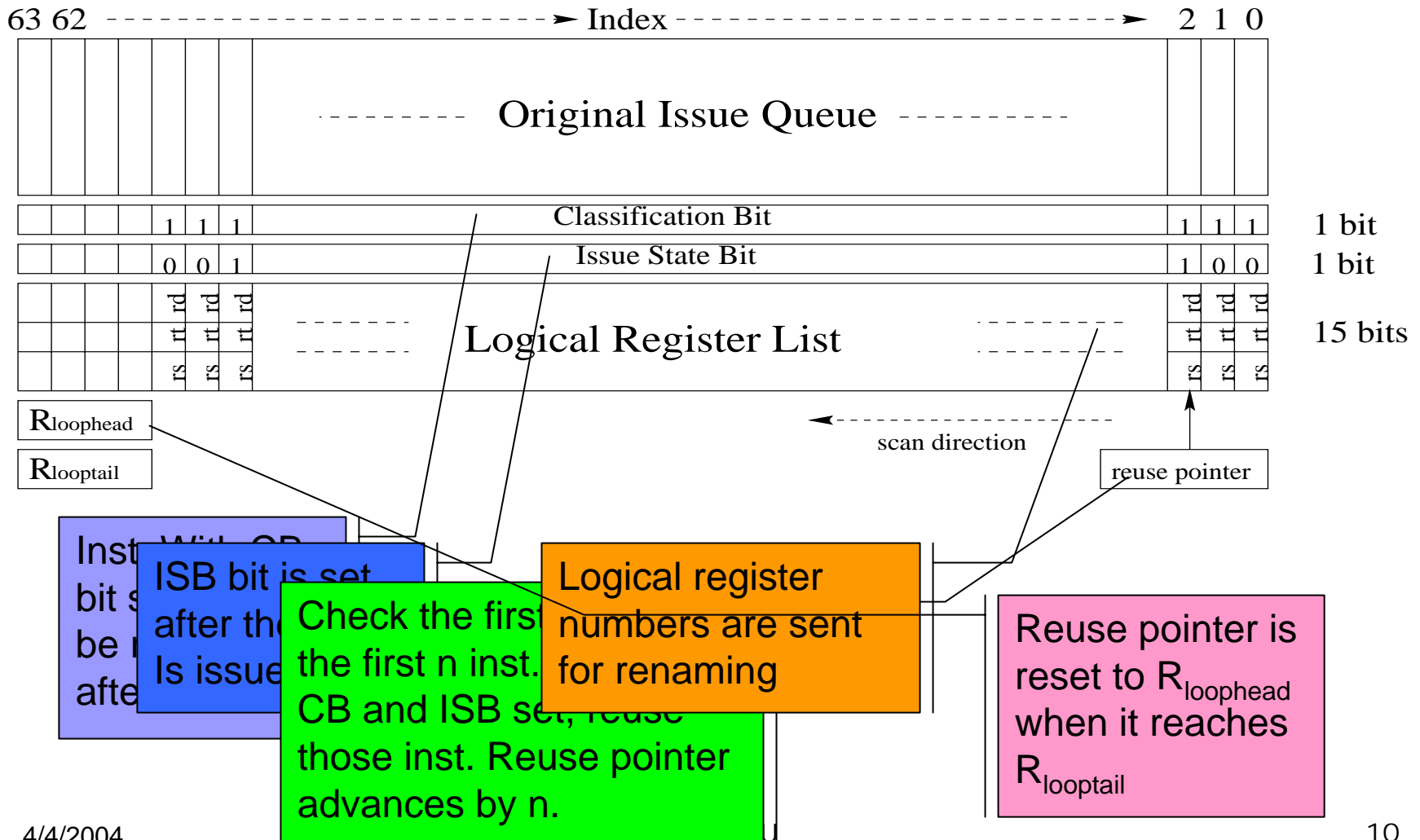


# Buffering Reusable Instructions

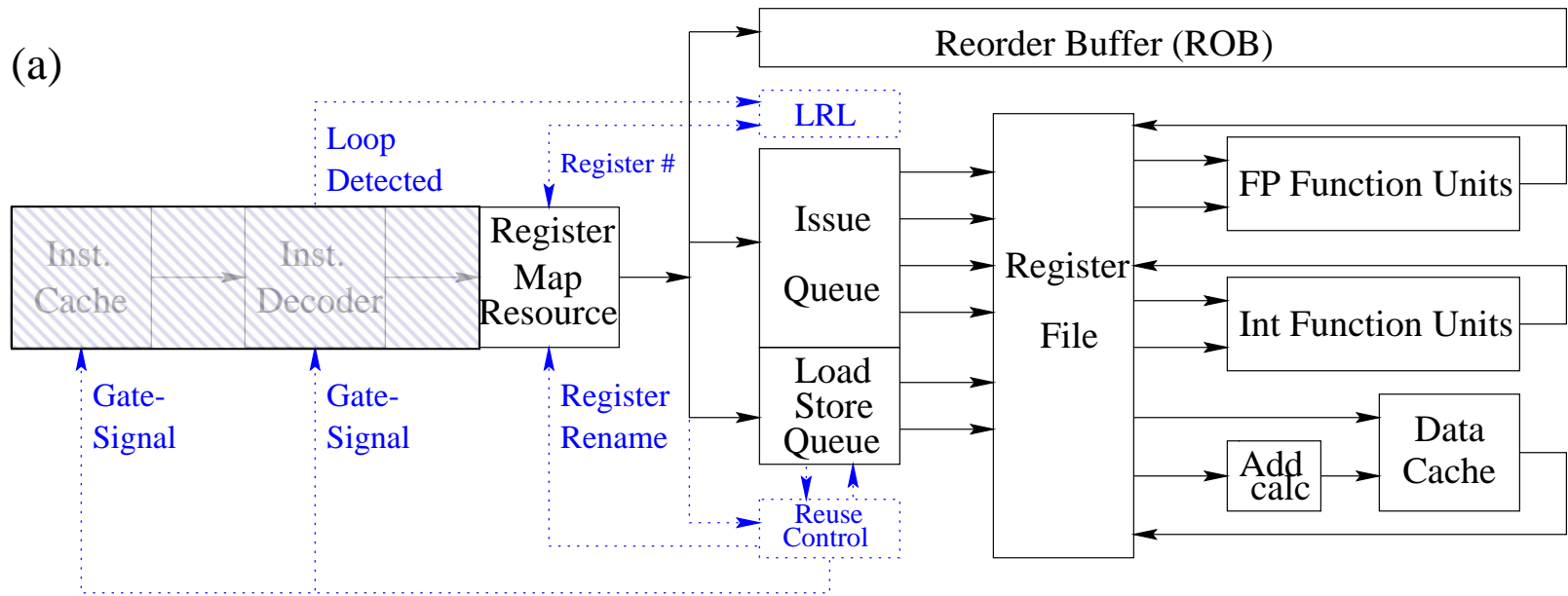
- Extended issue queue microarchitecture
- Reusable instructions are marked, logical register numbers are stored in LRA
- Buffering integer number of loops



# Reusing Buffered Instructions



# The New Datapath



(b)

Fetch	Decode	Rename Queue	Issue	Reg Read	Execute DcacheAcc	WriteBack	Commit
-------	--------	--------------	-------	----------	-------------------	-----------	--------

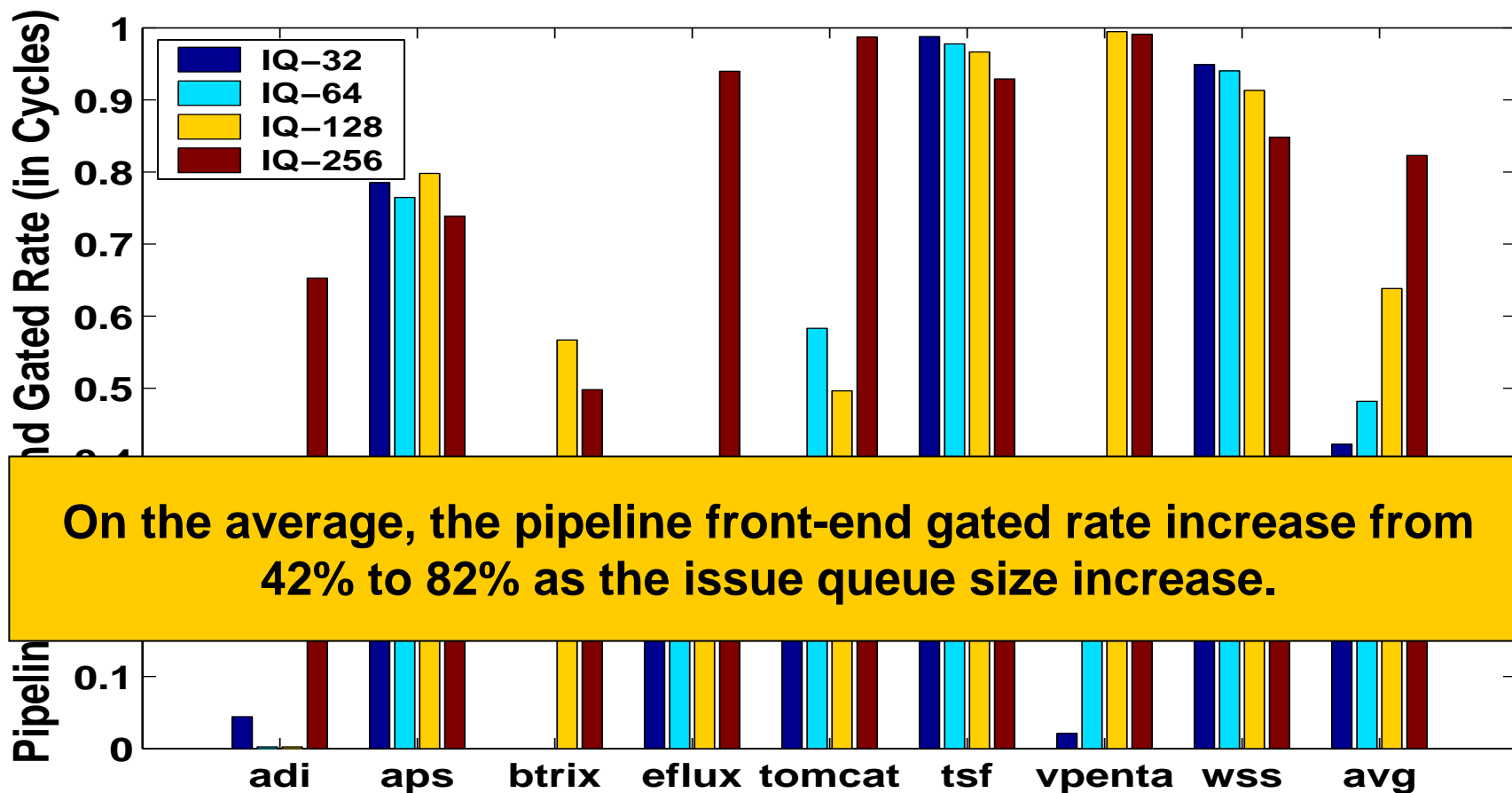
# Experiment Setup

Parameters	Configuration
Issue Queue	64 entries
Load/Store Queue	32 entries
ROB	64 entries
Fetch Queue	4 entries
Fetch/Decode Width	4 inst. per cycle
Issue/Commit Width	4 inst. per cycle
Function Units	4 IALU, 1 IMULT, 4 FPALU, 1 FPMULT
Branch Predictor	bimod, 2048 entries, RAS 8 entries BTB 512 set 4 way assoc.
L1 ICache	32KB, 2 way, 1 cycle
L1 DCache	32KB, 4 way, 1 cycle
L2 UCache	256KB, 4 way, 8 cycles
TLB	ITLB: 16 set 4 way, DTLB: 32 set 4 way 4KB page size, 30 cycle penalty
Memory	80 cycles for first chunk, 8 cycles the rest

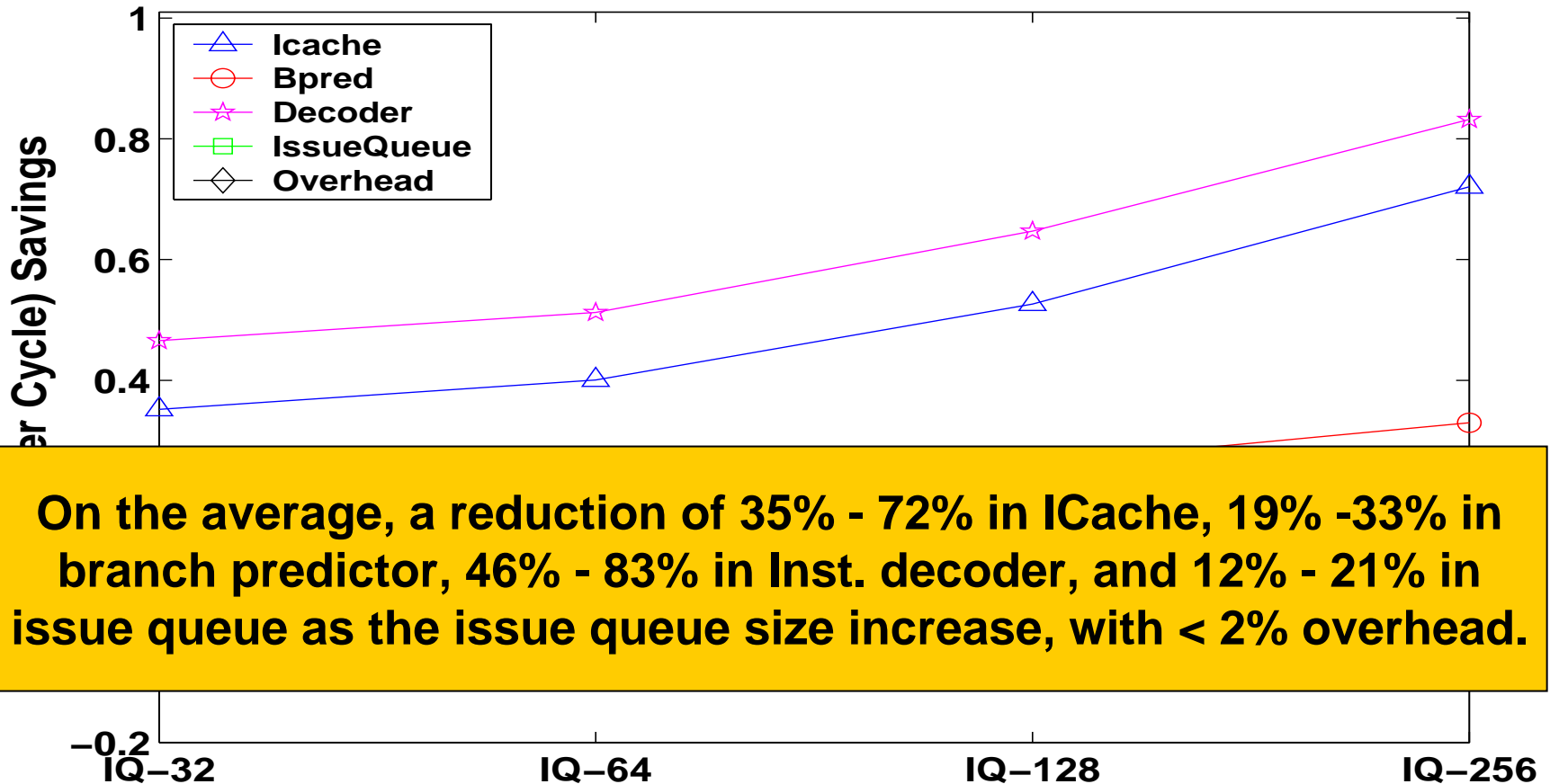
Bench	Source
Adi	Livermore
Aps	Perfect Club
Btrix	Spec92/NASA
Eflux	Perfect Club
Tomcat	Spec95
Tsf	Perfect Club
Vpenta	Spec92/NASA
wss	Perfect Club

Name	Simulator
Arch.	SimpleScalar 3.0
Power	Wattch

# Rate of Gated Front End

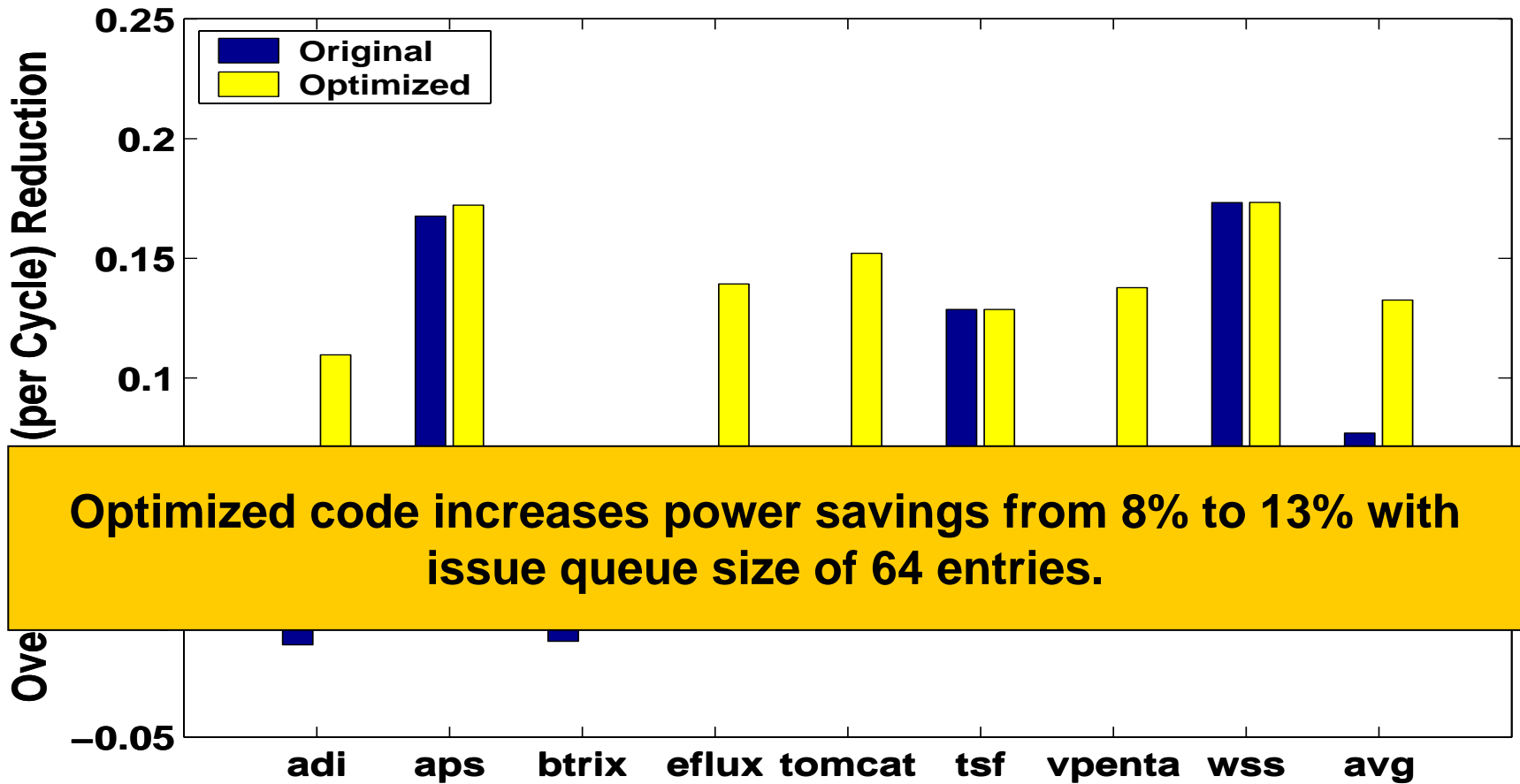


# Power Savings in Front-end



**On the average, a reduction of 35% - 72% in ICache, 19% - 33% in branch predictor, 46% - 83% in Inst. decoder, and 12% - 21% in issue queue as the issue queue size increase, with < 2% overhead.**

# Impact of Compiler Optimizations



# Conclusions

- Proposed a new issue queue architecture
  - Detect capturable loop code
  - Buffer loop code in the issue queue
  - Schedule the reusable loop inst. buffered
- Significant power reduction in pipeline front-end components while gated
- Compiler optimizations can further improve the power savings

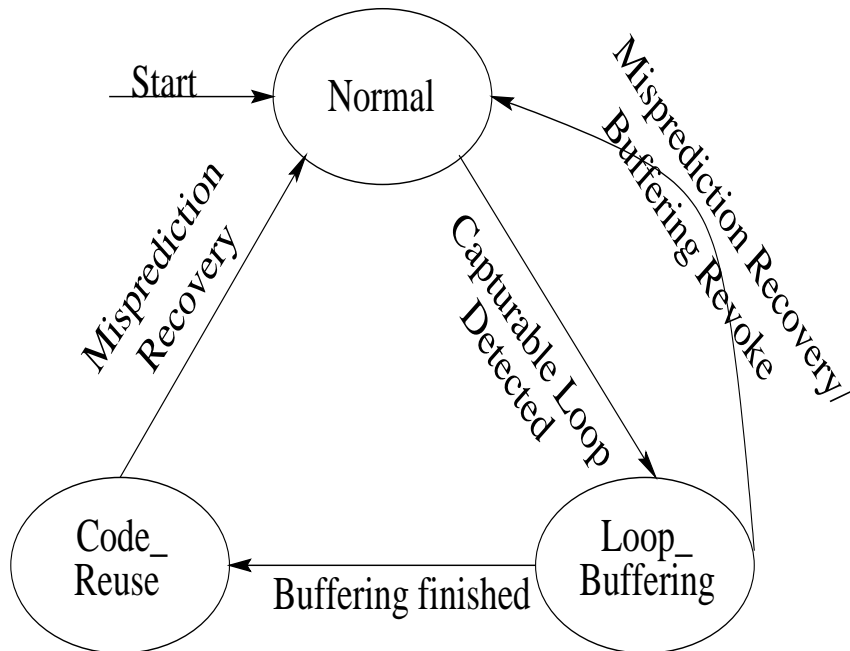


**Thank You!**

# Power: A Design Limiter?

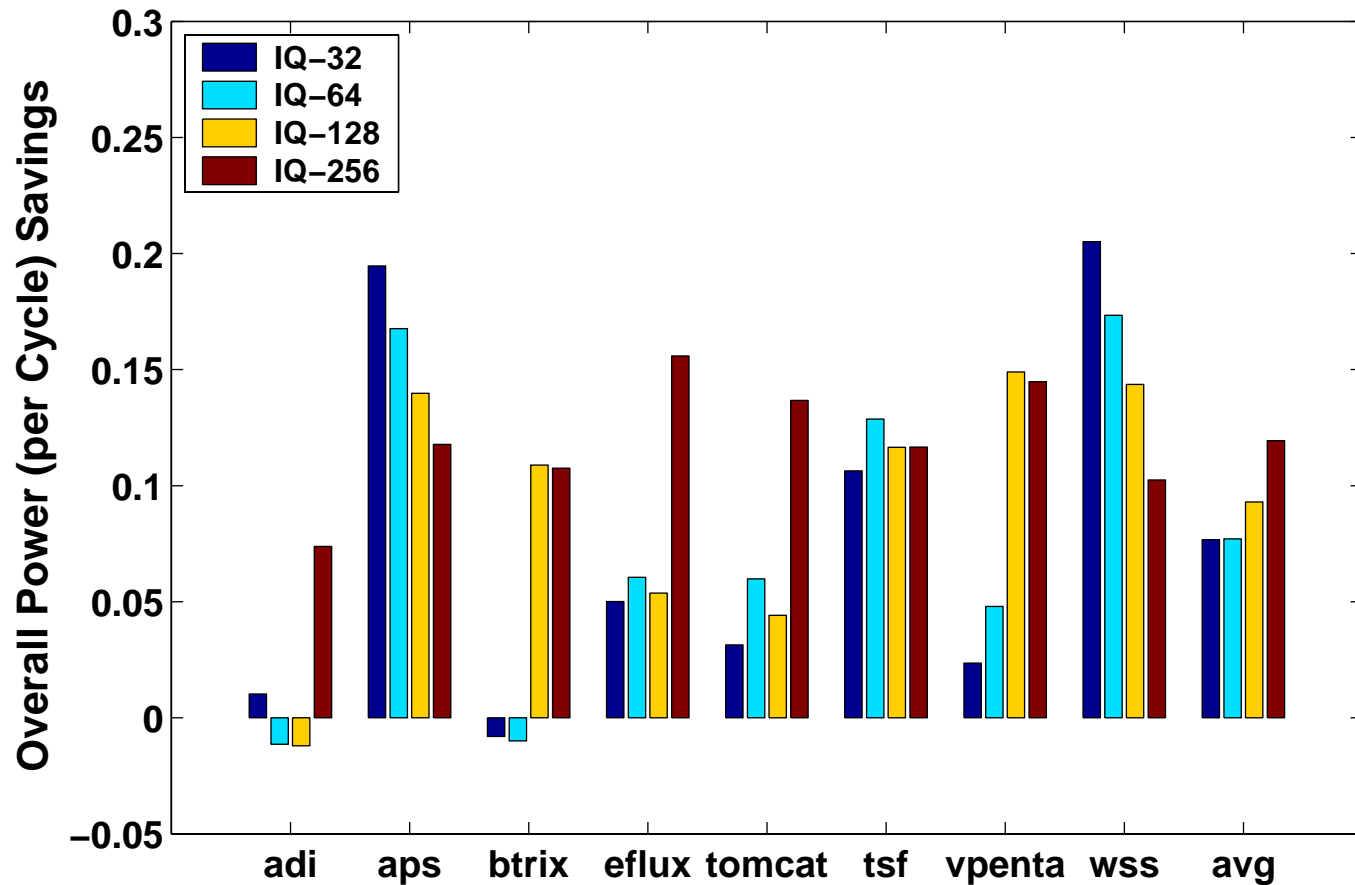
- A major constraint in embedded systems design
- Superscalar architecture is more likely used for performance
- Front-end: a power-hungry component
- Seek to optimize the front-end of datapath in embedded processors

# Issue Queue State Transition



- No impact on exception handling
- Non-successful buffering will be revoked
- Changing control flow in a buffering loop will cause buffering to be revoked
- Branch prediction is disabled & switched to static prediction during Code\_Reuse state
- Misprediction due to normally existing a loop will restore issue queue to Normal state

# Overall Power Reduction



# Performance Loss

