An Embedded Low Power/Cost 16-Bit Data/Instruction Microprocessor Compatible with ARM7 Software Tools

Fu-Ching Yang and Ing-Jer Huang

Dept. of Computer Science and Engineering.

National Sun Yat-Sen University, Kaohsiung Taiwan

fcyang@esl.cse.nsysu.edu.tw
ijhuang@cse.nsysu.edu.tw

12nd ASP-DAC, Jan. 26, 2007

Outline

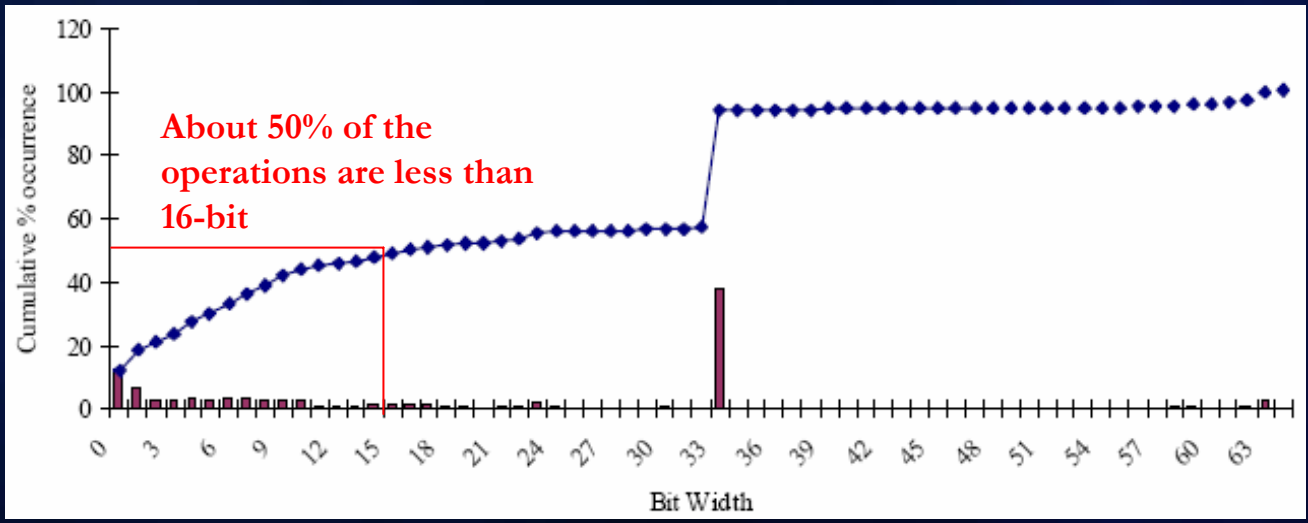
- Motivation
- Related Work
- Proposed Solution
  - Register file merging
  - Architecture modification
  - Software tool reuse
- Experimental Result
- Conclusion

Outline

- Motivation
- Related Work
- Proposed Solution
  - Register file merging
  - Architecture modification
  - Software tool reuse
- Experimental Result
- Conclusion

Motivation

- Short-precision oriented applications
  - Over 50% of the operations are less than 16 bits



Bitwidths for SPECint95 on 64-bit Alpha.

*Brooks, D. and Martonosi, M.,"*Dynamically exploiting narrow width operands to improve processor power and performance," *Proc. of Intl. Symposium on Fifth High-Performance Computer Architecture*, pp. 13-22, 1999.

Outline

- Motivation
- Related Work
- Proposed Solution
  - Register file merging
  - Architecture modification
  - Software tool reuse
- Experimental Result
- Conclusion

Related work

- Dynamic explore data bit width
- Static compiler analysis
  - ARM7/ARM9 cores:
  - 32-bit RISC+CISC instruction set
  - 16-bit RISC
  - 32-bit data
- What if the major data types in your SoC are 8 or 16 bits?
  - 32-bit data path may be an overkiller
    - bad for performance, power, chip size, *etc*.

Memory bandwidth bottleneck

- ARM7/ARM9 cores perform well
  - Memory bandwidth : 32/64
- What if your SoC can not afford such bandwidth
  - In cost-sensitive applications
  - 16-bit bus: 2 cycles/per instruction (or data) fetch
  - 8-bit bus: 4 cycles/per instruction (or data) fetch

Outline

- Motivation
- Related Work
- Proposed Solution
  - Register file merging
  - Architecture modification
  - Software tool reuse
- Experimental Result
- Conclusion

Proposed solution

- A 16-bit Thumb Microprocessor – SYS16TM
  - 16-bit instruction set: same as THUMB
  - 16-bit data path
  - An extra low cost ARM-based solution
  - Compatible with most ARM's existing development tools
- Architecture challenges
  1. Register file bank merging
  2. Pipeline Trimming
  3. Reuse ARM's develop environment

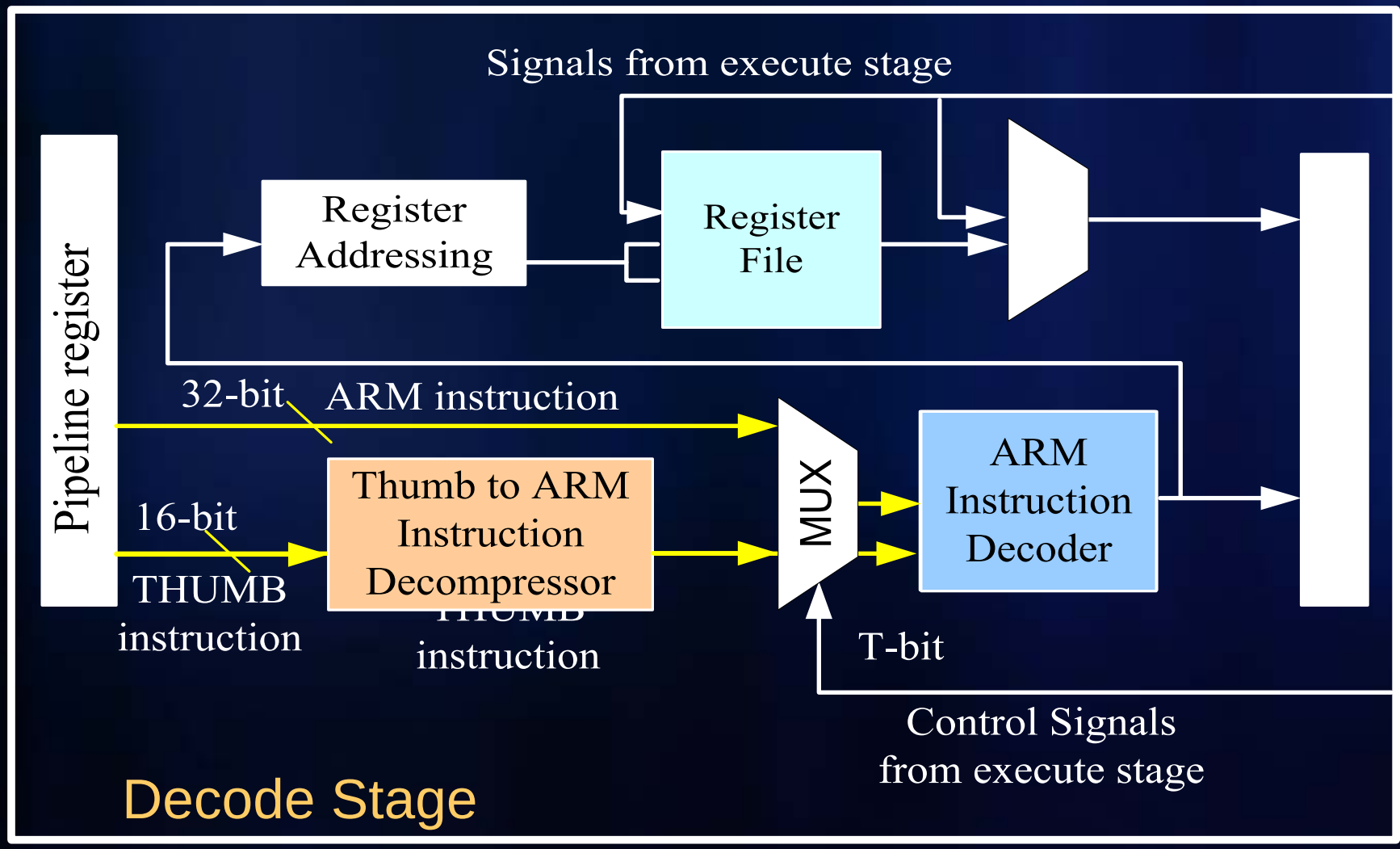
Register file bank merging

- R0~R12 : 16 bits
- SP, LR, PC : 32 bits
- CPSR, SPSR : 32 bits
- Define new instruction to access high half-word of SP, LR, PC

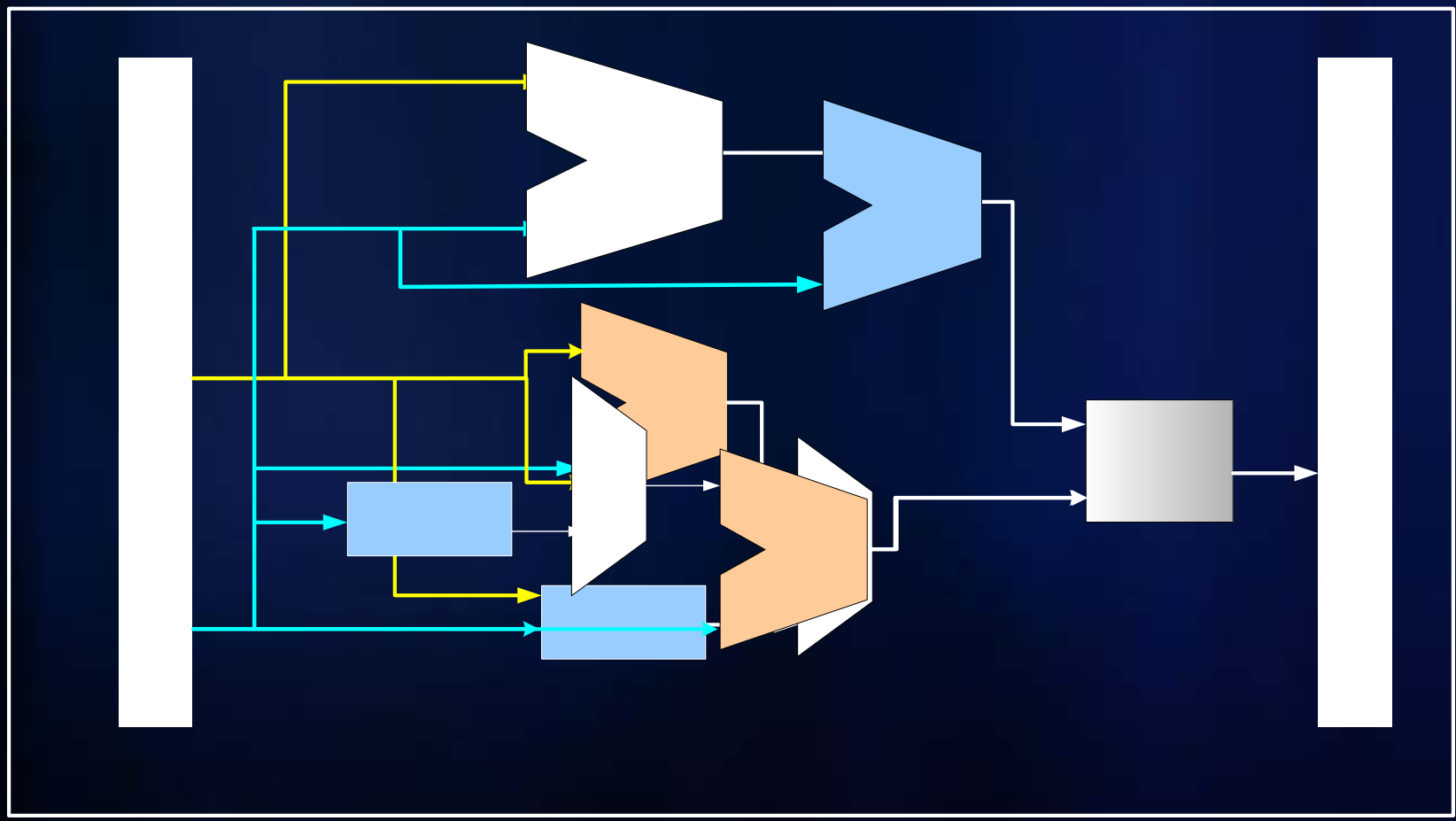
Architecture Changes

- Fetch stage
  - Replace the adder with added by 2
- Decode stage
  - Remove the THUMB decompressor
  - Modify the Decoder to decode THUMB directly
  - Modify the register file to single mode
- Execution stage
  - Modify multiplier
  - Remove the adder after multiplier
  - Cut the shifter before ALU path

Decode THUMB instruction directly



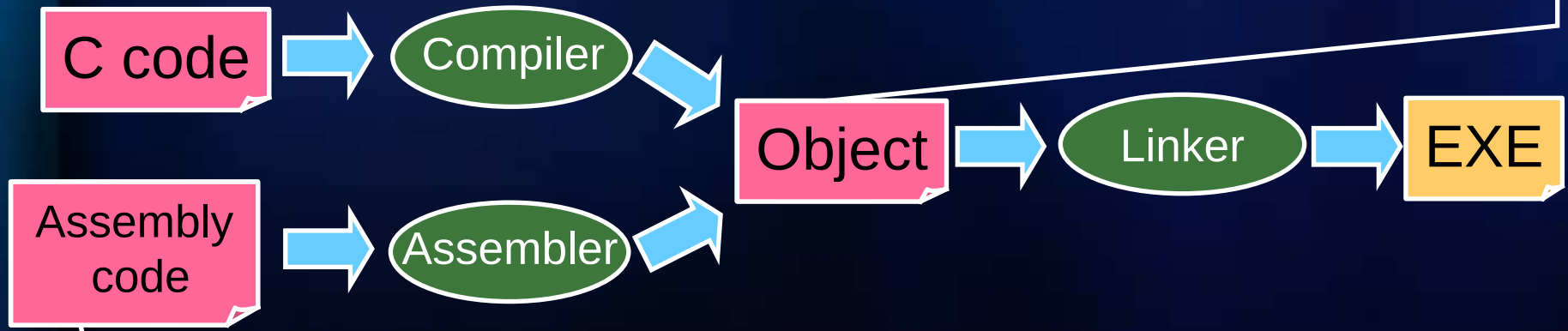
Execute stage modification



Coding rules for reusing ARM7's Software Environment

Declare "short " instead of "int" in C language
the registers in SYS16TM are 16 bits

- Remove shift instruction patterns
  - Generated by conventional 32-bit compiler
  - Clear upper 16 bits value
    - Emulating 16-bit computation with a 32-bit data path



- Interrupt table maintain the same addressing

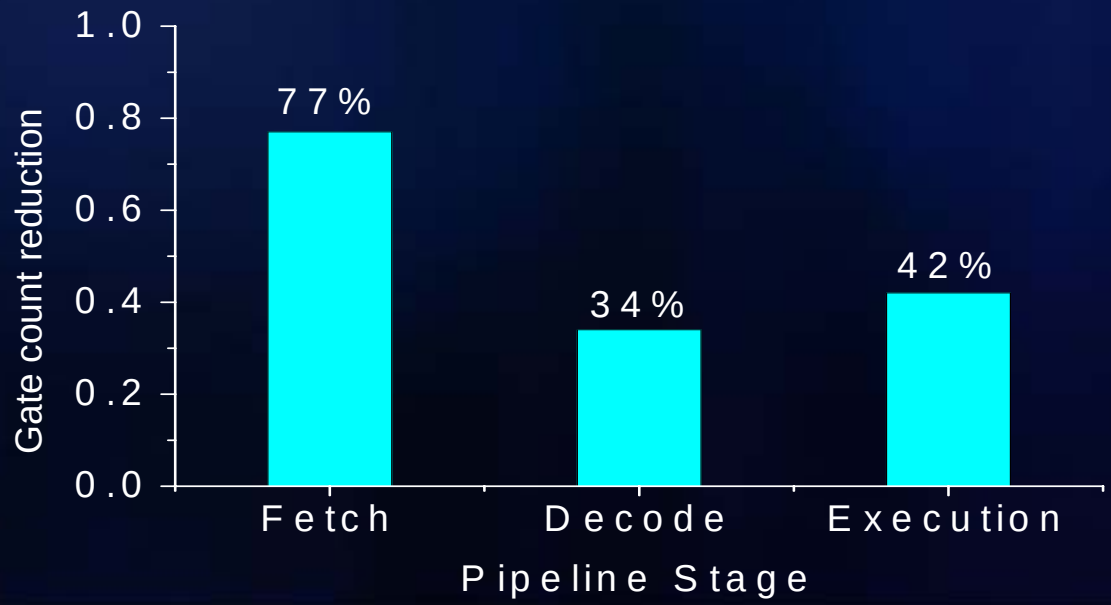
Outline

- Motivation
- Related Work
- Proposed Solution
  - Register file merging
  - Architecture modification
  - Software tool reuse
- Experimental Result
- Conclusion

Chip size, clock frequency

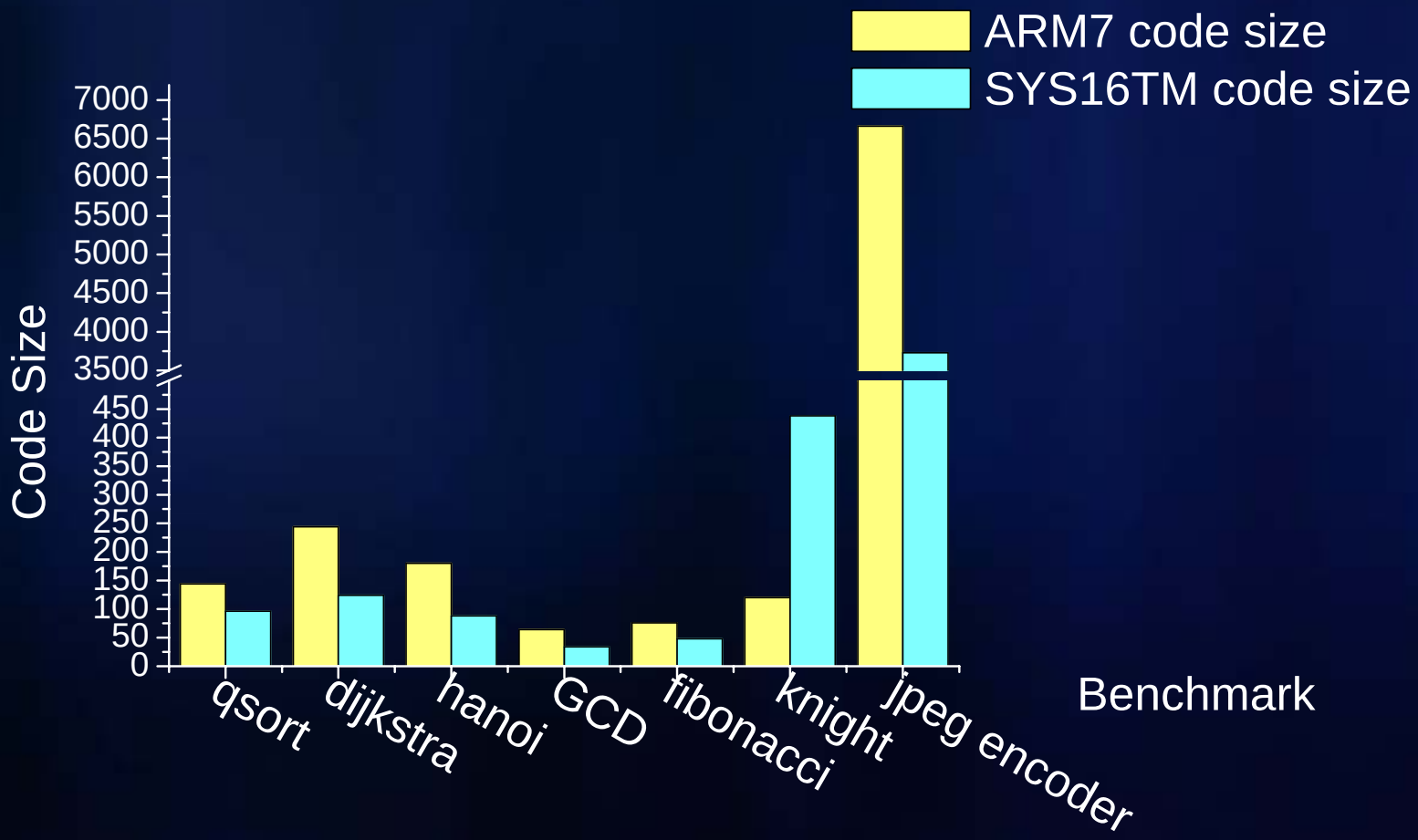
TSMC 0.18*um* Process	ARM7	SYS16TM	SYS16TM/ARM7
Gate count	**50,612**	**20,232**	**40%**
Max. Clock rate (MHz)	**91 MHz**	**137MHz**	**151%**
Power	**5.33 mW**	**2.7 mW**	**51%**

- The gate count reduction



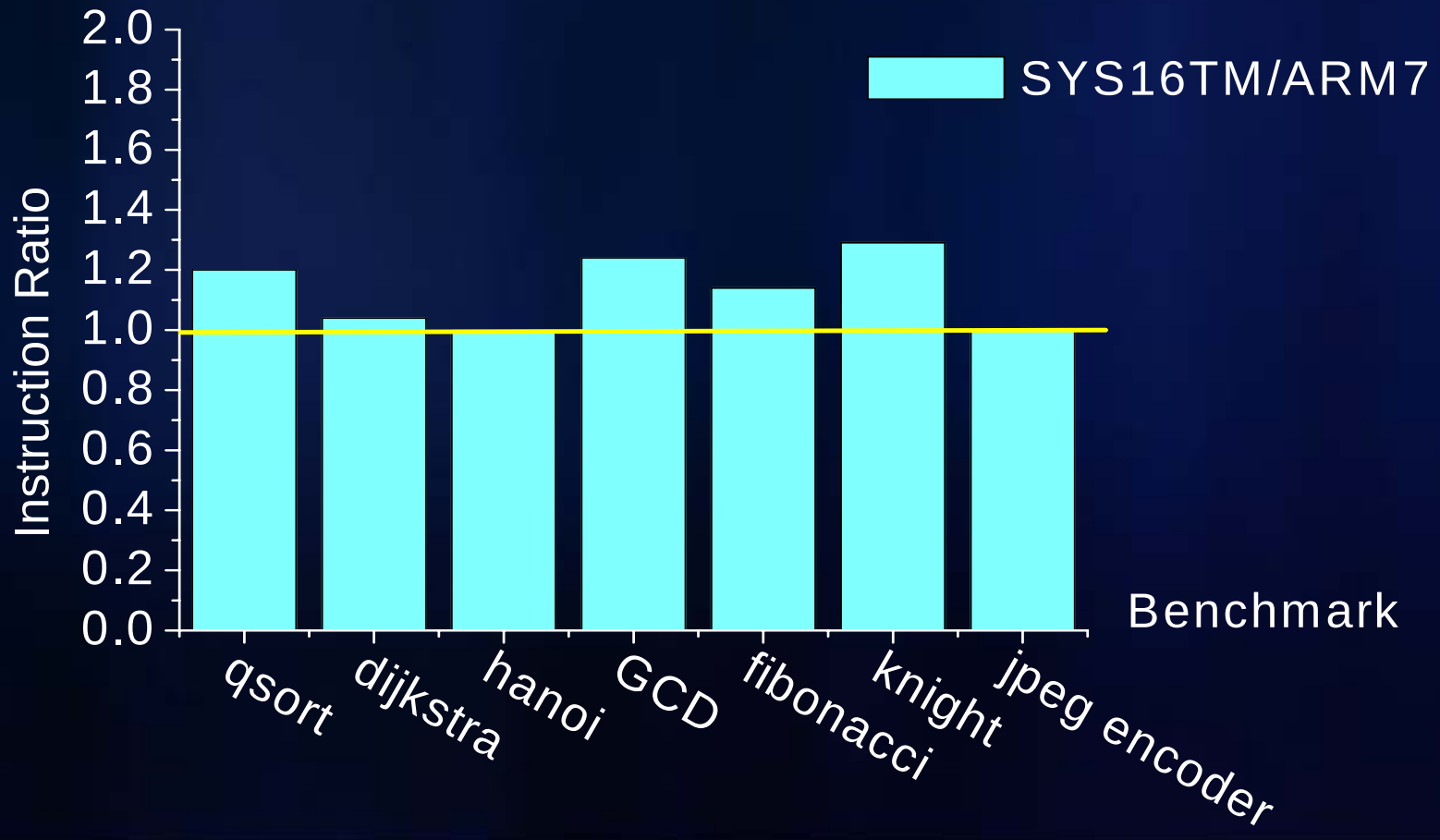
Static code Size : SYS16TM VS. ARM7

- SYS16TM's code size is smaller : 61%



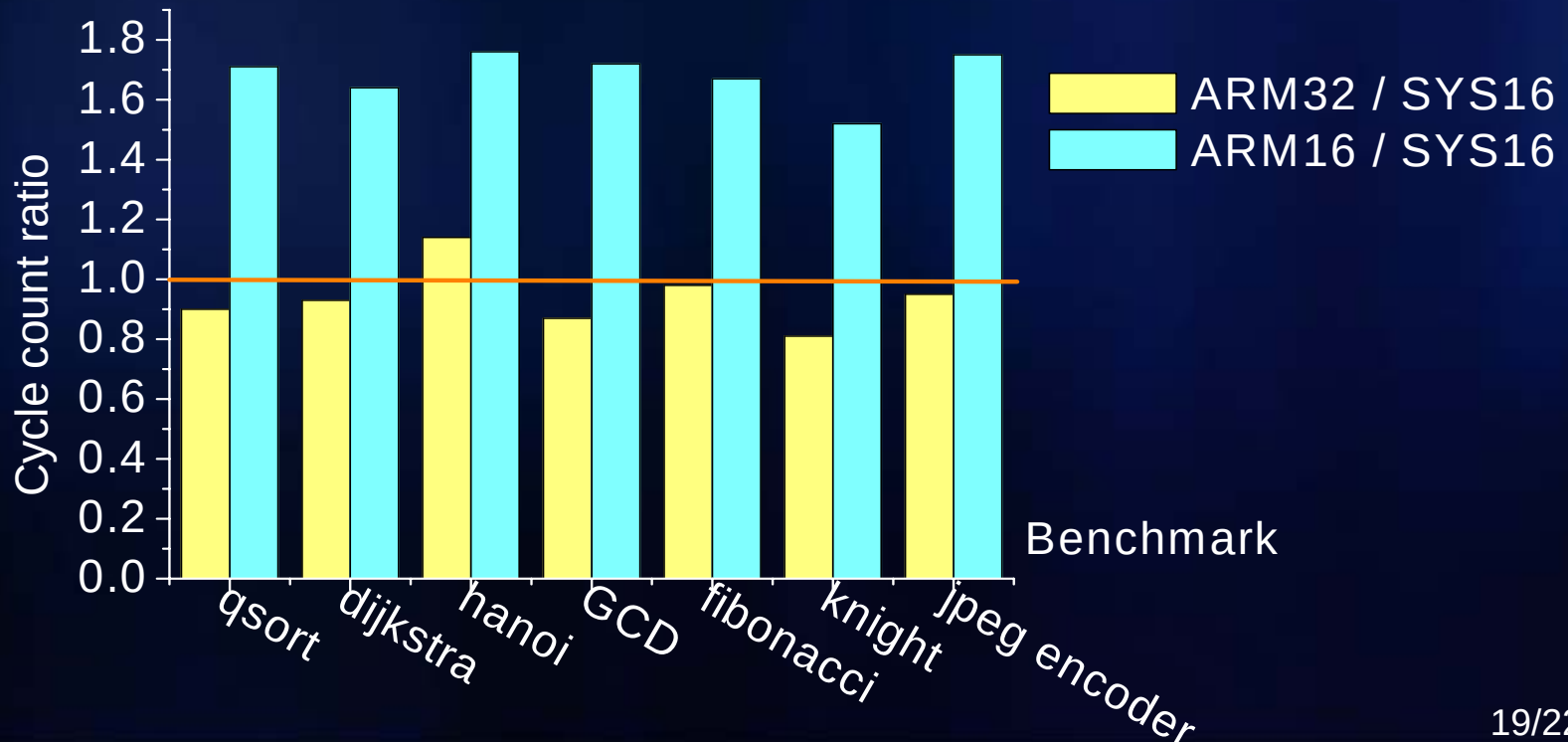
Executed Internal Instruction count

- SYS16TM requires 1.12 times instruction count comparing to ARM7 on average



When considering limited memory bandwidth

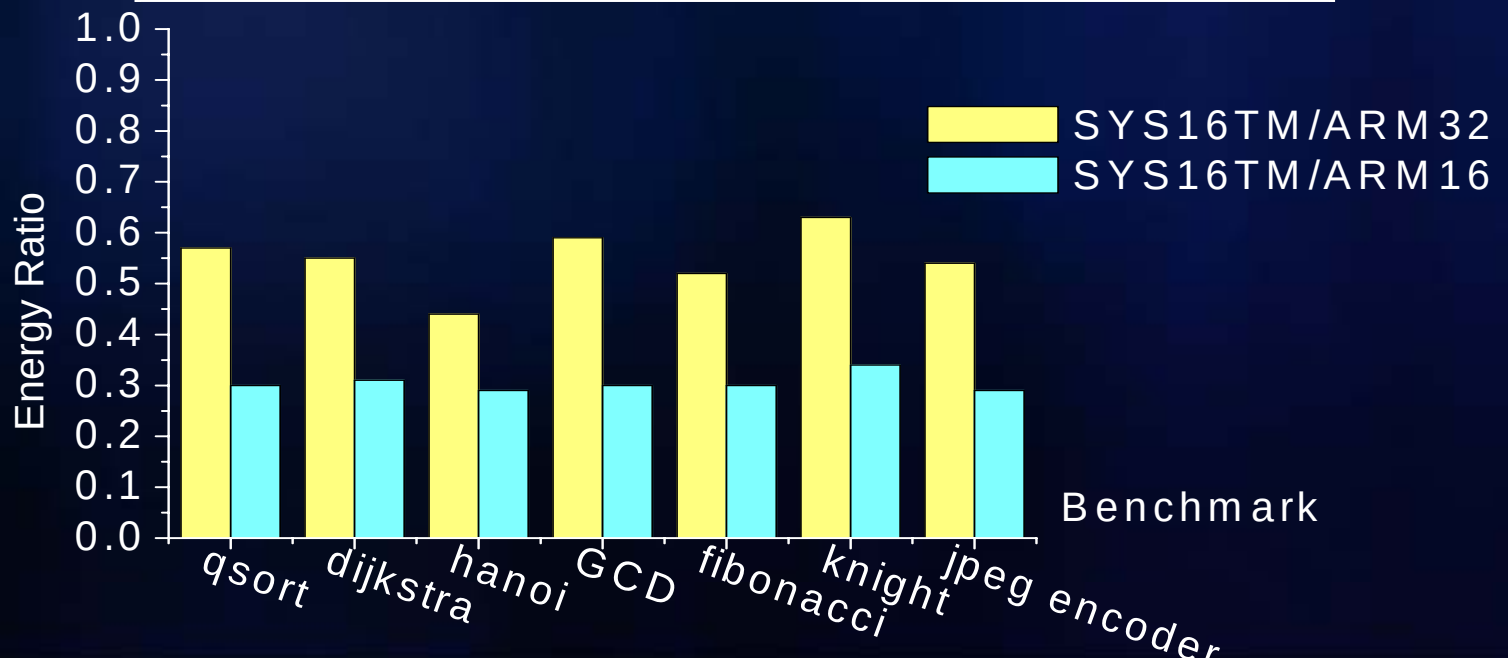
- With 32-bit memory bandwidth
  - ARM7 is better than SYS16TM by 6%
- With 16-bit memory bandwidth
  - SYS16TM is better than ARM7 by 64%



Microprocessor energy consumption

- Consider the real-time application
  - Assume both processors have to complete the applications at the same time

Memory bandwidth	Energy consumption ratio (SYS16TM/ARM7)
32-bit	55%
16-bit	30%



Outline

- Motivation
- Related Work
- Proposed Solution
  - Register file merging
  - Architecture modification
  - Software tool reuse
- Experimental Result
- Conclusion

Conclusions: SYS16TM VS. ARM7

- Proposed a 16-bit data 16-bit instruction THUMB microprocessor
- Features
  - 60% smaller in gate count
  - 39% smaller in the program size
  - faster in the maximal clock rate under the same technology
    - 51% for 0.18μm
  - 33% faster in cycles (@ same clock rate) under 16-bit memory bandwidth
  - More than 49% power reduction
  - 70% energy reduction on 16-bit memory bandwidth
  - Utilizing the existing ARM7's software development environment with minor patching

Thank You