Embedded Software Optimization for MP3 Decoder Implemented on RISC Core

Yingbiao Yao, Qingdong Yao,
Peng Liu, Zhibin Xiao
Zhejiang University
Information Science & Electronic Engineering Department, P.R. China

IEEE Transaction on Consumer Electronics
November 2004

Abstract

- *This paper proposes general software optimization techniques for embedded systems based on processors, which mainly include general optimization methods in high language and software & hardware co-optimization in assembly language. Then these techniques are applied to optimize our MP3 decoder, which is based on RISC32, a RISC core compatible with MIPS I instruction set. The last optimization decoder requires 48 MIPS and 49Kbytes memory space to decode 128Kbps, 44.1KHz joint stereo MP3 in real time with CPI 1.15, and we have achieved performance increase of 46.7% and memory space decrease of 38.8% over the original decoding software.*

Outline

- What’s the problem ?
- Related work
- Introduction to RISC32
- Embedded software optimization techniques
- MP3 decoding software optimization
- Experiment result
- Conclusion

What's the problem ?

- Low cost with fast time to market is the top requirement in embedded system
  - Develop based on **microprocessor architecture** by software and hardware co-design
- How to make software run efficiently on selected processor had become a main problem

Related works

- Many portable MP3 players adopt DSP core or dual core that are comprised by DSP and RISC

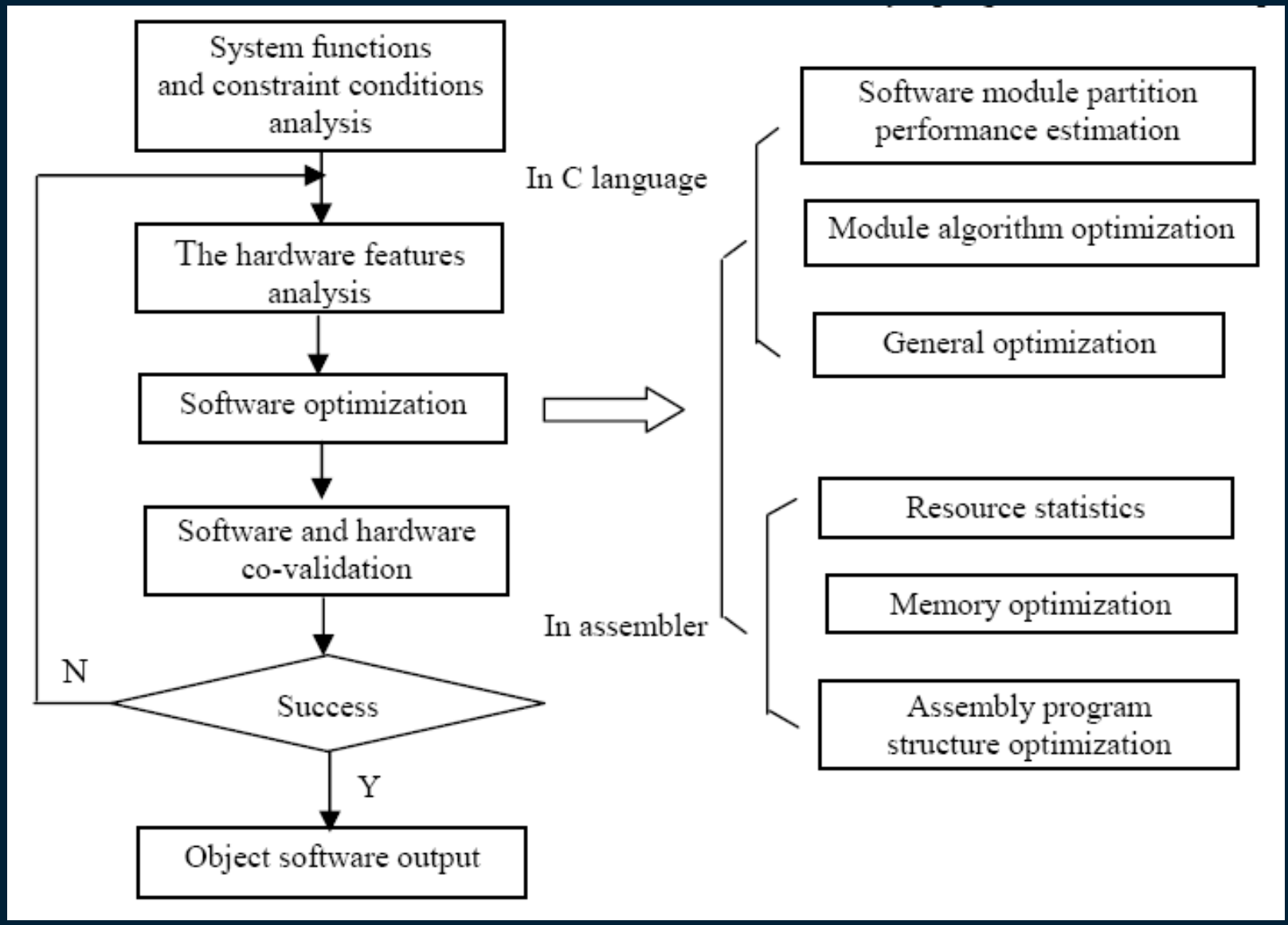
Introduction to RISC32

- 6 pipeline stages
  - Instruction fetch (IF)、instruction decoding (ID)、execute (EX)、data memory access (DM)、data tag comparing (TC) and write back (WB)
- Compatible with MIPS-I ISA
- Separate 16Kbytes ICache and 16Kbytes DCache by Harvard architecture
- Direct mapping, write-through cache strategy
- 16Kbytes on-chip RAM
- 200 MHz clock frequency
- 1mW power dissipation per MHz

Embedded software optimization techniques (1)

- The embedded software optimization can be divide into two parts
  - Algorithm optimization in high level language
    - Tradeoff among computation load , the complexity of data conveying and the size of coefficients
  - Code optimization in assembly level
    - Take the features of instruction set, micro-architecture and pipeline

Embedded software optimization techniques (2)



Embedded software optimization techniques (3)

- Phase 1
  - Analysis all kinds of constraint in the embedded system
- Phase 2
  - Analysis the feature of specific processor
- Phase 3
  - Software optimization in C and in assembly code
  - There have not ever existed a good complier for the embedded processor to meet the requirement suggested from phase 1
  - Embedded software must be close link to target processor
- Phase 4
  - Software and hardware co-validation to make certain to reach the goal

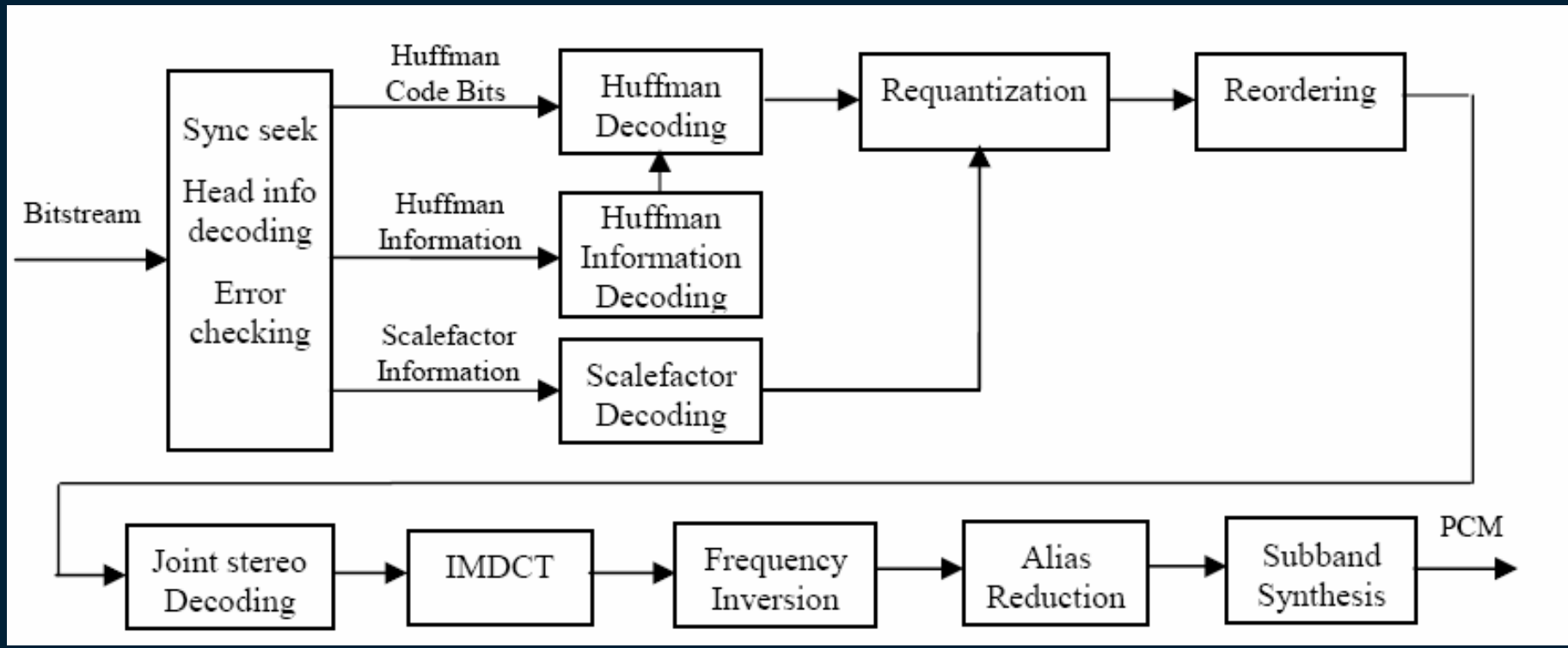
Software optimization in high level language

- Software module partition and performance estimation
  - Divide application software into small module
  - Run and get the profile information
- Module algorithm optimization
  - Modify algorithm to match with the used processor
- General optimization
  - Implement general optimization

Software optimization in assembly level

- Resource statistics
  - Memory sizes of objective code
  - Execution MIPS
- Memory size optimization
- Assembly program structure optimization
  - Take the features of instruction set, micro-architecture and pipeline into account

MP3 decode flow



Profiling of ISO reference decoder

- Employing **floating-point** computation

PROFILING OF ISO REFERENCE DECODER

Module	CPU Time(%)
Huffman decoding	4.1
Requantization	16.9
Stereo processing	1.2
IMDCT	17.9
Subband synthesis	58.2
Other	1.7

Performance analysis

- Integer computation-based

PERFORMANCE OF OUR INTEGER DECODER WITHOUT OPTIMIZATION

Module	MIPS ON ISA simulator	CPU Time (%)
Huffman decoding	4.1	6.1
Requantization	3.2	4.8
Stereo processing	1.5	2.2
IMDCT	19.2	28.7
Subband synthesis	33.4	50
other	5.4	8.1
total	66.8	100

Algorithm Optimization in C-code level -- IMDCT

- Using the Britanak & Rao's algorithm

MULTIPLICATION & ADDITION IN IMDCT

	36 points long block		12 points short block	
	Mul	Add	Mul	Add
ISO reference	648	630	72	66
Britank Rao's	47	165	13	39

Algorithm Optimization in C-code level – subband synthesis

- Two complexity
  - Matrix operation
    - Can be computed by a 32-point DCT and some data copy operation
  - Window filter
- Using Lee's DCT algorithm

MULTIPLICATION & ADDITION IN (1)

	Mul	Add	Coefficient Sizes
ISO reference	2048	1984	2048
Lee's	80	209	30

Algorithm Optimization in C-code level – zero value optimization

- Main idea
  - The result is always zero during zero value of input
- Set tags to indicate zero position
- Ex :

Without zero value optimization

For (i=0 ; i<576 ; i++)

Output[i] = function[huf$_i$]

With zero value optimization

For (i=0 ; i<Position_zero ; i++)

Output[i] = function[hufi]

For (i=Position_zero ; i<576 ; i++)

Output[i] = 0

Memory Size Optimization (1)

- The memory size of MP3 decoder related to
  - Coefficient sizes, bytes per coefficient and data spacing of decoding

$$xr_i = sign(huf_i) * |huf_i|^{\frac{4}{3}} * 2^{\frac{1}{4}(global\_gain[gr]-210)} * 2^{-(scale\_mul*(scale\_l[sfb][ch][gr]+preflag[gr]*pretab[sfb]))}$$

- It needs $|huf_i|^{4/3}$ and 2^x (x is not a integer number)

Memory Size Optimization (2)

- The value of huf_i varies from 0 to 8206
  - Need 16Kbytes to store these coefficients if a coefficient take 2 bytes
- Modify the computation
  - $| huf_i |^{4/3} = 16 \times | huf_i/8 |^{4/3}$
  - Setting upper bound
    - (huf_i > 256) ? 255 : huf_i
- It needs only 256 coefficients with unaware of distortion

Memory Size Optimization (3)

- Another problem : $2^{1/4(global\_gain[gr] - 210)}$,

  $0 \leqq global\_gain[gr] < 255$
- It can be decomposed as $2^N 2^{i/4}$
  - 2^N can be realized by shift
- Need only 4 coefficients

CPI Optimization

- $CPI = CPI_{idea} + CPI_{stall}$
  - eliminate the CPI_{stall}
- The main reasons causing pipeline stall
  - JBU (Jump Branch Unit) is placed at EX stage
  - Data cache access is separate into DM and TC stage
  - Cache miss penalty

CPI Optimization (cont.)

- Solutions
  - Modify the order of assembly program to eliminate control hazards and load stalls by delay slot techniques
  - Reduce program and data space
  - Rearrange data space

Experiment result

THE OPTIMIZATION RESULTS

Item	Results
Without optimization	90MIPS with CPI=1.35 and 80Kbytes memory
Algorithm optimization in C	Reduce 13.6MIPS 15.1%
General optimization in C	Reduce 7.2MIPS 8.0%
Zero value optimization	Reduce 9.0MIPS 10.0%
Code optimization in assembler	Reduce 12.2MIPS 13.6%
Programs after optimization	48MIPS with CPI=1.15 and 49Kbytes memory

Conclusion

- Proposed embedded software code optimization techniques
  - optimized the MP3 decoding program on RISC32 according to these techniques
- Algorithm optimization in C language for achieving smaller computation complexity
- Optimize target code for achieving low CPI value and small memory sizes on RISC32