A FIXED-POINT MPEG AUDIO PROCESSOR FOR LOW FREQEUNCY OPERATION

Yongseok Yi and In-Cheol Park

Department of Electrical Engineering and Computer Science, KAIST, Taejon, Korea
yslee@ics.kaist.ac.kr, icpark@ee.kaist.ac.kr

ABSTRACT[1]

A fixed-point pipelined processor optimized for decoding MPEG-1 audio layer III (MP3) is presented. Various examination and experiments on the decoding algorithm are exploited to lower the operating frequency. Accordingly, the dynamic scaling method is used to preserve the precision while computing in fixed-point arithmetic. Also, a novel instruction set and an addressing mode tuned for MP3 decoding are presented. The resulting cycle count required to decode a frame is so small that the proposed processor can operate with 12.8 MHz.

1. INTRODUCTION

Portable devices equipped with MPEG-1 audio layer III (MP3) decoding capability are becoming popular, where the power consumption problem is becoming a critical design factor. Judging from this point of view, lowering the operating frequency is valuable because it directly leads to lowering the power consumption.

Several previous works have implemented the MP3 decoding into VLSI circuits [1][2][3][4][5]. All the embedded or DSP processors employed in those implementations are developed based on conventional instructions and architectures, thus the processors are not tuned for MP3 decoding, leading to higher operating frequency and power consumption than expected.

This paper mainly focuses on the design of a fixed-point processor optimized for MP3 decoding. By conducting various examinations and experiments on the decoding algorithm, a novel instruction set is proposed to reduce the cycle count consumed in the computationally intensive stages, i.e. the IMDCT synthesis and the polyphase synthesis. The reduction of cycle count required to decode a frame is critical in determining the lowest frequency that a decoder operates at. In algorithmic and architectural view, lowering the frequency (f) and/or reducing the effective load capacitance (C_L) of the circuitry achieve low power consumption.

Using simple fixed-point arithmetic instead of floating-point arithmetic reduces the circuit area and lessens the time budget, resulting in the reduction of f and C_L. Reducing the bitwidth of memory and datapath also helps to reduce C_L. Since MP3 decoding is a real-time application, the time allowed for the decoding is fixed. If we reduce the cycle count for a given time interval, f will be reduced again.

This work was supported in part by the Korea Science and Engineering Foundation through the MICROS center at KAIST, Korea.

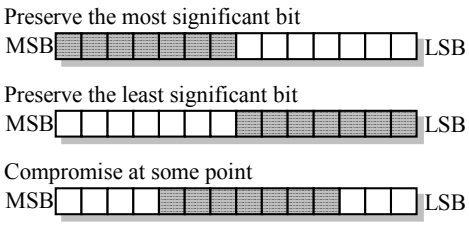

Fig. 1 Choices of scaling

2. FIXED-POINT IMPLEMENTATION

Although the fixed-point implementation results in an area- and power-efficient architecture working at a lower frequency, it suffers from the risk of precision loss. The bitwidth must be minimized to reduce hardware complexity and power consumption, while preserving the precision to comply the standard's requirement.

To determine the minimal bitwidth required for data-processing, we converted a floating-point version of an MP3 decoding program to a fixed-point version, where we have learned that it requires at least 70 bits to cover the whole dynamic range, 384 dB.

2.1. Dynamic scaling method

Since we are not allowed to use the bitwidth as wide as required, we have to somehow scale the values to fit in a rather short bitwidth. If the bitwidth is not wide enough to cover the dynamic range, some of the values are lost. The amount scaling determines which portion of the values would be lost. Two extreme choices of scaling are shown in Fig. 1. If the values are scaled to preserve the most significant bit, small values will be lost. On the other hand, if the values are scaled to preserve the least significant bit, large values will be saturated. Therefore we need to make a compromise between the two choices.

A granule, half of a frame, consisting of 576 sample data is the basic unit of MP3 decoding. Since not all the sample values reach the maximum range, the compromise is made at the point where at the point where the scaled values just cover the largest data in each granule. The scale factor, determined differently for each granule, is kept throughout the rest decoding procedures to be used for rescaling the output data at the final stage.

2.2. Finding minimum bitwidth

To find the minimum bitwidth with minimum precision loss, we ran the C model of MP3 decoding algorithm using fixed-point

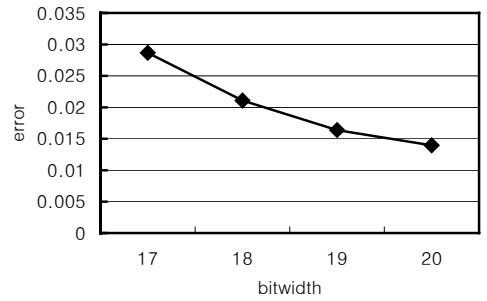

Fig. 2 The RMS error for varying bitwidths

arithmetic, varying the bitwidths. Fig. 2 depicts the result of the experiment.

The MPEG-1 audio compliance requires for a provided test sequence that the RMS level of the difference signal between the output of the decoder under test and the supplied reference output is less than 2^{-11}/sqrt(12) for a sine sweep (20 Hz – 10 kHz) with an amplitude of –20 dB relative to full scale [6]. As can be seen in Fig. 2, the bitwidth should be more than 20 to satisfy the requirement. The bitwidth of memory, register, and datapath is therefore set to 20.

3. INSTRUCTION SET

The minimum performance of an MP3 decoder must guarantee real-time decoding. Since a frame contains 1152 samples, when the highest sampling frequency, 48 kHz, is applied the decoder should be able to decode one frame in 1152/48000 = 0.024 seconds.

Once the time constraint is given, we need the target clock frequency. By the low-power strategy, we have first targeted 13.5 MHz, which is the system clock frequency of MPEG-1. Given this frequency the decoder must have the performance of 0.324 MOPS.

The instruction set of our audio processor was developed on the basis of the implementation of MP3 decoding algorithm using the *basic instruction set*, the set of instructions that might be supported by almost any architecture. When described using only the basic instruction set, the decoding algorithm's cycle count for a frame was 1,257,701 cycles, about 4 times the required 324,000 cycles. When all loops were unrolled, the instructions counted 563,789. This is the number of program memory access per one frame.

Based on the analysis of the assembly code, we modified the instruction set. First, NOT (1's complement), XOR (logical exclusive-OR), TST (test) and SHR (shift right logically) were removed because they were not necessary at all. Moreover, we found that the use of NEG (2's complement) instruction was restricted only to the case in which the operand and result is the same.

Second, the sequence [LDR (load a register from memory) → data processing instruction → STR (store a register in memory)] was found very often, i.e. 152223 times. The sequence includes [LDR→STR], which implies a move operation. This pattern is very natural because multimedia applications, such as MP3, are data-intensive. So rather than just relying on LDR and STR for an operand fetch, We enabled the data processing instructions fetch their operands directly from memory to remove the overhead of load and store instructions. This also led to the removal of LDR and STR because MOV took their place.

Third, the IMDCT stage and polyphase synthesis stage contained repetitive multiply-accumulate due to the linear convolution operations. To remove the branch overheads involved in these operations and make the multiplications and additions be pipelined, we added the SMAC (successive multiply-accumulate) instruction. It is able to execute up to 32 successive MAC's. It can use the operands only from memory, assuming the operands' addresses are supplied by AGU's.

Fourth, since the algorithm inherently contains many loops, we added the RPT (repeat) instruction to avoid the overhead of software looping. This instruction repeats a given number of following instructions for a given number of times. It does not accommodate nested loops to avoid the increase of hardware complexity.

The length of an instruction is fixed to 20 bits.

4. ARCHITECUTRAL DESIGN

The presented architecture is designed so as to accommodate the instruction set introduced in Section 3. It is fully pipelined and has two separate memory banks and an additional bitstream buffer to hold the incoming bitstream. The bitstream buffer is a dual buffer, thus it guarantees that the core can decode the bitstream continuously without having to wait for the buffer to be filled.

4.1. Pipeline stages

We used a DSP-like pipeline, [IF (instruction fetch) → ID (instruction decode) → OF (operand fetch) → EX (execution) → MEM (memory operation)]. Each stage is isolated by flip-flops using a single-phase clock. The cycle counts of the execution stage may vary with respect to the instructions, e.g. SMAC's iteration number can be up to 32 and then its execution stage extends to 33 cycles.

4.2. Datapath

Fig. 3 depicts the datapath along with other non-datapath elements. Detailed control signals are omitted to ease reading. Many of the signal lines going back and forth are the forwarding paths needed for fluent pipeline operation.

The execution stage is shown in more detail in Fig. 4. The 20 x 20 multiplier that can execute the multiplication in a single cycle is an array multiplier based on radix-4 Booth's algorithm. The multiplier used 4-to-2 compressors to sum up the partial products. We did not include the final adder in the multiplier to save area. Instead, the final addition is accomplished by a CLA (carry look-ahead adder) in ALU. Thus, the multiplier receives two 20-bit operands and emits a 40-bit sum vector and a 40-bit carry vector. These are held in the registers, op1 and op2, respectively. In the next cycle, they are added together in the ALU to produce the final result. So the multiplication takes two cycles even though it can be done in once cycle, relieving the time budget of the circuit. A MAC operation also consumes two cycles. Since the execution stage can perform a multiplication and an addition in parallel, the SMAC instruction can be fully

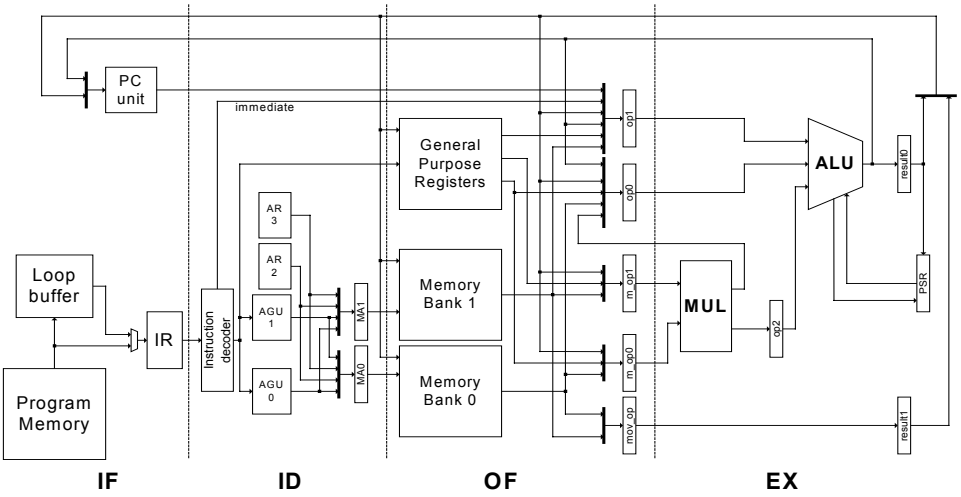

Fig. 3 Block diagram of datapath

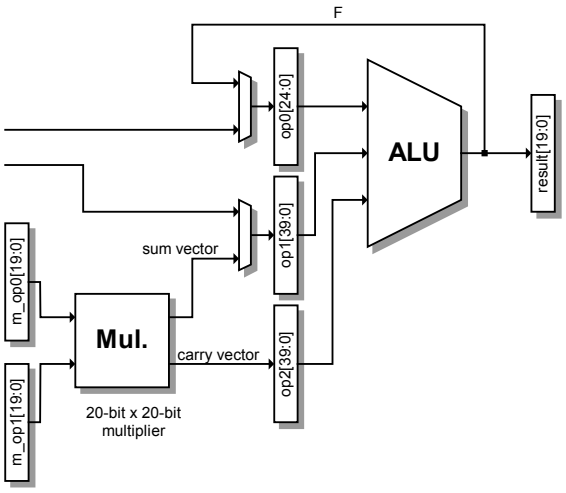

Fig. 4 Block diagram of execution stage

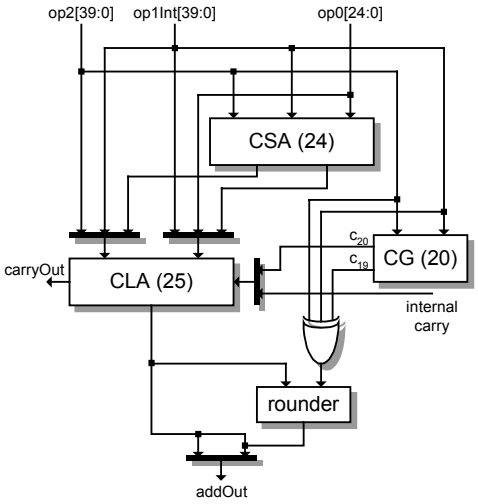

Fig. 5 Block diagram of adder

pipelined within the stage.

Fig. 5 shows the adder in detail. The adder adds three operands using 3-to-2 compressors. Only the upper 20 bits of the result are calculated because not all the 40 bits of the multiplication result are necessary. A 20-bit CG (carry generator) for the lower half, using the carry-look-ahead mechanism, is used to keep the result accurate.

4.3. Address generation unit

The presented processor basically supports register-indexed addressing as well as direct addressing. The address generation unit (AGU) calculates the data memory address according to the specified addressing modes.

In the alias reduction stage of MP3 decoding algorithm, a symmetrically increasing and decreasing address pattern is observed, as depicted in Fig. 7. The numbers in the boxes represent addresses and the numbers below them represent the order of accesses. This partially symmetric addressing is

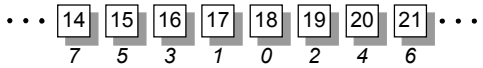

Fig. 6 Partially symmetric address pattern

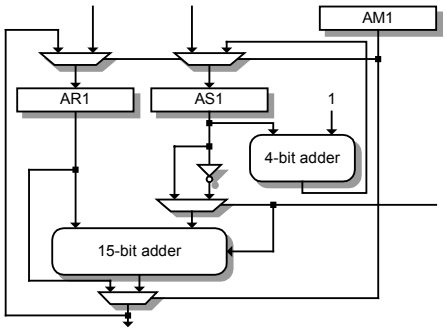

Fig. 7 The block diagram of AGU1

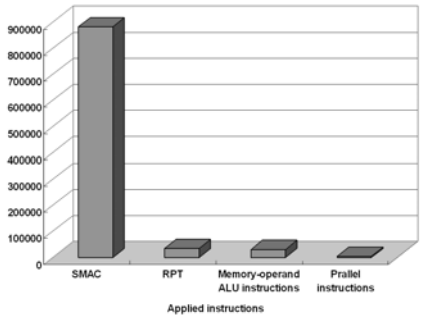

Fig. 8 Cycle count reduction

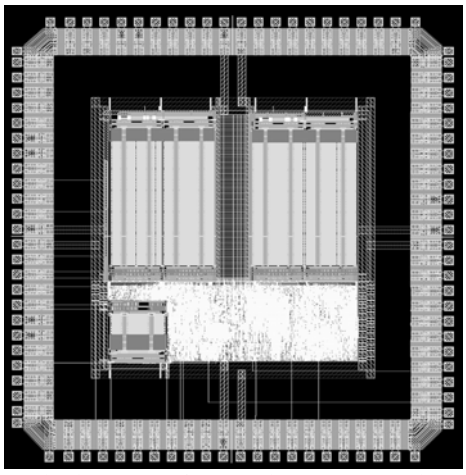

Fig. 9 Layout of the audio processor chip

performed by AGU1 (see Fig. 7). The partially symmetric addressing requires subtractions from the base address. So AGU1 is designed to perform the subtraction using an adder. Controlled by the control unit, the adder takes AS1 inverted and the carry in as 1 when a subtraction is needed. The subtraction and addition mode is toggled cycle by cycle. Along with this, AS1 is incremented using another 4-bit adder and the address register AR1 keeps being updated.

5. RESULTS AND COMPARISONS

The number of cycles to decode one frame is reduced due to the new instruction set. This is reported in Fig. 8. Apparently, most of the cycles reduced are due to the SMAC instruction.

With the final cycle counts, namely 306139 cycles, we can estimate the operating frequency. The estimated frequency is,

$$\frac{306139 \text{ cycles}}{0.024 \text{ seconds}} = 12.8 \text{ MHz} .$$

The result is lower than the target frequency by about 5.2 %.

In Table 4, the proposed decoder is compared with previous implementations. The bitwidth of the proposed decoder is equal to the previous minimum, and the resulting operating frequency is lower than any previous implementations but [3] which has little programmability.

A fully synthesizable Verilog HDL model of the presented audio processor is described in RTL level. The gate count of the synthesized circuit is 25,736. Fig 13 shows the layout of the circuit, where a 0.35-μm standard CMOS technology is used. The size of the core, including memory blocks, is 2.4 mm × 2.4 mm and the size of the die is about 4 mm × 4 mm. The chip is supplied by the power of 3.3 V.

TABLE 1
RESULT COMPARISON

	Freq. (MHz)	Bitwidth (bits)	Flexibility	Arithmetic
[1]	25-30	24	High	Not specified
[2]	20	20	High	Fixed-point
[3]	11.5	24	Low	Not specified
[4]	14.73	20	High	Not specified
[5]	27	32	High	Float-point
Proposed	12.8	20	High	Fixed-point

6. CONCLUSIONS

In this paper, the architecture of a low-frequency audio processor tuned for MP3 decoding has been presented. To achieve low operating frequency, we exploited fixed-point arithmetic and reduced the cycle count using a new instruction set optimized for the algorithm. To minimize the bitwidth of fixed-point arithmetic while preserving precision, dynamic scaling is proposed and implemented. In addition, the SMAC instruction that can perform 32 MAC operations serially in a pipelined manner and the partially symmetric addressing mode with corresponding address generation unit structure are proposed to reduce the cycle count. The resulting cycle count required to decode an MP3 frame is so small that the proposed processor can operate at 12.8 MHz.

7. REFERENCES

[1] G. Maturi, "Single Chip MPEG Audio Decoder," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 3, pp. 348-356, August 1992.

[2] F. –O. Witte, R. Backes, M. Klumpp, E. Schidlack and M. Ulrich, "Multipurpose Audio Signal Compander," *IEEE Transactions on Consumer Electronics*, vol. 39, no. 3, pp. 260-268, August 1993.

[3] T. –H. Tsai, T. –H. Chen, and L. –G. Chen, "An MPEG Audio Decoder Chip," *IEEE Transactions on Consumer Electronics*, vol. 41, no. 1, pp. 89-96, February 1995.

[4] *MAS3057D MEPG 1/2 Layer 2/3 Audio Decoder Data Sheet*, Micronas, Freiburg, Germany, March 2000.

[5] S. Hong, B. Park, Y. Song, H. Seo, J. Kim, H. Lee, D. Kim and M. Song, "A Full Accuracy MPEG1 Audio Layer 3 (MP3) Decoder with Internal Data Converters," *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference*, pp. 563-566.

[6] *ISO/IEC 11172-3, Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media up to 1.5 Mbits/s*

[7] I. Koren, *Computer Arithmetic Algorithms*, Prenitce-Hall International, 1993.