A PROGRAMMABLE DSP CORE DESIGN FOR SPEECH/AUDIO CODEC SOC

Chin-Teng Lin, Jen-Feng Chung, and Der-Jenq Liu,
Department of Electrical and Control Engineering, National Chiao-Tung University, Taiwan,
jfchung@falcon3.cn.nctu.edu.tw

DOI: 10.36724/2664-066X-2022-8-1-20-24

ABSTRACT

A novel programmable DSP core, called LASP24 (Low-cost Application-driven Speech Processor, with 24-bit data width), is developed. It is targeted as a wide-range platform of multimedia applications. The processor core is optimized to efficiently perform fundamental operations for speech/audio signal processing such as vector and matrix operations, and it can be easily embedded into the SOC platform. The design is fabricated with the UMC 0.18 m standard-cell technology in the total area of 6.5 mm2, and it can operate at 100 MHz. It has also been demonstrated that the MELP speech coding and the sound reverberation method could be executed in real-time on the LASP24 operating at 80 Mhz. The assembler and emulator environment have also been developed for designers to verify their algorithms.

KEYWORDS: *MELP, reverberation, vector, matrix, LPC, floating-point, emulator, autocorrelation.*

The article is reworked from unpublished 2nd IEEE International Conference on Circuits and Systems for Communications (ICCSC) materials.

I. Introduction

In recent years, the tremendous progress in VLSI technology allows the integration of an increasing number of transistors on a single chip. The continuous development of multimedia signal processing (DSP) applications makes the algorithms more sophisticated, and they also rely on more computing power for real-time implementation. The trend of consumer electronics is "portable" that means realizing the multimedia processing algorithms on a stand alone low-cost DSP processor [1],[6] within a System-on-Chip (SOC) platform instead of on a high-performance CPU in a PC.

For speech communication, a 2.4 kb/s Mixed Excitation Linear Prediction (MELP) algorithm [2] was standardized in 1997. MELP is similar to classical Linear Prediction Coding (LPC) but with additional features. The new standard provides equal or improved performance over 4.8 kb/s speech coder at only 2.4 kb/s. For audio effect, simulating an acoustic room means finding a system that describes the properties of sound beam propagation from the source of sound towards the listener in an open or closed room. Schroeder [3] implemented the first simulated room algorithm in 1961. Long reverberation times provide the feeling of a large hall, while short reverberation times give the impression of smaller rooms. These two applications can be embedded on the proposed programmable DSP core design.

DSP applications are generally characterized as computationally intensive with a large data set, accumulation-based operations, and loop-dominant control flow behavior. A general-purpose DSP can perform speech or audio processing with high flexibility, but the cost is relatively high. The optimal solution is an application-driven processor core [5], which has a lower cost than a general-purpose processor. In this paper, the implementation of an application-driven processor called LASP24 (Low-cost Application-driven Speech Processor, 24-bit data width) is proposed and it can be integrated into a 32-bit AMBA AHB system bus [4]. The specific addressing modes of LASP24 can quickly execute matrix operations without extra overhead. The advantages of this design include high flexibility in use, small area on silicon, high data throughput, fast portability to a wide range of mutimetia technologies, and integrated in the SOC design.

II. tem Architecture and Development

Accompanying the rapid development cycles, programmability is a fundamental requirement of a versatile platform designed for the new-generation multimedia applications and standards. The first step of developing a speech/audio codec SOC is to design a programmable DSP core. The proposed system architecture, as shown in Fig. 1, comprises a microcontroller and a programmable core. A 32-bit AMBA system bus connects all components to off-chip SDRAM memory via the SDRAM controller. In the AHB bus, the 8051 microcontroller and the LASP24 core are employed as the master and slave, respectively.

The development of the speech/audio processor is to meet the system demands that are based on sophisticated arithmetic algorithms and that emphasize on both hardware and software solutions. The verified tools offer the opportunity to trade off between software (for flexibility) and hardware (for functionality and performance). The development flow consists of two parts: hardware implementation and software development. These two tools, Assembler and Emulator, can quickly verify developed algorithms. The assembler can translate LASP24's assembly language into binary codes (or called machine codes). The emulator can emulate the computations of the processor hardware and verify the precision of different floating-point formats such as 32- or 24-bit.

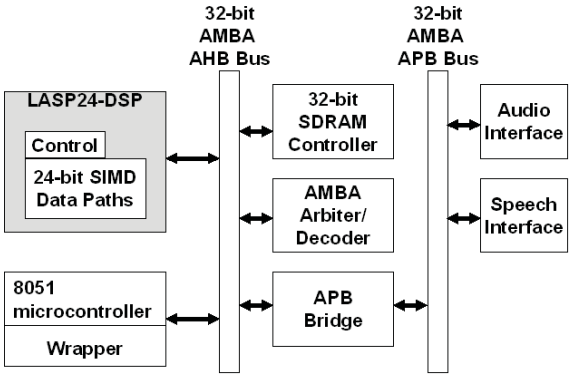

Fig. 1. Architecture of speech/audio codec SOC.

In order to verify and debug the DSP programs, a tool called DEFY-I is developed for functional emulation. The DEFY-I is an instruction-set-level hardware emulator for the processor core. With the emulator, the instructions could be taken out from the program memory and put into the instruction register for instruction analysis and execution. Finally, the execution results are written back to the register file or data memories. The flowchart of DEFY-I is shown in Figure 2. The whole emulator is constructed as the functional simulation kernel and could connect to other peripheral devices to perform the memory and display functions.

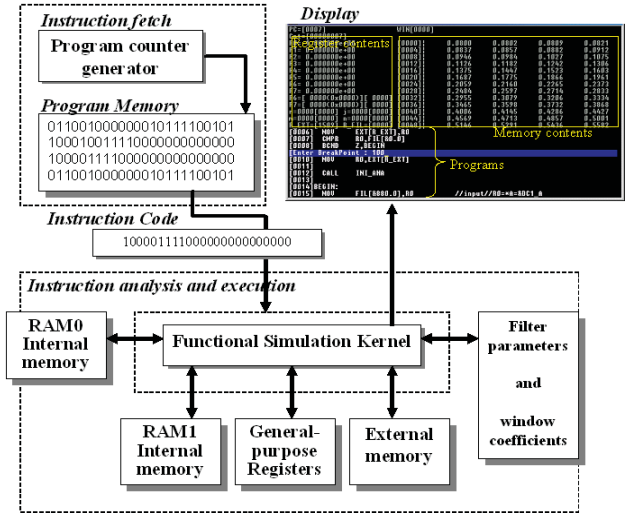

Fig. 2. The structure of DEFY-I for LASP24.

III. Design of LASP24 Core

The LASP24 is a efficiently parallel DSP core with a floating-point operating unit. Floating-point provides fast, accurate, and precise computations. The floating-point format is compatible with 24-bit of IEEE 754 standard. It has a 24-data-path single-instruction/multiple-data (SIMD) processor core controlled by five addressing modes, and a five-level pipeline executing engine. It is important to perform parallel multiplication and arithmetic operations in a single cycle. The effective execution time for most instructions is one cycle.

Figure 3 shows a block diagram of proposed LASP24 core. The core consists of a computation unit, which indicates ALU, multiplier, and accumulators, a program control unit (PCU), an external bus control dictating LASP24 external buses, a vector address generator computing the addresses which are used in vector operations, and two on-chip data memories. The program control unit performs instruction fetch, decoding, exception handling, and wait state supports. The PCU enerators the next address to the program memory and controls hardware loops. The on-chip memory provides an esay data exchange between data paths, which is required for speech/audio processing algorithms.

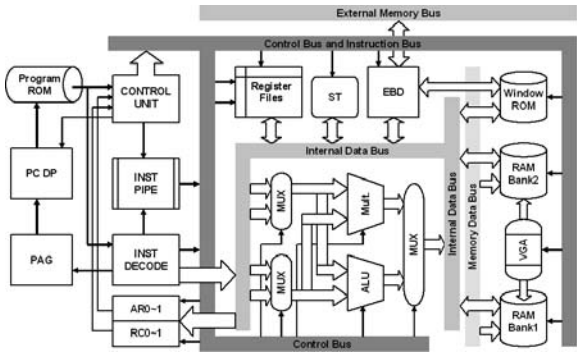

Fig. 3. The block diagram of the proposed DSP core.

LASP24 includes four register groups. The eight general-purpose registers (Register File) are capable of storing and supporting operations on 24-bit floating-point numbers. The two 8-bit auxiliary registers can be accessed by the processor and modified by the auxiliary register arithmetic unit. The primary function of the auxiliary registers is the generation of 8-bit addresses. They can also be used as loop counters or as matrix point register. The status registers contain information relating to the state of ALU and parallel multiplication. When the status registers is loaded, LASP24 sends out a busy signal, and executes the selected function. The two 8-bit repeat counters which used to specify the number of times are to be repeated when performing a block repeat. Concurrently, it has the ability of zero-overhead loop with a single-cycle branch.

A. Instruction Set and Addressing Mode

The ISA of LASP24 is defined as a fixed instruction length at 24 bits. A 24-bit instruction uses five bits each for addressing 8 general-purpose registers, two 3kb internal memories, and one 128kb

external memory. LASP24's instruction set is classified into three groups as data transfer, arithmetic, and control instructions. Each instruction supports 2- and/or 3-operand. The total of defined instructions is twenty-three.

The first group is "data transfer". LASP24 supports one load-and-store instruction (MOV) that can load or store a word (24-bit) between the memory and a register. The second group is "arithmetic and logic". There are 12 arithmetic and logic instructions including floating-point addition (ADD), subtraction (SUB), multiplication-and-accumulation (MAC), matrix multiplication (MPY), logic operations (AND, OR, and SHF), fixed-to-floating (FLOAT), floating-to-fixed (FIX), fixed-point addition (ADDI), and exponential operations (IEXP and EXP). The third group is "control". They control the data flow and perform the functions of no operation (NOP), loading auxiliary registers and filter coefficients (LDC and LDE), repeat and return blocks (RPB and RETB), unconditional branch (CALL), stack operations (PUSH and POP), comparison (CMPR), and conditional branch (BCND).

Most of speech and audio processing is related with auto-correlation, convolution, and FIR calculation. Hence, addressing modes are to enhance the hardware computing capability for the algorithms. Five types of addressing modes allow access of data and instruction words from memory and registers: register, direct, indirect, immediate, and vector addressing modes. Particularly, we design an auto-index method which uses auxiliary registers to address memory data as shown in Fig. 4. This method called vector addressing can easily get memory data in a single multiplier instruction. When the instruction decoder gets the vector address, the address would represent the coordinate of the matrix. Matrix multiplication is based on the operation of RAM0 and R3 (the third general-purpose register). The results are stored to the R3 register. An example for the equation of matrix multiplication is as

$$y = \sum_{j=r-1}^{0} x[k, r-j]h[j+1, r].$$

We can replace the above with the following LASP24 micro codes:

```
      RPB j, #r-1                  //set repeat block counter
L1:   MOV WIN[j+1, r], R3;         //move a coefficient to R3
      MPY R3, RAM0[AR0, r-j], R3   //matrix multiplication
      ADD R1, R3, R1               //R1=R1+R3
      RETB j, L1                   //if j≠0, return to L1
```

The index of a matrix coordinate is defined by auxiliary registers (AR0 and AR1). The index can automatically increase so that the pointer indicates the next matrix address. Hence, this addressing method enables a single-instruction matrix computation so that the size of program memory and the number of program memory access can be reduced.

Additionally, we also define a control mode to control data paths. The control mode is designed for program control and setting of repeat counters. By two auxiliary registers, LASP24 can execute four-level nested program.

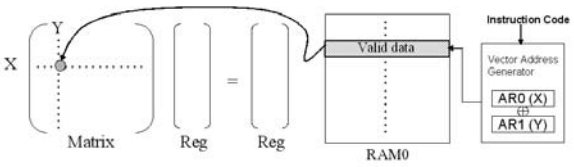

Fig. 4. Illustration for computing a matrix address with the vector addressing mode.

B. Special Vector Processing Techniques

The vector processing scheme [7] provides an approach to accelerating the processing of data streams. This technique can provide a significant speedup for communications, multimedia, and other performance-driven applications by using data-level parallelsim. The vector instruction format and addressing representation are shown in Fig. 5.

VC <= VA[AR_A] OP VB[AR_B]									
23 ~ 19	18~16	15~14	13~12	11~10	9 ~ 8	7 ~ 6	5 ~ 4	3 ~ 2	1 ~ 0
OPCODE	011	NU	FIL	EXT	RAM0	RAM1	VC	VA	VB

	FIL	EXT	RAM0	RAM1	VC	VA	VB
	13~12	11~10	9 ~ 8	7 ~ 6	5 ~ 4	3 ~ 2	1 ~ 0
00	FIL	EXT	AR0	AR0	RAM0	RAM0	RAM0
01	FIL+AR0	EXT+AR0	AR1	AR1	RAM1	RAM1	RAM1
10	FIL+AR1	EXT+AR1	AR0+AR1	AR0+AR1	EXT	EXT	WIN
11	FIL-AR0	EXT-AR0	AR1-AR0	AR1-AR0	R3	—	FIL

Fig. 5. The format of the vector addressing mode and the representation of vector addresses in LASP24, where OP indicates operation; VA, VB, and VC represent vector registers. FIL, EXT, WIN, RAM0, and RAM1 are memory symbols.

A vector multiplication instruction fetches data from RAM0 and RAM1 and feeds into ALU. ALU executes the "MAC" operation and adds the result to the accumulating register. The final results are stored to the external memory. An example of vector processing (100 points) is shown as:

```
L1:   MPY    RAM0(r), RAM1(r), EXT(r) //EXT(r) = RAM0(r) x RAM1(r)
      RETB   r, L1                    //r=r+1. if r=100, branch to L1.
```

The total execution time is about 200 clock cycles. Hence, we use a single instruction within a repeat block to execute the parallel multiplication-and-accumulation in the auto-correlation operation. The above example demonstrates that LASP24 has higher

performance in vector computation than the general-purpose DSPs.

IV. Physical Design

The processor core was developed by the synthesizable HDL discription. The layout view of LASP24 core is shown in Fig. 6. The total gate count is about 30,351 synthesized and estimated with UMC 0.18μm standard library (two 1024 by 16 data memories were not included). The silicon area required for this design is approximately 6.5 mm^2 and the 144-pin CQFP was adopted as the package.

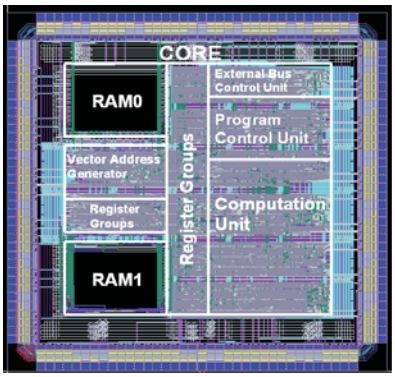

Fig. 6: Layout view of LASP24 core.

V. Results and Analysis

Complexity is measured using million instructions per second (MIPS), random access memory (RAM) and read only memory (ROM) measurements. MIPS are measured using the execution time and instruction counts. Linker memory maps are obtained with required sizes. As Table 1 shows, MELP complexity exceeds LPC and CELP in both processor and memory requirements. Additionally, the total performing cycles is listed for MELP, CELP, and reverberation algorithms.

Now LASP24 can perform the two practice applications in real time. We analyze the performance between them. For the MELP coder, the program performs 1338280 cycles in 60 MHz. The frame size is 22.5ms (180 samples) with a sampling frequency of 8000 Hz. Hence, the latency is about 21 ms (1338280×16.67 ns) for the encoder. As the result for the decoder, the latency is about 9.1 ms. Due to many filters used in the reverberation algorithm, the required execution time is larger. The program performs 1574430 cycles at 80 MHz. The frame size is 22.7 us (stereo channels) with a sampling frequency of 44100 Hz. The latency is about 19.67 us. Anyway, LASP24 can operate max frequency at 100 MHz. By the above analysis, it is able to satisfy all conditions with operating frequency 80MHz.

Table 1. Complexity comparison between processor (LASP24) and memory with optimization codes

Algorithm	MIPS	RAM	ROM	Cycles
		Unit: byte		
MELP Decoder	40	96K	10K	546449
MELP Encoder	60	96K	26K	1338280
CELP Decoder	30	14.8K	128K	364299
LPC	25	1.5K	805	71251
Reverberation	80	96K	30K	1574430

VI. Conclusions

In this paper, the LASP24 core developed for speech/audio codec SOC is presented. It can efficiently perform vector and matrix operations that are not usually supported by general-purpose processors. This design has been integrated in the total area of 6.5 mm^2 by using UMC 0.18μm 1P6M CMOS technology. It has also been demonstrated that the speech coding and audio processing algorithms can be executed on this programmable processor in real-time.

Acknowledgment

This paper was supported by the Brain Research Center, University System of Taiwan, under Grant 92B-711.

References

[1] H. R. Jang, S. H. Kim, and Y. H. Chang, "A digital signal processor for low power," in *Proc. AP-ASIC*, pp. 42-45, 1999.

[2] Federal Information Processing Standards Publication (Draft), *Specifications for the Analog to Digital Conversion of Voice by 2,400 Bit/Second Mixed Excitation Linear Prediction,* Jan. 1998.

[3] M. R. Schroeder and B. F. Logan, "Colorless artificial reverberation," *J. Audio Engineering Society*, vol. 9, 1961.

[4] ARM Ltd., *AMBA Specification Rev. 2.0*, http://www.arm.com, May 1999.

[5] J. Eyre and J. Bier, "The evolution of DSP processor: from early architectures to the latest developments," *IEEE Signal Processing Magazine*, Mar. 2000.

[6] M. Dolle, S. Jhand, W. Lehner, O. Müller, and M. Schlett, "A 32-b RISC/DSP microprocessor with reduced complexity," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1056-1066, Jul. 1997.

[7] C. G Lee and M.G. Stoodley, "Simple vector microprocessors for multimedia applications," *Microarchitecture, in Proc. 31st Annual ACM/IEEE Int. Symp.,* pp.25-36, Dec. 1998.