오디오 신호 처리를 위한 초저전력 DSP 프로세서

권기석, 안민욱, 조석환, 이연복, 이승원, 박영환, 김석진, 김도형, 김재현
삼성전자
kiseok.kwon@samsung.com

Ultra-low-power DSP for Audio Signal Processing

Kiseok Kwon, Minwook Ahn, Seokhwan Jo, Yeonbok Lee, Seungwon Lee,
Young-Hwan Park, Sukjin Kim, Do-Hyung Kim and Jaehyun Kim
Samsung Electronics

요 약

In this paper, we introduce SlimSRP, an ultra-low-power digital signal processor (DSP) solution for mobile audio and voice applications. So far, application processors (APs) have taken charge of all the tasks in mobile devices. However, they have suffered from short battery life problems to deal with complex usage scenarios, such as always-on voice trigger with continuous audio playback. From extensive analysis of audio and voice application characteristics, SlimSRP is designed to relive the performance and power burden of APs. It employs three-issue VLIW architecture, and the major low-power and high-performance techniques include: (1) an optimized register-file architecture friendly for constants generation, (2) a powerful instruction set to reduce the number of register file accesses and (3) a unique instruction compression scheme that contributes to saved memory size and reduced cache miss. An implementation of SlimSRP runs at up to 200MHz and the logic occupies 95K NAND2 gates in Samsung 28LPP process. The experimental results demonstrate that a MP3 decoder application with a 128kbps 44.1kHz input can run at 5.1MHz and the logic consumes only 22uW/MHz.

1. Introduction

With the keen attentions on the mobile devices today, it has become much more critical issue how to achieve the high performance, low power and small area at the same time, than for any other devices. So far, mobile devices employed a single application processor (AP) as a processing solution. However, as the functionalities of mobile devices are getting increased, and more peripherals are embedded into the mobile devices, complex usage scenarios are expected even during the screen-off state, such as device activation by voice detection in parallel with continuous audio playback and peripheral data processing. Such scenarios require higher processing capacity with low power consumption, but the conventional APs consume too much power. Thus we strongly need more optimized processor solutions for those scenarios with power and performance.

In this paper, we introduce an ultra-low-power digital signal processor (DSP) for mobile audio and voice applications, SlimSRP. As an assistant processor, SlimSRP is designed to relive the performance and power burden of APs in these applications. The architectural decision is made throughout an extensive architecture exploration and analysis of audio decoder and voice trigger applications utilizing our full-equipped tool environment. It achieves high instruction-level parallelism with three-issue VLIW architecture, as well as low power consumption with (1) optimized register file structure, (2) application-specific instructions and (3) instruction compression technique.

2. SlimSRP Architecture

Figure 1 shows the overall architecture of SlimSRP. The architecture comprises three functional units (FUs), a Data Register File (DRF), an Immediate Register File (IRF) and unified data memory system. The FUs are heterogeneous and optimized by extensive analysis of target applications such as audio decoder and voice trigger. In detail, while all FUs support basic operations, two of them support 32x32-bit multiplication and complex shift. Except for memory loads, all instructions complete within a single cycle. The data memory system allows three 32-bit accesses per cycle, two from the FUs and one from the system bus.

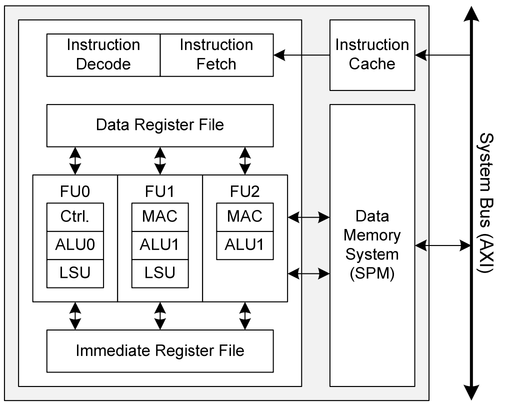

Fig. 1. Overview of SlimSRP Architecture

Throughout the following sections, we introduce the novel features and techniques applied to SlimSRP to achieve both low power and high performance.

2.1. Power-Optimized Register File Architecture

Register files are typical power- and area-consuming parts in modern multi-issue processors[1]. Accordingly, instead of using a single large register file, many DSPs employ several small register files each of which is dedicated to a specific data type, e.g. address register files for memory address and accumulators for bit-extended data[2]. However, this separation of register file prevents from efficient use of registers causing performance loss. Moreover, accumulators are incompatible with standard C data types resulting in inferior programmability[2].

In SlimSRP, we propose a distinctive register file architecture comprising DRF and IRF. While DRF consists of 32 entries for general 32-bit data, IRF holds only four entries to generate large constants. This architecture is decided from the deep insight for the common characteristic of the signal processing applications, where large constants are frequently used, e.g. coefficients of DCT, FFT and FIR filters. Also, C compilers handle addresses of global variables as large constants. In our experiment, we observed that up to 25% of instructions are used for large constants. Compared to DRF, IRF is very small and consumes much less power. Thus, our register file architecture can effectively contribute to reduce power consumptions for the target application.

Here, IRF does not replace but offloads a part of DRF' s role. The DRF still can handle large constants but with much more power. In addition, a constant belongs to one of the existing data types. Thus, this separation of register file does degrade neither register file usage nor programmability.

2.2. Application-Specific Instruction Set

An instruction count affects both performance and energy. If a given task can be performed with less number of instructions, it can save not only the execution time but also the power consumed by the instruction cache and the register files. In order to reduce instruction counts, SlimSRP provides a set of powerful instructions each of which performs a series of basic operations. The decision on the instruction set is the result of an extensive architecture exploration and analysis of signal processing applications.

One example of the powerful instruction is the dual-multiply-and-add instruction. The instruction performs two 32x16-bit multiplications, one 48-bit addition and one right shift in a single cycle. This instruction can be used to compute a sum-of-product type operation, which is widely used in transforms and filters. Another example is the bi-directional shift instruction. This instruction produces either a right-rounding-shift value or a saturating-left-shift value depending on the sign of the shift amount.

Additional speedup comes from improvements in the generation of large constants. In conventional RISC processors, at least three instructions are required to handle a large constant, since the instruction encoding space is limited to 32 bits. With the IRF and the unique instruction encoding scheme, we need only two instructions to handle a large constant in SlimSRP. The basic idea is to combine a part of constant generation into constant consuming instructions, such as add, compare, multiply, etc. Thus, prior to the computation, arithmetic instructions can create a 32-bit constant as an input operand. By reducing constant generation process, 25% speed up could be achieved in IMDCT kernel of the MP3 decoder.

2.3. Instruction Compression Scheme

As its name implies, a VLIW instruction is very long. For example, a single instruction of a three-issue 32-bit VLIW processor occupies 96 bits. Such a multi-issue architecture improves the performance with high instruction-level parallelism, but no-operations (NOPs) can be included in an instruction bundle.

In this work, we propose a novel instruction compression scheme that can enhance the performance and save power by improving the cache utilization and reducing the program code size. The compression scheme reduces the instruction size by removing the NOPs in an instruction bundle. In detail, each 32-bit sub-instruction has the two-bit compression field, where one bit represents the omitted NOP in prior to the sub-instruction, and the other bit represents the end point of the instruction bundle. The compiler fills this compression information and the hardware instruction decoder assigns NOPs to the corresponding FUs. Applying this compression scheme, about 26% reduction in code size is achieved in audio decoder applications.

3. Experimental Results

An implementation of SlimSRP runs at up to 200MHz and the logic occupies 95K NAND2 gates in Samsung 28LPP process. Simulation results show that a MP3 decoder with a 128kbps 44.1kHz input can run at 5.1MHz, an AAC decoder with a 128kbps 48kHz input can run at 5.4MHz, and a FLAC decoder with a 700kbps 48kHz input can run at 4.3MHz. In all cases, the dynamic power consumption is about 22uW/MHz. All the codes are compiled by the SlimSRP C Compiler without manual assembly coding.

Kenel	Cortex-M4	HiFi3	SlimSRP
FIR Filter	359,264	1,839,246	150,582
SAD	1,428,048	963,813	493,116
Median	1,750,100	1,384,825	965,651
Gaussian Filter	365,090	286,786	95,220
VLD	156,537	213,075	86,413

Table 1. Simulation Cycle Comparison (The smaller is the better.)

Table 1 shows execution cycles of some typical signal processing kernels on SlimSRP, Tensilica HiFi3 and ARM Cortex-M4. The result shows that SlimSRP achieves about 2x speedup over the popular processors in the embedded market.

4. Conclusion

In this work, we developed SlimSRP optimized for mobile audio and voice applications with an efficient register file architecture, a powerful instruction set and a novel instruction compression scheme. From the experiment results, SlimSRP showed satisfactory performance with very low power consumption. We believe that the proposed register file architecture is one of strong candidates in multi-issue DSP processors.

References

[1] R. Nagpal and Y. N. Srikant, Register File Energy Optimization for Snooping Based Clustered VLIW Architectures, In Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing, Oct. 2007.

[2] J. Glossner, et al., Trends in compilable DSP architecture, In Proceedings of 2000 IEEE Workshop on Signal Processing Systems, Oct. 2000.