

A Dual-core Embedded System-on-Chip Architecture for Multimedia Signal Processing Applications

Hong Yue

College of Computer, National
University of Defense Technology
Changsha, Hunan, P.R.China
yuehong_nudt@yahoo.com

Kui Dai

College of Computer, National
University of Defense Technology
Changsha, Hunan, P.R.China
daikui@chiplight.com.cn

Zhiying Wang

College of Computer, National
University of Defense Technology
Changsha, Hunan, P.R.China
zywang@nudt.edu.cn

Abstract - *This paper presents a dual-core embedded System-on-Chip for a wide range of application fields with particularly high processing demands, including general signal processing, video and audio processing, and a combination of these tasks. It integrates two processor cores and various interfaces onto a single chip, all tied to a 32-bit AMBA AHB bus. The RISC core coordinates the system and performs some reactive tasks, and the DSP core performs transformational tasks with more deterministic and regular behaviors, such as the small and well-defined workloads in multimedia signal processing applications. The DSP core is designed based on Transport Triggered Architecture (TTA) to reduce hardware complexity, get high flexibility and shorten market time. The processor is fabricated in 0.18 μ m standard-cell technology, occupies about 9.7mm², and operates at 266MHz while consuming 670mW average power.*

Keywords: dual-core, embedded multimedia processor, Transport Triggered Architecture, DSP, System-on-chip.

1 Introduction

The computing tasks of an embedded multimedia system can be roughly categorized into control-oriented and data-dominated, most of the times the computing platforms for media processing need to effectively handle both control-intensive and data-intensive tasks. Recent RISC architectures have been enhanced for data-intensive tasks by incorporating single-cycle multiply-accumulators, SIMD (MMX-like) datapaths, or specific functional units [1] to have the ability to perform data-intensive tasks, but the performance is still far behind that of a DSP with similar computing resources [2]. This is because data-intensive tasks are very distinct from general-purpose computations. Similarly, DSP is enhanced to have some GPP features but it is not effective.

This paper presents a dual-core embedded System-on-Chip for both RISC and DSP tasks. It integrates two processor cores and various interfaces onto a single chip, all tied to a 32-bit AMBA AHB bus. One core is a RISC core which coordinates the system and performs reactive tasks such as user interface; the other is a novel VLIW DSP core designed based on Transport Triggered Architecture

(TTA) which performs data-intensive tasks with more deterministic and regular behaviors. Applications are mapped to the two cores according to the characteristics, small and well-defined DSP algorithms are executed on DSP core, and the others are executed on RISC core.

This paper is organized as follows. In section 2, the proposed dual-core processor architecture is presented. The DSP core designed based on TTA are described in section 3. In section 4, we will describe inter-processor communication mechanism in detail. In section 5, the simulation and implementation results are reported. Section 6 concludes the paper.

2 SoC Architecture Overview

Dual-core SoCs are attractive candidate architectures for multimedia processing as multimedia schemes in general can be partitioned in control-oriented and data-dominated functions, which can all be processed in parallel on different cores. Each core can be adapted towards a specific class of algorithms, and individual tasks can be mapped efficiently to the most suitable core. The OMAP media processor of Texas Instruments company and the Tricore processor series of Infineon company are both this kind of processor [3][4].

In general, different approaches are adapted to accelerate execution on embedded processors. In most cases, some kind of parallelization technique is employed on instruction level (e.g., very long instruction word, VLIW), data level (e.g., single instruction multiple data, SIMD), or on task level (e.g., simultaneous multithreading) [5]. Even combination of these techniques is used. Another very powerful means to accelerate multimedia processing is to adapt processors to specific algorithms by introducing specialized instructions for frequently occurring operations of higher complexity [6].

The SoC architecture, shown in Figure 1, comprises two cores that have been specifically optimized towards a particular class of tasks by employing different architectural strategies. The RISC processor is a 32bit processor compliant with the SPARC V8 architecture; it is optimized towards control-oriented tasks such as bit stream processing or global system control. The TTA-DSP processor is an eight-issue VLIW processor. It is particularly optimized towards high-throughput DSP-style

processing, such as FFT, IDCT or filtering. The TTA-DSP processor core is designed based on TTA architecture; it offers high data level parallelism and high programming flexibility, more details are described in the next section.

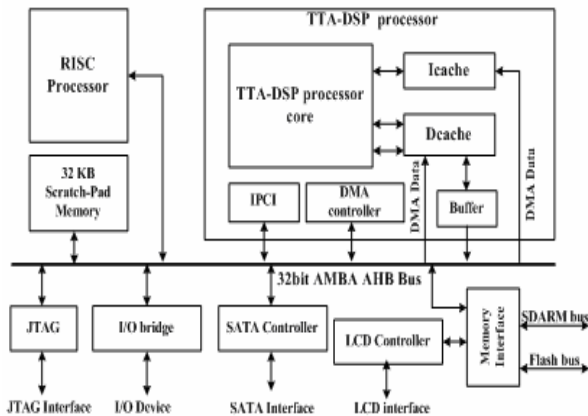


Figure 1 Dual-core system-on-chip architecture

A 32-bit AMBA AHB system bus [7] connects the two cores to off-chip SDRAM memory and flash memory via a 32-bit memory interface, to JTAG interface for debug, to SATA interface for access to the hard disk, and to the I/O bridge for access to different peripherals. The direct change of data between TTA-DSP Cache and external memory without placing a burden on the TTA-DSP core is supported by the DMA tunnels. A 32kB Scratch-Pad memory exists to store some constant data that are used frequently in one procedure. If these data are always loaded from external memory, it is very time-consuming. Buffer between TTA-DSP DCache and system bus is to hide the memory access latency. The inter-processor communication is handled by IPCI (Inter-Processor Communication Interface), the detailed mechanism will be described in Section 4.

3 TTA-DSP

TTA-DSP is a highly parallel DSP core with VLIW architecture. The processor core is designed based on TTA (Transport Triggered Architecture). Simple design flow, customized function units and flexible data level programmability make it very suitable for embedded DSP applications. The autonomously operating DMA unit can perform instruction prefetch and data prefetch transfers to ICache and DCache.

3.1 Transport Triggered Architecture

The main difference of TTA compared to traditional, operation triggered architecture is the way the operations are executed. Instead of triggering data transports, in TTA operations occur as a side effect of data transports, i. e., the execution begins when data is written to operand registers. This design implies that one data transport is needed to be explicitly appointed in the instruction. Figure 2 shows the basic TTA processor architecture [8].

The data path of TTA is organized as a set of function units (FU) and register files (RF). The data transfers between these units are done through an interconnection network that consists of desired number of buses. The architecture is very flexible because the number of FU, RF, RF ports, buses and bus connections can be changed unlimitedly. For example, by adding buses and connections, more data transfers can be executed in parallel and thus the execution time is reduced. The compiler supports all these changes [8].

In addition to the flexible basic architecture, TTA allows the designer to add application specific operations into the instruction set [9]. This is a very efficient way to improve performance since quite complicated operations may become trivial.

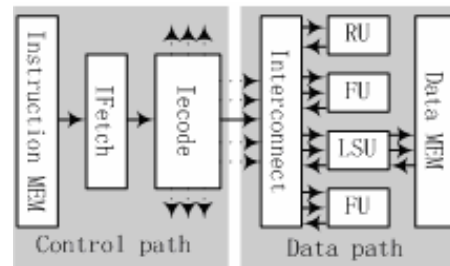


Figure 2 TTA architecture

The user software tools for TTA processor development include a compiler to translate high-level programming language into sequential code, a scheduler to schedule the sequential code and produce parallel code, and a simulator to verify and evaluate both the sequential and parallel codes. Then, a design space explorer can be used to test different TTA configurations for the application. Finally, synthesizable Verilog code can be automatically generated for the chosen configuration using the Processor Generator.

3.2 TTA-DSP Architecture

Because of the flexibility and scalability features of TTA, it is very suitable for application specific processor design. TTA-DSP is designed based on TTA according to the characteristics of multimedia signal processing applications. So many function units are customized to execute some specific operations. That also means special instructions are added to the instruction set. Figure 3 shows the TTA-DSP architecture. There are 4 integer function units (IFU) and 2 floating-point function units (FFU), 2 load/store units (LSU), 2 integer register files (IRF) and 2 floating-point register files (FRF). Another TriU unit is a function unit specific for trigonometric function calculation. All these function units are connected to 8 buses. That implies 8 data transfers can be executed simultaneously provided that there is parallelism in the application.

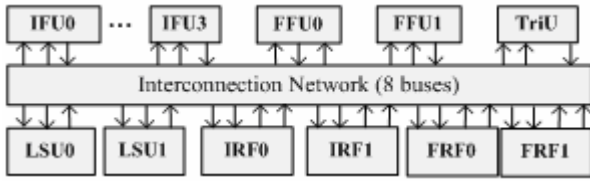


Figure 3 TTA-DSP architecture

To thoroughly exploit data level parallelism, SIMD data path is designed in the IFU. IFU integrates rich integer functions. In addition to the basic operations such as add, sub, multiply, subword operations and multimedia data manipulation operations such as mix, shuffle, pack and unpack, are also implemented in it. It means that multimedia instruction extension is done to the instruction set. Figure 3 shows the structure of an IFU. 4 IFUs are set to avoid hardware resources conflict. So at the same time, one IFU is executing multiply-accumulate, another IFU is executing add operation of 8 bytes (4 bytes packed in one 32bit register adds another 4 bytes packed in one 32bit register). Another function unit should be mentioned, that is TriU. It is designed based on CORDIC algorithm, so it can calculate many trigonometric functions in 32 cycles instead of other time-consuming or place-consuming ways [10].

In VLIW machines, the complexity of the register file grows rapidly as more and more FU are integrated on a chip. For N concurrent FU, the area of the centralized register files grows as N^3 , the delay as $N^{3/2}$, and the power dissipation as N^3 [11]. Thus, the register file will soon dominate the area, the delay and the power dissipation in the multi-issue processors as the number of FU increases. To solve this problem, TTA provides partitioned register files. 4 register files in TTA-DSP can be accessed by any FU if needed. So the complexity decreased without performance penalty. And data can be transferred from one FU to another directly for each FU has its own registers, so lots of middle results saving are bypassed.

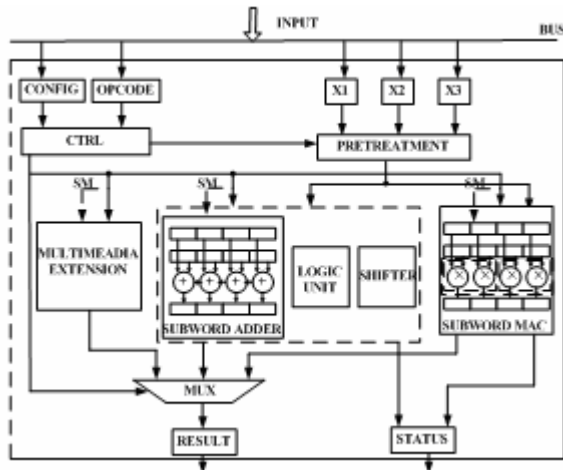


Figure 4 Integer Function Unit Structure

3.3 Software Development

For TTA-DSP, optimizing compiler is available that support its hardware features and perform code optimization, instruction scheduling, and VLIW parallelization. Based on the compiler, C programs can be easily mapped to the TTA-DSP hardware architecture, whereas some computation-intensive core algorithms are typically written in assembly language and scheduled by hand for full exploitation of all data path features. The TTA-DSP compiler is being extended to support subword data types for SIMD.

During the architecture definition phase of TTA-DSP cores, cycle-accurate simulator have been developed for the optimization of architectural features and instruction sets. The simulator continues to be used for application development. But the integrated simulator for both RISC core and TTA-DSP is under way.

4 Inter-Processor Communication Mechanism

There are many well-defined DSP kernel algorithms, which are executed on the DSP processor. A complete application begins to run on the RISC processor, for the control-oriented part is always first executed, and the DSP kernel algorithms will be called if needed. So the inter-processor communication is always happened during application running. The RISC processor and the DSP processor communicate via an interrupt mechanism. This mechanism provides a flexible software protocol between the two cores.

There are 6 registers in IPCI used for the communication mechanism. Three are from RISC to DSP, another three are from DSP to RISC. The two sets almost have the same meaning. One register is command register, the interrupting processor can use this register to pass a command to the interrupted processor. The second register is data register, the interrupting processor can use this register to pass a data or an address or status information to the interrupted processor. The third register is a flag register. If the interrupt is responded, the flag will be cleared.

On writing to the command register, an interrupt is generated to the other processor and the flag register is set. The interrupted processor must acknowledge the interrupt by reading the data register (if necessary) and the command register for the associated interrupt. The interrupt is reset and the flag register is cleared when the command is read by the interrupted processor. If the interrupt is masked in the interrupt handler when the command register is written, no interrupt is generated and sent to the processor. However, the flag is set. If the interrupt is unmasked at a later time, an interrupt is generated to the processor.

The following TTA-DSP start procedures are provided as an example.

- 1) The RISC processor writes to the RISC2DSPd (command register) the start address.
- 2) The RISC processor writes to the RISC2DSPc (data register) a "DSP start" command (predefined and understood by both processors). And the RISC2DSPf is set. This write issues the interrupt to the TTA-DSP.
- 3) In response to the interrupt, the interrupt handler acknowledges the interrupt by reading the registers. It reads the start address first and then read the command register. And the RISC2DSPf (flag register) is cleared.
- 4) The interrupt handler calls an interrupt service routine (ISR) which examines the data and command words and processes the work necessary. Here the TTA-DSP will begin to run from the start address which is specified in the RISC2DSPd.
- 5) The interrupt handler returns and the interrupted normal processing continue.

5 Implementation Results

5.1 Performance Evaluation

To fully exploit the data-level parallelism of TTA-DSP, some DSP kernel algorithms in assembly code are hand written. The complete multimedia application simulation on both cores is a hard task partition work, it is under way. So this section only presents the performance evaluation results of the TTA-DSP.

Figure 5 summarizes the performance comparisons between TTA-DSP and the state-of-the-art high-performance DSP processor TI TMS320C64xx. The results of TTA-DSP are get by our simulator, and the results of TMS320C64xx are get by TI development environment CCS2.0 [12][13]. Figure 6 summarizes the performance comparisons between TTA-DSP and the general purpose embedded processor LEON3 [14]. The results of LEON3 are got by LEON3 simulator [14]. All the 6 benchmarks are selected from TMS320C64x DSP Library. The parameters of the benchmarks are listed in table 1. They are measured in the execution cycles.

Table 1 Benchmarks Description

No.	Name	Brief Description
1	FFT	16-bit 256 points radix 4 Fast Fourier Transform
2	FIR	16-bit samples for 16-sample 40-tap FIR filter
3	IDCT	Inverse Discrete Cosine Transform on 8*8 block of 16-bit pixel data
4	MAT_MUL	Multiply of two 4*4 matrix
5	IIR	IIR filter of 500 output samples
6	MAXIDX	Get the index of the maximum value of a vector

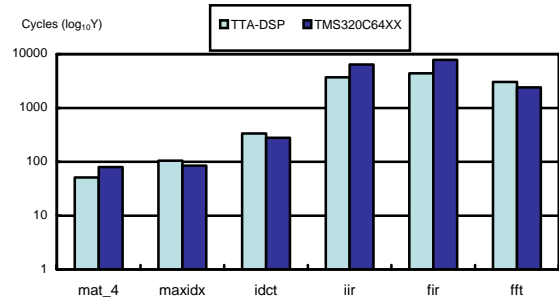


Figure 5 Performance Comparisons with TMS320C64xx

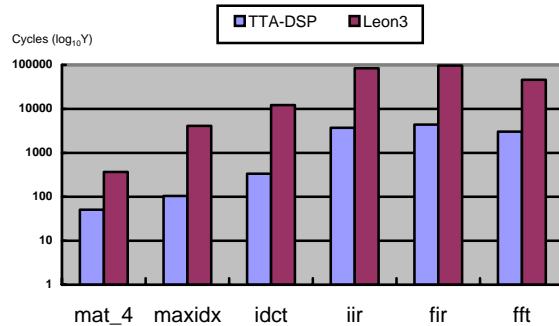


Figure 6 Performance Comparisons with LEON3

The simulation results show that the performance of our proposed TTA-DSP processor is better than that of the current popular DSP processors for various benchmarks, especially for the algorithms which contain trigonometric computations. The performance is also much better than that of the current popular general purpose embedded processors.

5.2 Silicon Implementation

We have implemented the SoC in Verilog RTL code, which is verified with Modelsim. The design is synthesized using Design Compiler from Synopsys with the 0.18um standard-cell library. The net-lists are then placed and routed using SoC Encounter from Cadence for the 1P6M CMOS technology. Figure 7 gives the layout of the proposed SoC. The area is about 21.7mm² and it can operate at 266MHz and consume 670mW average power.

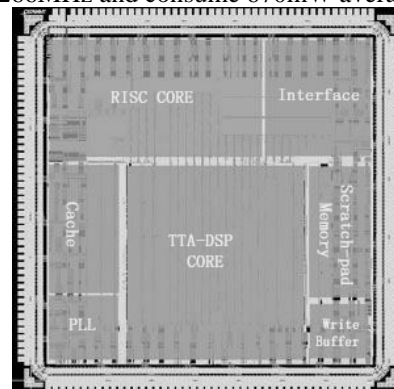


Figure 7 Layout of the proposed SoC

6 Conclusions

This paper presents the design and the silicon implementation of the proposed Dual-core System-on-Chip for multimedia signal processing applications. It contains a RISC processor and a DSP processor. The RISC core coordinates the system and performs control-oriented tasks, and the DSP core performs data-intensive tasks with more deterministic and regular behaviors, in the form of optimized assembly code. With effective mapping techniques, a wide range of multimedia signal processing applications can be fast executed on it. We are now studying the application partition techniques on two processors and the code optimization techniques for TTA-DSP architecture. An integrated simulator and development environment for both cores are developed.

7 Acknowledgements

This work was supported by Chinese NSF project No. 90407022 and some other funds. The authors would like to thank Andrea Cilio and his colleagues in Delft University of Technology in Netherlands for their great help and support to our research work.

8 References

- [1] M. Levy, "ARM picks up performance," *Microprocessor Report*, 4/7/03-01
- [2] R. A. Quinnell, "Logical combination? Convergence products need both RISC and DSP processors, but merging them may not be the answer," *EDN*, 1/23/2003
- [3] OMAP5910 Dual Core Processor – Technical Reference Manual, Texas Instruments, Jan,2003
- [4] Siemens, "TriCore Architecture". White Paper, 1998
- [5] Hans-Joachim Stolberg et al., "HiBRID-SoC: A Multi-Core System-on-Chip Architecture for Multimedia Signal Processing Applications," in *Proc. DATE'03*, pp.20008, March, 2003.
- [6] M. Berekovic, H.-J. Stolberg, M.B. Kulaczewski, P.Pirsch, et al., "Instruction set extensions for MPEG-4 video," *J. VLSI Signal Processing Syst.*, vol. 23,pp. 27 - 50, October, 1999.
- [7] "ARM Ltd. AMBA Specification Rev. 2.0. [Online]," Available: www.arm.com
- [8] Corporaal H., "Microprocessor Architecture from VLIW to TTA," John Wiley & Sons Ltd., West Sussex, England, 1998
- [9] Hoogerbrugge J., "Code Generation for Transport Triggered Architecture," PhD Thesis, Delft University of Technology, Delft, The Netherlands, 1996
- [10] Hong Yue, Kui Dai, Zhiying Wang, "Key Technique of Float Point Function Unit Implementation based on CORDIC Algorithm", in *Proc. Annual Conference of Engineer and Technology of China*, pp.102-105, April 2005.
- [11] S. Rixner, W.J. Dally, B. Khailany, P. Mattson, et al., "Register organization for media processing," in *Proc. HPCA-6*, 2000, pp.375-386.
- [12] TMS320C64x DSP Library Programmer's Reference, Texas Instruments Inc., Apr 2002
- [13] TMS320C64x CPU and Instruction Set Reference Guide, Texas Instruments, Inc, USA, 2000
- [14] LEON3 home page, <http://www.gaisler.com/>
- [15] TTay-Jyi Lin, Chie-Min Chao, "A Unified Processor Architecture for RISC&VLIW DSP", *GLSVLSI'05*, 2005.
- [16] T. Kumura, M. Ikekawa, M. Yoshida, and I. Kuroda, "VLIW DSP for mobile applications," *IEEE Signal Processing Mag.*, pp.10-21, July 2002
- [17] R. K. Kolagotla, et al, "A 333-MHz dual-MAC DSP architecture for next-generation wireless applications," in *Proc. ICASSP*, 2001, pp.1013-1016
- [18] T. Takayanagi et al., "A Dual-Core 64b UltraSPARC Microprocessor for Dense Server applications," *ISSCC Dig. Tech. Papers*, Feb. 2004, pp. 58-59
- [19] W. Cesario et al., "Colif: A Design Representation for Application Specific Multiprocessor SOCs," *IEEE Design & Test of Computes*, Sep-Oct 2001.