A Novel Data-Parallel Coprocessor for Multimedia Signal Processing

Lai Mingche, Dai Kui, Lu Hong-yi, Wang Zhi-ying
School of Computer National University of Defense Technology
Changsha, P. R. China.
mingchelai@chiplight.com.cn

ABSTRACT

Realizations of demanding embedded applications particularly in the field of signal processing often require high processing performance and low energy consumption which are far beyond what can be delivered by DSPs nowadays. A novel data-parallel coprocessor basing on the Transport Triggered Architecture （TTA） is presented in this paper. The coprocessor consists of two powerful arithmetic clusters, the stream memory as well as the optimized data transport network, and is good at exploiting the data parallelism in the computation intensive multimedia applications. Also, the solution of the clock-gating in a gradual way is advanced for the low power dissipation. Then, a SoC chip involving the coprocessor is implemented using 0.18μm CMOS process. Experimental results show that this coprocessor has good performance improvement for multimedia applications.

Index Term -- Data-parallel Coprocessor, Multimedia Signal Processing, Transport Triggered Architecture.

1. INTRODUCTION

Today, with the increasing computational complexity of the developing algorithms and the real-time constraints due to the increasing data rate, the relative importance of some embedded applications especially in the field of the multimedia signal processing continues to increase. These applications exhibit large amounts of parallelism, and specifically data level parallelism. But most embedded processors nowadays do not have enough hardware resources or performance to support these applications. Thus, many modern computer systems are designed to exploit data level parallelism to achieve high performance. These data-parallel architectures execute similar or identical operations concurrently on multiple data elements, allowing them to achieve high execution rates while tolerating high hardware complexity, low flexibility and configurability. On the other hand, another critical problem puzzling the designers still remains. The energy consumption that accompanies the high performance is often orders of magnitude beyond typical embedded power budgets. Obviously, it has already become the bottleneck of the development of the microprocessor nowadays.

Given the requirement of the high performance and the low energy consumption, a more reasonable approach to keep up with the real time requirement of sophisticated signal processing applications seems to use a high performance coprocessor in conjunction with a low power host processor. In this paper, we present our data-parallel coprocessor (DPC) which follows the TTA [1]. Thereby, the relative design space exploration and cost estimation of the TTA have been provided before the design, and the DPC core only represents the optimized architecture which is characterized by the high computational performance, low energy consumption and flexible programming ability.

The paper is organized as follows. Section 2 describes the TTA architecture. Section 3 presents the coprocessor including the function units, interconnect and the stream memory system. Then, the SoC implementation is presented in section 4. In section 5, we present experimental results with a set of benchmarks, illustrating the advantages of high performance and low power consumption. Finally, the paper concludes with section 6.

2. TRANSPORT TRIGGERED ARCHITECTURE

TTA proposed by Henk Corporal [1] can be viewed as a superset of traditional VLIW architecture. The major difference between TTA and traditional operation triggered architecture is the way that the operation is executed. Instead of triggering data transports, TTA operations occur as a side effect of data transports. At the time that the data is written into the register, the execution just begins. This means that there is only one data move operation in the TTA programming. Moreover, any instruction in TTA usually consists of several moves, depending on the interconnection network. By adding inner busses, more move operations can be executed in parallel and thus the execution time is reduced.

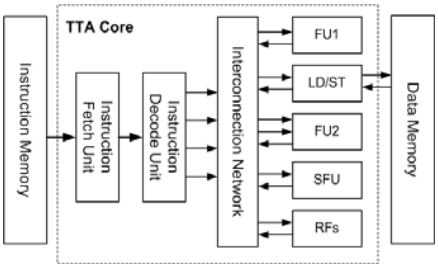

Fig.1. TTA architecture

The structure of the TTA core is organized as shown in Fig.1. There are an abundance of functional units (FU) and register files (RF) which are connected to the interconnection network by sockets. Every FU or RF has one or more operator registers, result registers but only one trigger register. Data being transferred to the trigger register will trigger the function unit to work. All operations, including load/store and branch/jump/link occur solely through moves. FUs operate concurrently and may incorporate pipelined execution. Thus, compared with the traditional VLIW architecture, TTA makes parallelism more fine-grained and flexible, which suits to the more efficient schedule in the instruction level as well as the data level.

3. COPROCESSOR

In the following sections, an overview of the DPC architecture is

proposed. Thereby, we start from the system view and then to the component level.

3.1 System view

The core of the data-parallel coprocessor is depicted in Fig.2. It consists of a coprocessor control unit (CCU), an instruction cache, two arithmetic clusters as well as the stream memory system. The DPC core has three interfaces, the control interface (Control IF) for control and synchronization tasks which are handled by the CCU, the 64bits instruction interface (Instruction IF) for the microinstructions and the 64bits data interface (Data IF) for the data exchanges between the local memory and the off-chip memory. Then, the DPC core is deployed with an 8KB pipelined instruction cache with the support of the DMA, which is responsible for the prefetch of the microinstructions [2]. Moreover, for the data-intensive characteristic of the stream data in the multimedia application [3], the DPC core also provides four local data memories, which comprise in total 16KB, to exploit the stream-like access patterns. The usage of the scratch-pad ram is to protect the processor state or to store some scalar and array variables from the off-chip memory to minimize the total execution time, while the stream caches which support the 2-way configurable prefetch technique are introduced for the regular array access patterns. Thereby, these four local memories can be used in a flexible way to minimize the memory accesses, which results to the good performance improvement for the multimedia applications.

As shown in Fig.2, the heart of the core consists of two arithmetic clusters, which follow the TTA to satisfy the high performance requirement in the data density applications. Moreover, each cluster has seven parallel SIMD data paths and an abundance of the computational resources, comprising four integer ALUs, three multipliers, one float units, one compare unit and one specific CORDIC unit (only in the first cluster). In each cycle, the specified data transfer in the transport network can trigger the operations of these functional units with the effective support of the 64-bits load/store unit. This allows for an extremely high utilization of the silicon area and suits for the different signal processing algorithms ranging from mobile communication systems to stream media applications.

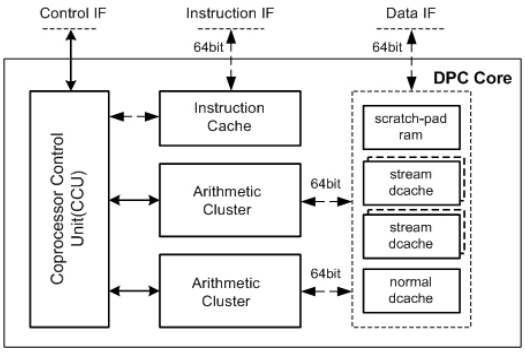

Fig.2. DPC Core Overview

3.2 Function Units

The arithmetic clusters in the core adopt the TTA, where the operation is triggered by the data transfer. In general, the typical function unit contains three types of the registers: the operand register (OR), the trigger register (TR) and the result register (RR). The computation is done by transferring values to the operand register and starting an operation implicitly via a move targeting the trigger register associated with the command. The generic function unit follows the organization shown in Fig.3. The operands may arrive from the input sockets or the shadow register (SR) which serves as the OR's shadow. In addition, the operands may be the output of its final stage or neighbors. Note that the data paths from the neighbor or the output are used as the bypass which is controlled by the compiler. Several cycles later after the triggered operation, the result will be written into the result register.

Several key function units are studied in the following.

1. Integer ALU performs the common operations including the arithmetical and logical ones. Then, it also supports the sub-word parallelism on byte or half-word entities with an extensive instruction set optimized with respect to media processing. This arrangement results to the operations in parallel and achieves to a high media processing throughput.
2. Pipelined multiplier supports 32-bit integers multiply operations with 2 cycle latency. Typically, it also performs the 2-way 16x16-bit or 4-way 8x8-bit multiplication.
3. FPU performs operations on single precision floating-point operands. All operations are IEEE-754 compliant. The FPU is fully pipelined and a new operation can be started every clock cycle. The only exceptions are the FDIV and FSQRT which require between 15 and 24 clocks, and which are not pipelined.
4. Compare unit not only supports compare operations but also returns a result, which can be used to predicate the conditional transfers by the condition codes. This makes if-conversion and conditional data flows [4] possible.
5. Sine/cosine unit bases on the parallelization of the original CORDIC algorithm and adopts a relatively simple prediction scheme through an efficient angle recoding. The proposed implementation, a pure combined logic without pipeline operation, has a delay of 27 clocks.

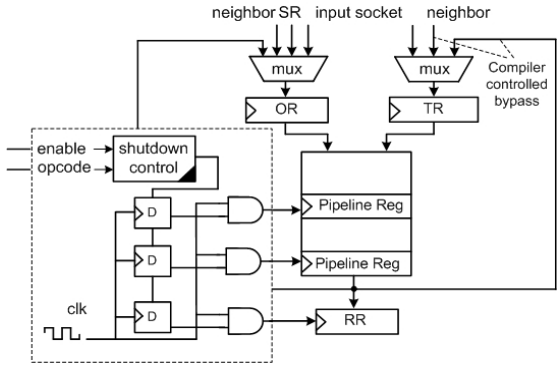

Fig.3. Function Unit Architecture

As shown in Fig.3, another distinguishing feature of the unit is that the CCU can manage its activity cycle by cycle with the support of the clock-gating. By the way, the clock-gating in Fig.3 works in a gradual way to reduce the inductive noise. During the normal operation mode, each stage gets the same clock. When the unit needs to be turned off, the enable signal travels down the flip-flops chain and each stage get clock-gated one per cycle. Then, this scheme also requires less hardware overhead in terms of the enable gates and shutdown controller. Once the function units are gated off, it involves less cycle-to-cycle current variation which could introduce large transient power.

3.3 Interconnect and Communication

For the core, a data transport network approach is proposed to exploit the parallelism between the abundant resources and to simplify the control logic in the CCU. As shown in the Fig.4, each cluster is deployed with seven 32-bits data paths. The connections between the units and the data transport network go through the input and the output sockets. By the way, a functional unit can connect several registers to one socket, but any socket can transfer only one data item each cycle.

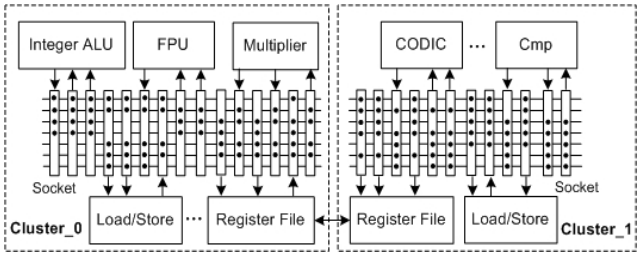

Fig.4. Optimized interconnect in the core

Note that the data transport network in the clusters is not necessarily fully connected. Although the fully connected network holds the maximal capacity of the data transfer, it complicates the CCU design and widens the microinstruction, which degrades the system performance on the reverse. Thus, the proposed scheme is to reduce the redundant connections in terms of the observation on some multimedia processing applications, but keeping a high connectivity. Then, assuring the sufficient parallelism in the targeted applications, the connectivity in the DPC core is optimized to shorten the microinstruction. In addition, one common occurrence is that some transfers are so frequent that they exclusively occupy the data path for a long time and lead to the interconnect conflicts. In such case, the network also supplies 8 bypasses controlled by the compiler to attain feasible schedule. However, only two bypasses are allowed every cycle.

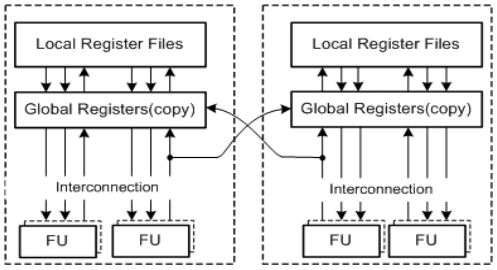

Fig.5. Shared register address for communication

Then, the DPC uses shared register address for communication, and the registers corresponding to the shared address can be both read and written in the two clusters. As shown in Figure.5, the function units read the shared registers from the local copy registers via the connection network, whereas the writes to the shared registers are lead to another cluster. Besides the 4 global registers, each cluster also deploys 176 local registers, consisting of 4 register sliding windows. Moreover, the two implementation issues here were to complete the communication behavior in only one cycle and solving the problem of the critical path on the one hand and on the other hand to restrict the number of writes to the shared register per cycle to ease the pressure of write ports on the global registers.

3.4 Stream Memory System

The stream memory system in the core comprises a 4KB scratch-pad ram, an 8KB normal data cache supporting the write-back scheme and two specific stream caches as well. The stream memory architecture and its non-uniform address space are illustrated in the Fig.6. The separated memory address segments are mapped into different memories, and any access request from the clusters is routed to one of the memories based on the mapped address. As we known, the usual data access patterns in the multimedia applications involve a lot of vectors, matrixes and scalar elements. In general, the frequent cross-interferences between them are inevitable no matter in the normal direct-mapped or set-associative caches. However, the relative problem can be solved elegantly if we include a scratch-pad ram in the architecture. It can be used to store some scalar variables or array variables with high frequency, i.e. the loop index or the small array frequently accessed. In addition, the scratch-pad ram can also supply the space for the state protection. As a result, it not only reduces the access to the off-chip memory but also eases the pressure of the register allocation in the compiler [5].

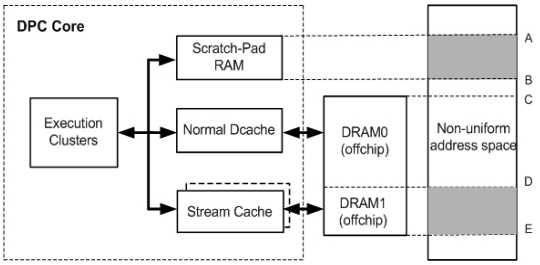

Fig.6. Stream memory architecture and its non-uniform space

Considering the regularity of the access pattern, DPC is also deployed with two stream caches to suit for the characteristic of the stream data in the multimedia applications. Each stream cache module is parameterized by 2 banks with 32B line size, which are determined by the memory access pattern analysis. Moreover, each bank supports its own prefetch which can be configured through its stride register inside the stream controller. If a memory access occurs to a stream cache, the stream cache controller issues a prefetch request of the next address that is the sum of the current address plus the stride value. Thus, one of the advantages of the stream caches is that it supports the flexible prefetch towards to the regular patterns.

```
If hit in bank0 or bank1 {
        Return data and prefetch next;}
else {
        if (previous == 1) {
        fill the bank0 and prefetch bank0;
        previous = 0; }
        else {
              fill the bank1 and prefetch bank1;
              previous = 1; }
         }
```

Fig.7. Scheme of the stream cache

As illustrated in Fig.7, the stream cache adopts a reasonable scheme to arrange the two banks to support the n-way cross access patterns. Where, the flag *previous* indicates which bank is used previously. Followed the scheme above, our stream caches can

support for the 4-way array access at the same time. Therefore, with the combination of the cross-conflict reduction and the configurable prefetch scheme, the stream memory system can improve the processing performance significantly.

4. IMPLEMENTATION

The first implementation has been fabricated in 0.18μm six-metal CMOS with a clock frequency of 250 MHz. The DPC is integrated into a LEON-based SoC with a complete set of peripherals for the multimedia applications. Figure.8 shows a die photo of the SoC, which results in the area of 4.6×4.6mm^2. Note that LEON is nearly 1.64mm^2, while the area of the DPC is 6.62mm^2.

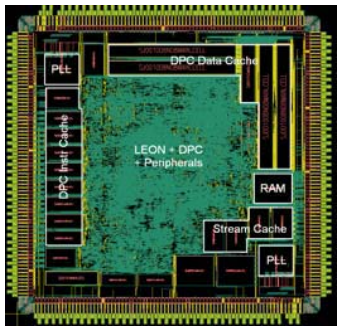

Fig.8. die photo of the SoC

5. EXPERIMENT AND PERFORMANCE

Each instruction of the core allows 14 data transfers and 2 bypasses to exploit the sub-word parallelism. Supposing the clock frequency of 250MHz, the core can provide a peak performance of 14GOPS on 8-bit operations or 7GOPS on 16-bit ones.

The benchmarks listed in Table.1 are selected from application areas ranging from image processing, audio, and signal processing. Some of the benchmarks are kernels, and others are applications for which only a portion can be executed in the DPC. Table.1 presents the performance numbers, assuming a 1-cycle and a 14-2-cycle memory access timing (1 cycle for ram, 14 cycles for the first 64-bits access to the SDRAMs, but 2 cycles thereafter). Thus, the results highlight the ability of the processor to achieve the significant speedup with the realistic memory systems.

Table.1. Performance comparisons on benchmarks（cycles）

Algorithm	LEON	LEON + DPC	Speedup
IDCT	10,992	1,104	10.0
FFT	30,130	2,204	13.7
FIR	127,659	5,857	21.8
JPEG encode	16.8×10^6	2.33×10^6	7.2
MPEG4 encode	13.2×10^6	3.23×10^6	4.1

IDCT: IEEE 118-1990 compliant, 8×8 block of 16-bits pixels
FFT: A 64-points complex FFT, 16b/input, normally ordered
FIR: 128-sample, 32taps, 16b/input complex FIR
JPEG encode: QVGA 320 x 240 Frame size
MPEG4 encode: QCIF 176x144 Frame size, on average

Our benchmark effort consists of running code on LEON processor by itself, profiling the code and then programming or compiling the kernels running on the DPC. Especially, with the utility of the scratch-pad ram and the stream cache, the coprocessor decouples from the off-chip memory to a certain extent, which results to an overhead reduction of the processor. As shown in Table.1, compared to LEON alone, LEON with our DPC coprocessor achieves a speedup in the range of approximately 10-22 to kernels and 4-7 to applications. In most cases, the speedup is ultimately limited by the fractions that are data parallelism in nature and can be accelerated on the coprocessor.

In addition, using the Spice simulations of the 0.18μm technology, the implementation (Fig.9) is evaluated as follows. Fig.9 shows the synergistic effect of introducing the gated clock in the 250MHz frequency. This graph shows that on average this saves 16.3% power when compared to the 0.49 Watts dissipation without clock gating. Then, it can be also concluded typical power consumption of the processor is nearly 0.41 Watts.

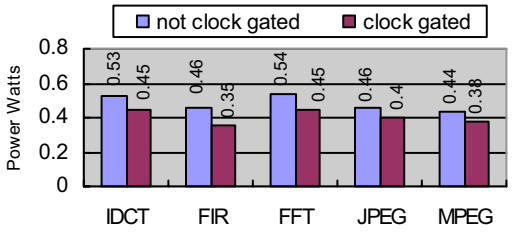

Fig.9. Impact of the gated clock

6. SUMMARY

This paper advanced a novel data-parallel coprocessor for multimedia processing. The DPC core not only provides an abundance of parallel computational resources, but also introduces the gated clock to reduce the power dissipation. The core is realized in the 0.18μm technology and operates at 250MHz. Its powerful processing ability comprising 14 data paths and 2 bypasses in parallel as well as the sub-word parallelism can achieve a satisfactory performance. In the future, it is possible to enlarge the level of parallelism from 2 to 8 clusters if needed. Also, a second-generation implantation in 90nm is underway in order to further improve the performance.

ACKNOWLEDGMENT

The authors would like to thank Andrea Cilio and his colleagues in Delft technology university of Netherlands for their great help.

REFERENCES

[1]. H. Corporaal. Microprocessor Architectures, From VLIW to TTA. John Wiley, 1998.

[2]. H. Sbeyti, S. Niar, L. Eeckhout, "Adaptive Prefetching for Multimedia Applications in Embedded Systems." DATE'04, EDA IEEE, Paris, France, 16-18, February 2004.

[3]. Y. Wu and EY Chang, Optimal Multimodal Fusion for Multimedia Data Analysis, ACM International Conf. on Multimedia (MM), pp.572-579, New York, October 2004.

[4]. Ujval J. Kapasi, William J. Dally and Scott Rixner, "Efficient Conditional Operations for Data-parallel Architectures," Proc. Intl. Symp. on. Microarchitecture, pp. 159-170, Dec. 2000.

[5]. Lian Li, LinGao and Jingling Xue, Memory coloring: a compiler approach for automatic scratchpad memory management. PACT'05, page 329--338, Saint Louis, 2005.