PERFORMANCE EVALUATION OF RING-STRUCTURE REGISTER FILE IN MULTIMEDIA APPLICATIONS*

Tay-Jyi Lin, Chin-Chi Chang, Tsung-Hsun Yang, Yu-Ming Chang, Chien-Hung Lin, Chen-Chia Lee, Hung-Yueh Lin, and *Chein-Wei Jen*

Department of Electronics Engineering
National Chiao Tung University, Taiwan

ABSTRACT

As the concurrent functional units in media processors increase continuously to meet the performance needs, the required access (i.e. read or write) ports of the centralized register file (RF) multiply rapidly and cannot be efficiently implemented. We propose a novel ring-structure RF, which is composed of register sub-blocks identical to the RF for a single functional unit. Data exchanges among functional units occur on the switch network of the ring registers. The proposed RF has been integrated into a four-way VLIW DSP processor successfully that demonstrates its effectiveness in DSP kernels. The synthesis result shows that our proposed ring-structure RF saves 91.88% silicon area of the centralized one, while reducing its access time by 77.35%.

1. INTRODUCTION

Multimedia processing in audio, image, and video applications demands extremely high computing power for the real-time constraints. More functional units (FU) are integrated, which operate concurrently to achieve the performance requirements. Programmable solutions are attractive for their less development efforts, the upgradability to support new standards and possibly filed software patches. These factors together effectively reduce the time-to-market, extend the time-in-market, and thus make the greatest profit. High-performance processors with multiple concurrent FU exploit finer granularity of parallelism than multi-processor systems, which can improve the performance of DSP kernels more effectively. The major distinction between multi-processor systems and microprocessors resides in their inter-FU communications. Microprocessors have a direct and efficient data exchange mechanism among FU through a centralized register file (RF).

The centralized RF provides storage for and interconnects to each FU in a general manner and each FU can read from or write to any register location. For N concurrent FU, the silicon area of the centralized RF grows as N^3, the delay as $N^{3/2}$, and the power dissipation as N^3 [1]. Thus, RF will soon dominate the area, the delay, and the power dissipation as the number of FU increases. Restricting the communication between FU and registers, so that each FU can only read and write a limited subset of registers, can significantly reduce the RF complexity with small performance penalty. In this paper, we propose a novel ring-structure RF, which partitions the centralized RF into $2N$ sub-blocks. Each FU is able to access two sub-blocks simultaneously, one of which is private (i.e. dedicated to the FU) and the other is dynamically mapped to facilitate inter-FU communications. Therefore, each register sub-block requires the access ports for a single FU only. The shared sub-blocks are organized in a ring and globally controlled to reduce the overheads. The synthesis results of the proposed ring-structure RF saves 91.88% silicon area of the centralized one, while reducing its access time by 77.35%. The proposed ring-structure RF has been integrated into a four-way VLIW DSP processor to evaluate the performance degradation due to the access restrictions, where the port mapping is under the direct control of programmers. The instruction-set simulation shows that the performance of our proposed DSP processor is comparable with the state-of-the-art high-performance DSP processors.

The rest of this paper is organized as follows. Section 2 summarizes the features of the four-way VLIW DSP processor and discusses the ring-structure RF. Section 3 describes the programming model with explicit port mapping and compares its performance with commercially available high-performance DSP processors. The silicon implementations of the proposed ring-structure RF and the VLIW DSP processor are available in Section 4. Finally, Section 5 concludes this paper and outlines our future works.

2. VLIW DSP PROCESSOR WITH PROPOSED RING-STRUCTURE REGISTER FILE

Instruction parallelism is exploited to improve the speed of the high-performance microprocessors. Compared to the dynamic hardware scheduling of superscalar processors, VLIW machines have low-cost compiler scheduling with deterministic execution time and thus become the trends of high-performance DSP processors. In this section, we will introduce a novel four-way VLIW DSP processor of two load/store and two ALU/MAC units with SIMD capability. We propose the 2-tier instruction processing, which manipulates data and control separately and effectively smoothes the instruction flow to the DSP datapath. With the proposed ring-structure RF for efficient data exchange among FU, the proposed DSP processor can easily achieve its

* This work was supported by the National Science Council, Taiwan under Grant NSC91-2218-E009-011

peak performance of 4 16-bit data operations/cycle, or 15 RISC-type operations/cycle (including 4 effective data manipulations, 4 data generations, 4 address updates, and 3 branch controls).

2.1. 2-tier Instruction Processing

Fig 1 depicts our proposed 2-tier instruction processing with separate control and data manipulations. The instruction dispatcher handles unconditional branches (e.g. jumps and traps) and zero-overhead looping transparently to the datapath, which can thus receive an instruction word (including four instructions) per cycle, regardless of the control flow. Conditional branches (data-dependent control) are resolved in cooperation with the control/LS FU (instruction field 0) in the datapath.

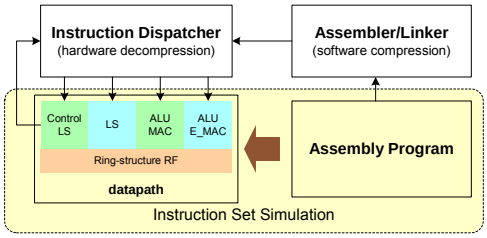

Figure 1 2-tier Instruction Processing

VLIW processors are notorious for their poor code density, for the unused instruction slots must be filled by NOP. The situation gets worse when the parallelism is limited. ***Variable-length VLIW*** [2] eliminates NOP with alternative FU codes for run-time instruction dispatch and decoding, other than the fixed position codes in conventional VLIW processors (i.e. each FU has a corresponding field in the long instruction word). The instruction fetch unit needs queues to regulate the varying rate of instruction consumption, which also work as the instruction buffer for zero-overhead looping. ***Indirect VLIW*** [3] uses short instructions to address the internal micro-instruction memory (i.e. the programmable VIM) for the long instruction words. Existing instructions (inside an instruction word) can be reused to synthesize new instruction words, and thus reduce the instruction bandwidth. ***Differential VLIW encoding*** [4] eliminates repeated instructions in neighboring codes, which is very efficient in software pipelining [5] and our explicit programming model for dynamic port mapping, which will be described in Section 3. In addition to general assembling/linking, our code generation tool is also responsible for the instruction compression in the 2-tier instruction processing. The instruction dispatcher decompresses the instruction word dynamically. By the way, it is worthy to note that the symbolic instruction-set simulation is independent from the instruction encoding and compression schemes in the processor implementation.

To simplify the target address calculation for control flow, the compressed and variable-length instruction word is re-coded for a fixed-length header and the remnant code. These two parts are then stuffed into the head and the tail of a large fixed-length instruction packet respectively [6]. The linker assigns each label of the user's assembly program the packet address with the offset of the target instruction word, instead of the target address of the instruction word directly.

2.2. Ring-Structure Register File

Imagine the four concurrent FU in our VLIW DSP processor as individual RISC, each of which has a 16-element RF. Each RF is partitioned into a private and a shared sub-block, each of which has 8 registers. The latter is used for data exchanges among FU. Every FU can access two sub-blocks with total 16 registers simultaneously. The four shared sub-blocks are concatenated as a ring with a 2-bit offset to reduce the context for dynamic port mapping.

The shaded region in Fig 2 depicts the proposed ring-structure RF. Each sub-block has 4 access ports (i.e. 2 reads and 2 writes respectively). All ring registers (i.e. the shared sub-blocks) are identical and each has 8 (R8~R15) 32-bit elements. The local registers (i.e. the private sub-blocks) of the control/LS FU have 8 32-bit elements for general-purpose uses and memory addresses, while those of ALU/MAC have 8 40-bit accumulators. A 2-bit ring offset is attached to every instruction word without state. The instruction dispatcher is responsible for the dynamic port mapping, which is transparent to the FU.

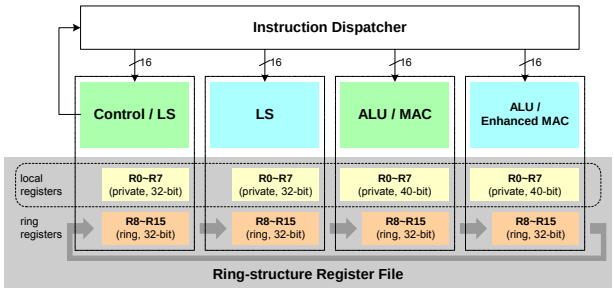

Figure 2 Proposed VLIW DSP Datapath

2.3. SIMD Capability

Besides two parallel 16-bit ALU operations, the ALU/MAC can simultaneously perform two 16-bit MAC operations with 40-bit accumulators with the instruction

`MAC_V` r_i, r_m, r_n

which performs $r_i \leftarrow r_i + r_m.Hi \times r_n.Hi$, and $r_{i+1} \leftarrow r_{i+1} + r_m.Lo \times r_n.Lo$. This instruction has four concurrent accesses to the RF (2 reads and 2 writes respectively). The index i must be even with i+1 implicitly specified.

Our DSP also supports very powerful double load (store) instructions of the form

`LW_D` $r_m, r_n, (r_i)+j$

which performs $r_m \leftarrow Mem[r_i]$, $r_n \leftarrow Mem[r_{i+1}]$ (memory accesses), $r_i \leftarrow r_i + j$, and $r_{i+1} \leftarrow r_{i+1} + j$ (address updates) in parallel. This instruction requires 6 RF accesses simultaneously (including 2 reads and 4 writes). Fortunately, the four write accesses do not conflict because r_i and r_{i+1} are local registers to store memory addresses while r_m and r_n are ring registers to deliver data to ALU/MAC. They locate in independent sub-blocks.

The enhanced MAC in our DSP supports a single-cycle 16-bit complex MAC/MUL or a single-cycle 32-bit MAC/MUL. These instructions exhaust all multiplication resources (i.e. our DSP has total 4 16-bit multipliers) and prevent the other ALU/MAC unit from any operation involving multiplication.

```
1   0;  MOV r0,COEF;          MOV r0,COEF;          MOV r0,0;          MOV r0,0;
2   0;  MOV r1,X;             MOV r1,X+1;           NOP;               NOP;
3   0;  MOV r2,Y;             MOV r2,Y+2;           NOP;               NOP;
4       RPT 512,8;
5   0;  LW_D r8,r9,(r0)+2;    LW_D r8,r9,(r0)+2;    MOV r1,0;          MOV r1,0;
6       RPT 15,2;
7   2;  LW_D r8,r9,(r0)+2;    LW_D r8,r9,(r0)+2;    MAC_V r0,r8,r9;    MAC_V r0,r8,r9;
8   0;  LW_D r8,r9,(r0)+2;    LW_D r8,r9,(r0)+2;    MAC_V r0,r8,r9;    MAC_V r0,r8,r9;
9   2;  LW_D r8,r9,(r0)+2;    LW_D r8,r9,(r0)+2;    MAC_V r0,r8,r9;    MAC_V r0,r8,r9;
10  0;  MOV r0,COEF;          MOV r0,COEF;          MAC_V r0,r8,r9;    MAC_V r0,r8,r9;
11  0;  ADDI r1,r1,-60;       ADDI r1,r1,-60;       ADD r8,r0,r1;      ADD r8,r0,r1;
12  2;  SW (r2)+4,r8;         SW (r2)+4,r8;         MOV r0,0;          MOV r0,0;
```

Figure 3 Example: 64-tap FIR Filter

3. PROGRAMMING MODEL & PERFORMANCE EVALUATION

We have constructed the instruction-set simulator for our VLIW DSP processor [7] with the proposed ring-structure RF. The assembly syntax for our simulator starts with the ring offset, followed by the four instructions to form an instruction word –

ring offset; instr 0; instr 1; instr 2; instr 3;

The data memory subsystem uses half-word addressing. Fig 3 is an illustrating example of a 64-tap finite-impulse response (FIR) filter that produces 1,024 outputs. The input and output data are 16-bit fractional and 32-bit fixed-point numbers respectively. `RPT` (the repeat instruction; line 4 and line 6) is carried out in the instruction dispatcher and consumes no execution cycle of the datapath. Note that only two-level loop nesting is allowed in our implementation.

The inner loop (line 7-8) loads four 16-bit inputs and four 16-bit coefficients every cycle into two 32-bit `r8` and two 32-bit `r9` respectively with the two LS units. The four address registers (two `r0` and two `r1`) are updated simultaneously. In the meanwhile, the two ALU/MAC units perform 16-bit SIMD MAC operations of the form

```
MAC_V r0,r8,r9
```

for four taps (i.e. r0←r0+r8.Hi×r9.Hi, and r1←r1+r8.Lo×r9.Lo for each ALU/MAC). After summing up the 32 32-bit products with 40-bit accumulators, `r0` are `r1` are added together and rounded to the 32-bit `r8` in the ring registers. Finally, two 32-bit outputs are stored in the memory subsystem by the two LS units via `r8`. In this 64-tap FIR example, the outer loop (line 5 and line 7-12) produces two filter outputs in 35 cycles. In other words, the proposed DSP processor can compute 3.66 taps every cycle.

In the proposed VLIW DSP processor, conditional branches evaluate the data values (i.e. the conditions) through the register ports from control/LS FU (i.e. instruction field 0), which execute in parallel with the succeeding instruction word. NOP must be inserted if the access ports conflict.

Table 1 summarizes the performance comparisons of state-of-the-art high-performance DSP processors with the proposed VLIW DSP processor with the ring-structure RF. The second row shows the number of cycles required for *N*-sample, *T*-tap FIR filtering. The third row lists the performance of the radix-2 256-point fast Fourier transform (FFT), which is estimated in the number of execution cycles. Maximum numbers of add-select-compare (ACS) operations per cycle are given in the fourth row, which is the kernel of the Viterbi algorithm. The numbers in parentheses show the results that account for the load/store overheads when the depth is 16. TI C'64s DSP has a specific Viterbi coprocessor and is not included for the comparison. The last row compares the performance of the motion estimation algorithm with MAE (mean absolute error) criteria, which is measured in the maximum number of pixels per cycle.

Table 1 Performance Comparison

	TI C'55x [9]	TI C'64x [2]	NEC SPXK5[8]	Intel/ADI MSA [10]	Proposed
FIR	*NT*/2	*NT*/4	*NT*/2	*NT*/2	*NT*/4
FFT	4,768	2,403	2,944	3,176	2,340
Viterbi	1 (0.4)	N.A.	1 (1)	1 (N.A.)	1 (0.84)
ME	N.A.	2	2	4	2

4. SILICON IMPLEMENTATION

We have implemented in Verilog RTL the ring-structure RF and the centralized one with the same number of registers and the required access ports for the proposed VLIW DSP processor. The designs are synthesized using Synopsys with 0.35μm cell library and automatically placed and routed in 1P4M technology using Apollo. Table 2 summarizes the implementation results. Our approach reduces the delay and the area by factors of 4.42 and 87.37 respectively. PowerMill is used to estimate the power dissipation of the ring-structure RF to perform FFT at 100 MHz. We do not have the power estimation for the centralized RF because of the limited tool capability.

Table 2 Implementation Results

	Centralized RF	Ring-Structure RF
Delay	38.46 ns	8.71 ns
Gate Count	591K	48K
Area	17.76mm×17.76mm	1.9mm×1.9mm (2.34mm×1.53mm)
Power	N.A.	356mW @3.3V 100MHz

Fig 4 shows the trial implementation of our 4-way VLIW DSP datapath with the ring-structure RF. The core size of this single-cycle implementation is 2.34mm×2.34mm. Its maximum operating frequency is 20MHz, which is going to be pipelined at least into three stages.

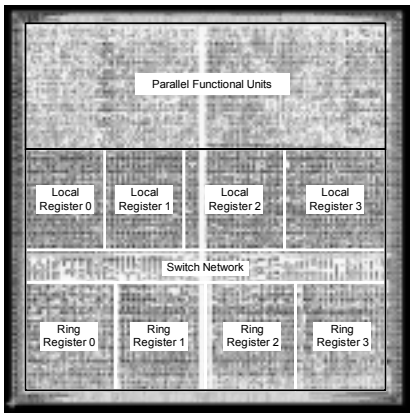

Figure 4 Layout of the DSP Datapath

5. CONCLUSIONS

As the number of concurrent FU increases, the complexity of the centralized RF grows exponentially. For N concurrent FU, the area grows as N^3, the delay as $N^{3/2}$, and the power dissipation as N^3. We propose a novel ring-structure RF, which consists of an N-by-N switch network and $2N$ register sub-blocks with access ports only for a single FU. The synthesis result shows that our approach saves 91.88% gates of the centralized one, while reducing the access time by 77.35%. The ring-structure RF has been successfully integrated into a 4-way VLIW DSP processor with explicit port mapping to demonstrate its effectiveness in various DSP kernels.

We are going on the full-custom designs of the 2R/2W register sub-blocks and the 4-by-4 switch network to improve the area and the power consumption of our first chip realization. Clock gating will also be investigated to further reduce the power dissipation. Besides, more DSP kernels and benchmark programs will be developed to verify the effectiveness of our proposed VLIW DSP processor with the ring-structure RF.

REFERENCES

[1] S. Rixner, W. J. Dally, B. Khailany, P. Mattson, U. J. Kapasi, and J. D. Owens, "Register Organization for Media Processing," *International Symposium on High Performance Computer Architecture (HPCA)*, pp.375-386, 2000

[2] *TMS320C64x DSP Library Programmer's Reference*, Texas Instruments Inc., Apr 2002

[3] G. G. Pechanek and S. Vassiliadis, "The ManArray Embedded Processor Architecture," *Euromicro Conference*, vol.1, pp.348-355, September, 2000

[4] G. Fettweis, M. Bolle, J. Kneip, and M. Weiss, "OnDSP: A New Architecture for Wireless LAN Applications, *Embedded Processor Forum*, May 2002

[5] J. L Hennessy, and D. A. Patterson, *Computer Architecture – A Quantitative Approach*, 3rd Edition, Morgan Kaufmann, 2002

[6] H. Pan and K. Asanovic, "Heads and Tails: A Variable-Length Instruction Format Supporting Parallel Fetch and Decode," *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES 2001)*, Nov 2001

[7] *DSP Architecture v.2 Instruction Set Manual* [Online], available: http://twins.ee.nctu.edu.tw/dspv2

[8] T. Kumura, M. Ikekawa, M. Yoshida, and I. Kuroda, "VLIW DSP for Mobile Applications," *IEEE Signal Processing Magazine*, pp.10-21, July 2002

[9] *TMS320C55x DSP Programmer's Guide*, Texas Instruments Inc., July 2000

[10] R. K. Kolagotla, et al, "A 333-MHz Dual-MAC DSP Architecture for Next-Generation Wireless Applications," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, vol. 2, pp.1013-1016, May 2001

[11] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, *DSP Processor Fundamentals – Architectures and Features*, IEEE Press, 1996

APPENDIX

Instructions of the VLIW DSP with Ring-Structured RF [7]

Mnemonic	Action
	INSTRUCTION DISPATCHER
RPT	Repeat m instructions n times; zero-overhead looping
J	Jump; unconditional branch
TRAP	Trap; system call
	COMMON TO ALL FIELDS
NOP	No operation
ADDI	Add immediate
MOV	Move; pseudo instruction
	SPECIFIC TO FIELD 0
BNEZ	Branch on not equal zero
JAL	Jump and link
JR	Jump register
	COMMON TO FIELD 0/1
LH	Load half word
LW	Load word
LH_D	Double load halfword*
LW_D	Double load word
LH_V	Load halfword vector (SIMD)*
SH	Store halfword
SW	Store word
SH_D	Double store halfword
SW_D	Double store word
SH_V	Store halfword vector (SIMD)
	SPECIFIC TO FIELD 3
CMUL	16-bit complex multiply
CMAC	16-bit complex multiply & accumulate
MUL32	32-bit multiply
MAC32	32-bit multiply & accumulate
	COMMON TO FIELD 2/3
MUL	16-bit multiply
MAC	16-bit multiply & accumulate
ADD	Add
SUB	Subtract
AND	AND
OR	OR
XOR	Exclusive OR
SLL	Shift left logical
SRL	Shift right logical
SRA	Shift right arithmetic
BF2	Radix-2 butterfly
MUL_V	Two 16-bit multiply (SIMD)*
MAC_V	Two 16-bit multiply & accumulate (SIMD)*
ADD_V	Two 16-bit add (SIMD)
SUB_V	Two 16-bit subtract (SIMD)
MIN_V	Compare & store the small one (subword)
MAX_V	Compare & store the large one (subword)
ABS_V	Two absolute value (SIMD)
PACK	Merge low 16-bit of two registers

* See Section 2.3 for instruction syntax