Architecture of SIMD Type Vector Processor

Mohammad Suaib
National Institute of Technology Hamirpur, India

Abel Palaty
National Institute of Technology Hamirpur, India

Kumar Sambhav Pandey
National Institute of Technology Hamirpur, India

ABSTRACT
Throughput and performance are the major constraints in designing system level models. As vector processor used deeply pipelined functional unit, the operation on elements of vector was performed concurrently. It means the elements were processed one by one. Improvement can be made in vector processing by incorporating parallelism in execution of these concurrent operations so that these operations can be performed simultaneously. This paper presents a design and implementation of SIMD-Vector processor that implements this parallelism on short vectors having 4 words. The operation on these words is performed simultaneously i.e. the operation on these words is performed in one cycle. This reduces the clock cycles per instruction (CPI). To implement parallelism in vector processing requires parallel issue and execution of vector instructions. Vector processor operates on a vector and superscalar processor issues multiple instructions at a time. This means parallel pipelines are implemented and then made these to support vector data. SIMD-Vector processor will operate on short vector say 4 words vector in a superscalar fashion i.e. 4 words will be fetched at a time and then executed in parallel. This requires redundant functional units e.g. if addition is to be performed on two vectors multiple adders are needed. We have designed the architecture of SIMD type Vector processor. All the designing parameters are explained.

Keywords
SIMD type Vector processor, vertical and horizontal parallelism, ILP.

1. INTRODUCTION
Parallel processing is the need of today's architectures. Parallel processing reduces the execution time taken by any program. The execution time taken by any program is determined by three factors: First, the number of instructions executed. Second, number of clock cycles needed to execute each instruction and the third is the length of each clock cycle. Here we shall try to reduce the number of clock cycles by introducing a new processor named SIMD type of vector processor. Superscalar and VLIW architectures improve the performance by reducing the Cycles Per Instruction (CPI). This architecture take the advantages of superscalar processor as well as vector processor. SIMD-Vector architecture supports In-order issue with out-of-order completion. All the vector instructions are issued in-order and kept in Instruction cache. After checking the structural and data hazard all the vector instructions are executed in out-of-order sequence. Reorder buffer is used to write the output in-order. Hence we get the correct output sequence.

Technology is changing rapidly and significantly in past few years. For microprocessor technologies multimedia applications are the main stream computing. In this scenario we can improve the performance of the processor by exploiting data level parallelism (DLP) and instruction level parallelism (ILP). To exploit DLP, instructions are executed in single instruction multiple data (SIMD) fashion. We adopt the SIMD processors into general purpose processors [2]. Multimedia processors has a lot of inherent parallelism so it can be easily exploited by SIMD instructions at low cost and energy overhead.

Here we can see a lot of superior theoretic performance. But practically it is not possible due to some limitations. If we add more processing unit into our SIMD-Vector architecture then it sufficiently increase the hardware cost as well as complexity of the processor. So as a result we worked on short vector. SIMD-Vector architecture supports the instructions of vector length 4. In this architecture we assume that all the instructions are vector and should be of the length of four. This architecture has 4 execution units. All the four vector elements are processed on four different processing units. This execution is performed parallel in one clock cycle. Hence we can reduce the clock cycles to perform multimedia applications. To reduce the complexity of the system chaining is not used to improve the performance of vector processing. If some instructions have the vector length less than four then available vector elements are sent to execution engines and remaining are circuited to ground. Short vector implementation introduces large parallelization overhead such as loop handling and address generation [1]. There are many examples of SIMD processors such as IBM's VMX, AMD's 3D Now!, Intel's SSE and Motorola's Altivec. In these processors we can embed vector processing with taking the advantage of 4 way superscalar processor.

The SIMD-Vector architecture brings new levels of performance and energy efficiency. Organization of paper is as follows. In section 2 the motivations of this work is introduced. Section 3 describes the SIMD-Vector architecture. SIMD-Vector is compared with other conventional vector architecture in section 4. Then the evaluation result is shown in section 5. Section 6 describes the conclusion of whole work. Finally section 7 gives the future work.

2. MOTIVATION
A vector ISA packages multiple homogeneous, independent operations into a single short instruction which results into a compact code. The code is compact because a single short vector instruction can describe N operation. This reduces instruction bandwidth requirements.

Reduction in instruction bandwidth: A single vector instruction comprises of N operations thereby reducing the instruction bandwidth. In the proposed scheme throughput and performance can be enhanced by introducing parallelism. It can be done by incorporating superscalar issue in vector processing.

Hardware reduction: In vector instruction N operations are homogeneous. This saves hardware in the decode and issue stage. The opcode is decoded once and all N operations can be issued as a group to the same functional unit. In our proposed scheme, this is taken as the basic design constraint.

SIMD extensions and vector architecture are quite similar. The principle difference is that how the instructions control is implemented and communication between execution unit and memory unit. With the help of pipelining technology vector processor can overlap computation, load, store operations on vector elements. So vector length may be long and variable. This kind of parallelism is called vertical parallelism. Instruction latency is bigger than one cycle per vector element. While SIMD extension duplicates the execution units to perform the parallel execution. This type of parallelization is called horizontal parallelism. Due to limitation of hardware cost we cannot add much execution units so the vector length should be fixed and short.

```
for (int a=0;a<64;a++)
    {
      z[a]=x[a]+y[a]:
     }
```

(a) Scalar form

```
for (int a=0;a<64;a+=4)
    {
     z[a+3:a]=x[a+3:a]+y[a+3:a]
     }
```

(b) SIMD-Vector form

For above given example there are 64 iterations in scalar architecture. Scalar architecture takes one clock cycle instruction latency. While using SIMD-Vector architecture four vector instructions can be executed in one clock cycle simultaneously. So instruction latency is just greater than 16.

3. SIMD TYPE VECTOR PROCESSOR

In this section we describe the architecture of SIMD-Vector processor, pipelining and working of proposed architecture.

3.1 Proposed Architecture

In proposed architecture SIMD unit is the functional unit to perform the vector operations. It is similar as conventional SIMD unit. Architectural overview of proposed scheme is given in Figure 1.

For a given set of vector operations each time SIMD unit executes one vector instruction at a time concurrently as vector instruction has four vector element only. To handle the long vector operations we need the smart compiler for vectorizing the instructions. All the vectorized instructions should be of length 4. We add a additional unit called vector code cache (VCC) to handle the long vector operations. We restrict the size of VCCache to 1 KB that can store 256 operations of 32 bit instruction encoding that is enough for most of the multimedia applications. Loop controller generates the loop control signal to complete long vector operations with keeping in mind that 4 operation can be done in one clock cycle. It is very tedious to provide the memory location to all the vector element using conventional memory system. To support the strided memory location to vector elements we need an address generator unit [3]. This address generator unit is connected to vector register file and memory via load-store unit. And all remaining units are as conventional with standard meaning. Figure 2 shows the SIMD unit having 4 execution units that can execute 4 operations in parallel in one clock cycle.

Table 1. Architectural parameter

Parameter	Explanation	Bit Size
B_S	Bit size of SIMD unit	128
B_{VRF}	Bit size of vector register file	128
B_{LS}	Bit size of load store unit	128
B_{VE}	Bit size of vector element	32
L_V	Vector length	4

We have described some parameters for SIMD type Vector processor that are listed in table 1. Our vector register should support 4 vector element of 32 bit each. So length of vector register file (VRF) would be 128. Generally we take the SIMD unit of 128 bit length. Memory unit that is load-store unit would also be 128 bit long. These type of architecture is well supported by IBM's Altivec ISA [4] and Intel's SSE ISA. We are taking 32 bit long vector element. Our proposed architecture would support the instructions of vector length 4.

3.2 Pipelining In SIMD Type Vector processor

In Figure 3 it is shown that how pipeline technology is exploited in SIMD-Vector architecture. At x axis clock cycle is plotted and y axis vector instructions (VI) are shown. Five stage pipelines are shown in Figure 3. By seeing pipeline structure it is easily understood there are four functional unit that can be operated simultaneously on 4 vector element in one clock cycle.

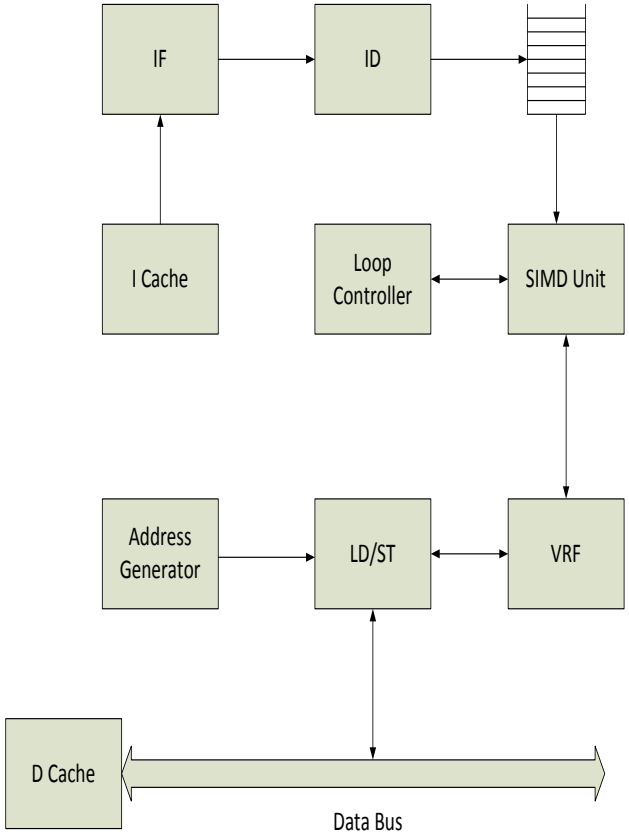

Fig 1: Proposed Architecture of SIMD type Vector Processor

3.3 Working Of SIMD Type Vector Processor

In SIMD-Vector, superscalar implementation is converted to support vector data instead of scalar data. To implement parallel operations on vector redundant functional units are needed. SIMD-Vector behavior is shown in figure 4.

4. COMPARISON WITH OTHER ARCHITECTURE

In this section we have compared SIMD-Vector architecture with SIMD extensions and vector architecture. Proposed architecture take the advantages of SIMD as well as vector processors. The width of SIMD-Vector VRF file is much smaller than vector architecture implemented in recent single chip processors [5,6].

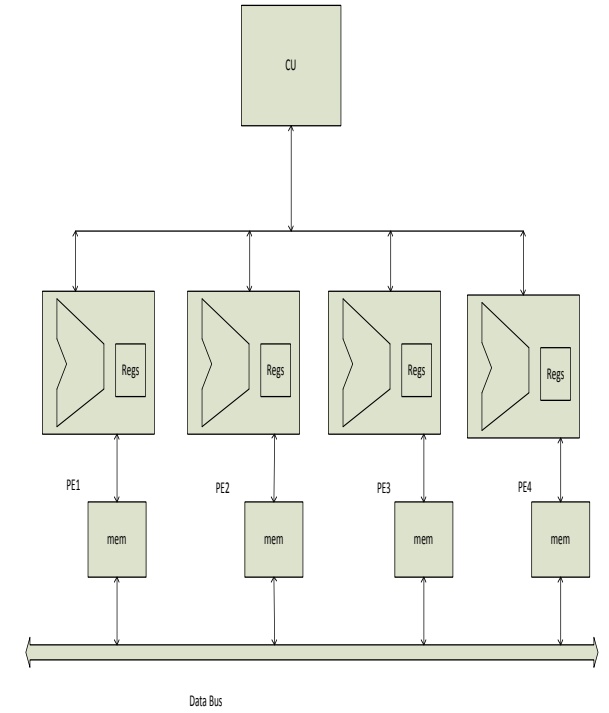

Fig 2: SIMD unit

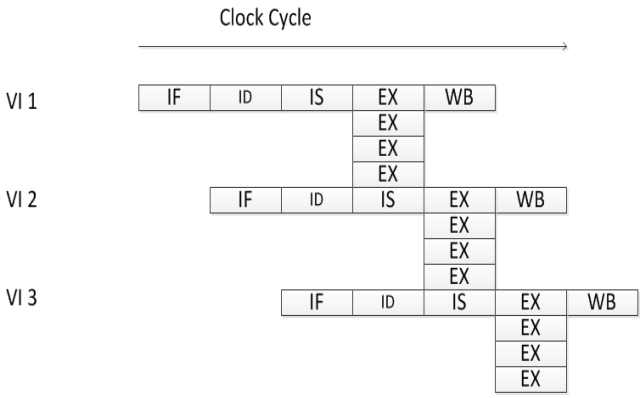

Fig 3: Pipelining in SIMD type Vector Processor

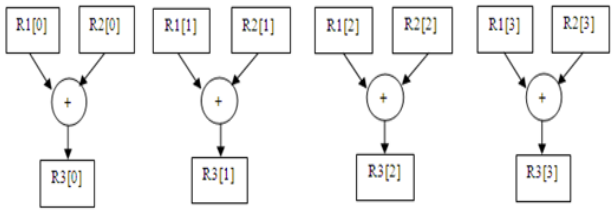

Fig 4: Working of SIMD-Vector processor

Table 2. Architecture Comparison

Feature	SIMD-Vector	SIMD	Vector
Vector Length	4	32	>=64
Memory access	Automatic address generation	Sequential access	Strided access
Instruction latency	1 cycle per vector element	1 cycle per instruction	1 cycle per element
Parallelism	combined	Vertical	Horizontal

5. EVALUATION

By using proposed SIMD-Vector architecture we can enhance the performance of the system. We have analyzed instruction counts on many multimedia operations like fast fourier transform, matrix multiplication, finite impulse response filter infinite impulse response filter using scalar, SIMD and SIMD-Vector architecture. Response of the analysis is shown n the figure 5. This figure completely shows that when we use SIMD-Vector architecture number of instructions are fairly less.

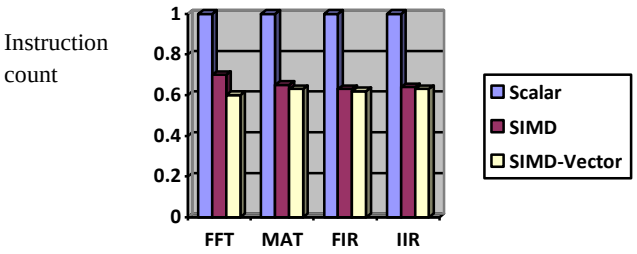

Fig 5: Comparison of instruction counts

6. CONCLUSION

SIMD-Vector processor implements parallelisms on shorts vector having four words. The operation on these words is performed simultaneously i.e. the operation on these words is performed in one cycle. This reduces the clock cycles per instruction (CPI). The parallelism in vector processing requires superscalar issue of vector instructions. Above paper gives the architecture of proposed processor that can be exploited in many multimedia applications.

7. FUTURE WORK

In the future, the parallelism in operation can be enhanced to support longer vectors having more words. This leads to an increase in the hardware as more parallelism requires more functional units.

8. REFERENCES

[1] Shin, J., Hall, M.W., Chame, J.: Superword-Level Parallelism in the Presence of Control Flow. In: CGO 2005, pp. 165–175 (2005).

[2] Lee, R.: Multimedia Extensions for General-purpose Processors. In: SIPS 1997, pp. 9–23 (1997).

[3] Talla, D.: Architectural techniques to accelerate multimedia applications on general-purpose processors, Ph.D. Thesis, The University of Texas at Austin (2001).

[4] Diefendorff, K., et al.: Altivec Extension to PowerPC Accelerates Media Processing. IEEE Micro 2000 20(2), 85–95 (2000).

[5] Corbal, J., Espasa, R., Valero, M.: Exploiting a New Level of DLP in Multimedia Applications. In: MICRO 1999 (1999).

[6] Kozyrakis, C.E., Patterson, D.A.: Scalable Vector Processors for Embedded Systems. IEEE Micro 23(6), 36–45 (2003).

[7] K. Yeager, "The MIPS R10000 Superscalar Microprocessor", in Proceedings of IEEE Micro, Vol. 16, No. 2, pp. 28-41, April 1996.

[8] James E. Smith, Gurindar S. Sohi, "The Microarchitecture of Superscalar Processors", in Proceedings of IEEE, Vol. 83, No. 12, pp. 1609-1624, December 1995).

[9] Open SystemC Initiative (OSCI), www.systemc.org.