A Micro-architecture Simulator for Multimedia Stream Processor

Fang-Ju Lin and Herming Chiueh
Department of Communication Engineering,
College of Electrical and Computer Engineering,
National Chiao Tung University, Hsin-Chu 30050, Taiwan
maggie.cm92g@nctu.edu.tw, chiueh@mail.nctu.edu.tw

***Abstract*— Recent research has proposed using stream processors for media applications. Since a programmable Stream Processor could utilize various hardware micro-architectures for diverse media applications, decision on a suitable micro-architectures to achieve efficiencies and hardware cost is critical. In this paper, a micro-architecture simulator for stream processor is implemented. The simulator evaluate the performance of media application executed on various micro-architectures, and then to analyze the utility rate of hardware and consumption of memory. By comparing the performance of media application executed on diverse micro-architectures, the optimized hardware micro-architecture can be determined for specific application and suitable micro-architecture of Stream Processor can be implemented on later VLSI for different targeting systems.**

I. INTRODUCTION

Media applications are characterized by large available parallelism [1], little data reuse and a high computation of memory access ratio [2]. While these characteristics are poorly matched to conventional micro processor micro-architectures, recent research has proposed using streaming micro-architecture by fit modern VLSI technology with lots of ALUs on a single chip with hierarchical communication bandwidth design to provide a leap in media applications. Relative topics of recent research are Image Stream Processor [3], Smart Memories [4], and Processing-In-Memory [5].

In order to achieve computation rates, current media processor often uses special-purpose [6], fixed-function hardware tailored to one specific application. However, special-purpose solutions lack of the flexibility to work effectively on a wide application space. The demand for flexibility in media processing motivates the use of programmable processors [6]. To bridge the gap between inflexible special special-purpose solutions and current programmable micro-architectures, stream processor architecture has been proposed [1] which directly exploit the parallelism and locality exposed by the stream programming model [7] to achieve high performance. Figure 1 illustrates a stream processor block diagram.

Since, various stream micro-architecture with different ALU clusters are suitable for media applications, mapping multimedia applications to adequate stream programming model becomes essential. This paper proposed a "micro-architecture simulator" as a software solution to evaluate the number of hardware needed for dedicated application. The proposed solution simulates performance on different virtual stream micro-architectures and compares the performance between the architectures. By doing so, the best hardware organization, which has to fully optimize the usage of the hardware resources and reach better performance, is obtained.

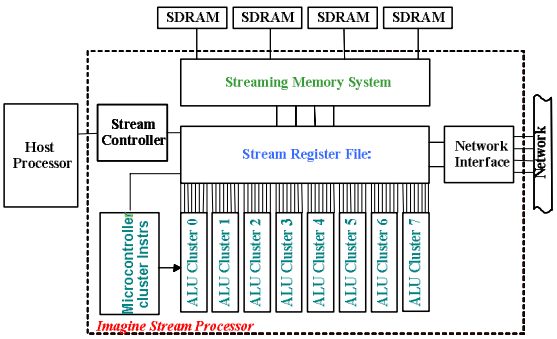

Figure 2 Stream processor block diagram

As shown in Figure 3, for different micro-architecture, application can be mapped into binary stream programming codes according to the ISA and instruction format of different functional units. Stream programming codes can be put into simulator to simulate the operation of which in stream processors, then the simulation result will be generated, which can be used to compare to the simulation result of other organization of micro-architectures and adjust the parameters that will affect the organization of stream micro-architecture till the optimal organization of micro-architecture is discovered. On the other hand, simulation result can also be used to verify the correctness of the stream programming codes. The simulation results will show: (1) Utilization rate of functional units (2) Utilization rate of memory hierarchies. (3) CPU time of specific application.

This research was partially supported by national Science Council, Taiwan (Contract number: NSC95-2219-E-009-018, 95-2220-E-009-002) and Ministry of Education, Taiwan (MoE ATU Program).

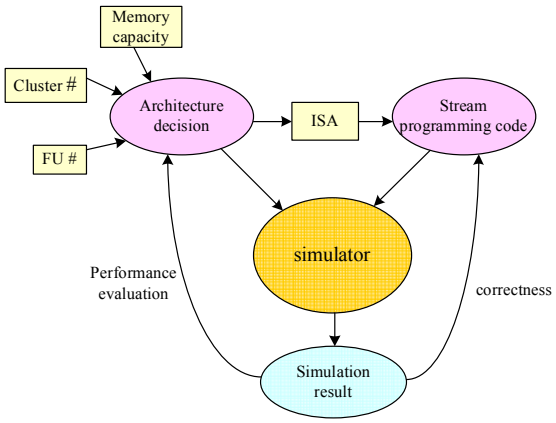

Figure 3 Role of the micro-architecture simulator

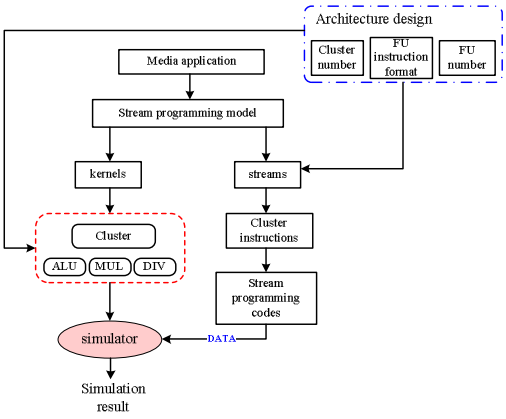

Figure 4 Simulation flow

Up to now, the stream micro-architecture has been widely explored to many media applications and explicitly demonstrated in many references, such as 3-D polygon rendering [8], MPEG-2 encoding [3], stereo depth extraction [9], and fast Fourier transform (FFT). In this paper, the FFT [10] is choose as the benchmark to simulate the performance for specific hardware micro-architecture and evaluate the performance and circuitry complexity tradeoff from different micro-architectures.

The remainder of this paper is organized as following: In Section II, the design methodology is presented. In Section III, experimental results are provided and a comparison to conventional micro-architecture is presented. Finally, future work and conclusions are presented in Section IV.

II. MICRO-ARCHITECTURE SIMULATOR

Micro-simulator is composed of a controller, many arithmetic clusters where the number of the clusters will differ from diverse application, three-tiered memory hierarchy (SRF, SP, and LRF), and instruction memory. It should be noted that the number of the components in the simulator is not fixed. It would depend on the demands of diverse media applications.

Micro-architecture simulator is used to simulate the operation of the stream processor and the performance of the hardware micro-architecture. As shown in the flow of Figure 4, first, a multimedia application and then its hardware micro-architecture, including the number hardware and the memory size have to be chosen. Then, this application is translated into the stream programming model, including kernel and stream, respectively. In addition, the data being processed must be transformed into cluster instructions (stream) and then translate the cluster instructions into binary expression according to the instruction format defined by hardware. Cluster instructions will be copied to instruction memory through File I/O. Controller will fetch instruction from instruction memory and equally distribute these cluster instructions to cluster for executing. Three kinds of registers, SRF, SP, and LRF, are used to save temporary results during computation process.

A. Cluster

Cluster includes two ALU units, two MUL units, one DIV unit, and one 64 32-bit register, which is used to save data exchange between different units in the same cluster. The size of SP register could vary according to the demands of diverse applications. The main function of the component Cluster is to divide the received cluster instruction into multiple instructions that are processable by function units according to the number of functional units.

B. ALU, MUL, DIV

Main spirits for designing ALU, MUL, and DIV are basically the same. The only difference between these three components is the bit of "Opcode" needed for expressing the operation in the instruction format.

- ALU is a two stage pipeline computation unit. The function of ALU unit could be cataloged into three parts: data read, calculate, and write back. Since ALU requires more diverse operations, it demands 4-bits to express.
- MUL is a four-stage pipeline computation unit. The way MUL Unit deal with instruction is the same as that of ALU. The only difference is that MUL Unit just deal with one operation "multiple".
- DIV is a six stage pipeline computation unit. Again, the way DIV Unit treats instruction is the same as that of ALU. The only difference lies in the operations executed by DIV Unit and ALU Unit.

C. Memory hierarchy

- Instruction memory: Vary with different instruction number and cluster length.
- SRF, SP, LRF: Assume each SRF, SP, and LRF is a sixty-four 32-bit register. It could vary with the demands of diverse media applications.

D. Mechanism of the micro-architecture simulator

The performance simulation flow of media application in micro-architecture simulator should be delineated in this section. In this paper, FFT is selected as the simulation benchmark.

The application of Simulator can be roughly classified into two stages: (1) Determination of the simulation micro-architecture of the application. This step contains a number of parameters setups. (2) Plugging cluster instruction (stream) into simulator for simulating, and performance estimation.

- Parameter definition: The number of ALU, MUL, and DIV in a cluster, the number of cluster, the number of SRF, SP, and LRF and the consumed clock cycles executed by every functional unit should be decided from hardware design, in this case is the timing information from EDA tools from matching micro-architecture.
- Map application into dedicated binary stream programming codes: Based on the parameter definition stage, ISA of the micro-architecture and the instruction format of different mirco-architecture is decided. As shown in Figure 5, a compiler or hand-coding can used to scheduling the instructions of the application according into cluster instructions, then the cluster instructions is mapping into stream programming codes that can be executed on the stream processor.
- Simulation: The final stage is to load the code and simulate in the simulator, which contain following stages: (1) Load the cluster instructions saved in the files into instruction memory using File I/O. (2) Controller would fetch the cluster instructions in the instruction memory, and then equally distribute these instructions to clusters for dealing with instruction. (3) Cluster is used to divide the cluster instruction into instructions of the same number as the number of functional unit in the cluster, and then submit these instructions into ALU/MUL/DIV for executing. (5) Finally estimating performance including CPU Time, memory access times and the amount of each memory hierarchy level being used.

III. Performance Evaluation

In the paper, 32-point Fast Fourier Transform (FFT) is selected for the benchmark and translated as application into stream programming model to evaluate the performance of the micro-architecture defined.

A. Benchmark

The formulation of selected benchmark is shown in Equation 1. FFT algorism has to be mapped into stream programming model. By casting media applications as stream programs, hardware is able to take advantage of the abundant parallelism, computational intensity, and locality in media applications.

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{kn}, 0 \le k \le N-1 \qquad (1)$$
$$W_N = e^{-j \cdot 2\pi / N}$$

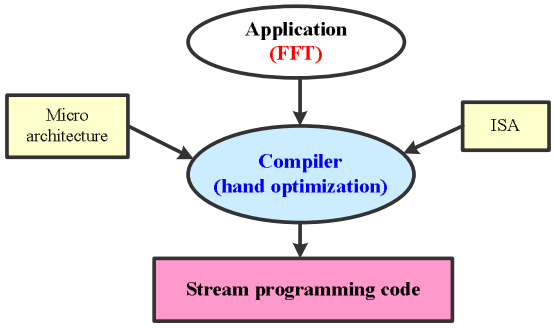

Figure 5 Map the application into stream programming code

B. Experimental set-up

- Parameter definition: Before simulating FFT on simulator, a couple of parameters for determining stream micro-architecture have to be defined, including number of function unit in a cluster, cluster number and required clock cycles for functional unit executing operation.
- Translate FFT algorism into stream code: As the simulation flow illustrated in Figure 4 shows, the selected media application is mapped into stream programming model. Therefore, FFT is translated into stream code for the convenience of simulating on simulator.
- Data loading from main memory: In addition to determining parameters mentioned above, simulator will in advance load the necessary memory data into SRF, and LRF. the memories that data consumes when performing FFT, i.e., the thirty-two x[n] in DFT, have been loaded on SRF in advance, and a number of indispensable parameters during computation have been loaded on LRF in advance, as well. As an example, during operation of FFT, there are many mathematical formulae regarding sin and cosine being used. Thus, a couple of corresponding values of sin and cosine will be loaded into LRF in advance. After the necessary parameters and loading for memory data are determined, the simulator is ready to evaluate the performance.

C. Performance Evaluation

Figure 6 shows the analysis chart of 32-point FFT performance evaluation. The x-axis represents different stream micro-architectures, including 1-cluster, 2-cluster, 4-cluster, and 8-cluster; while the y-axis denotes performance.

Since the number of accessing memory hierarchy is fixed when executing FFT, the number of SRF being used increases proportionally with the number of clusters, which linearly increases from one to four. However, the number of consumed LRF shows linear decrease. In addition, when increasing the number of cluster from 4 to 8, the number of accessing SRF suddenly doubles and the number of accessing LRF show noticeable decrease, which represents

that data exchange between clusters, i.e., utilizing SRF to exchange data, becomes frequent. On the other hand, the number of data exchange inside every cluster, i.e., using LRF to do data exchange, would decrease.

From 1-cluster, 2-cluster, to 4-cluster micro-architecture, the performance all doubly increases. However, the performance only shows little improvement from 4-cluster to 8-cluster micro-architecture. It cause huge data exchange from high-bandwidth LRF to low-bandwidth SRF since the bandwidth of LRF > SP > SRF. In this case, the advance in performance does not follow that in expensive hardware.

Figure 7 shows the memory usage, where x-axis represents different level of memory hierarchy, while y-axis denotes memory usage in every level. The demand for memory in every level does not hold close relationship between performances.

The demands for SRF usage in 1-cluster, 2-cluster, 4-cluster, and 8-cluster micro-architecture are roughly the same. It does not increase with the number of clusters. The capacity of SP usage for four micro-architectures is quite different. However, it does not positive relation with the number of cluster. The capacity of LRF would linearly increase with the number of cluster. In the 3-tiered memory hierarchy, only the demand of LRF holds positive relation with cluster number.

It could be observed that when the number of cluster increase from 1 to 2 and 2 to 4, CPU Time would double, and the number of SRF being used shows "linear" increase while the that of LRF present linear decrease. However, when the cluster number of the simulator goes from 4-cluster to 8-cluster, the progress of CPU Time is limited and causes large amount of SRF being used, i.e., huge data exchange between clusters. The design of stream processor is trying to have data calculated inside cluster as possible. When necessary, data would be exchanged between clusters through SRF. However, the simulation results of 8-cluster micro-architecture do not fit our expectation.

Therefore, 4-cluster micro-architecture has been chosen as the most suitable one for executing FFT in stream processor. In addition, the usage of memory hierarchy is SRF : SP : LRF = 64 : 20 : 212, respectively.

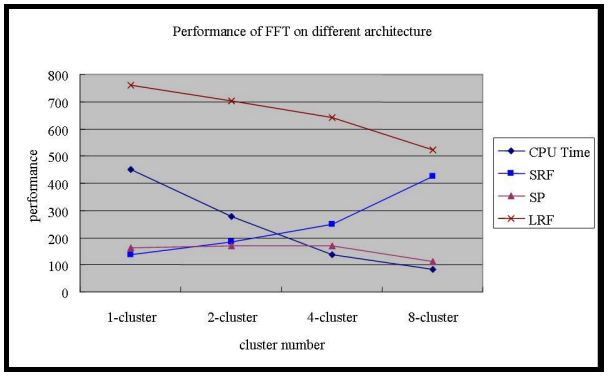

Figure 7 Performance of FFT on different micro-architecture

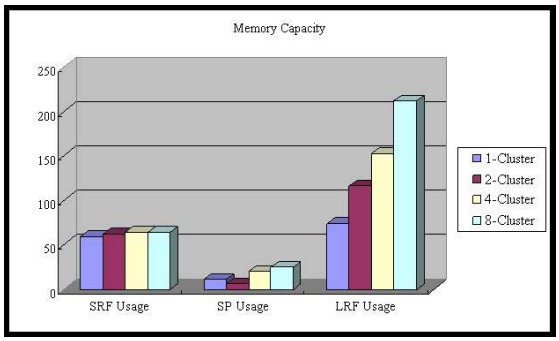

Figure 8 Memory capacity of memory hierarchy

IV. CONCLUSION

This paper presented a micro-architecture simulator for stream processor. The simulator evaluate the performance of media application executed on various micro-architectures, and then to analyze the utility rate of hardware and consumption of memory. By comparing the performance of media application executed on diverse micro-architectures, the optimized hardware micro-architecture can be determined for specific application and its tradeoff to different VLSI implementation can be evaluate before the costly circuitry prototype implementation.

REFERENCES

[1] Rixner, Scott, et al., "A Bandwidth-Efficient Micro-architecture for Media Processing," In Proceedings of the International Symposium on Micromicro-architecture (December 1998), pp. 3-13.

[2] U. J. Kapasi, P. Mattson, W. J. Dally, J. D. Owens, and B. Towles, "Stream scheduling," Concurrent VLSI Micro-architecture Tech Report 122, Stanford University, Computer Systems Laboratory, March 2002.

[3] U.J, Kapasi et al., "Programmable Stream Processors," Computer, August 2003, pp. 54-62

[4] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, M. Horowitz, "Smart Memories: A Modular Reconfigurable Micro-architecture," Computer Micro-architecture, 2000, Proceedings of the 27th International Symposium on 2000, Pages161-171.

[5] J. Draper, et al, "The Micro-architecture of the DIVA Processing-in-Memory Chip," to appear at International Conference on Supercomputing, June 2002.

[6] Brucek Khailany, "The VLSI Implementation and Evaluation of Area- and Energy-Efficient Streaming Media Processors," PhD thesis, Stanford University, June 2003.

[7] Dally W. J. et al., "Stream processors programmability with efficiency," ACM QUEUE, March 2004.

[8] Owens JD, Dally WJ, Kapasi UJ, Rixner S, Mattson P, Mowery B, "Polygon rendering on a stream micro-architecture," In: Proc, of the Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2000. 23~32.

[9] William Dally et al, "Stream Processors: Programmability with Efficiency," ACM Queue, March 2004, pp. 52-62.

[10] Ujval J, Kapasi, William J, Dally, Scott Rixner, John D, Owens, and Brucek Khailany, "The Imagine Stream Processor," Proceedings of the International Conference on Computer Design, Sep. 2002.

[11] http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html