Implementations of FIR for MONARCH Processor

Jinwoo Suh and Janice O. McMahon
University of Southern California – Information Sciences Institute
3811 N. Fairfax Drive, Suite 200, Arlington, VA 22203
{jsuh, jmcmahon}@isi.edu

Introduction

Stream processing exploits the properties of the stream applications such as parallelism and throughput-oriented nature of the applications. One research on the streaming processing is implementing data flow architecture. One example is MONARCH architecture being developed at Raytheon and USC/ISI. We implemented an FIR bank using ALU clusters in the MONARCH architecture using its simulator and performed optimizations and analysis of the results.

MONARCH Processor

The MONARCH processor consists of six RISC processors and 12 ALU cluster. These are interconnected using networks. Figure 1 shows a block diagram of the MONARCH processor.

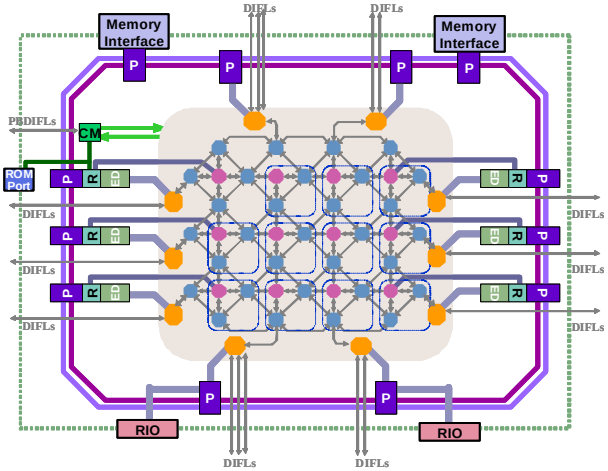

Figure 1: MONARCH processor

The ALU clusters provide 64 GOPS peak performance. The expected operating clock frequency is 333 MHz.

The authors gratefully acknowledge the extraordinary support of the Raytheon MONARCH team for the use of their compilers, simulators, and their generous help. The authors also appreciate Susan Reckitt for her proofreading.
This effort was sponsored by Defense Advanced Research Projects Agency (DARPA) through the Dept. of Interior National Business Center (NBC), under grant number NBCH1050022. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsement, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), Dept. of Interior, NBC, or the U.S. Government.

FIR bank

FIR bank is one of the kernel benchmark suite [1] specified by Lincoln Labs for Polymorphic Computing Architecture, a DARPA program. The FIR bank implements a set of M FIR filters and each FIR filter m, $m \in \{0, 1, \ldots, M\text{-}1\}$, has a set of impulse response coefficients $wm[k]$, $k \in \{0, \ldots K\text{-}1\}$. It is mathematically specified as:

$$y_m[i] = \sum_{k=0}^{K-1} x_m[i-k]w_m[k], \text{ for } i = 0,1,\ldots,N-1.$$

The filters are implemented in arithmetic clusters. Since the data is complex, the number of multiplications and additions per (input data * coefficient) are both four. This perfectly matches the configuration of the arithmetic clusters since the numbers of multipliers and adders in the arithmetic clusters are the same.

Figure 2 shows a basic data flow pipeline implemented using arithmetic clusters. The input data is connected to delays, and then connected to multipliers through multiplexers. The multipliers multiply the input data and coefficients. The output data is sent to adders. One output data from an adder is connected to the next adder such that as data flows through the adders, the data is accumulated. On the input side of multipliers, there are increasing numbers of zeros used to clean the pipeline. Input data is delayed to match the flow speed in the adder pipeline.

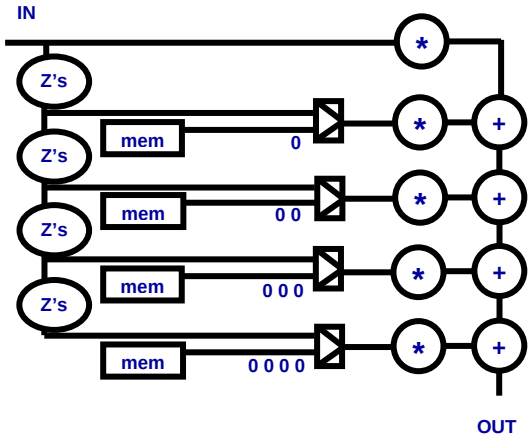

Figure2: A floating point FIR implementation

Figure 2 shows FIR for complex data. The complex FIR can be implemented using the floating point FIR implementation. There are four pipelines: input real * coefficient real, input real * coefficient imaginary, input imaginary * coefficient real, and input imaginary *

coefficient imaginary. These are combined at the end of the pipeline.

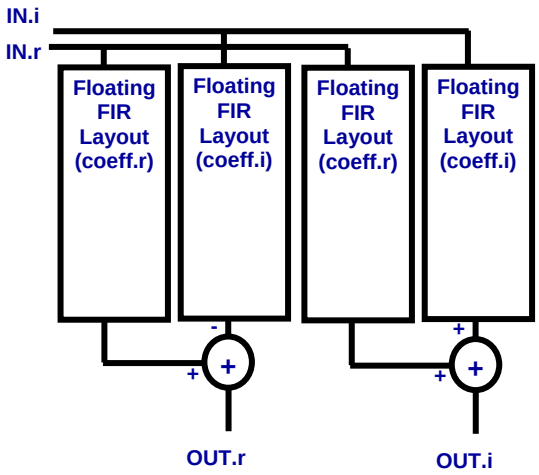

Figure 3: A complex FIR implementation

Implementation Results

In this paper, we implemented FIR bank using the assembly language for MONARCH. The RISC was not used for this implementation. Figure 4 shows the results when the number of coefficients is 24.

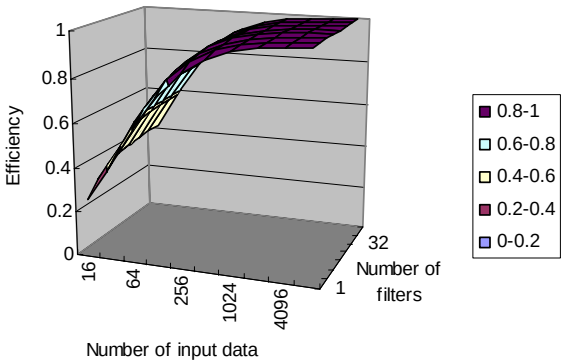

Figure 4: Efficiency when number of coefficients is 24

The figure shows that the efficiency is increased as the number of input data is increased. This is because the initial startup cost for the pipeline is amortized over the length of the input data. The efficiency is relatively constant as a function of the number of filters because the pipeline simply repeats number of filter times the same way and it does not change efficiency. When the number of input data is large, the efficiency is 99.9%. The efficiency is such a high level that no conventional processor can achieve easily.

Figure 5 shows the efficiency when the number of input data is 1024.

The figure shows that the efficiency varies as a function of the number of coefficients. The reason for this is that the FIR is implemented as a unit of 24 coefficients. For example, if the number of coefficients is 25, it puts 48 coefficients with 23 (= 48 – 25) zero coefficients. Thus, it almost reduces the efficiency to half. However, as the number of coefficient increases, the efficiency also goes up and get a best efficiency when the number of coefficients is a multiple of 24.

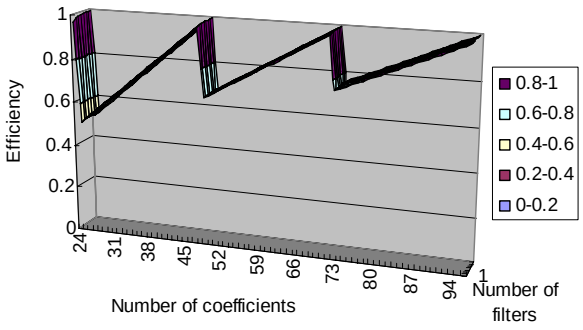

Figure 5: Efficiency when input data size is 1024

Note that the reduced efficiency is merely due to the simple implementation of algorithm and it is not due to the characteristics of MONARCH architecture. With better implementations of FIR that utilize the ALU clusters more efficiently using different level of optimizations, it is possible to obtain higher efficiencies. Estimated efficiencies for two better algorithms are as shown in Figure 6.

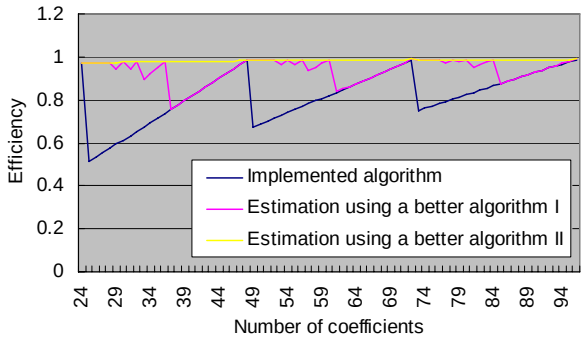

Figure 6. Estimated efficiency using a better algorithm

Conclusion

The authors have presented implementation results of FIR for MONARCH ALU clusters. Our results show that it is possible to obtain very high utilization that is close to the peak performance for FIR whose characteristics matches the MONARCH ALU cluster.

We used MONARCH assembly language to code the FIR and the cost of coding is pretty high compared with using high level languages. However, we expect the coding cost will be reduced significantly when more tool chains are available and mature.

References

[1] J. Lebak, A. Reuther, and E. Wong, "Poloymorphous Computing Architecture (PCA) Kernel-Level Benchmarks," Project Report PCA-KERNEL-1, MIT Lincoln Laboratory, January 2004.