Mapping Signal Processing Kernels to Tiled Architectures

Henry Hoffmann and James Lebak (presenter)
MIT Lincoln Laboratory[1]
05 May 2004

A key feature of many new computer architectures is that they are composed of multiple tiles, each of which is a fully capable processor. Tiled architectures are attractive alternatives to monolithic computer architecture designs because they allow a larger design to be built from smaller modules and limit the number of wires that need to span the entire chip. Examples of tiled architectures include many under development for the DARPA/IPTO Polymorphous Computer Architectures (PCA) program, including the MIT Raw machine [13], the Stanford Smart memories project [8], and the University of Texas TRIPS machine [9].

The ***decoupled systolic architecture*** (DSA) represents a canonical abstract machine that encompasses many of the key features of single-chip tiled architectures [4] including the PCAs and other emerging architectures such as Scale [6], Wavescalar [12], and Synchroscalar [10]. ***Stream algorithms*** are defined as the family of algorithms which can achieve $100\,\%$ computational efficiency on the DSA. The DSA and stream algorithms provide a rigorous analytical framework for reasoning about the performance of algorithms on modern architectures. This framework is unique in that it explicitly penalizes algorithmic implementations that make use of long wires and/or large local memories while rewarding those algorithms that can efficiently execute using only a small, bounded amount of local memory and a next-neighbor interconnect network. Thus, this framework makes an excellent match for evaluating architectures faced with the growing physical concerns of wire delay [3] and the energy dissipation of on-chip memory [2, 5].

Stream algorithms are therefore important because the existence of a stream algorithm for a particular problem implies a scalable, computation, energy, and area efficient solution to that problem on many real-world architectures. Stream algorithms decouple memory access from computation, performing memory access on tiles on the periphery of the chip and performing computation in a systolic fashion on the tiles in the interior of the chip. For a problem of size N on an $R \times R$ array of tiles, the efficiency of the problem is the total number of operations $C(N)$ divided by the product of the number of cycles $T(N,R)$ and the total number of memory tiles $M(R)$ and compute tiles $P(R)$,

$$E(N,R) = \frac{C(N)}{T(N,R) * (M(R) + P(R))}. \tag{1}$$

For a conventional architecture, the total number of tiles is equal to 1. A necessary condition for $E(N,R)$ to scale with the size of the array is that $M(R)$ be asymptotically smaller than $P(R)$. An algorithm that meets the requirement that $P(R) = o(M(R))$ is *decoupling efficient*, because it efficiently decouples memory accesses from computation [11]. An algorithm is *computation efficient* if $\lim_{\sigma,R\to\infty} E(\sigma,R) = 1$ where $\sigma = N/R$. Computation efficient algorithms implemented on an array of fixed size scale toward an asymptotic limit on performance as data size increases, and this asymptotic limit becomes larger as the array size R increases. Stream algorithms are therefore those algorithms that meet the computation efficiency condition. Stream algorithms for matrix multiplication, QR factorization, convolution, and other applications have been discovered and implemented on the Raw cycle accurate simulator [4]. Comparison of these algorithms with conventional implementations on conventional architectures such as the PowerPC G4 [7] shows that stream algorithms have the potential to achieve higher efficiency on many different problems.

This presentation focuses on understanding when a stream algorithm exists for a given kernel. We do so by considering the directed acyclic graph (DAG) for a particular implementation of the kernel. Nodes in the DAG represent inputs, outputs, or intermediate products of the algorithm, and edges from node A to node B in the DAG show that A is used to compute B. We can characterize the DAG for an algorithm by the ratio of inputs, W, to the number of intermediate products, Q, for which any one value is directly required. For example, in an algorithm to multiply two $N \times N$ matrices A and B, element i,j of the output matrix C is computed as $c_{i,j} = \sum_{k=1}^{N} a_{ik} b_{kj}$. That is, for each output, there are $W = 2N$ inputs used and a total of $Q = N$ intermediate products (the partial sums) computed. The stream algorithm implementation of matrix multiply meets the compute efficiency condition. Matrix multiplication is an example of a kernel with a *constant ratio* of W to Q. All known algorithms – including QR, SVD, convolution – with a constant ratio of W to Q have implementations that meet the compute efficiency condition.

In contrast, consider an algorithm to compute the FFT of a length-N vector. To compute any particular output of the FFT, all N inputs are required, and (as is well known) each input directly contributes to $\log_2(N)$ intermediate products.

[1]This work sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

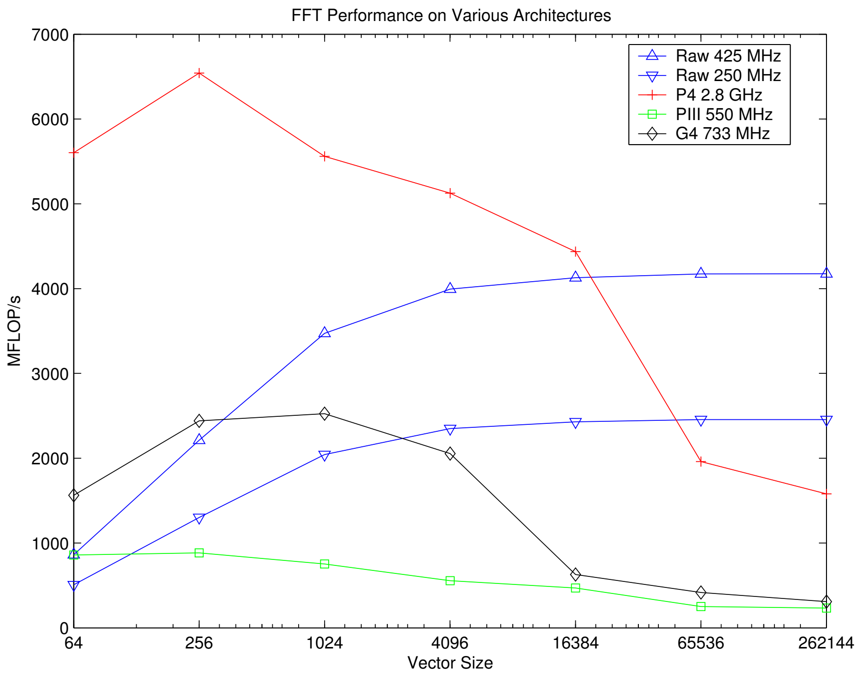

Figure 1: Comparison of Raw FFT throughput, measured in the Raw cycle-accurate simulator, to FFTW throughput on the PowerPC G4 and Xeon.

For the radix-2 FFT algorithm, the ratio of $W = O(N)$to $Q = O(\log_2(N))$ is asymptotically *greater* than a constant. Because any stream algorithm for the FFT must meet the decoupling efficiency condition, we cannot use local memory to buffer the large number of inputs. Instead inputs must reside in the network while compute tiles are working. For the FFT, with a $W/Q = O(N)/O(\log_2(N))$, this implies that the maximum distance that any piece of data must travel is greater than the number of intermediate calculations in which the data is used. Therefore, communication costs cannot be effectively amortized in the systolic implementation on a tiled architecture. The factor T in the denominator of the efficiency expression (1) will have a lower bound that is limited by the size of the array, meaning that the efficiency cannot approach a limit of 1 as the array size R increases. A stream algorithm implementation for the FFT is still an open research problem. Stream algorithm techniques can be used to implement an efficient implementation of the radix-4 FFT for a 4x4 tile array, but this implementation is not scalable and performance will be worse on larger Raw systems. Simulated throughput of this algorithm is compared to the throughput of FFTW [1] on the 2.8 GHz Pentium 4 and 733 MHz G4 in Figure 1. The Raw FFT achieves high performance for large data sizes, and offers performance that is more stable across a range of data sizes.

In this talk, we will describe the implementation of FFT, QR factorization, and CFAR kernels on the Raw simulator and Raw board. We examine the performance of these kernels and compare to conventional implementations on the Pentium and G4 architectures. Finally, we characterize the DAG of each kernel and discuss how the DAG influences the implementation on Raw and on tiled architectures in general.

References

[1] Matteo Frigo and Steven G. Johnson. FFTW: An adaptive software architecture for the FFT. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1381–1384. IEEE Signal Processing Society, May 1998.

[2] Ricardo Gonzalez and Mark Horowitz. Energy Dissipation in General Purpose Microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, September 1996.

[3] Ron Ho, Kenneth W. Mai, and Mark A. Horowitz. The Future of Wires. *Proceedings of the IEEE*, 89(4):490–504, April 2001.

[4] Henry Hoffmann. Stream Algorithms and Architecture. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2003.

[5] Jason Sungtae Kim, Michael B. Taylor, Jason Miller, and David Wentzlaff. Energy Characterization of a Tiled Architecture Processor with On-Chip Networks. In *ACM Symposium on Low Power Electronics and Design*, pages 424–427, Seoul, Korea, August 2003.

[6] Ronny Krashinsky, Christopher Batten, Mark Hampton, Steve Gerding, Brian Pharris, Jared Casper, and Krste Asanovic. The Vector-Thread Architecture. In *31st International Symposium on Computer Architecture*, München, Germany, June 2004. (publication pending).

[7] James Lebak, Hector Chan, Ryan Haney, and Edmund Wong. Polymorphous computing architectures (PCA) kernel benchmark measurements on the PowerPC G4. Project Report PCA-Kernel-2, MIT Lincoln Laboratory, Lexington, MA, January 2004.

[8] Ken Mai, Tim Paaske, Nuwan Jayasena, Ron Ho, William J. Dally, and Mark Horowitz. Smart Memories: A Modular Reconfigurable Architecture. In *28th Annual International Symposium on Computer Architecture*, pages 161–171, June 2000.

[9] Ramdass Nagarajan, Karthikeyan Sankaralingam, Doug C. Burger, and Steve W. Keckler. A Design Space Evaluation of Grid Processor Architectures. In *34th Annual International Symposium on Microarchitecture*, pages 40–51, December 2001.

[10] John Oliver, Ravishankar Rao, Paul Sultana, Jedidiah Crandall, Erik Czernikowski, Leslie W. Jones IV, Dean Copsey, Diana Keen, Venkatesh Akella, and Frederic T. Chong. Synchroscalar: A Multiple Clock Domain Power-Aware Tile-Based Embedded Processor. In *31st International Symposium on Computer Architecture*, München, Germany, June 2004. (publication pending).

[11] James E. Smith. Decoupled Access/Execute Computer Architectures. *ACM Transactions on Computer Systems*, 2(4):289–308, November 1984.

[12] Steven Swanson, Ken Michelson, Andrew Schwerin, and Mark Oskin. WaveScalar. In *36th International Symposium on Microarchitecture*, pages 291–302, San Diego, CA, December 2003. IEEE Computer Society.

[13] Michael B. Taylor, Jason Kim, Jason Miller, David Wentzlaff, Fae Ghodrat, Ben Greenwald, Henry Hoffmann, Paul Johnson, Jae-Wook Lee, Walter Lee, Albert Ma, Arvind Saraf, Mark Seneski, Nathan Shnidman, Volker Strumpen, Matt Frank, Saman Amarasinghe, and Anant Agarwal. The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs. *IEEE Micro*, 22(2):25–36, March/April 2002.