Project Updates

Lecture #15: Thursday, 23 May 2002
Lecturer: Project Groups
Scribe: Ayodele Thomas
Reviewer: Mattan Erez

1 Group 1: Aspect Ratio

Group Members: James Bonanno, Suzanne Rivoire, Rex Petersen

Three key areas are being explored to evaluate the impact of varying aspect ratios on the Imagine architecture. 1) Cost Model 2) JPEG Compression Algorithm 3) Definition of configurations

1) Cost Model. The cost model is based on the performance per unit area. Approximations are used to calculate the affect of different configurations on the chip area. The aspects being varied include the number ALUs per cluster, the ALUsize, the number of clusters, and the number of threads.

2) JPEG. The application being used to evaluate the architectures is a JPG-like algorithm that uses the discrete cosine transform (DCT) and run length coding (RLC).

3) Architecture Configurations. The limiting factors are being explored to determine the best configurations to exploit DLP, ILP, and TLP.

One important question that was raised during the comment period was "Is it really realistic for the area to be only 15% greater than the current Imagine with a single cluster containing 48 ALUs." The effect of such components as the scratchpad in calculating their areas was apparently ignored.

2 Group 2: Viterbi Algorithm

Group Members: John Davis, Andrew Lin, Njuguna Njoroge, Ayodele Thomas

A general description of the Viterbi algorithm was given. The most interesting aspect of implementing the algorithm on Imagine is dealing with dependencies in the feedback path.

Several approaches are being taken to extract parallelism. 1) Extract ILP. The problem here is the detrimental affect of intercluster communication.

2) Approach 2 extracts DLP and TLP by implementing multiple decoders. DLP is extracted by breaking a single message into frames to break the dependency chain which are encoded on separate clusters. In order to break the dependency chain, the frames are padded such that they begin and end in a predetermined state. This is key because without the padding, the dependency chain will go back to the beginning of the input stream. TLP is extracted by encoding a separate message on each cluster.

The Viterbi algorithm is also being benchmarked with a DSP running assembly code so that a comparison can be made with the common implementation of the algorithm. The assembly code is functional and being profiled. The StreamC/KernelC code is in the debug stage. A Matlab version has been implemented to create data streams and verify results.

Several optimizations are being studied including using bit packing to reduce the size of the storage footprint.

The work on coding the Viterbi will be used to give insight on how to manage feedback dependencies in general on the Imagine architecture that is reflected in such algorithms as IIR filters and Delta-Sigma.

3 Group 3: Compiler

Group Members: Jayanth Gummaraju, Ahbishek Das, Mattan Erez

The compiler group commented that their work is looking more like translation than compilation. They are converting the stream aspect of Brook into StreamC using the Brook metacompiler as a base.

One problem that must be addressed is the fact Brook has an infinite stream length while StreamC must have a stream size declared. The syntax for calling kernels has been completed. Currently, files can be split automatically (headers, kernels, etc).

4 Group 4: Stream Cache

Group Members: Timothy Knight, Arjun Singh, Jung Ho Ahn

The simulation environment has been completed for evaluating stream caches. The effect of having a cache vs. not having a cache is being explored. If an application is not memory limited, a cache should give a benefit. The locality that can be exploited by the stream cache is due to small kernels that are called repeatedly.

Two versions are being implemented. Version one is a memory system cache where the cache is located between the memory and the SRF. The second version is a cluster cache with caches located between the SRF and the clusters.

5 Group 5: IP Address Lookup

Group Members: Henry Fu, Yeow Cheng Ong, Harn Hua Ng

The coding of the IP Address lookup application with a single kernel has been completed. It has been tested successfully with 2 Imagine processors. However there are problems making it work with more than two processors and it will currently only work in debug mode - not with Isim.

Two implementations are being considered which differ in the way that the lookup table and data traffic are distributed. In the first implementation, the lookup table is split in half and distributed to the two clusters. The same traffic streams through both clusters until the best match is found within each cluster and then those two matches are compared for the best result. In the second implementation, the traffic is split in two and the two clusters share a single copy of the table. In this implementation, each cluster selects the best match for different traffic.

The current experimental lookup table is small such that the entire table fits in the scratchpad. It was not clear how a realistic table size ($> 30K$) that would certainly overflow the scratchpad would be handled.

6 Group 6: Unstructured Mesh

Group Members: Nuwan Jayasena, Yangjin Oh, Hsaio Heng Lee, Anand Ramalingam, Francois Labonte

This group is looking at how Imagine can be used to code an unstructured mesh. Conditional streams are used to help to fetch the appropriate data.

One addition to the architecture that is being considered is a cache. A cache would be beneficial if the same values are fetched repeatedly. Another addition is a hardware buffer. A random dataset is being simulated and bank and buffer size are modified to reduce the number of potential conflicts.

Two hashing functions are being used, one which uses the division method and one which uses a multiplication method. Three apps will be used to evaluate the architecture including Motor, Tire, and Gear.

7 Group 7: Legacy Architectures

Group Members: Chaiyasit Manovit, John J Kim, Sanjit Zubin Biswas, Zi-Bin Yang

The focus for the Legacy Architecture group is using compiler optimizations to model array accesses with affine indices in the Imagine architecture. A goal is to minimize prefetching by using locality analysis and software pipelining.

Accesses are likely to be cache misses are tried to be determined up front. Loops are split into prologue, body and epilogue in order to schedule the data appropriately. Three types of reuse are exploited: 1) Spatial reuse - same cache line is used 2) Temporal reuse - same location accessed soon after previously accessed 3) Group reuse - former uses latter

Loop transformations and scheduling are used to unroll the loop and prefetch a certain number of records. For this experiment, streams are being modeled as arrays.

The compiler is being implemented using GCC 3.1 and performance is being measured for a square matrix multiplication.

8 Group 8: Mapping Stream Stencils to Imagine

Group Members: Jacob Chang, Alex Solomatnikov, Jae-Wook Lee, Nate Hill

The Stencil group is looking at both 1D and 2D mapping of stream stencils to Imagine. For 1D and a stencil size of 0 - 8, history is kept in the local registers and communicated across clusters. For sizes of greater than 8, history is kept in the scratchpad and the performance will be limited by communicating with the scratchpad. If the history is too large for the scratchpad, it is assumed that the problem can be strip-mined into sizes that can fit into the scratchpad. In this case, immediate values will have to be computed and held. A 2D version was discussed too briefly for any significant comments.

One problem that was raised was how to prevent variables from overflowing the scratchpad. One solution was to use a temporary stream to hold the history values.

9 Group 9: Cellular Automata

Group Members: Dan Bentley

This experiment has been broken down into three areas:

1) Making Code The Game of Life and an unstructured grid have been/will be implemented. 2) Fun Fun entails exploring how to express cellular autonoma idiomatically and how to provide the compiler with more information automatically on optimizations such as strip-mining and loop unrolling.

3) Usefulness Documentation will be written to give a philosophical look at streams and how to approach programming in streams.

An aspect that is being explored is an idiomatic way to say "all of my neighbors" in the language construct (i.e. for loops). A larger goal is to make people understand how to use and program the Imagine architecture.

10 Group 10: Vector

Group Members: Mehdi Baradaran Tahoori, Paul Wang Lee

This group is exploring how to transform vector code into stream code. Vector codes are currently being written in StreamC and KernelC. The dimensions being examined are vector size and how to extract basic blocks. The basic blocks will indicate how to split the code into a kernel. Performance will be evaluated for varying vector sizes and the number of ALUs.