Project Brainstorm

Lecture #8: Tuesday, 30 April 2002
Lecturer: Prof. Bill Dally
Scribe: Rex Petersen
Reviewer: Mattan Erez

Announcements:

- The due date for HW1 has been postponed until this Thursday 5/2
- Project proposals are due next Tuesday on 5/7
- Next lecture will be a Brook Tutorial. There will be two handouts: *Brook: A Stream Processing Language* and *Brook QuickSpec*. Note that the QuickSpec is the most up-to-date if there are discrepancies between the two.

During this lecture we discussed ideas for final projects for the course. Professor Dally gave some general guidlines and direction for the projects and answered questions as we discussed different ideas from the handout on suggested project topics.

1 General Ideas that Apply to Most Projects

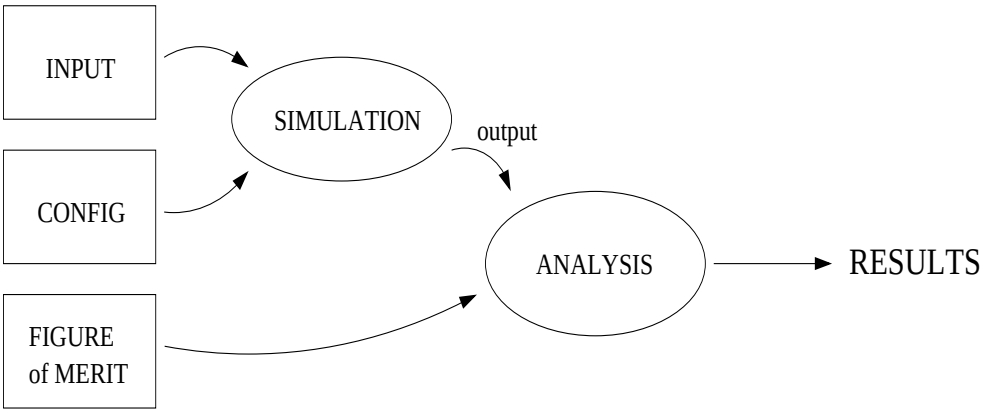

Figure 1: General Flow for a Project

Most projects will require the identification of inputs and configurations that will go into a test or simulation. It is also necessary to decide how the output will be analyzed. This requires some figure of merit to determine which output is better than another.

There were three items identified that each project team should be able to define:

- *INPUT:* A set of applications to be tested, code to be compiled or scheduled, etc.
- *CONFIG:* The types of configurations to be explored such as a different number of ALUs per cluster, scheduling algorithms, compile techniques, etc.
- *FIGURE of MERIT:* This is the criteria used to evaluate the results such as shortest execution time, lowest cost based on a cost model, smallest area, or the most effective utilization of resources.

Professor Dally recommended doing a combination of brainstorming on your own as well as reading what others in industry have done already. He cautioned that reading everything out there first can sometimes dampen your creativity and you will tend to go in the same directions that others have gone even though there may be a better way to approach the problem.

A student commented that the best project ideas are those that are tractable, interesting, and load balanced. Professor Dally expanded on this comment and recommended that students look for low hanging fruit that can produce some early initial results since we only have 5 weeks for the project. It should lay some initial groundwork in an area. "You need to be a little bit fearless and dive right in."

2 Legacy Architecture

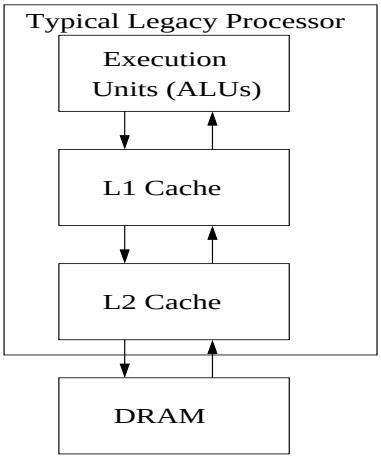

Figure 2: Legacy Cache

A project in this area would investigate how stream processing could improve the performance of an existing processor such as the Intel P4. For example, use the concepts of stream scheduling to make better use of the cache.

The cache on a legacy processor shown here in Figure 2 is very reactive and waits until it misses and then does something. A more efficient design may be more "stream-like" where data is moved into the cache in a way that will allow for fast access and evictions could be forced on data when it is known that that data will not be used again.

3 Mapping Legacy Code to Streams

Select an interesting application. Find out its issues such as bandwidth demands, memory footprint, etc. You don't have to convert the entire application into a stream language to do this study. Professor Dally stated that the outcome might be: "Here's a better way to build a stream processor that would be better for this application".

4 Aspect Ratio

Given some number of ALUs in a stream processor, what is the best way to distribute these resources across the three axes of parallelism: DLP (more clusters), ILP (more ALUs per cluster), or TLP (more independent execution engines)?

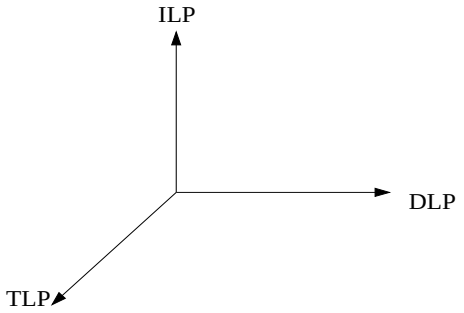

Figure 3: Aspect Ratio project would explore each axis of parallelism

The control case for this project could be 1 ALU, 1 cluster, and 1 thread. You will need to decide on a cost model for adding more ALUs or clusters and look at options that lie on the constant cost plane. The figure of merit could be execution time.

- *Question: What part of the configuration do we have control over?*

There is an *.md file that allows you to specify the number of ALUs per cluster. There is a #define for the number of clusters but many locations in the code assume that this is set to 8 (which is the number of clusters on Imagine). It is possible to experiment with less than 8 clusters but it will require more work to go above 8. There is also some ability to link multiple stream processors together using networking, which is working but not heavily tested.

5 High Level Language Issues

What should a stream language look like? Should it have retained state? This is a debated issue with the trade-offs being ease of coding vs. complexity and efficiency of

the compiler. Machine specific issues such as the number of clusters should not show up in a high level language.

A few comments were made about Brook which has no retained state aside from reduction variables. More details on Brook will be covered next time.

6 Time vs. Space Multiplexing

Investigate the advantages of both time and space multiplexing. The Imagine and RAW stream processors have provided two different examples of using time and space. Design an experiment to explore the continuum of options between the two extremes. The results should be useful for those designing the next generation of stream processors. It was noted that there are many differences between RAW and Imagine that make it difficult to clearly evaluate the performance impact of each feature separately. In this project it is recommended that you focus on just one issue such as the use of time vs. space.

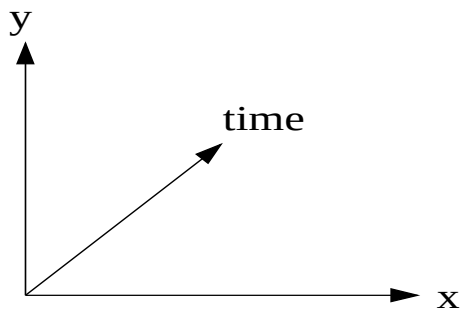

Figure 4: The challenge is mapping kernels into both space and time.

Combining both concepts of time and space multiplexing involves mapping kernels onto the graph shown here in Figure 4. One approach might be to consider how long each kernel may take and the bandwidth of the kernel output. On the RAW project they use *kernel fission* (where long kernels are split) and *kernel fusion* (where small kernels are combined).

It was pointed out that when looking at the best option for a processor in the future, it is important to consider the impact of process scaling and technology improvements that will effect power, delay, and area in the target timeframe.

7 Dealing with Irregular Data Structures

Investigate methods for executing programs with irregular data structures on a stream processor. An example of an irregular grid was discussed as shown in Figure 5.

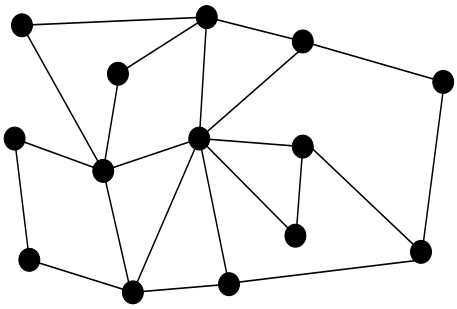

Figure 5: Example of an irregular grid

Consider an operation where the average of all neighboring points must be calculated at each point on the grid. Note that the point in the center has 7 neighbors, while other points have only 2.

The brute force method of doing this calculation would be to preprocess the grid and determine a value for MaxNeighbors, then assume each point has 7 neighbors some of which will be null. One problem with this method is that it wastes memory resources.

8 Bandwidth Heirarchy

An idea was presented to improve performance when dealing with the case where multiple computational units need the same piece of data from the SRF. One possible solution would be to put an extra cache between the SRF and the execution clusters.

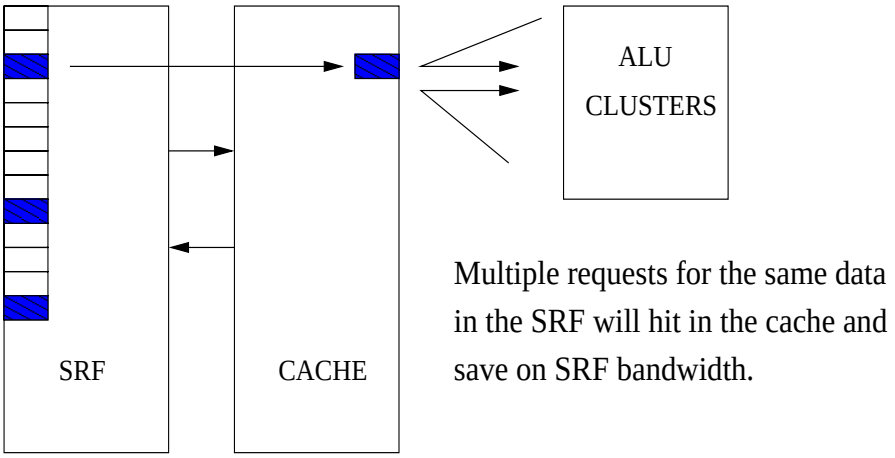

Figure 6: A cache could cut demand on SRF bandwidth

This would cut the demand on the SRF bandwidth since multiple requests would hit in the cache. This would be helpful in the irregular grid problem where the same neighbor value is read many times to compute the average at nearby points.