A Novel Approach to Sparse Matrix Factorization using Unroll & Re-association

Siddhartha **Supervisor: Dr Nachiket Kapre**
School of Computer Engineering, Nanyang Technological University

Introduction

Sparse matrix factorization is a common engineering problem that is often a computational bottleneck in many scientific applications. In this project, we explore techniques such as unrolling and re-association in an attempt to parallelize the front- solve algorithm found in the LU decomposition method. The front-solve is a major computational bottleneck in the KLU sparse matrix factorization algorithm, which can be found in applications such as SPICE, an EDA tool for designing and testing hardware. We have developed a preliminary sparse matrix pre- processor software model in Java to generate compute graphs of matrix solve operations using the novel technique, which are then used to benchmark the improvements in performance over the existing methods.

Methods

Unrolling

The aim of unrolling is to remove any data dependencies between each row solve for a Ly = b solve (front-solve). Figure 1 below an example of a small 4x4 dense lower-triangular matrix equation, and Equations 1 below show how we would solve this example by applying the unrolling technique.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ L_{2,1} & 1 & 0 & 0 \\ L_{3,1} & L_{3,2} & 1 & 0 \\ L_{4,1} & L_{4,2} & L_{4,3} & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

Figure 1: 4x4 dense lower-triangular matrix equation

$$\begin{aligned} y_1 &= b_1 \\ y_2 &= b_2 - L_{21}b_1 \\ y_3 &= b_3 - L_{31}b_1 + L_{32}L_{21}b_1 - L_{32}b_2 \\ y_4 &= b_4 - L_{41}b_1 + L_{42}L_{21}b_1 + L_{43}L_{31}b_1 - L_{43}L_{32}L_{21}b_1 + L_{43}L_{32}b_2 - L_{42}b_2 - L_{43}b_3 \end{aligned}$$

Equations 1: Solving Figure 1 example with unrolling

Re-association

By unrolling, we introduce many extra computations in the form of multiply chains. These multiply chains have to be added together and in turn result in long add chains. To reduce the critical compute latency, we can perform re-association on the compute chain to transform it into an efficient binary tree structure. This has the potential to reduce the critical compute latency from an order of O(N) to O(log N). This is especially beneficial for long chains, which is where we would observe greatest savings in compute latency. Figure 2 shows how re-association can be done on a simple 4 input multiply chain.

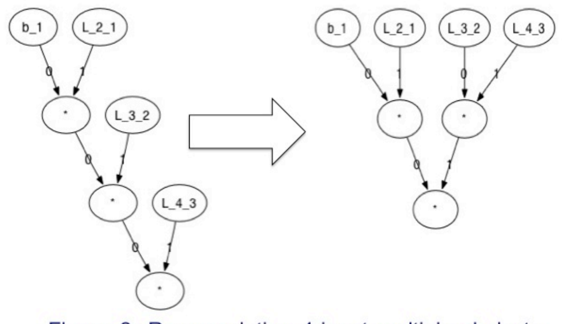


Figure 2: Re-associating 4-input multiply chain to save 1 compute cycle latency

Acknowledgements

I would like to thank Dr. Nachiket Kapre for his patience and guidance on this project.

References

1. Nachiket Kapre, SPICE[2] A Spatial Parallel Architecture for Accelerating the SPICE Circuit Simulator. California Institute of Technology, 2010.

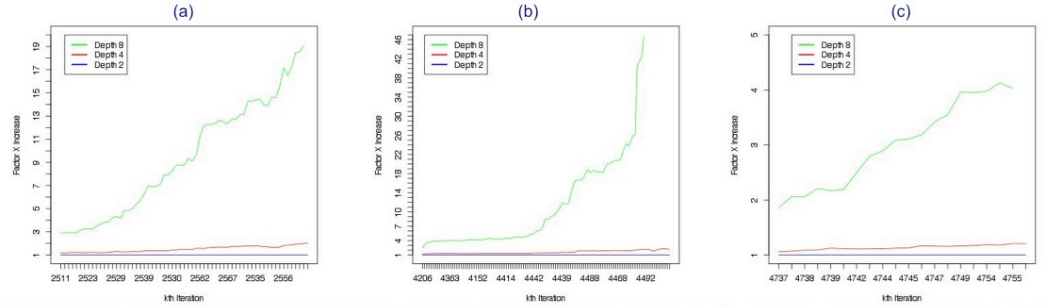
Preliminary Results

Depth Unroll

Performing a full unroll on a large benchmark matrix is undesirable, as we discovered from our experiments. The number of extra computations that must be carried out is significantly increased such that an unachievable amount of parallelism would be required to cope with the added workload. Hence, it was advisable to control the depth of our unroll such that the increase in the number of computations is not too large.

$$\begin{aligned} y_1 &= b_1 \\ y_2 &= b_2 - L_{21}b_1 \\ y_3 &= b_3 - L_{32}y_2 - L_{31}y_1 \\ y_4 &= b_4 - L_{43}b_3 + L_{43}L_{32}y_2 + L_{43}L_{31}y_1 - L_{42}y_2 - L_{41}y_1 \end{aligned}$$

Equation 2: Performing a depth unroll of two on Figure 1 Example. The variables in red highlight the data dependencies in these row solve operations.



Figures 3(a)-(c): # of nodes as depth of unroll is varied from 2-8 (a) Bomhof1, (b) Bomhof2, (c) Simucad_ram2k

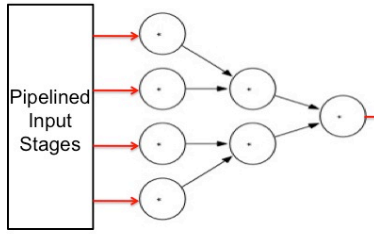


Figure 4: A 4-to-1 multiply reduction tree kernel architecture, with pipelined inputs

Figures 3(a)-(c) above show the increase in the number of compute nodes as we increase the depth of the unroll for 3 different benchmarks. *Bomhof1* is the smallest benchmark in terms of the input matrix size, while *Simucad_ram2k* is the largest.

Hardware Design

Due to the significant increase in the parallelism requirements after unrolling, we discover that a packet switch network is unable to meet the demands due to the communicate latency penalty between processing elements. Hence, a new pipelined-input reduction tree architecture is chosen to handle the parallelism requirements. Figure 4 shows an example of an 4-input multiply reduction tree.

Maxeler Technologies

Maxeler systems is an application accelerator hardware solution that exploits the parallelism potential offered by FPGAs. The Maxeler IDE allows us to describe hardware kernels with ease using a high-level language like Java, while the hardware synthesis, place & route and optimizations are handled by the compiler. We would be developing a hardware solution on the Maxeler systems to measure actual speedups.

Conclusion

We have conducted preliminary experiments and determined the computational challenges that we face with real-world benchmark examples. In the future, we aim to develop a simulated performance model to estimate the performance specifications and eventually develop hardware circuits on the Maxeler systems to measure actual performance observed.