Multi-FPGA based High Performance LU Decomposition

Arvind Sudarsanam, Seth Young, Thomas Hauser, Dasu Aravind.
Utah State University
asudarsanam@cc.usu.edu, sey@cc.usu.edu, thauser@engineering.usu.edu, dasu@engineering.usu.edu.

Abstract

LU Decomposition is a linear algebra routine that is used to bring down the complexity of solving a system of linear equations with multiple RHS. Its application can be found in computational physics (modeling 2-D structures), image processing, and computational chemistry (design and analysis of molecular structures). This paper investigates the hardware software co-design of large scale block-based LU decomposition algorithm on the Starbridge Hypercomputer. Results are shown for a double precision complex matrix of size 1024x1024 implemented on a system comprising of a single PC connected to 'N' FPGAs via a single PCI bus. Performance results and comparisons with a cluster will be provided at the time of presentation at the conference.

Introduction

In the recent past FPGAs seem to indicate a promise for large scale data-path parallelism on-chip and hence potential for significant speedup over conventional microprocessor systems. There have been a crop of several hardware systems on the market such as the Cray XD1, Nallatech boards, SRC machines and Starbridge Hypercomputer. The programmability of these systems also seems to have a large variety such as conventional VHDL or Viva to program the FPGAs and C/C++ to program accompanying microprocessors. But there has also been a healthy level of skepticism on how these systems will perform with scientific applications that involve floating point and complex data types. In this paper, we are investigating the hardware-software co-design of large scale LU decomposition algorithm on the Starbridge Hypercomputer. We evaluate the potential speed up the system can offer with respect to a cluster of AMD Opteron cluster, measure the compute and communication limits/tradeoffs of the system and the scientific friendliness of the GUI based synthesis environment Viva. The lessons learnt from this process are also being integrated into a system level multi-FPGA-CPU performance prediction tool to design application specific systems or evaluate suitability of off-the shelf systems for specific applications.

LU Decomposition is a matrix-based scientific application that decomposes a square matrix into two triangular matrices [1]. One of them is a lower-triangular matrix, with all the elements above the main diagonal equal to zero, and the other matrix is an upper-triangular matrix, with all the elements below the main diagonal equal to zero.

Block-based algorithms are conventionally used for parallel processing of large matrices. Linear algebra based applications that are optimized towards distributed computing make use of block-based algorithms. In such algorithms, a top-level controller breaks down the input matrix into multiple sub-matrices and assigns the processing of the different sub-matrices to different nodes. This is done due to the following reasons: (1) Each computation node may not have enough local memory to hold the entire input matrix and process it. (2) Block-based algorithms are designed such that the inter-block communication is kept to a minimum.

Block-based algorithms for LU decomposition [2] provide a large amount of inter-block parallelism. This characteristic makes it an ideal candidate for distributed computing platforms. By analyzing the computation model at the level of a single bxb block, it was found that the bulk of computations happened in a triple-nested loop, thereby making it an ideal candidate for hardware acceleration. Also, the amount of data to be transferred is $O(b^2)$, whereas the number of arithmetic operations to be executed is $O(b^3)$. This alleviates the need for a high bandwidth network to exist between memory and the hardware device. However, bandwidth limitations become a major issue for large distributed computing platforms, with multiple computing nodes connected to a single PC (number of computing nodes >> b). This paper provides an overview of the blocked LU algorithm, and discusses the co-design methodology used to implement the same. Results will include performance numbers for different input configurations (block size, matrix size etc.).

Algorithm overview

In the proposed LU decomposition implementation based on a block-based algorithm, the entire matrix is split into multiple equal-sized sub-matrices and each block is processed separately. Pseudo code for the entire algorithm is provided below.

Algorithm 1: Top-level pseudo-code

```
/* Inputs: Data matrix A[N][N], Matrix size (N), block size(b) */
1.  [A11[1][1],A12[1][N/b-1],A21[N/b-1][1],A22[N/b-1][N/b-1]]=
Partition(A,b);
2. [L11, U11, temp] = sub-step1(A11,b);
3. invL11 = inverse(L11);
4. For X = 1 to N/b-1
L21[X][1] = sub-step2(A21[X][1],U11,temp);
5. For X = 1 to N/b-1
        U12[X][1] = sub-step3(A12[X][1],invL11)
6. For X = 1 to N/b-1
        For Y = 1 to N/b-1
                A22new[X][Y] = sub-step4(A22[X][Y],
L21[X][1], U12[1][Y]);
7. N = N – b;
8. A = Combine(A22new[N/b-1][N/b-1]);
9. If (A is not empty) goto step 1.
10. Exit
```

Design Setup

Hardware acceleration of LU decomposition uses multiple hardware and software resources developed at Starbridge Systems [3]. Hardware platform used is a Hypercomputer with multiple XC2V6000 FPGAs on a single board connected to an x86 machine via PCI bus. We used the graphics-based hardware design tool Viva as the front-end and Xilinx synthesis tools as back-end to generate the bitstreams. C APIs at Starbridge systems were used to control data communication between the PC and board.

Methodology

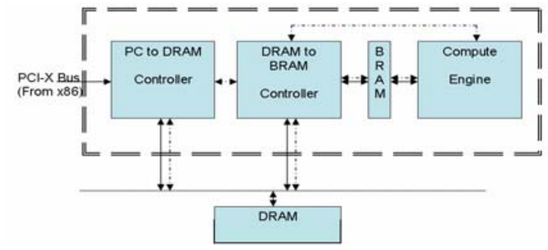

Figure 1: Top-level FPGA design.

The block-based algorithm is implemented using a C program that can handle the top-level control. This C program also handles the configuration and reconfiguration of the target FPGA hardware and data communication between the hard disk holding the input matrix and the DRAM memory of all the FPGAs. This program runs on the host PC controller and any data transfer between the host PC and the FPGA accelerator board is performed through the PCI bus. In the proposed implementation, processing of a single 'bxb' block of data is localized to a single FPGA, thereby removing any need for inter-FPGA communication. The inner-most loop is identified and the computation data path found inside this loop is implemented using FPGA hardware. Multiple data paths can be implemented based on the availability of hardware resources on the FPGA. There is considerable overhead due to the control and memory access modules that are required to support the main data paths.

Data paths found in the LU decomposition algorithm consists of double precision floating point complex arithmetic computations, and are extremely expensive in terms of hardware resources. Hence, we found it difficult to realize multiple data path units inside a single FPGA, though a lot of intra-block parallelism was available in the LU algorithm. Parallelism was explored at the inter-block level and exploited using multiple FPGA units.

The host PC controller is responsible for data to be made available at the PCI bus and for reading back the processed data from the PCI bus. Figure 1 shows the major functional components involved in our design. Such a modular design helps us to develop a general technique towards hardware design of any block-based algorithm (commonly found in matrix algebra).

PC to DRAM controller is the Viva module that handles data transfer between the PCI bus and the DRAM. Once the PC is ready to provide data on the PCI bus, it sends a control signal that is interpreted by this controller and controller is set to a state that enables it to transfer data on the PCI bus onto the DRAM. Once all the data provided by the PC is written onto DRAM, the PC to DRAM controller sends a 'Done' signal to the DRAM to BRAM controller. The DRAM to BRAM controller is now set into a state that enables it to read data from DRAM and provide it to the compute engine, and also sends a control signal to the compute engine, that enables the compute engine to fill up relevant block RAMs using the data provided by the DRAM to BRAM controller. Once all the data is written onto BRAMs, the compute engine starts processing the data. Once the processing is completed and the BRAMs are filled with the output data, the compute engine sends a 'Done' signal to the DRAM to BRAM controller and starts providing data. This process continues till all input data is processed and written back to PC memory.

Results and Analysis

The proposed design for LU decomposition can handle varying block sizes and input matrix sizes. Table 1 shows the resource utilization of the three different steps of our design for a matrix of size 1024x1024 and a block size of 16. The compute engine was implemented with a single data path available for performing the computations in the inner-most loop. As seen in Table 1, the resource utilization for steps 3 and 4 are nearly 50% of a single FPGA. Hence, it is possible to increase the number of data paths to two or three and obtain further speed-up. We are currently generating the results for varying matrix and block sizes. Results will be made available during the presentation.

Sub-step	No. of slices	No. of 18x18 multipliers	No. of BRAMs
#2	25056	39	20
#3	16277	39	28
#4	16553	39	32

Table 1: Resource utilization for LU decomposition

Summary and Future Work

This paper evaluates a multi-FPGA based system design for large-scale LU decomposition. This design is scalable to any number of FPGAs, any matrix size, or any block size. Initial results show lack of resources within a single FPGA for implementing multiple data paths (more than 3) for complex double precision numbers. Proposed system exploits inter-block parallelism to speed-up the algorithm. Future work involves optimization of the arithmetic units and use of PCI express for PC to FPGA data transfer.

References

[1] W. H. Press, et. al. Numerical Recipes in C. Cambridge University Press, 1992.

[2] Seonil Choi et. al. "Time and Energy Efficient Matrix Factorization using FPGAs", Proceedings of ICFPT, 2003.

[3] Starbridge Sytems, www.starbridgesystems.com.