DESIGN STUDY OF (2×2) CORE ARCHITECTURE FOR MATRIX MULTIPLICATIONS VIA PROGRAMMABLE GRAPH ARCHITECTURE

Jun-Hee Mun, Muling Peng, Sangjin Hong, Alex Doboli, K. Wendy Tang

Department of Electrical and Computer Engineering
Stony Brook University - SUNY, Stony Brook, NY 11794-2350

ABSTRACT

This paper presents a 2×2 core architecture for matrix multiplications via the Programmable Graph Architecture approach proposed earlier. A larger matrix-matrix multiplication can be carried out through sub-matrix decomposition. The iterative operation is completely performed with simple arithmetic operations and memory accesses. The core architecture is structurally described using *Verilog* and its functionality has been verified. Performance of the operation and factors influencing the execution are analyzed.

1. INTRODUCTION

In this paper, we consider the Programmable Graph Architecture (PGA) [1] approach for matrix multiplications based on previous work on Cayley Graph theory [2]. The details of the PGA approach can be found in [1]. The following is a brief summary.

The operation is based on two basic assumptions: (1) Given two $N \times N$ matrices where N is a power of 2. (2) Elements of matrices are integers. Performing the matrix multiplications involves several steps. First a larger matrix multiplication is iteratively broken down into a set of 2×2 matrix multiplications. The operation can be used with the other small unit of matrices, such as 4×4, 8×8. A matrix multiplication is further broken down iteratively to a set of multiplications of singular and/or non-singular Z_p matrices through modular p arithmetic [2]. For example, a 2×2 matrix multiplication is given by

$$AB = \begin{bmatrix} 40 & 20 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 2 & 9 \\ 41 & 16 \end{bmatrix} \quad (1)$$

where

$$A = \begin{bmatrix} 10 & 5 \\ 1 & 1 \end{bmatrix} p + \begin{bmatrix} 0 & 0 \\ 1 & 2 \end{bmatrix} \quad (2)$$

$$B = \begin{bmatrix} 0 & 2 \\ 10 & 4 \end{bmatrix} p + \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}. \quad (3)$$

Then,

$$AB = (Q_A Q_B p + Q_A R_B + R_A Q_B)p + R_A R_B \quad (4)$$

where $Q_A Q_B$, $Q_A R_B$, $R_A Q_B$, and $R_A R_B$ can be computed in parallel. These four matrices are further broken down to Z_p matrices. Again, all products are computed in parallel. When the break down process is completed, all matrices are either singular or non-singular Z_p matrices. For singular matrix multiplications, each matrix is expressed as sum or difference of non-singular matrices. Then for all non-singular matrix multiplications, we use the Programmable Graph Architecture (PGA) [1] approach.

This work has been supported by the NSF under award CCR-0325584.

For illustration of a multiplication of Z_p matrices via Cayley Graph, let's consider $p = 3$, $n = 48$ nodes where

$$V = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \cdots \right\}, \quad (5)$$

then, 4 generators (it can be other even numbers) are picked as

$$G_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, G_2 = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}, G_1^{-1} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}, G_2^{-1} = \begin{bmatrix} 0 & 2 \\ 1 & 1 \end{bmatrix} \quad (6)$$

for GCR representation with $q = 6$ classes.

The advantage of the PGA approach is that when it is applied to matrix multiplication, all of the calculations can be simplified to table-look-up operations where these tables can be pre-stored in memory. Because every characteristic matrix multiplication can be considered as a route from one vertex to the other through certain vertices on the graph, we have to map each matrix to a specific integer representing the vertex on Cayley graph. Then they can be utilized to find the solution through routing and the result is simply addition and subtraction of these characteristic matrices multiplied by fixed power of p. The number of integers, represented as n is equal to that of vertices on Cayley graph which is a function of parameter p, then $n = (p^N - 1)(p^N - p)(p^N - p^2) \cdots (p^N - p^{N-1})$ where N is the order of the matrix.

2. PROGRAMMABLE GRAPH ARCHITECTURE

2.1. Core Operation

Figure 1 illustrates the main operation of the matrix multiplications. The inputs to this core process are sub-matrices multiplications that are obtained from the decomposition. The core operation involves accessing 5 tables. Two input matrices, A and B, are mapped to integers through the Integer Mapping Table, Table 1, and generate two input integers $\#A$ and $\#B$. The number of entries of this table is n. Then, the core operation performs two basic operations. The first operation generates partial results from a sub-matrices multiplication based on the routing path. This is an iterative operation and becomes the critical path of the core operation. The other operation generates a remainder, which will be subtracted from the results of the routing process before obtaining the result of the multiplication.

Initially, the routing path is obtained with $\#B$ by referencing the Routing Table, Table 2. The number of entries in this table is n. Each entry corresponds to a routing path, which contains a set of generator matrices. Both #A and #B are also used to look up the Table 5, which is not shown in the figure containing the pre-calculated matrices $A \times Q_B \cdot p$ (i.e., used on the *Verilog* model to simplifies the process). This table has n^2 entries and the entry in this table corresponds to a remainder matrix that was mentioned

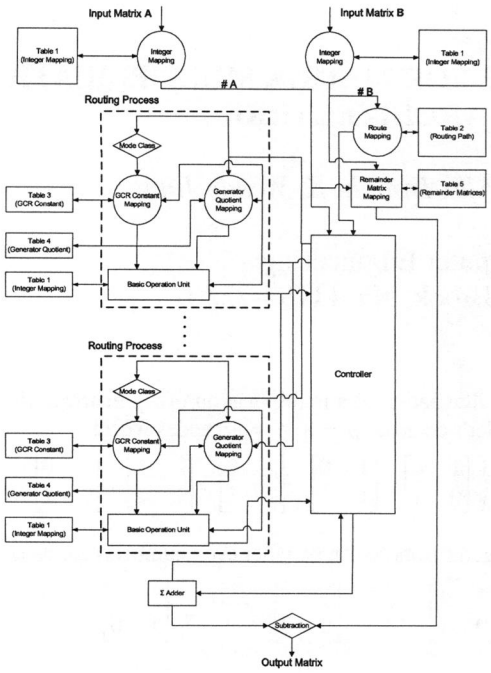

Figure 1: Illustration of matrix multiplication of decomposed matrices.

above. Generally the routing process takes certain iterations (i.e., the number of generators), therefore the remainder matrix from Table 5 will be stored and wait until the entire routing process is done.

The operation within the dotted part in Figure 1 illustrates one iteration of the routing process. The inputs of the process are the $\#A$ and a set of generators matrices. First, the input integer number (which the initial number is $\#A$) will go through a modulo q operation to obtain the remainder. The divisor, q, corresponds to the number of classes of the GCR (General Chordal Ring) representation [2]. The remainder and an input generator become the reference to access the GCR constant in Table 3. The number of entries in this table is qD where D is the number of generators as illustrated in Eq. (6). Meanwhile, the input integer and the generator are also used to access the Quotient Matrices of Generators Table, Table 4. The number of entries in this table is nD. In the routing process, the Quotient Matrix will be checked to see if it is singular. A singular matrix will be decomposed into two non-singular matrices. The Quotient Matrix will be mapped back to integer through the Integer Mapping Table. The GCR constant will be combined with the input integer and then divided by n to obtain the remainder which represents another vertex in integer.

Every routing process will multiply the Quotient Matrix by a scaling factor p. The routing process generates three partial results per iteration where they are collected and accumulated by the $\Sigma Adder$. Before the accumulation, these partial results are transformed back to matrices. The result of $\Sigma Adder$ is then subtracted with the remainder matrix from the Table 5 and the result of this matrix is the result of sub-matrices multiplication. The results are composed for the final output.

2.2. Core Processing Unit

Figure 2 shows the architecture of the Core Processing Unit. The Core Processing Unit consists of Input Transformation Module, Routing Process Module, Controller, and Output Generation Module. The entire unit takes two inputs *InputA* and *InputB*, and generates *Output*. Both Input Transformation Module and Routing Process Modules have several memory look-up tables and simple arithmetic and logic operators. Note that no multiplier is used.

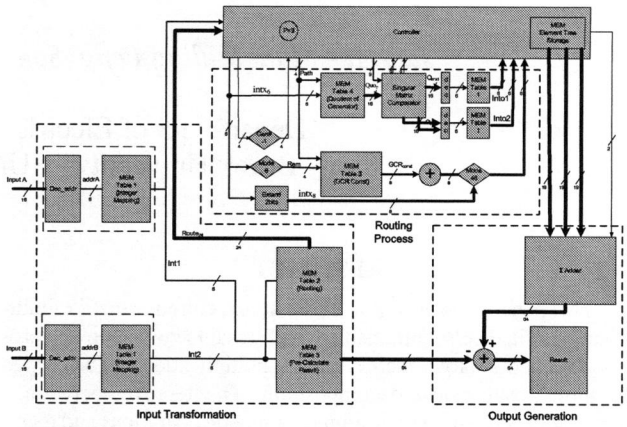

Figure 2: Illustration of Core Processing Unit architecture.

3. DESIGN EVALUATION

We evaluate complexity of the architecture as a function of the matrix elements. We investigate how many sub-matrices multiplications are needed as a function of elements. Each element is randomly selected and the number of decomposed multiplications is plotted for 4-bit, 8-bit, and 16-bit elements. Figure 3 illustrates the number of decomposition process represented by Eq. (4). In the figure, three bars correspond to the number of singular cases, the average Core Process Unit used, and the maximum possible number of Core Process Unit used. The computational requirement is analyzed for 8-bit, 12-bit, and 16-bit element sizes. The values of the elements in the matrix are randomly generated. There are significant numbers of the core process computation due to singular matrices. The percentage of the singular cases can be reduced for larger value of p. However, the core processing unit will be larger when the value of p increases.

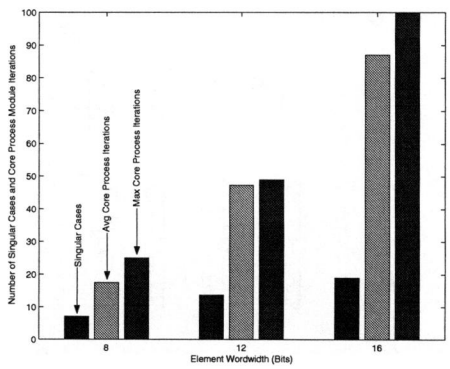

Figure 3: Illustration of the computation requirements.

4. REFERENCES

[1] K.W. Tang and A. Y. Oruc, Programmable Graph Architecture for Matrix Operations, the 37th Annual Conference on Information Sciences and Systems, Johns Hopkins University, Baltimore, Maryland, on March 12-14, 2003.

[2] B. W. Arden and K. W. Tang, "Representations and Routing of Cayley Graphs," *IEEE Transactions on Communications*, 39(11):1533-1537, November, 1991.