A Review of Different Methods for Matrix Multiplication Based on FPGA

AKSHITA MISHRA, HEMIKA YADAV, SARITA RANI, SHIVANISAXENA
M.TECH VLSI Design, Banasthali Vidyapith, Rajasthan
Department of Electronics, Banasthali Vidyapith
akshitamishra27@gmail.com, hemikayadav90@gmail.com, saritarathi19@yahoo.in
shivani.19saxena@gmail.com

ABSTRACT
In this paper, we present the design methods and Field Programmable Gate Array (FPGA) implementation of matrix multiplier in image and signal processing applications. This paper gives the detailed review of matrix product implementation on FPGA where area, energy, speed, power dissipation, latency and time efficient methods are compared for different Xilinx families. Here we are also presenting parallel architecture and parameterized system for matrix product used in FPGA. In this way, this literature will help in making performance characterization to implement high performance matrix multiplier on FPGA.

Keywords: ASIC, distributed arithmetic, FPGA, Systolic architecture.

[1] INTRODUCTION
Matrix multiplication is a fundamental computation intensive operation used in many algorithms in scientific computation. Coding, digital signal processing, image processing, graphics, robotics and various control algorithms like adaptive control, model predictive control (MPC) uses matrix operations extensively. It is the core engine of various algorithms used in applications. For improving the performance of these applications efficient matrix multiplication is required. The goal of this paper is develop efficient architectures ideally suited for fast computation of matrix multiplication using an FPGA based parametriables system and parallel architecture. Also, systolic architecture (SA) and distributed arithmetic (DA) design methodologies have been discussed for implementation of matrix product [3]. The importance of these two approaches have been described and shown that DA approach provides better performance in terms of speed and area with SA approach. For high performance application, matrix multiplication operation must be realized by parallel architecture based on FPGA [5].

[2] ROLE OF FPGA IN MATRIX MULTIPLICATION
Traditionally, matrix multiplication operation can be either realized by software on fast processor or dedicated hardware like ASIC (Application Specific Integrated Circuit). The software based matrix multiplication is slower and can become bottleneck in overall system operation. But hardware based FPGA design of matrix multiplier provides speedy computation time and flexibility as compare to software and ASIC base approaches respectively. FPGA has become a platform of choice for hardware realization of computation intensive applications and have been improved considerably in terms of speed, density and functionality which make them ideal for system on a programmable chip design for wide range of applications. FPGA technology allows easy reprogram ability less development time with respect to full custom VLSI design as a result being time efficient and resource competitive. The main motive for using FPGA is to reuse the resource and input the data in serial fashion from which hardware cost can be dramatically decreased. But more time for computation will be needed. This method provides a new way to improve the module in order to save the resources (energy, area, speed, power dissipation and latency). The basic idea is to reuse the RAMs and MACs. The control logic is needs to select input line or column by which less hardware is required but time taken for computation will be more.

[3] PERFORMANCE METRICS
FPGA based designs are usually evaluated using following performance metrics: energy efficiency and latency, area, power dissipation and speed.

[3.1] Energy efficiency and latency
FPGAs can multiply two n x n matrices with both lower energy and lower latency than the other types of devices. This makes FPGA the ideal choice for matrix multiplication in various applications. To calculate energy consumption in n x n matrix multiplication, the first task is to calculate the latency. In each case latency is calculated first in cycles and then converted to seconds by dividing the clock frequency. Once the latency is calculated we can use it and the power dissipation of each device to calculate the consumption of energy. Each device requires different method for performing the mentioned two tasks. The latency of the design in [11] is 0.57µs while the other design takes 0.15µs using 18% less area [10]. The algorithm computes the matrix product in parallel.
efficiently in terms of latency and energy, by cleverly moving the entries of input matrix through the linear array.

The energy for performing matrix multiplication is given by:

\[ E = L( pP_M + p(p/16)P_S + 4P_R + 2P_{IO} + P_{IC}) \]

Latency can be calculated from [10]

Where L is the latency when PEs are used; \( P_M \), \( P_S \), \( P_R \), \( P_{IC} \) and \( P_{IO} \) are powers used by a multiplier, an on chip memory cell, a register, the interconnection network and the data transfer respectively. Values obtained by performing low level simulation of Virtex II FPGA are following:

Table 1. Energy parameters in terms of power [10]

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>POWER(mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_M )</td>
<td>17.00</td>
</tr>
<tr>
<td>( P_S )</td>
<td>8.39</td>
</tr>
<tr>
<td>( P_R )</td>
<td>2.34</td>
</tr>
<tr>
<td>( P_{IO} )</td>
<td>11.31</td>
</tr>
<tr>
<td>( P_{IC} )</td>
<td>10.00</td>
</tr>
</tbody>
</table>

[3.2] Area

Area is of prime concern for matrix multiplication based on FPGA as compactness is widely demanded in the upcoming technology. This literature survey talks about the FPGA family which consumes lesser area than the previous families. The architecture proposed in this paper was targeted to Xilinx XC2000E of Virtex–E FPGA family. This design improves the previous designs in [12] and [13] in terms of area metrics. The calculated area matrix for the designs in [12],[13] and [14] are 14.45,4.93 and 2.35 respectively. For a 4×4 matrix multiplication which is least of the above three. The performance improvements tend to grow with the problem size. The area of this design is smaller by 11%-46% compared with the best known systolic design (basic principle by replacing a single PE by a regular array of PEs and by carefully orchestrating the flow of data between the PEs, high throughput can be achieved without increasing memory bandwidth requirement) based on [1] with the same latency of the sizes 3×3 - 12×12. Partially reconfigurability feature was exploited for the first time for computation of matrix multiplication by Jianwen et al. in [15] partially reconfigurable devices offer the possibility of changing the design implementation without stopping the whole execution process. The design was evaluated in terms of latency and area and it was found that area is reduced by 72% -81% for the matrix sizes 3×3 - 48×48 as compared to [16] and the performance further improves for larger matrices.

[3.3] Power Dissipation

It has been increasingly important that the systems are also energy efficient and consume low power. There are two types of power consumption in FPGAs: static and dynamic. We will consider dynamic power only. Dynamic power dissipation is caused by signal transitions in the circuits. A higher operating frequency leads to more frequent signal transitions and results in increased power dissipations. In FPGA devices, major chunk of power are consumed by the programmable interconnects, while the remaining power is consumed by the clocking, logic and I/O blocks.[6,8,9] Choi et al. developed new designs and architectures for FPGAs which minimize the power consumption along with the latency and area[1]. They used linear systolic architecture to develop energy efficient design. For linear array architecture the amount of storage per processing element affects the system with wide energy. Thus, they use maximum amount of storage per processing element and minimum number of multipliers to obtain energy efficient matrix multiplier.

The below figure shows comparison of power consumption and performance for different Xilinx families. Fig 1 shows there is a tradeoff in between power and performance. Contribution of different types of power in various technologies of Xilinx (45nm and 28nm), are shown in fig 2.
The most important performance metrics for FPGA based designs is the speed. It has become increasingly important that the operation of matrix multiplication should be fastest. Therefore speed must be as high as possible[1].

In this literature we will consider speed as an important parameter. DA based design provides better performance in terms of speed and area as compared to SA based design. The I/O bandwidth required by the design is directly proportional to the problem size. The design presented in [3,17] were restricted to small matrix size. For multiplying large matrices (n=128,256 and 512), Bensaali et al. design an FPGA based co-processor[4]. The designed co-processor first partitions the input matrices into smaller sum matrices and then calculates the product.

The latency is defined as the time between reading the first element from input matrices, A and B, and writing the first element C to the result matrix. The total computation time is the time elapsed between reading the first element from the input matrices A and B, and writing the final result matrix element C to memory, as shown in fig 3. In order to optimise FPGA architecture resource use, the data from the input matrices A and B should be reused.

Optimal data reuse when data is read from memory for matrices A and B exactly works. By simultaneously reading one column of matrix A and one row of matrix B, and performing all multiplying operations based on those values before additional memory reads, optimal data reuse occurs. Data read in this sequence allows one partial term of every element in output matrix C to be computed per clock cycle.

TABLE OF CONTENTS

[3,4] Speed
The most important performance metrics for FPGA based designs is the speed. It has become increasingly important that the operation of matrix multiplication should be fastest. Therefore speed must be as high as possible[1].

In this literature we will consider speed as an important parameter. DA based design provides better performance in terms of speed and area as compared to SA based design. The I/O bandwidth required by the design is directly proportional to the problem size. The design presented in [3,17] were restricted to small matrix size. For multiplying large matrices (n=128,256 and 512), Bensaali et al. design an FPGA based co-processor[4]. The designed co-processor first partitions the input matrices into smaller sum matrices and then calculates the product.

The latency is defined as the time between reading the first element from input matrices, A and B, and writing the first element C to the result matrix. The total computation time is the time elapsed between reading the first element from the input matrices A and B, and writing the final result matrix element C to memory, as shown in fig 3. In order to optimise FPGA architecture resource use, the data from the input matrices A and B should be reused.

Optimal data reuse when data is read from memory for matrices A and B exactly works. By simultaneously reading one column of matrix A and one row of matrix B, and performing all multiplying operations based on those values before additional memory reads, optimal data reuse occurs. Data read in this sequence allows one partial term of every element in output matrix C to be computed per clock cycle.

[4] METHODOLOGY
Design methodology for the hardware realization of computation intensive algorithm is a combined effort of EDA tools, methods and FPGA technology that enables to produce the optimise circuit for many applications. A perfect combination of FPGA hardware, designed IP core and EDA tools will definitely enhance the efficiency of the design methodology. By design methodology, we imply the step by step process of FPGA design. The FPGA design methodology is used as a guideline for the hardware realisation of algorithms. The EDA tools the Xilinx integrated software environment (ISE), Altera’s, Quartus II and Mentor Graphics FPGA advantage plays a very important role in obtaining and optimised digital circuit using FPGA. A design methodology for
Indexing in Process - EMBASE, EnCARE, Electronics & Communication Abstracts, SCIRUS, SPARC, GOOGLE Database, EBSCO, NewJour, Worldcat, DOAJ, and other major databases etc.,