# Introduction to FPGAs - Matrix Multiplication

Philip Leong & Duncan Moss

July 7, 2014

## 1 INTRODUCTION

Matrix Multiplication is one of the principle concepts of Linear Algebra with its applications crucial to the fields of finance, science, engineering and medicine. The aims of this tutorial are:

1. Develop a practical and intuitive understanding of hardware designs
2. Practise translating mathematical descriptions into hardware designs
3. Explore the area and performance trade-offs when developing custom hardware
4. Develop a Matrix-Vector and Matrix-Matrix Multiplier

Golub .et .al published 'Matrix Computations' in 1996 and is considered the industry standard when representing matrix computations both mathematically and algorithmically [1].

### 1.1 VECTOR NOTATION & DOT PRODUCT

An $n$ dimensional vector can be represented as :

$$\mathbf{x} \in \mathbb{R}^n \quad \Longleftrightarrow \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad x_i \in \mathbb{R} \tag{1.1}$$

Additionally, The *dot product* (or *inner product*) of two $n$-vectors involves $n$ multiplication and $n$ additions. Concretely, the elements of each vector are multiplied pair-wise then accumulated to obtain the final answer. This has several different mathematical representations.

$$c = \mathbf{x}^T \mathbf{y} \tag{1.2}$$

$$c = \sum_{i=1}^{n} x_i y_i \tag{1.3}$$

Algorithmically this can be represented as:

```
c = 0
for i = 1:n
    c = c + x(i)y(i)
end
```

## 1.2 Matrix-Matrix Multiplication

A matrix is a collection of *nxm* elements in row column format.

$$A \in \mathbb{R}^{nxm} \iff A = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix} \quad x_{ij} \in \mathbb{R} \tag{1.4}$$

There are several ways to represent matrix-matrix multiplication, for this tutorial we will be considering the *dot product* formulation. First consider a 2-by-2 matrix-matrix multiplication:

$$\begin{aligned} AB = C \iff & A \in \mathbb{R}^{2x2} \ B \in \mathbb{R}^{2x2} \ C \in \mathbb{R}^{2x2} \\ & \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix} \end{aligned} \tag{1.5}$$

Generalising this we produce:

$$AB = C \iff A \in \mathbb{R}^{mxp} \ B \in \mathbb{R}^{pxn} \ C \in \mathbb{R}^{mxn} \tag{1.6}$$

Notice that each element of the resulting matrix involves a *dot product*. Algorithmically, this is represented as:

```
for i = 1:m
    for j = 1:n
        c_ij = a_i^T b_j + c_ij
    end
end
```

Alternativily, the algorithm can be represnted in the $ijk$-format:

```
for i = 1:n
    for j = 1:p
        for k = 1:m
            C(i,j) = A(i,k)B(k,j) + C(i,j)
        end
    end
end
```

## 2 TUTORIAL QUESTIONS

In answering these questions, marks will be awarded not only for correctness but also understandability and elegance of the solution.

- Dot Product Block (15%): Complete the provided *dot product* block. There is a C program provided as a reference as well as a testbench which will verify the correctness of your block.
- Parallel Matrix Multiplication Block (30%): Using the *dot product* block you made in part 1, make a parallel implementation of a Matrix Multiplication Block. i.e., each *dot product* runs in parallel. As with the previous part there is a C program and testbench.
- Serial Matrix Multiplication Block (30%): Now transform the parallel Matrix Multiplication Block to a serial implementation. This requires that only a single *dot product* computation runs each cycle.
- Area and Performance Measure (25%): Take the 3 architectures developed in parts 1,2 and 3 and generate the synthesis report to extract the area and timing values.

## REFERENCES

[1] Gene H Golub and Charles V. Van Loan. *Matrix Computations*, volume 3rd. 1996.