UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Processor Design
(PD)

Microprocessor Selection

1st Report

Jordi Solà, Ying hao Xu

Contents

1 Introduction

The objective of this session is to select an available microprocessor (either written in Verilog or VHDL) to use throughout the course.

In our case, instead of using a currently available processor, we will build a standalone accelerator based on a systolic array. As part of the testing environment, stub modules will also be developed in order to test the accelerator.

2 Define Selection Criteria

Selection criteria:

- **RTL Language:** SystemVerilog
- **Accelerator communication:** Custom protocol (memory mapped)
- **Testing environment:** Modelsim Free Version and Verilator.

The reason why we are choosing SystemVerilog over VHDL is due a pure language syntax preference (Verilog is C-like syntax and less verbose). On the other hand, VHDL is strongly typed and very deterministic compared to Verilog, that is why instead of using Verilog we are going to use SystemVerilog, which can be seen as an extension of Verilog, adding enhanced procedural blocks (that ensure that only the intended type of behavior occurs), new data types (e.g. structs) or module interfaces.

3 Define Baseline Accelerator

We are going to built a systolic array targeting Matrix-Matrix multiplication. The data type we are going to use is int8 in order to simplify the implementation.

Systolic arrays were invented in 1978, described as an homogeneous network of tightly coupled data processing units (nodes). Each node computes a partial result using data coming from upstream neighbors and the result is passed downstream.

Nowadays, systolic arrays became very popular as they fit really well as an accelerator for machine learning workloads, however, the main pros and cons are:

Pros:

- Higher computation throughput without increasing memory bandwidth.
- Scalable design.
- Less inefficient data movements.

Cons:

- Highly specialized.
- Not all algorithms can be mapped into a systolic array.
- Higher silicon area due its distributed computation nature.

A very popular accelerator which uses a systolic array as the basic computation element is Google's TPU and achieves really high speedup compared to a CPU or a GPU (15X - 30X)[1].

4 Systolic array introduction

Systolic arrays were introduced by Kung *et al.* in 1978[2], these systems are composed of several small processing elements (PEs) interconnected collaborating in order to perform different computations in a highly parallel way. The control is only involved in the injection timings of the data and the data flows through the computing units in rhythmic pulsations, hence the analogy with the human heart systoles.

There are several implementations of PEs and different interconnections, depending on the problem you want to solve, as far as we have found, the list covers:

- Hexagonal Arrays:
  - LU decomposition[2]
  - Matrix multiply[2]
- Binary Trees:
  - Sorting[3]
- Torus
  - Transitive closure[4]
- Linear Arrays
  - Convolution[5]
  - Correlation[6]
  - Fast Fourier Transform[7]
  - Matrix-Vector multiplication[2]

- Rectangular Arrays
  - Matrix multiplication[2]

On each interconnection pattern, the internal PEs are fed by their neighbours and it's the boundary ones those who manage the input/output of the array.

4.1 Bidirectional Linear Systolic Arrays

As an introduction, we will try to explain a basic linear array, used for Matrix-Vector multiplication, in a configuration called bidirectional linear systolic array (BLSA). It was introduced by Kung *et al.* in their original publication in 1978 [0].

Figure 1 shows a BLSA processing element (PE) interfaces.

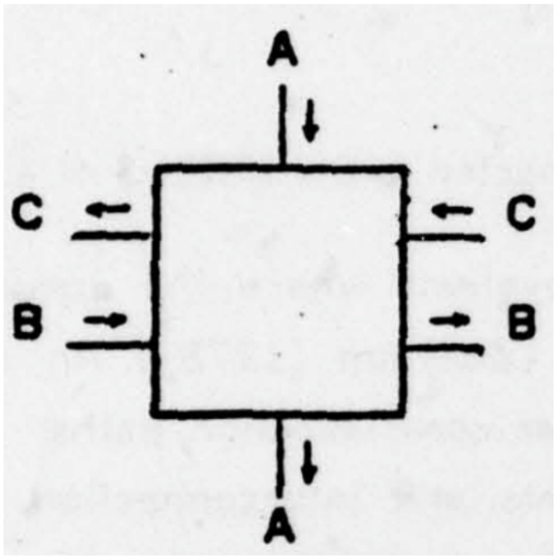


Figure 1: BLSA Processing Element (PE)

Each of the PEs performs the same computation, and can be modeled as:

$$A_{out} = A_{in}$$
$$B_{out} = B_{in}$$
$$C_{out} = C_{in} + A_{in} \times B_{in}$$

For the BLSA configuration, we connect those processing elements linearly, one next to the other, in a way in which both ports C and B traverse all the cells, with one entry and the exit of both ports in the opposite side of the array, while ports A are left as a direct access per each PE, and not producing any output, like shown in Figure 2.

In this array, the components of the final result are initialized as 0 and fed from the rightmost PE through the C port and traverse the array until they get to the leftmost PE, which will output the final computed components. On the other hand, the components of the vector to be operated are fed from the leftmost PE and traverse to the rightmost PE (without being modified) and the components of the matrix are fed diagonal by diagonal from the top of each PE.

The biggest limitation of this configuration is that the maximum computable diagonal length accepted by the system is predetermined by the amount of computing elements in the system.

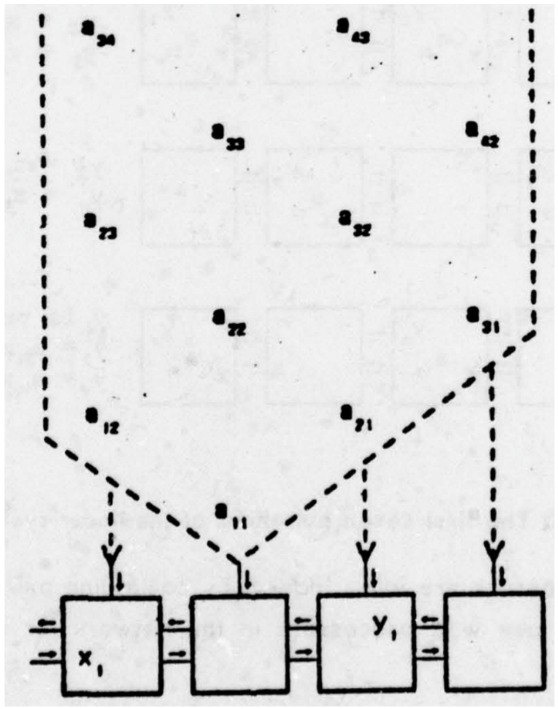


Figure 2: BLSA Processing Element connection distribution

So, as an illustration, if we want to compute the following multiplication:

$$\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} o_0 \\ o_1 \\ o_2 \end{bmatrix}$$

$$o_0 = a_0 \times w_{00} + a_1 \times w_{01} + a_2 \times w_{02}$$
$$o_1 = a_0 \times w_{10} + a_1 \times w_{11} + a_2 \times w_{12}$$
$$o_2 = a_0 \times w_{20} + a_1 \times w_{21} + a_2 \times w_{22}$$

With this BLSA array:

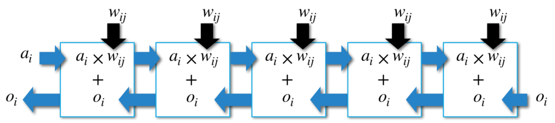
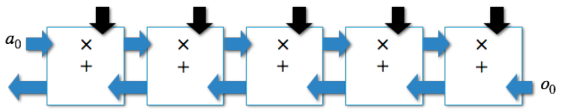


Figure 3: BLSA array

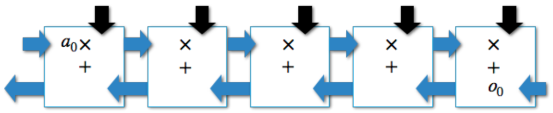
We would have to perform the following steps:

Cycle 0:



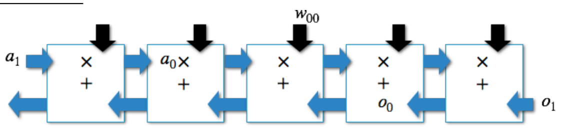
$$o_0 =$$
$$o_1 =$$
$$o_2 =$$

Cycle 1:



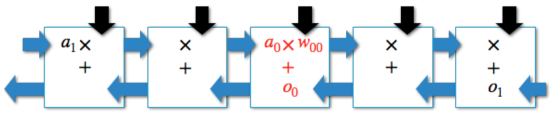
$$o_0 =$$
$$o_1 =$$
$$o_2 =$$

Cycle 2:



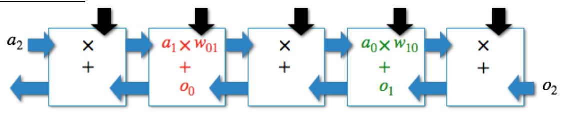
$$o_0 =$$
$$o_1 =$$
$$o_2 =$$

Cycle 3:



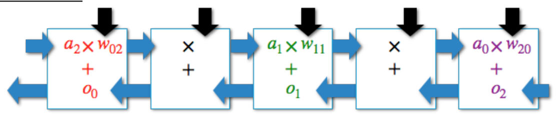
$$o_0 = a_0 \times w_{00}$$
$$o_1 =$$
$$o_2 =$$

Cycle 4:



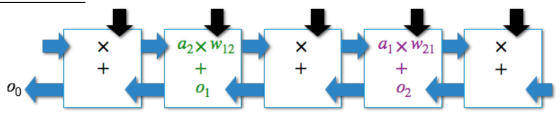
$$o_0 = a_0 \times w_{00} + a_1 \times w_{01}$$
$$o_1 = a_0 \times w_{10}$$
$$o_2 =$$

Cycle 5:



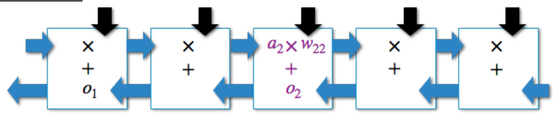
$$o_0 = a_0 \times w_{00} + a_1 \times w_{01} + a_2 \times w_{02}$$
$$o_1 = a_0 \times w_{10} + a_1 \times w_{11}$$
$$o_2 = a_0 \times w_{20}$$

Cycle 6:



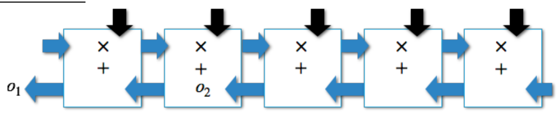
$$o_0 = a_0 \times w_{00} + a_1 \times w_{01} + a_2 \times w_{02}$$
$$o_1 = a_0 \times w_{10} + a_1 \times w_{11} + a_2 \times w_{12}$$
$$o_2 = a_0 \times w_{20} + a_1 \times w_{21}$$

Cycle 7:



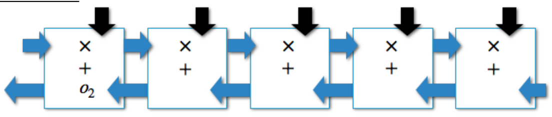
$$o_0 = a_0 \times w_{00} + a_1 \times w_{01} + a_2 \times w_{02}$$
$$o_1 = a_0 \times w_{10} + a_1 \times w_{11} + a_2 \times w_{12}$$
$$o_2 = a_0 \times w_{20} + a_1 \times w_{21} + a_2 \times w_{22}$$

Cycle 8:



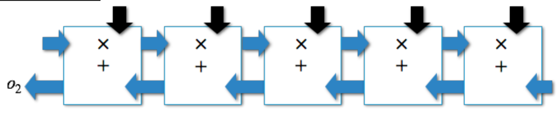
$$o_0 = a_0 \times w_{00} + a_1 \times w_{01} + a_2 \times w_{02}$$
$$o_1 = a_0 \times w_{10} + a_1 \times w_{11} + a_2 \times w_{12}$$
$$o_2 = a_0 \times w_{20} + a_1 \times w_{21} + a_2 \times w_{22}$$

Cycle 9:



$$o_0 = a_0 \times w_{00} + a_1 \times w_{01} + a_2 \times w_{02}$$
$$o_1 = a_0 \times w_{10} + a_1 \times w_{11} + a_2 \times w_{12}$$
$$o_2 = a_0 \times w_{20} + a_1 \times w_{21} + a_2 \times w_{22}$$

Cycle 10:



$$o_0 = a_0 \times w_{00} + a_1 \times w_{01} + a_2 \times w_{02}$$
$$o_1 = a_0 \times w_{10} + a_1 \times w_{11} + a_2 \times w_{12}$$
$$o_2 = a_0 \times w_{20} + a_1 \times w_{21} + a_2 \times w_{22}$$

Due to the cycles spent on the placement of the initial coefficients and acquisition of the final one (which cost us 3+3=6total cycles), the computation took more cycles than it theoretically needed, leaving the effective computation cycles to 5. In fact, considering that there are empty slots, it will be possible to perform another parallel computation interleaved with this one, thus doubling the throughput of the system.

Even though the scaling of this configuration is not really optimal for straightforward dense matrix multiplications, it really shows its potential when operating with banded matrices with a band width of $n - 1$ elements (being n the amount of PEs in the system). For this type of matrices, BLSA can perform $n/2$ sub-fma operations of the same matrix-vector multiplication per cycle after the initialization cycles and before the final phase.

Given those specifications and limitations, it looks optimal to transform dense matrices into banded matrices in order to fully exploit the potential of systolic arrays, as suggested by Navarro *et al.*[8][9] which we will try to exploit further into the project.

References

[1] Norman P Jouppi et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit". In: (2017). DOI: `10.1145/3079856.3080246`. URL: `https://doi.org/10.1145/3079856.3080246`.

[2] H.T. Kung and C.E. Leiserson. "Systolic Arrays for VLSI". In: *Sparse Matrix Proceedings* (1978).

[3] Lang et al. "Systolic Sorting on a Mesh-Connected Network". In: *IEEE Transactions on Computers* C-34.7 (July 1985), pp. 652–658. ISSN: 0018-9340. DOI: `10.1109/TC.1985.1676603`. URL: `http://ieeexplore.ieee.org/document/1676603/`.

[4] H.-W. Lang. "Transitive closure on an instruction systolic array". In: *[1988] Proceedings. International Conference on Systolic Arrays.* IEEE Comput. Soc. Press, pp. 295–304. ISBN: 0-8186-8860-2. DOI: `10.1109/ARRAYS.1988.18070`. URL: `http://ieeexplore.ieee.org/document/18070/`.

[5] Ivan Z. Milentijević, Mile K. Stojčev, and Dejan M. Maksimović. "Configurable digit-serial convolver of type F". In: *Microelectronics Journal* 27.6 (Sept. 1996), pp. 559–566. ISSN: 0026-2692. DOI: `10.1016/0026-2692(95)00115-8`. URL: `https://www.sciencedirect.com/science/article/pii/0026269295001158`.

[6] U. Peisl. "Image correlation using a bit level systolic array". In: *1988., IEEE International Symposium on Circuits and Systems.* IEEE, pp. 2689–2693. DOI: `10.1109/ISCAS.1988.15494`. URL: `http://ieeexplore.ieee.org/document/15494/`.

[7] Izidor Gertner and Moshe Shamash. "VLSI Architectures for Multidimensional Fourier Transform Processing". In: *IEEE Transactions on Computers* C-36.11 (Nov. 1987), pp. 1265–1274. ISSN: 0018-9340. DOI: `10.1109/TC.1987.5009467`. URL: `http://ieeexplore.ieee.org/document/5009467/`.

[8] J. J. Navarro et al. "Computing size-independent matrix problems on systolic array processors". In: *ACM SIGARCH Computer Architecture News* 14.2 (June 1986), pp. 271–278. ISSN: 01635964. DOI: `10.1145/17356.17388`. URL: `http://portal.acm.org/citation.cfm?doid=17356.17388`.

[9] J.J. Navarro, José M. Llaberia, and Mateo Valero. "Partitioning: An Essential step in mapping algorithms into systolic array procedures". In: (). URL: `https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/stamp/stamp.jsp?tp=&arnumber=1663622`.