Final Project: Systolic Array Matrix Multiplier

Finally we made to this point. Make sure you read the requirements carefully before proceeding to the layout.

In this final project, we are going to build a fun yet essential component with a technique that has been sleeping for decades. Systolic array was introduced in the late 1970s. It is very powerful performing linear algebra computations, such as matrix multiplication and vector convolution. It was not widely used in the past ten years due to the complexity to efficiently run general programs on such architecture.

However, thanks to the fact that neural network operations involve large amount of repeating multiplication and accumulation, systolic array finds where it shines again. It has been brought back to people's attention recently by the tech giant, Google, who designed the well known TPU AI accelerator. Google even called it "The heart of the TPU" in their technical blog.

Systolic Array

Systolic Array consists of an "array" of computational nodes, which we call them Data Processing Unit, or in short, DPU. Each DPU processes the data in exactly the same way. They are capable of receiving data from the left and top neighbors and pass it to the right and bottom DPUs. Of course, data passing in other directions is possible. Some systolic array have data connection in the diagonal direction.

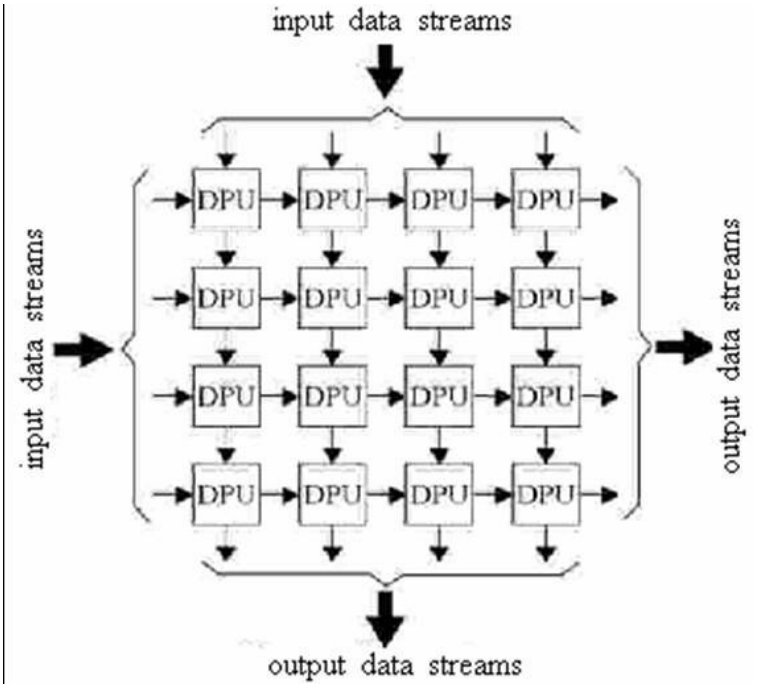

Figure 1 An example systolic array.

Matrix Multiplication using Systolic Array

Let us take a quick recap of a two 4x4-matrix multiplication. Figure 2 shows the operation break down. As we can see, each output consists of four multiplications and four accumulations.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

$$\begin{aligned}
c_{11} &= a_{11} \bullet b_{11} + a_{12} \bullet b_{21} + a_{13} \bullet b_{31} + a_{14} \bullet b_{41} \\
c_{12} &= a_{11} \bullet b_{12} + a_{12} \bullet b_{22} + a_{13} \bullet b_{32} + a_{14} \bullet b_{42} \\
c_{13} &= a_{11} \bullet b_{13} + a_{12} \bullet b_{23} + a_{13} \bullet b_{33} + a_{14} \bullet b_{43} \\
c_{14} &= a_{11} \bullet b_{14} + a_{12} \bullet b_{24} + a_{13} \bullet b_{34} + a_{14} \bullet b_{44} \\
c_{21} &= a_{21} \bullet b_{11} + a_{22} \bullet b_{21} + a_{23} \bullet b_{31} + a_{24} \bullet b_{41} \\
c_{22} &= a_{21} \bullet b_{12} + a_{22} \bullet b_{22} + a_{23} \bullet b_{32} + a_{24} \bullet b_{42} \\
c_{23} &= a_{21} \bullet b_{13} + a_{22} \bullet b_{23} + a_{23} \bullet b_{33} + a_{24} \bullet b_{43} \\
c_{24} &= a_{21} \bullet b_{14} + a_{22} \bullet b_{24} + a_{23} \bullet b_{34} + a_{24} \bullet b_{44} \\
c_{31} &= a_{31} \bullet b_{11} + a_{32} \bullet b_{21} + a_{33} \bullet b_{31} + a_{34} \bullet b_{41} \\
c_{32} &= a_{31} \bullet b_{12} + a_{32} \bullet b_{22} + a_{33} \bullet b_{32} + a_{34} \bullet b_{42} \\
c_{33} &= a_{31} \bullet b_{13} + a_{32} \bullet b_{23} + a_{33} \bullet b_{33} + a_{34} \bullet b_{43} \\
c_{34} &= a_{31} \bullet b_{14} + a_{32} \bullet b_{24} + a_{33} \bullet b_{34} + a_{34} \bullet b_{44} \\
c_{41} &= a_{41} \bullet b_{11} + a_{42} \bullet b_{21} + a_{43} \bullet b_{31} + a_{44} \bullet b_{41} \\
c_{42} &= a_{41} \bullet b_{12} + a_{42} \bullet b_{22} + a_{43} \bullet b_{32} + a_{44} \bullet b_{42} \\
c_{43} &= a_{41} \bullet b_{13} + a_{42} \bullet b_{23} + a_{43} \bullet b_{33} + a_{44} \bullet b_{43} \\
c_{44} &= a_{41} \bullet b_{14} + a_{42} \bullet b_{24} + a_{43} \bullet b_{34} + a_{44} \bullet b_{44}
\end{aligned}$$

Figure 2 Multiplication of matrices of size 4x4

Wavefront Systolic Multiplication

A typical way to introduce systolic array in such calculation is the wavefront systolic multiplication. Figure 3 demonstrates how the input sequence is fed into the array and how it propagates as the way like wavefronts.

DPU only does four things every time the data shifts in:

- It calculates a x b ;
- It adds the result to the previous value c_{ij} , and stores the result in c_{ij} ;
- It sends a to $P_{i,j+1}$ (right), unless j = 4; and
- It sends b to $P_{i+1,j}$ (below), unless i = 4.

Repeat the same procedure for seven times with the input array shown in Figure 3, we will have all the results stored in c_{ij}.

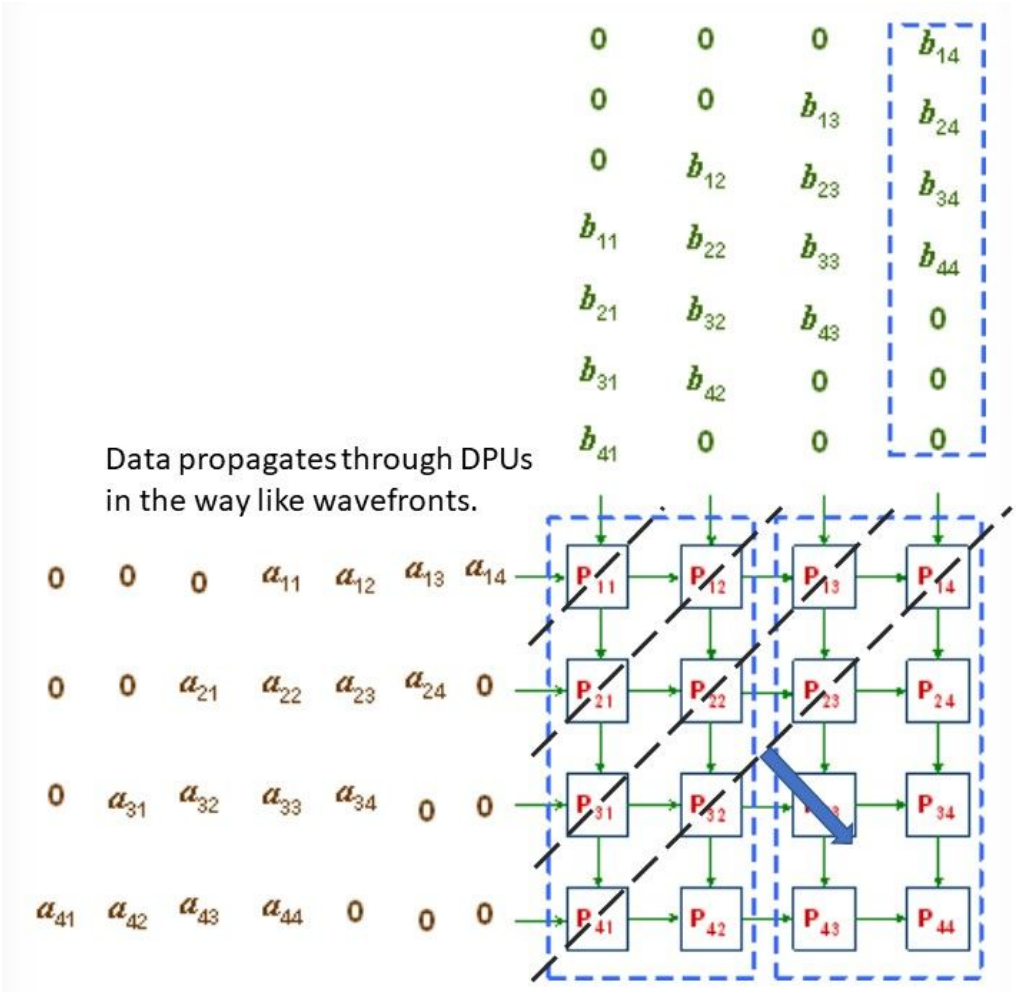

Figure 3 Input sequence of a 4x4-matrix multiplication. Row/column data from the two input matrices are offset and zeros are inserted to fill the blanks. Each time, one row of data on top, as well as one column of data on the left, is shifted into the DPU arrays.

Project Objectives

Build a matrix multiplier using the systolic array architecture. Design the schematic, layout and reasonable testbenches showing each component works as it should. **Total area**, **maximum available clock speed** and **overall throughput in time** will be considered as the building quality.

The project will be graded based on complexity as well, we have two levels of target applications. Choose one as your project topic:

Matrix-Vector Multiplier (90% complexity)

- Capable of multiplying a 4x4 matrix by a 4x1 vector.

- Input:
    - 4-bit unsigned. 5 inputs in total (4 for A & 1 for B)
    - Every input has to be driven by a 4-bit register (D-FF array).
- Output:
    - 12-bit unsigned. 4 inputs in total (4 for C)
    - Every output has to be stored in a 12-bit register. The register can be included in the DPU.
- 4 DPUs. Each DPU consists of a 4-bit x 4-bit multiplier and a 12-bit accumulator.

Matrix-Matrix Multiplier (100% complexity)

- Capable of multiplying a 4x4 matrix by a 4x4 matrix.
- Input:
    - 4-bit unsigned. 8 inputs in total (4 for A & 4 for B)
    - Every input has to be driven by a 4-bit register (D-FF array)
- Output:
    - 12-bit unsigned. 16 inputs in total (16 for C)
    - Every output has to be stored in a 12-bit register. The register can be included in the DPU.
- 16 DPUs. Each DPU consists of a 4-bit x 4-bit multiplier and a 12-bit accumulator.

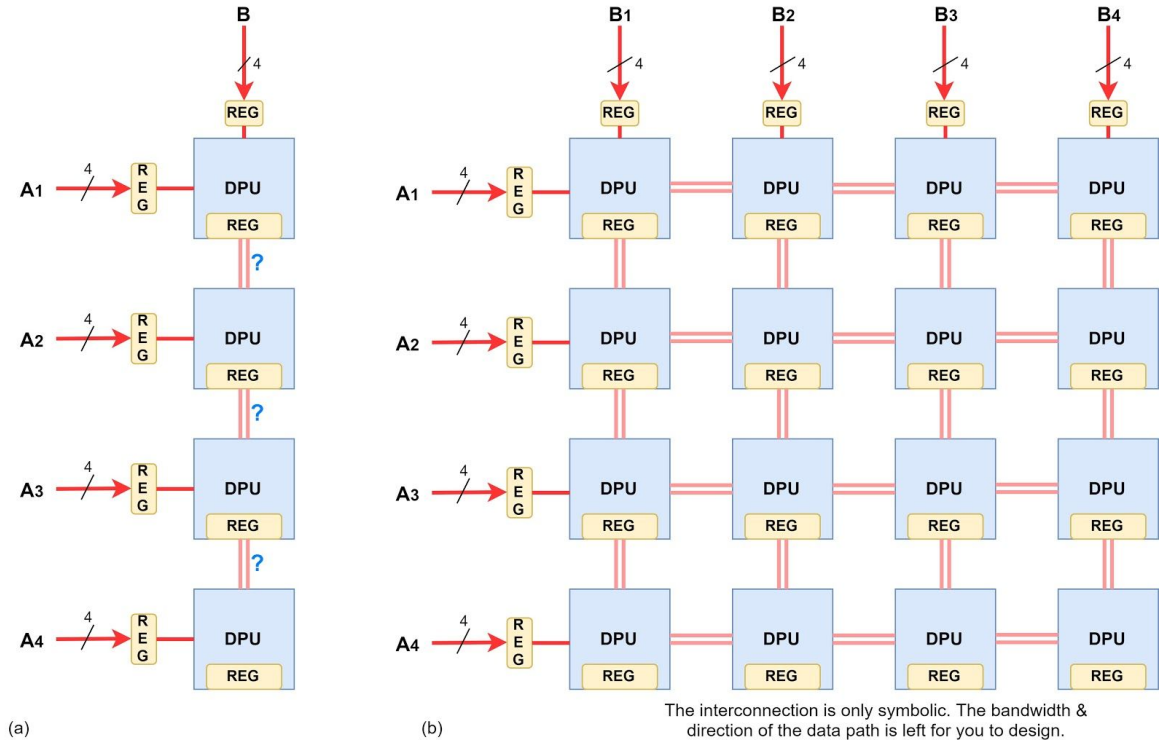

Figure 4 Potential architecture candidates for the (a) Matrix-Vector Multiplier & (b) Matrix-Matrix Multiplier. The bandwidth and direction of the interconnection is left for you to design.

Design Tips

Architecture Breakdown

Regardless which target application chosen, you might have noticed the DPUs look all the same. Recall that DPU only does four things every time the data shifts in:

- It calculates a x b ;
- It adds the result to the previous value c_{ij} , and stores the result in c_{ij} ;
- It sends a to $P_{i,j+1}$ (right), unless j = 4; and
- It sends b to $P_{i+1,j}$ (below), unless i = 4.

As highlighted, you will need one 4bit x 4bit multiplier to calculate axb. The product should be 8-bit. Then you need an adder to add the product back to the previous result. Also a 12-bit register is required to store the final outcome. Notice the adder should also be 12-bit to align with the register.

Combining the components of the DPU and peripheral, these are some suggested modules you want to build first. Please note that ***you are free to design the microarchitecture of the DPU***. Be creative and do not be constrained by the list:

- DPU
  - 4-bit x 4-bit multiplier
  - 12-bit adder
  - 12-bit register (D-FF array)
- Input
  - 4-bit register
- Output
  - register included in DPU
- Miscellaneous
  - buffer with different drive strength

As you can see many modules can be reused from your previous lab assignments. For example, 4-bit adders can be cascaded to build a 12-bit one. And you can redesign any components to maximize performance and/or minimize the area.

Collaboration in Cadence Virtuoso

You are going to build the project in a group of 2 ~ 3 people. It is important that everyone can share their designed unit with others. Here are the steps to grant your teammates access to your libraries:

```
1) cd /directory/to/your/library
2) permit teammate_username
3) Follow the steps on screen and enable write/edit/delete…
4) Go back to your teammate's session
```

5) Edit cds.lib using text editors at where they usually open Virtuoso
6) Add "DEFINE library_name /directory/to/your/library" at the end of the file.
7) Reopen Virtuoso. Now they can read and edit your library.

Project rubrics

Part I: Data Processing Unit Implementation (50pts)

1. [35pt] Schematic & Layout of the Data Processing Unit
   Design the data processing unit that is capable of multiplication and accumulation. Optimize it for the ***maximum operating clock speed*** and ***minimum area***. (Minimize the clock cycle-area product.) Include screenshots of the schematic and layout of all building blocks. **You can use four metal layers (M1, M2, M3, & M4) in this project.** Also think of how you are going to wire and put the DPUs together in a systolic array. Place pins with caution.

2. [15pt] Test Bench and Simulation
   Create a testbench for the DPU and give at least one example input to justify the correctness. Use the same clock configuration as lab 3.

Part II: Complete Systolic Array for Matrix Multiplication (30pts)

1. [20pt] Systolic Array
   Draw the schematic and layout of the complete systolic array with your choice of target application.

   - Matrix-Vector Multiplier (maximum 10pts)
   - Matrix-Matrix Multiplier (maximum 20pts)

   You must wire the clock, all signals, and power rails. Put the pins on the edge of the final design. Include screenshots of the complete circuit and mark the functionality of each building blocks. Attach rulers on the final bounding box to show the area.

2. [10pt] Simulation & Analysis
   Create a testbench for the DPU and give at least one example input to justify the correctness. Analyze the minimum working clock cycle. Compute the data throughput in (bit/s) and latency in (ns).

Part III: Project Presentation (20pts)

We will have a project presentation on **12/14 (Sat.) at 7:45 - 9:45 AM**. Each group has to present their project design in 5-8 minutes. **Please upload the slides by midnight on 12/13 (Fri.)**. Detail of the presentation will be announced.

Submission

This is a group assignment. Please register yourselves in one final project group on Canvas. Make sure you and your teammate's names are clearly stated in the report. There are two submission deadlines:

- Submit the ***slides*** for presentation by **12/13 (Fri.) 11:59 PM**.
- Submit a ***tgz file*** of your library along with ***the written report*** to Canvas by **12/14 (Sat.) 11:59 PM**.

Tips for creating tarball:

```
1) Go to the directory where you can see your library folder.
2) tar -czvf final_project_group_X.tgz name-of-your-library
```

Please replace group_X with your group number on Canvas and the name of your virtuoso library as name-of-your-library. For example, if your group number is 7, and your virtuoso library name for the assignment is systolic_array, then the command should be:

```
tar -czvf final_project_group_7.tgz systolic_array
```

Then you should submit the file name final_project_group_7.tgz.

Hint: if you're using ssh.ece.vt.edu to access the software, you can use `<CTRL><SHIFT><ALT>` key combination to bring out Guacamole Menu, where you can upload and download files from the remote server by drag and drop.