ECE627 Lab2 Report

Group Members:

- Paulos Tegegn (ptegegn@uwaterloo.ca)
- Sreagni Banerjee (s33banerjee@uwaterloo.ca)
- Tushar Garg (tushar.garg@uwaterloo.ca)
- Zhuanhao Wu (zhuanhao.wu@uwaterloo.ca)

Q1. RTL design

The vector is streamed one-element at a time and so is the matrix. The assumption is that either the vector is streamed completely before the matrix on the AXI stream bus OR the vector is streamed simultaneously with the matrix on AXI stream bus. We have tested our code on **PYNQ** for both of the mentioned assumptions and we are getting the desired results. We use 1-multiplier and 1-adder for this design as the input matrix/vector is being streamed one-element-at-a-time. The current multiplication and addition is happening in 1-clock cycle so we need 1-extra clock cycle to output the valid registered result. Assume that the multiplication is between an $N \times N$ matrix and an $N \times 1$ vector, then we would have the following data:

- Latency: $(N + 1) * N$
- Throughput: $\frac{1}{(N+1)*N}$

Q2. Waveforms

The waveform in Wavedrom is shown in figure 1.

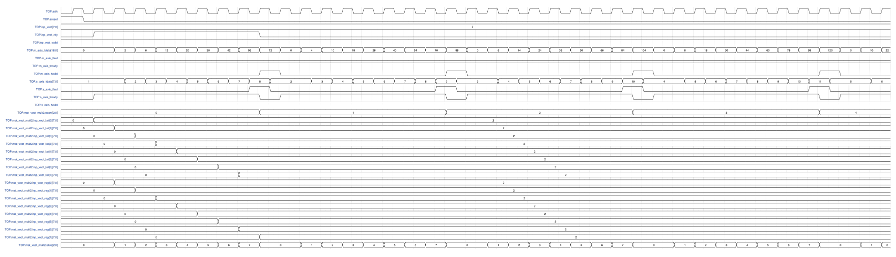

Figure 1: Waveform of circuit of Wavedrom.

The screenshot of VCD file is shown in figure 2

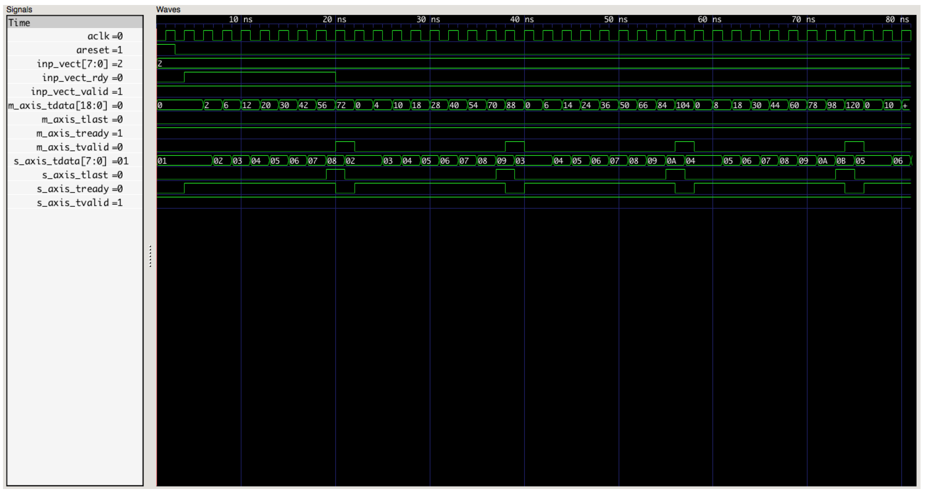

Figure 2: VCD dump.

Q3. Area-time tradeoffs

Resource Utilization

The utilization is reported using `yosys` and the script for generating the report is `run_yosys.ys`.

```
=== mat_vect_mult2 ===

   Number of wires:                326
   Number of wire bits:            416
   Number of public wires:          19
   Number of public wire bits:      77
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                393
     FDCE                           38
     LUT2                           24
     LUT3                           26
     LUT4                            7
     LUT5                           10
     LUT6                          180
     MUXCY                          16
     MUXF7                          62
     MUXF8                          13
     XORCY                          17
```

By default, `yosys` does not use DSPs and instead use LUTs to implement the addition and multiplication logic.

The following result is generated using `vivado` and the `edif` file generated by `yosys`:

```
+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| LUT6     |  133 |                 LUT |
| MUXF7    |   56 |               MuxFx |
| LUT5     |   43 |                 LUT |
| FDCE     |   38 |        Flop & Latch |
| LUT3     |   26 |                 LUT |
| LUT2     |   24 |                 LUT |
| MUXF8    |   13 |               MuxFx |
| LUT4     |    8 |                 LUT |
| CARRY4   |    5 |          CarryLogic |
+----------+------+---------------------+
```

With the optimization introduced by `vivado`, the number of `LUT6` primitive is less than that in `yosys`.

Trade-offs

The problem statement requires that the data is streamed into the design one at a time, and the vector is streamed into the design at the beginning. Every element of the matrix will be used to calculate one multiplication and the results will be used in at most one addition. Thus, even if we can use multiple adders and multipliers, the data stream of matrix will limit the parallelism we can achieve and multiple adders/multipliers will not bring us any benefits. The best we can do in this design is to split the multiplier and adder in 2-clock cycles, which increases the number of flip-flop usage but also increases the clock frequency on which the design can operate. For our design, we perform the addition and multiplication in the same clock cycle, hence we are saving flip-flops but our design will be slightly slower than the above mentioned design.