EEE 525 VLSI Design **Athi Narayanan Parameswaran**

ASU ID: 1211377931

LAB 05 Convolution & Max-Pooling Engine Design

❖ Architecture:

Basic Architecture:

1. Convolution and Max-Pooling Engine design has 4 input ports namely, Image (128 bits), Kernel (72 bits), Shift amount (2 bits) and Clock. One output port y (8 bits).
2. At start of every cycle, we get 128 bits of input image of size 512x512 and it is realized as a 4x4 matrix, while the kernel is realized as a 3x3 matrix.
3. The inputs are flopped and provided to the engine for computation.
4. The image is then split into 4 sub-matrices each of size 3x3. Each of the sub-matrix is then convoluted with the kernel matrix to get the weighted tap values of each submatrix.
5. Since, image pixels can only be in the range of 0 to 255, the weighted values are restricted to the range and the value is then shifted based the amount specified in Shift register (0,1,2,3).
6. In max-pooling method, the maximum of the calculated weighted values is taken and replaced in the input image.
7. The max-pooled output is flopped and written in conv_pool.output file.

Optimized Architecture:

- **Stage 1 – Image Acquisition**
  - ❖ Image (128 bits), Kernel (72 bits) and Shift (2bits) are provided as inputs and they are flopped in the first stage.
  - ❖ Image is split into 16 8-bit values (img1 to img 16) and Kernel is split into 9 8-bit values (kernel1 to kernel9) for easy computation.
- **Stage 2 - Multiplication**
  - ❖ The design consists of 4 multiplication stages to compute the partial products of Image and Kernel which are running parallelly with 9 multipliers in each stage.
  - ❖ All 36 multipliers are run parallelly in the same clock and the partial products of all are flopped and passed on to next stage (a01 to a39).
- **Stage 3 – Addition/Shift/Floor**
  - ❖ The flopped partial products (af01 to af39) are then split into 4 addition stages with each having 8 adders to add the 9 partial products.
  - ❖ The sign bit of total sum values (sum0, sum1, sum2, sum3) are checked and if Sign bit is set (if negative), the sum value is forced to zero.
  - ❖ The floored sum values (v0, v1, v2, v3) are then shifted based on the shift amount and then flopped to next stage.
- **Stage 4 – Ceil/Max-Pooling**
  - ❖ The weighted sum values (v0, v1, v2, v3) are then checked if they are greater than 255, then, the values are limited to 0 to 255.
  - ❖ The final outputs of the convolution engine are c0, c1, c2, c3, which are passed as inputs to max-pooling engine.
  - ❖ In Max-pooling, the maximum (c0, c1, c2, c3) is computed and the resultant value is flopped and written in the output file as (y).

EEE 525 VLSI Design **Athi Narayanan Parameswaran**
ASU ID: 1211377931

❖ High Level Diagram

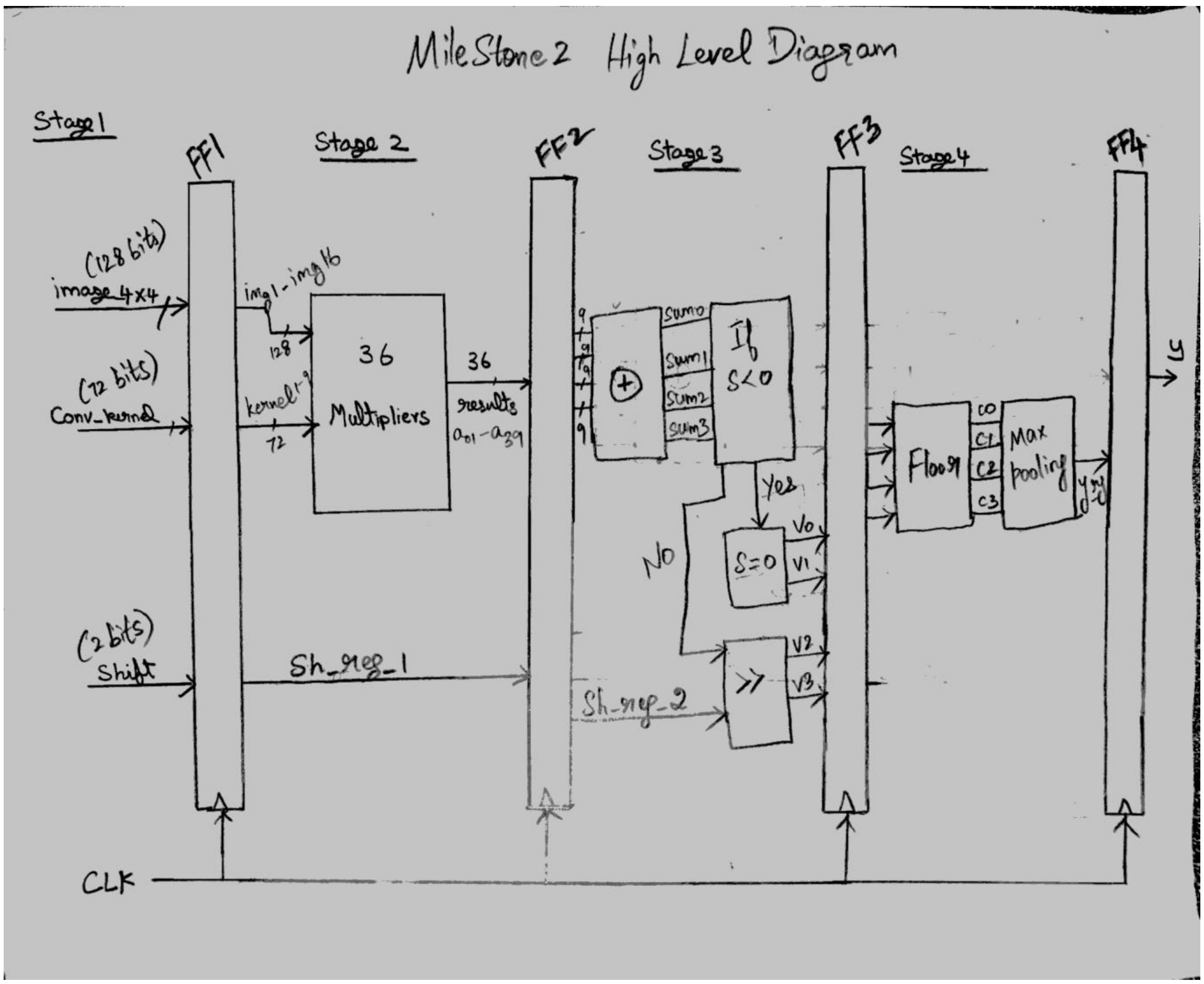


Figure 1: High level block diagram of the pipelined architecture

❖ Design Decisions

1. Pipelining is adding registers in between my design to partition my design into stages where each stage can work independently.
2. Also, my architecture has only one data path which follows a single unique direction.
3. The worst-case critical path in my basic architecture was from multiplier to the output flopped value y.
4. To break the critical path, I have introduced pipeline stages in between the multiplier and adder.
5. I broke the architecture into 4 stages as Image Acquisition, Multiplication, Addition/Shift and Max-pooling.
6. After pipelining, I have my worst-case path from in the shift logic which can only be removed if I further optimize my shift logic or the comparator module.

❖ Problems faced

- Initially, I was planning to optimize my multiplier by using a Booth encoded multiplier instead of the combinatorial multiplier, since, combinatorial multiplier takes more number of cycles to compute the product.
  - Although, Booth multiplier reduces my clock period (total latency), it occupies more area and it will have a huge impact in my Quality metric, so I dropped the idea to implement booth multiplier.
- Secondly, I don't have a worst-case path, linking my adder, but if I add another stage to break my adder stage.
  - This will further improve my latency but it will also increase area/power and might add hold time violations which require buffers to be inserted.
- The Shift/Floor/Ceil logic was the only obvious choice to introduce pipeline stages and hence, I have added a stage in between shift and floor/ceil logic to improve overall latency.

❖ Final Results:

- **Total Latency** = 27569010 ps => **0.027569010 ms**
- **Power** = **6.895 mW**
- **Area** = 226.3*228.5 = 51709 umsq = **0.051708 mmsq.**
- **Density** = **0.77308**

Table1: Final results table

Total Latency (ms)	Power (mW)	Area (mmsq)	Density
0.02756901	6.895	0.051708	0.77308

❖ Results

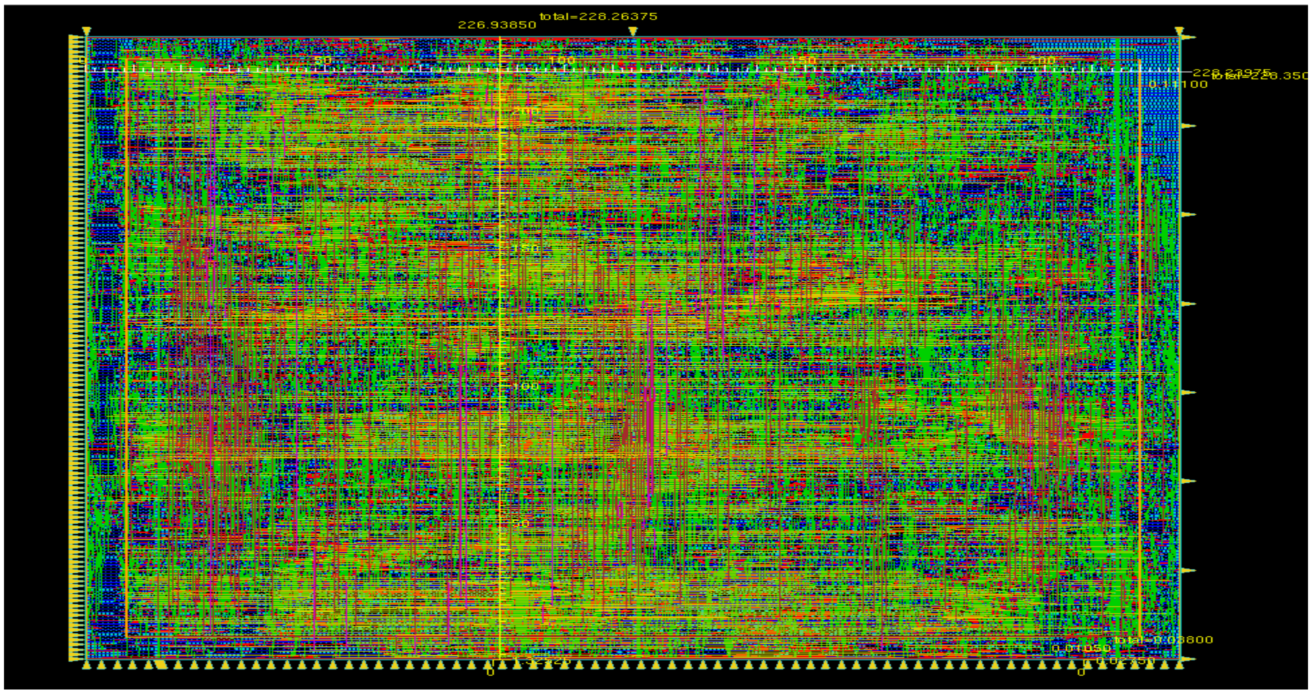
1. On comparing this architecture with the architecture used in Milestone 1, we could clearly see some improvements based on frequency of operation and area.
2. The pipelined architecture, increases the area of the used by design due to addition of registers between stages.
3. At the same time, by adding registers, we are improving the throughput of the design, by making the same design work in a faster clock period or lower frequency.
4. Designs running at lower frequencies usually utilize more power with the extra overhead due to registers.

Table 2: Comparison of performance between milestone1 and milestone2

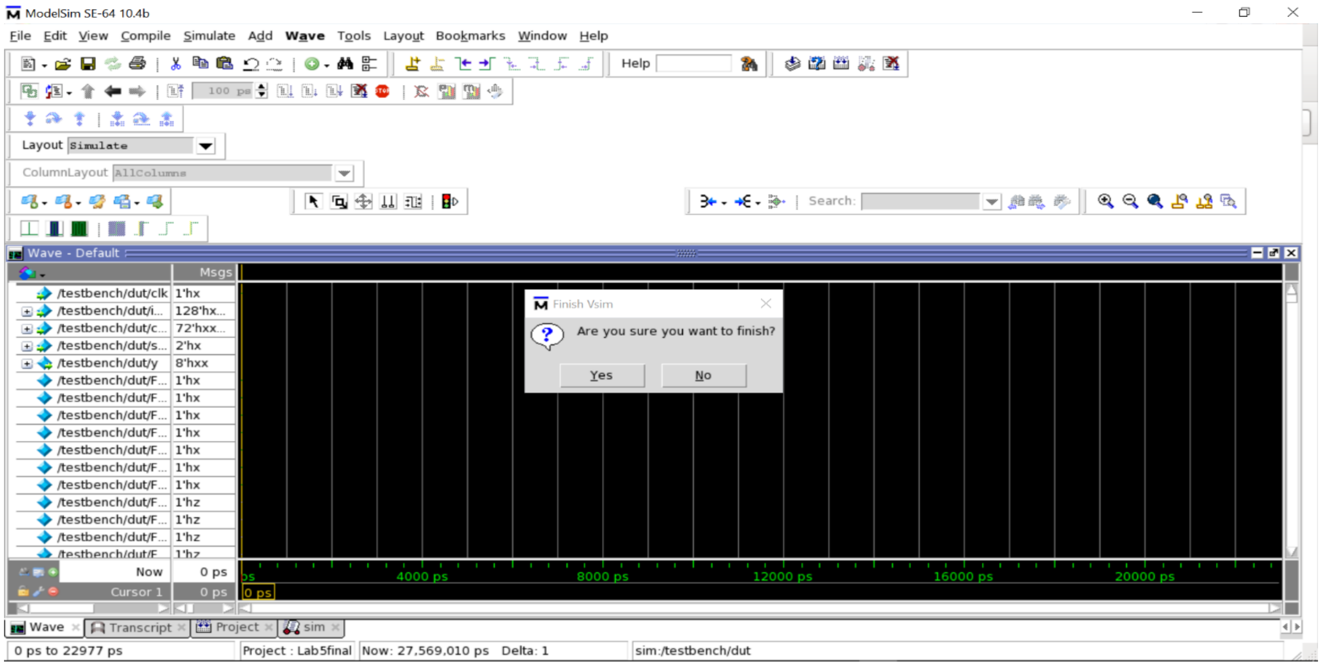
Architecture	Clock Period
Milestone1	1500ps
Milestone2	420ps

Athi Narayanan Parameswaran
ASU ID: 1211377931

❖ Final Layout of my architecture:



❖ Screenshot of Model Sim



```
                          Msgs
/testbench/dut/clk        1'hx
/testbench/dut/i...       128'hx...
/testbench/dut/c...       72'hxx...
/testbench/dut/s...       2'hx
/testbench/dut/y          8'hxx
/testbench/dut/F...       1'hx
/testbench/dut/F...       1'hx
/testbench/dut/F...       1'hx
/testbench/dut/F...       1'hx
/testbench/dut/F...       1'hx
/testbench/dut/F...       1'hx
/testbench/dut/F...       1'hx
/testbench/dut/F...       1'hz
/testbench/dut/F...       1'hz
/testbench/dut/F...       1'hz
/testbench/dut/F...       1'hz
Now                       0 ps
Cursor 1                  0 ps

ps   4000 ps   8000 ps   12000 ps   16000 ps   20000 ps

0 ps to 22977 ps   Project : Lab5final   Now: 27,569,010 ps   Delta: 1   sim:/testbench/dut
```

❖ Conclusion

- The limitation to this architecture lies in the number of pipeline stages that could be added in the design. As the number of stages, increases frequency drops, but power and area steadily increases.
- Hence while designing, we have to consider the requirements and purpose of the architecture.