SIMD and Weight-Output Reconfigurable 2D Systolic Array-Based AI Accelerator and Mapping on Cyclone IV GX

Jesse Vernallis , Sankalpa Hota , Ned Bitar , Stanley Pan , Rohon Ray , Madeleine McSwain
Electrical and Computer Engineering (ECE)
ECE 284: Low Power VLSI for Machine Learning

I. PART 1: VANILLA VERSION

In the vanilla version we trained a 4 bit version of VGG16 to 92%, which squeezes down layer 27 to be an 8 x 8 channel convolution. When mapped to hardware, with padding included, this resulted in 36 nij input pixels and 16 o_nij output pixels. After loading our kernel weights into our systolic array we would execute the dot product of each nij with each kernel's kij. This was read into an OFIFO below the array and then written to a PSUM memory. Once all kernel execution was completed, a LUT within our special function processor (SFP) would read the required PSUM from memory and accumulate each final output o_nij value across the 8 output channels. These were then finally passed through a ReLU and our sfp_out was verified by our testbench.

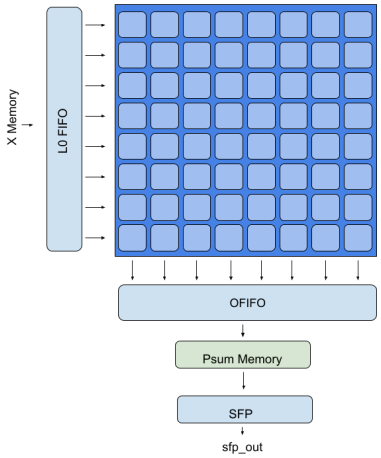

Figure 2.1 : Weight Stationary Array Hardware

II. PART 2: 2BIT AND 4BIT LANE RECONFIGURABLE SIMD SYSTOLIC ARRAY

Instead of using 4-bit activations, the model was trained with 2-bit activation quantization, achieving 90% accuracy, while the weights were maintained at 4-bit precision. In the VGG-16 network, layer 27 was compressed to a 16×16 channel convolution.

From an architectural perspective, the MAC tile was redesigned to operate over two cycles instead of a single-cycle execution. The tile consists of two MAC units. In a single cycle, each MAC processes two 2-bit activations packed into a 4-bit input. Over two cycles, this is equivalent to processing two 4-bit activations, enabling correct partial-sum (PSUM) accumulation.

Additionally, the control signals were enhanced to allow runtime configurability, enabling the user to select between 2-bit and 4-bit activation modes. Correspondingly, the SFU (Special Function Unit) was updated to correctly support and process both activation precisions, ensuring continuous operation across quantization modes.

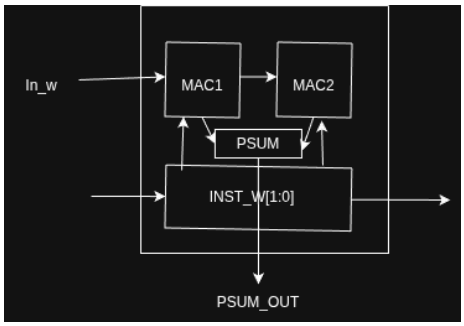

Figure 3.1 : 2Bit 4 bit activation MAC Tile layer

III. PART 3: WEIGHT STATIONARY AND OUTPUT STATIONARY RECONFIGURABLE SYSTOLIC ARRAY

We implemented an OS (Output-Stationary) dataflow by introducing a dedicated control signal, os. In this

mode, the c_q register is allocated to accumulate the partial sums of the incoming activations and weights.

With OS mapping enabled, weights are routed from the north direction instead of the west, while activations continue to stream from the west. To support this data movement, the weights are stored in an L1 FIFO located at the top of the array, analogous to how activations are buffered on the left side.

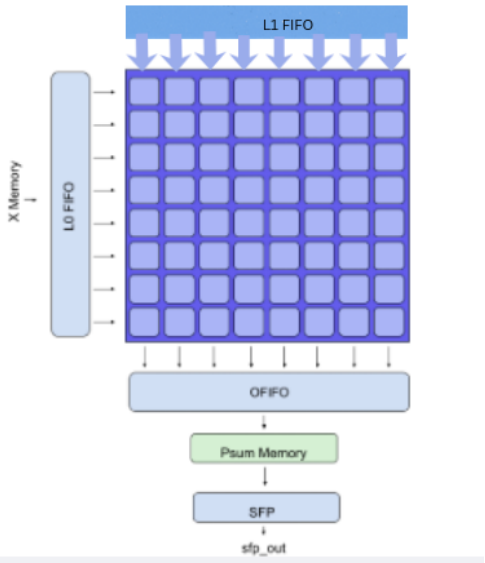

Figure 3.1 : Output Stationary Array Hardware

IV. PART 4: ALPHAS

A. Alpha 1 : ResNet

Although VGG and ResNet are both convolutional networks, their computational structure differs, which significantly impacts how efficiently they map onto a systolic array. For example, ResNet features layers which have skip (residual) connections (hence "Res"Net) that resolve the vanishing gradient problem [1].

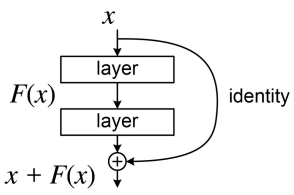

Figure A.1: Identity layer referenced to calculate fc layer.

In the proposed architecture, we squeezed the third layer's conv2 to an 8 x 8 channel convolution, a similar process to squeezing a deeper layer in VGG. The accuracy achieved compared to VGG was similar, but ResNet exposes better hardware optimization due to its structured dataflow and activation reuse. Although this may not directly affect the speed of the hardware, capturing these activations help with the accuracy of the ML pipeline.

B. Alpha 2 : Clock Gating

In a systolic array, the majority of power consumption occurs from clock toggling and FIFO register chains, even if the FIFOs are full or empty. Without clock gating, idle hardware consumes more dynamic power (as activity factor remains high).

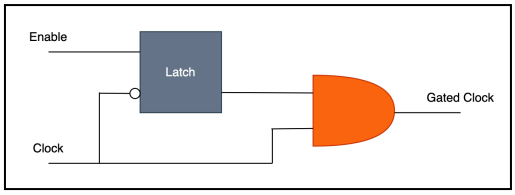

Figure B.1: visual concept of clock-gating. Here, Enable would be defined as our FIFO condition.

Alpha 2 introduces local clock enabling to suppress unnecessary switching, based on the activity of the FIFO. Here, our gating was defined on `(~ofifo_full | ofifo_ready)` enabling the clock of the MAC array.

	Before clock gating	After clock gating
Total Power Dissipation (mW)	244.05	245.43
Core Dynamic Thermal Power Dissipation (mW)	34.93	35.37
Core Static Thermal Power Dissipation (mW)	118.76	118.77
I/O Thermal Power Dissipation (mW)	90.36	91.29

	Before clock gating	After clock gating
Max Clock Frequency (MHz)	135.94	137.17

Figure B.2: table of clock-gating results

After synthesis, our results were able to achieve slightly faster clock speeds, while maintaining negligible power dissipation. The results were not significant to guarantee use on our hardware, but in more intense FIFO conditions, the accelerator is able to significantly reduce its dynamic power.

C. Alpha 3: Pruning

To further reduce computation and energy consumption, pruning is applied to the ResNet model to remove redundant parameters, resulting in less

computations that can be skipped in hardware. There are two methods to use:

- *Unstructured Pruning:*
  - Great for compressing weights and potentially lowering energy if you have sparse-aware hardware.
- *Structured Pruning:*
  - Lower latency, and smaller on-chip buffers because the network's dimensions (channels, filters, branches) shrink, making it better fit for fixed-size accelerators and systolic arrays.

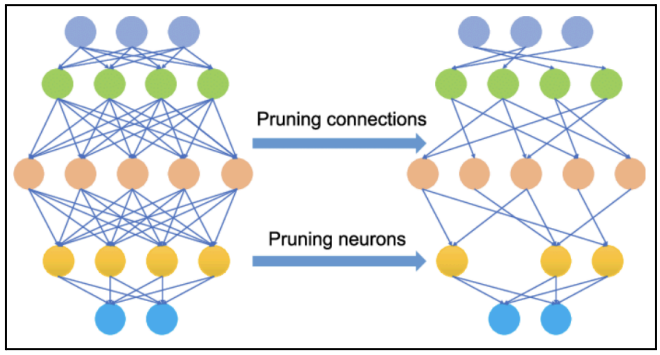

Figure C.1: Combined Pruning

By using both pruning methods, we can minimize the number of calculations while maintaining reasonable sparsity and accuracy, as we preserve regular computation patterns.

Method	*Accuracy*	*Sparsity*
Combined	*82%*	*~0.8*

Figure C.2: Results after Pruning and Training

D. Alpha 4 : Nij LUT and In Place PSUM

After completing part 1, we noticed that the weight stationary systolic array computed many partial sums that were never used in the final accumulation. To reduce runtime, we decided to only compute the MAC operations required for the convolution by filtering our activations. This was done by reusing the LUT from our weight stationary special function processor to read the required activations for each kij to the l0 FIFO. Due to the 16 output pixels of our convolution, only 16 activations were required for each kernel load. This resulted in a 60% speedup of our systolic array execution.

This filtered execution resulted in only 16 elements in our OFIFO after each kernel execution. These could have been accumulated after all kernel execution using the PSUM SRAM, but changes in the SFP LUT would have been required to acquire the correct result. However, accumulation in the SFP was not needed if we used the systolic array's in_n input to accumulate in place. Additionally, the removal of the SFP would result in a reduction of around 150 bits in registers and 8 16 bit adder circuits. In order to do this the OFIFO was changed to an I/O FIFO with a configurable read input. This configurable read allowed the fifo to both read row wise and all at once. The row wise read was required to match the pipelined nature of the systolic array to the partial sums from the previous kernel. This was done on all but the first kernel, which set in_n to zero. Once all kernels were complete, the I/O FIFO would read all at once and pass through our channel-wise ReLU block.

This in-place accumulation resulted in the removal of a PSUM memory. Additionally, accumulation time after the kernel execution now only required reading the I/O FIFO for its 16 o_nij. Therefore speeding up accumulation by 1153%. Both of these in tandem resulted in a 45% speedup of the entire convolution, now largely bottlenecked by memory access.

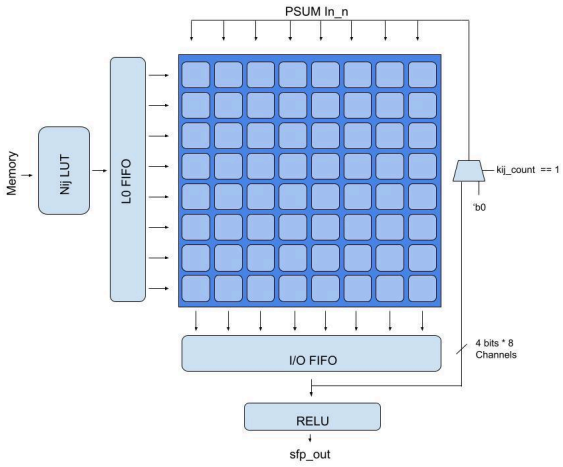

Figure 5-D.1 : Nij LUT and In Place PSUM Hardware

Section	*Weight Stationary Baseline*	*Nij LUT and Inplace PUSMs*
Execution	*486 clocks*	*306 clocks*
Accumulation	*238 clocks*	*19 clocks*
Total	*1265 clocks*	*875 clocks*

Figure 5-D.2 : Clock Reduction Table

E. Alpha 5 : FIFO Depth Reduction

The 64 deep l0 and OFIFOs used in our Part 1 baseline were larger than required for either FIFO's function. After Alpha 4, both FIFOs were only required to be 16 deep. This resulted in a 75% reduction in FIFO registers from 10Kb to 2.5Kb.

F. Alpha 6: Combined Part 2 + Part 3

Integrated all four features by combining the 2-bit and 4-bit lane reconfigurable SIMD systolic array with reconfigurable weight-stationary (WS) and output-stationary (OS) dataflows. The MAC tile supports runtime-selectable 2-bit and 4-bit activation modes through enhanced control signals. In 2-bit mode, the design operates over two cycles for PSUM generation, using two MAC tiles, where each MAC processes two 2-bit activations packed into a 4-bit input per cycle. Over two cycles, this is equivalent to processing two 4-bit activations, enabling correct PSUM accumulation with 4-bit weights. In contrast, in 4-bit mode, PSUM generation is completed in a single cycle, and only one MAC tile is active, directly processing 4-bit activations with 4-bit weights. The SFU was updated to correctly support and process both activation precisions, ensuring continuous operation across quantization modes. In addition, Alpha 6 supports both WS and OS mappings through a dedicated os control signal. In OS mode, the c_q register is allocated to accumulate the partial sums of the incoming activations and weights. With OS mapping enabled, weights are routed from the north instead of the west, while activations continue to stream from the west. To support this data movement, the weights are stored in an IFIFO at the top of the array, analogous to how activations are buffered on the left side. This unified architecture enables seamless switching between OS and WS dataflows while maintaining correct PSUM accumulation across both 2-bit and 4-bit activation modes.

Logic Util. (in ALMs)	*12,692 / 56,480 (23 %)*
Total Registers	*4089*
Total Pins	*473*
DSP Blocks	*30*

Figure F.1: Results of Synthesis

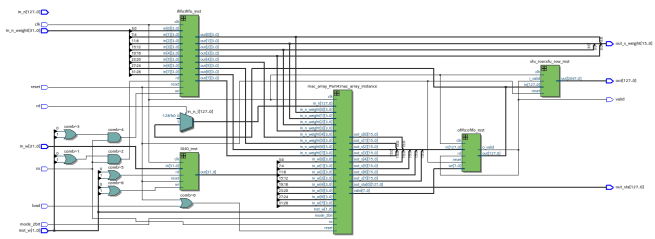

Figure F.2: RTL Netlist of Alpha 6 design using Quartus

G. Alpha 7: FPGA Mapping

Most of these alphas were able to be achieved through simulation and calculation, so our team decided to map it to hardware to see the feasibility of this project under physical constraints. Here, we used a Cyclone V; we were able to successfully synthesize the systolic array onto the FPGA, suggesting that this design meets resource and timing requirements of the Cyclone V (and likely its predecessor), alongside the additional optimizations, such as clock gating, FIFO reduction, etc.

Logic Util. (in ALMs)	*6,407/29,080 (22%)*
Total Registers	*12,049*
Total Pins	*199/364 (55%)*
DSP Blocks	*69/150 (49%)*

Figure G.1: Results of Synthesis

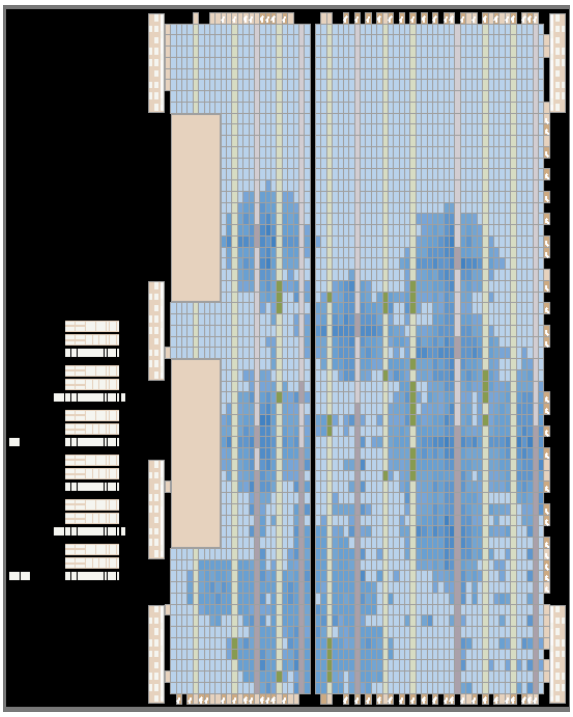

Figure G.2: Mapping of combinational blocks on Cyclone V

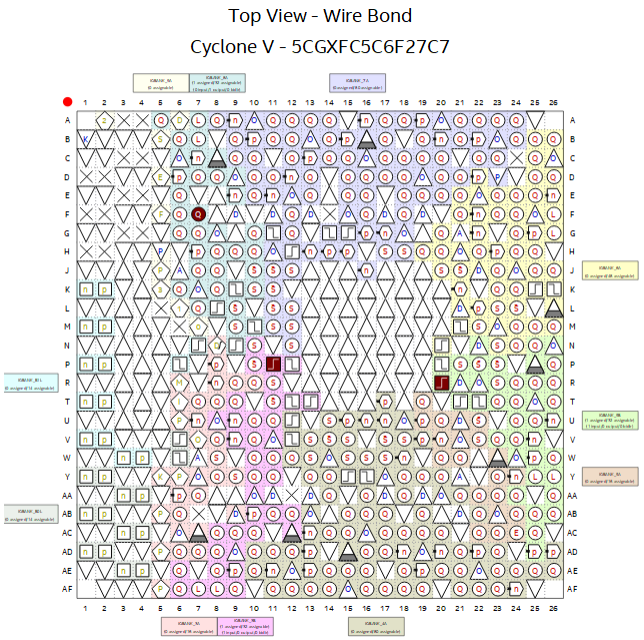

Figure G.3: Pin assignment of clk, reset

Although the design was successfully synthesized, full hardware validation using the testbench was not completed due to time constraints associated with configuring data transfer over USB for the FIFO-based pipeline. As a result, on-hardware functional testing was deferred. Future work includes completing the data interface and performing end-to-end validation to fully demonstrate correct operation under physical hardware constraints.

H. Alpha 8 : 2-bit Activation + 2-bit Weights

This alpha is a slight modification of the Part 2 systolic array for Weight-Stationary operation. Reducing the weights to two bits allows for simpler (though more numerous) MAC operations, smaller partial sum storage, and reduced FIFO widths during operation. This should ideally reduce flip-flop utilization and dynamic power due to fewer bit switches. The MAC layer was updated with two different MAC units, which operate as one MAC for 4-bit operations and two MACs for 2-bit operations. The data always arrives as 4 bits, but for the 2-bit mode, the MAC layer splits the data into two parts while maintaining the sign convention.

Although increased quantization can impact model accuracy, prior quantization-aware training ensures that accuracy degradation remains limited, giving a different version for us to experiment with. The results are not inherently better, it allows us to see the trend between bit resolution and efficiency of the systolic array, compared to the vanilla and part 2 versions. In Table H.1 we can observe the design effectively reduced the Logic unit (ALM) count to 16%.

Logic Util. (in ALMs)	*8,834 / 56,480 (16 %)*
Total Registers	*12298*
Total Pins	*563 / 268 (210 %)*
DSP Blocks	*128 / 156 (82 %)*

Figure H.1: Results of Synthesis

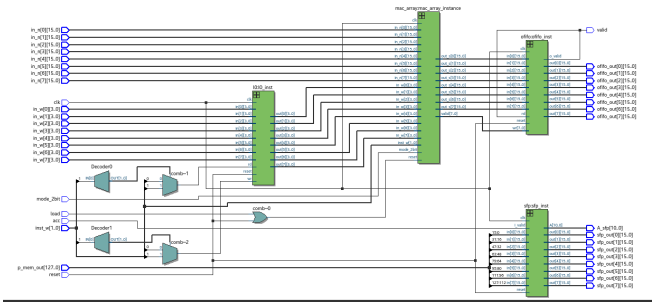

Figure H.2: RTL Netlist of Alpha 8 design using Quartus

V. REFERENCES

[1] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 6, no. 2, pp. 107–116, 1998.

[2] P. Varasala, B. Karapa, and K. Maddu, “Intelligent clock gating for FPGA-based RISC architectures: A novel approach to switching activity and dynamic power reduction,” International Journal of Computer, vol. 51, no. 1, pp. 79–89, Jul. 2024.