Final Project Report

CPR E 487/587

Kevin Dickey & Varun Jain

1.0 Introduction – Aims & Motivation

Since the emergence of IoT, researchers have been working on various accelerator designs to minimize power consumption. While most approaches are generally domain and application specific, many high-level concepts can be taken and adapted across different domains. In this work, we've decided to focus on a key component of these accelerators – the MAC (Multiply-Accumulate) unit. As MAC operations constitute the majority of the computational workload in accelerators, designing a high-performance, low-power MAC unit is essential for developing truly efficient and versatile accelerators.

Our objective is to explore and evaluate a few different MAC unit designs, balancing both power and performance. Our hope going into the future is to be able to use these designs to further develop a low-power accelerator suitable for implementation on various edge devices. This will expand the range of AI and ML applications in a person's daily life, enabling broader adoption in IoT and other resource-constrained environments.

For the purposes of this project, we specifically design out MAC units so that they are implementable on the BOBBER platform made at Iowa State University [7]. The platform uses a combination of a low power non-volatile FPGA with a microcontroller for testing different Neural Networks and Accelerator Designs on a Batteryless System. Since we are aiming to make a low-power-high-throughput MAC unit, BOBBER is a good test system.

2.0 Design Approach

All of our designs have 4 assumptions: 1. The inputs will be provided as a concatenated 16-bit value, 2. the quantization level is provided, 3. the system outside the MAC unit (C++ code) will handle truncation of the final output of the MAC unit, and 4. the bias addition is handled outside the MAC unit.

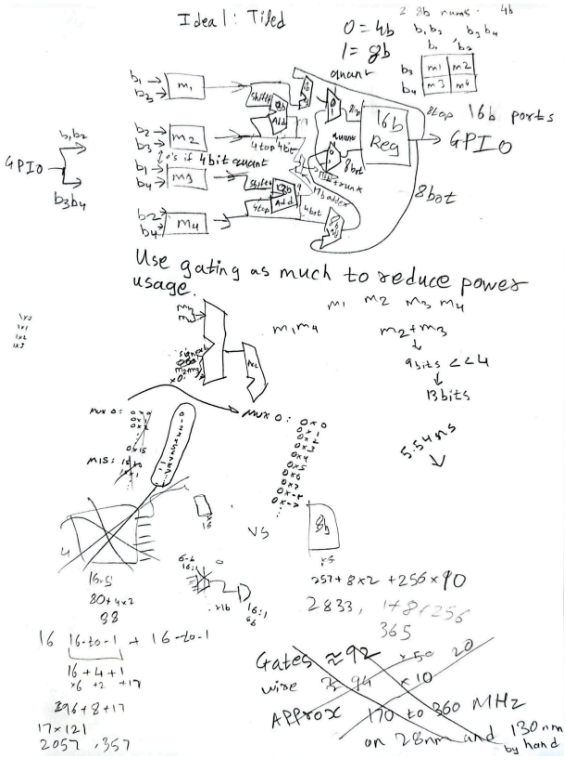

Figure 1: Tiled MAC Initial Concept.

2.1 Tiled MAC Unit

This MAC unit (initial design in Figure 1) is designed to work with 8-bit and 4-bit quantization. It divides the inputs into two 4-bit numbers each and multiplies them using our 4x4 MUXiplier Units (Section 2.3) and table 1 below. IN_A and IN_B are 8-bit numbers, with H and L representing the higher and lower 4-bits respectively.

Table 1:

IN_A_H * IN_B_H	IN_A_L * IN_B_H
IN_A_H * IN_B_L	IN_A_L * IN_B_L

This design took a few attempts! Thankfully, it shares quite a bit of overlap in design with the DRMAC. The rough schematic for our final Tiled MAC Design is in Figure 2. Some of the key optimizations of the MAC unit are as follows:

- Carry-Lookahead Adder
- Quantization Gating
- 4x4 MUXiplier Unit

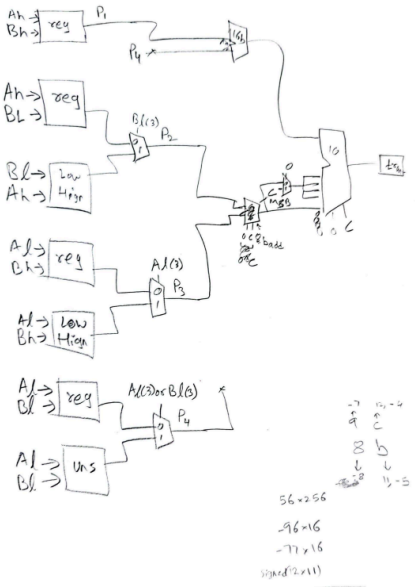

Figure 2: Revised Tiled MAC Design.

2.1.1 Pipelined Approach

Our approach for the pipelining the Tiled MAC unit was to insert a pipeline register just before adding the final accumulated value. We chose this location for our pipeline register to ensure it could effectively be used in both the 8-bit and 4-bit quantization options.

2.2 Dual-Range MAC Unit

This MAC unit is designed to work with 8-bit and 6-bit quantization. It masks some of the input bits based on the quantization, preventing the switching of unnecessary bits. According to [1], a similar design with 24-bit and 16-bit options decreased the power consumption by 56%. We believe that when employing this to 8-bit and 6-bit quantization, we should get about 20% power savings.

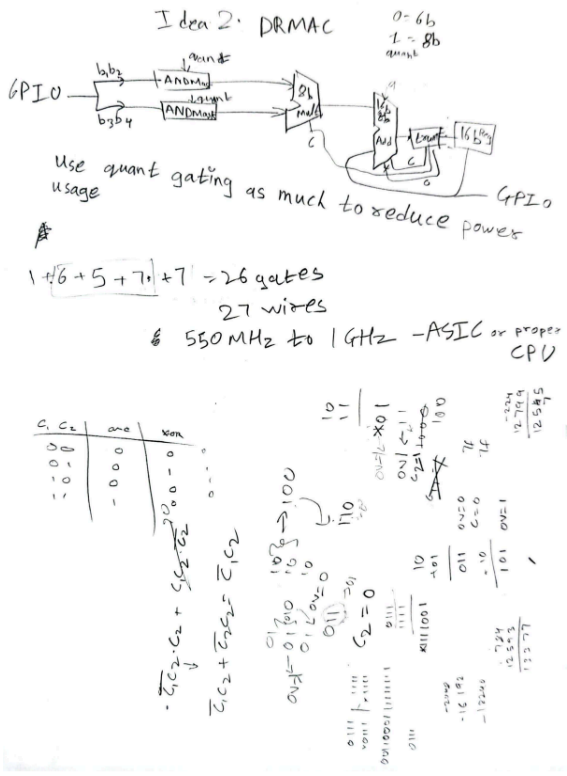

Figure 3: DRMAC Initial Concept.

Some of the key optimizations of the MAC unit are as follows:

- Carry-Lookahead Adder
- AND Masking for 6-bit Quantization
- 8x8 MUXiplier Unit

2.2.1 Pipelined Approach

Our placement for the pipeline register in the DRMAC was the same as in the Tiled MAC – just after the 16-bit accumulation adder. This was to ensure that our 8-bit and 6-bit options were able to fully utilize the pipeline register efficiently.

2.3 MUXiplier Units

A key component of our MAC designs were our MUXipliers as we have recently decided to name them. Figure 4 shows our 4x4 MUXiplier implementation, with the inputs on the row of 16 MUXes being hard coded to the possible 4-bit x 4-bit multiplications. The 8x8 MUXiplier contains a total of 7 of these 4x4 MUXiplier units, alongside three Carry-Lookahead adders (2x 16-bit, 1x 8-bit).

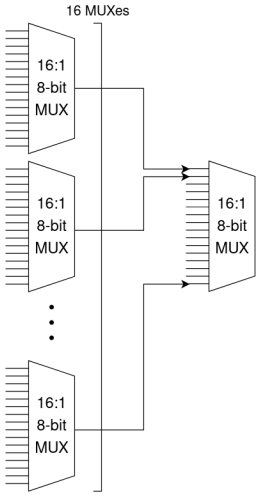

Figure 4: 4x4 MUXiplier Design.

2.3.1 Modified MUXipliers

Due to design issues mentioned in 7.0, we similarly needed to implement a few other types of MUXipliers. Namely, we implemented mux_4x4_lh.vhd and mux_4x4_uns.vhd. These units followed the same structure as the regular mux_4x4.vhd MUXiplier; however, they implement some special logic necessary for the correct multiplication. The

mux_4x4_lh.vhd file handles this special logic we derived through a lot of debugging:

```
-signed((-signed(A_high) * signed('0' & B_low)));
```

Similarly, the mux_4x4_uns.vhd file handles 4-bit x 4-bit unsigned multiplication for these operations:

```
signed((unsigned(A_low) * unsigned(B_low)));
```

To help us derive the results from all of these multiplication results, we used a few different python scripts which can be found commented out at the bottom of each mux_4x4 unit.

2.4 Scrapped Ideas

2.4.1 DRMAC-Based Accelerator

This was our initial project premise; however, given its scope and our lack of lab 6 completion we unfortunately had to change gears. The first paper we researched was regarding an energy efficient DCNN processor for intelligent IoE systems [1]. The key point of this design was its use of multiple low-power Dual-Range MAC units, a row-stationary dataflow, and a few other optimizations for reducing off-chip memory accesses. From this paper we developed the singular thought – we need to make this DRMAC unit, it sounds GOATED. Thankfully we still implemented the DRMAC so we could eventually go forward with making an entire accelerator based off of it, but for now it's on hold.

2.4.2 Booth's Algorithm

Another one of the ideas we had for our MAC units was implementing Booth's Multiplication Algorithm into our multiplier [9]. This algorithm would allow our multiplication to be more efficiently calculated – meaning better performance and less power consumption. Unfortunately, we have decided to put this idea on hold due to time constraints, but we believe it would be well worth implementing.

2.4.3 Approximate Multiplier

One of the last ideas we ended up scrapping was regarding an approximate 8-bit multiplier [10]. Part of what made this unit particularly interesting was that it was input-aware. That is to say, it often approximated the given inputs such that they were often instead powers of two – meaning its multiplication can be accomplished simply by shifting. In cases where it

was too egregious to approximate it as a power of two, it gives a rough approximation for the result. We've included a few snippets that we found particularly interesting from the paper below.

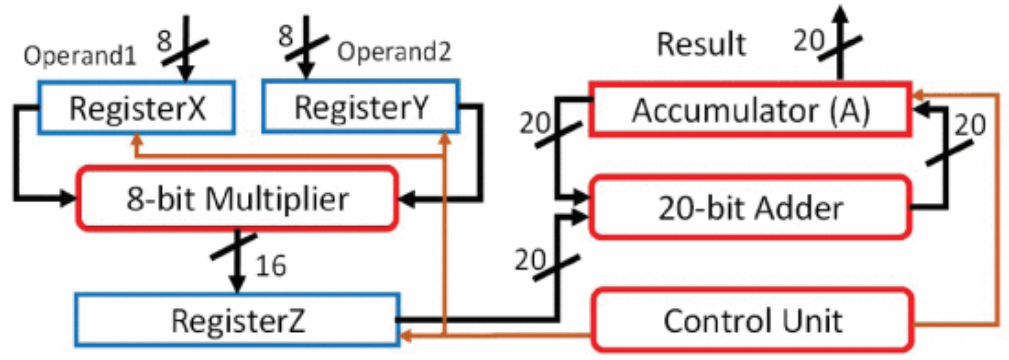

Figure 5: Schematic of MAC with Approximate Multiplier.

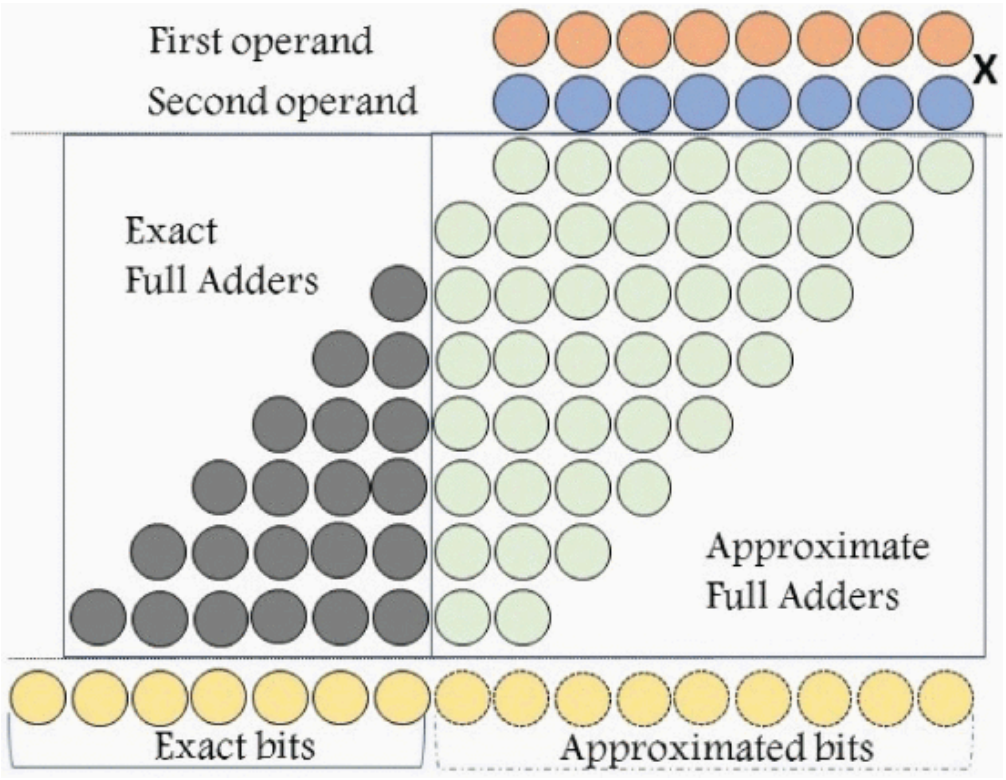

Figure 6: Approximate Multiplier.

3.0 Literature Search

Sim et al. [1] propose a deep convolutional neural network (DCNN) processor that achieves 1.42 TOPS/W for intelligent IoE systems by integrating energy-efficient Dual-Range MAC (DRMAC) units, a row-stationary dataflow, and reduced off-chip memory access. The DRMAC units support both 24-bit and 16-bit quantization, resulting in a 56% reduction in power consumption. While their work demonstrates the feasibility of dual-range MAC designs, our project adapts this concept to support 8-bit and 6-bit quantization. We anticipate approximately 20% power savings through techniques such as AND masking and optimized multipliers (MUXipliers), balancing power and performance while laying the groundwork for a complete accelerator system in the future.

Chen et al. [2] introduce Eyeriss, an energy-efficient and reconfigurable accelerator for deep convolutional neural networks (DCNNs). Eyeriss minimizes power consumption through optimized dataflow and reduced data movement, which are major contributors to energy cost. It employs a spatial architecture with row-stationary dataflow, emphasizing local data reuse to minimize off-chip memory accesses. This highlights the importance of energy-efficient MAC designs, as MAC operations dominate the workload in accelerators. Inspired by Eyeriss, our project incorporates quantization gating and our MUXipliers to optimize energy efficiency and performance in power-constrained environments.

Du et al. [3] present a reconfigurable streaming accelerator for DCNNs designed for Internet of Things (IoT) applications. The accelerator employs a streaming architecture to minimize memory access and maximize data reuse, thereby reducing power consumption. Additionally, the reconfigurable MAC units support precision-aware operations to adapt to varying workload requirements. This aligns with our focus on precision-aware low-power MAC designs, where we leverage quantization gating and our MUXipliers to achieve energy efficiency and throughput for edge-based IoT systems.

Lin et al. [4] examine low-power, ultra-small edge AI accelerators for image recognition tasks using convolutional neural networks (CNNs). Their work emphasizes energy-efficient strategies, such as precision reduction, optimized MAC unit architectures, and minimizing switching activity to achieve power savings. This aligns with our approach of designing quantization-aware MAC units with our MUXipliers to balance power consumption and performance. The relevance of Lin et al.'s focus on resource-constrained edge AI systems further reinforces our project’s objective to enable efficient neural network processing on edge devices.

Narayanan et al. [7] propose BOBBER, a prototyping platform for testing batteryless intermittent accelerators. BOBBER combines a low-power non-volatile FPGA with a microcontroller, enabling efficient execution of neural network designs in energy-harvesting environments. While our project aimed to implement and test low-power MAC units on BOBBER, time constraints prevented full integration. Nevertheless, BOBBER remains a

relevant platform for future validation of our MAC designs in batteryless and intermittent systems.

Hartstein and Puzak [8] explore the relationship between pipeline depth and optimal power-performance trade-offs. They demonstrate that deeper pipelines can improve performance but often increase power consumption, highlighting the need to balance these factors. Their insights emphasize the importance of optimizing computational units to achieve both high throughput and low power usage. Similarly, our project implements quantization gating and MUXipliers to strike this balance within our MAC unit designs, ensuring efficiency in constrained environments.

Suriya and Rani [9] investigate power consumption in MAC units using the modified Booth algorithm. Their findings show that the algorithm reduces power usage by minimizing partial products generated during multiplication, which lowers switching activity. This aligns with our initial idea of incorporating Booth's algorithm into our MAC designs to improve energy efficiency. While this approach was ultimately postponed due to time constraints, it remains a promising direction for future work aimed at further reducing power consumption.

Masadeh et al. [10] propose an input-conscious approximate MAC unit that enhances energy efficiency by simplifying operations when inputs are close to powers of two. This adaptive approach reduces overall energy consumption based on the input values. While our current work does not incorporate approximate multiplication due to time constraints, this idea closely aligns with our exploration of energy-efficient MAC designs. The input-aware approach offers an interesting avenue for future optimizations that could further reduce power consumption in resource-constrained systems.

4.0 Synthesis & Implementation Methodology

For each of our MAC units implemented, 2 synthesis runs, and 3 implementation runs per synthesis were done. We synthesized the designs with Vivavado's Synthesis Defaults as well as the Flow_PerfOptimized_high (PerfOptHigh) strategy. For implementation, we used Vivado's Default Implementation strategy, Performance_ExploreWithRemap (PerfOpt) strategy and Power_ExploreArea (PowerOpt) strategy.

5.0 Results

Key	1st Place	2nd Place	3rd Place

MAC Type	Synth Type	Impl Type	WNS	Total Power (W)	LUTs + FFs	BOPS	BOPS^3/W	BOPS^2/W	BOPS/W
SC_DRMAC	Default	Default	-0.507	0.167	225	0.250	0.0931	0.3729	1.494
		PerfOpt	-0.467	0.166	219	0.252	0.0965	0.3828	1.519
		PowerOpt	-1.213	0.164	225	0.212	0.0582	0.2745	1.294
	PerfOpt High	Default	-1.011	0.166	227	0.222	0.0656	0.2960	1.335
		PerfOpt	-0.320	0.170	216	0.262	0.1055	0.4031	1.540
		PowerOpt	-1.149	0.163	227	0.215	0.0611	0.2839	1.320
P2_DRMAC	Default	Default	-0.757	0.158	220	0.235	0.0820	0.3492	1.487
		PerfOpt	-0.479	0.153	212	0.251	0.1037	0.4128	1.643
		PowerOpt	-0.924	0.154	220	0.226	0.0750	0.3318	1.468
	PerfOpt High	Default	-0.685	0.156	226	0.239	0.0875	0.3660	1.532
		PerfOpt	-0.451	0.155	213	0.253	0.1046	0.4133	1.633
		PowerOpt	-0.763	0.153	221	0.235	0.0844	0.3596	1.533
SC_TILED	Default	Default	-1.330	0.146	272	0.207	0.0608	0.2936	1.418
		PerfOpt	-1.025	0.145	271	0.221	0.0744	0.3368	1.524
		PowerOpt	-1.587	0.143	272	0.197	0.0531	0.2702	1.375
	PerfOpt High	Default	-1.375	0.147	273	0.205	0.0587	0.2862	1.395
		PerfOpt	-0.647	0.148	266	0.241	0.0947	0.3929	1.629
		PowerOpt	-1.790	0.142	273	0.189	0.0476	0.2517	1.331
P2_TILED	Default	Default	-1.197	0.147	277	0.213	0.0656	0.3083	1.448
		PerfOpt	-1.015	0.146	273	0.221	0.0744	0.3360	1.517
		PowerOpt	-1.366	0.142	267	0.206	0.0611	0.2974	1.447
	PerfOpt High	Default	-1.056	0.152	288	0.219	0.0696	0.3169	1.444
		PerfOpt	-1.041	0.148	287	0.220	0.0722	0.3277	1.488
		PowerOpt	-1.398	0.142	272	0.204	0.0599	0.2935	1.438

Table 2: Results of MAC Synthesis and Implementation.

6.0 Discussion of Results

Looking at the results of our various synthesis and implementation runs, there's a pretty clear winner in terms of performance/power for our two MAC unit designs – the DRMAC. The main performance/power metric we're looking at here is $BOPS^3/W$ (Billion Operations per second / Watt) [8]. Our very best design in terms of $BOPS^3/W$ is shown to be our single-cycle DRMAC, with High-Performance Optimized synthesization and Performance Optimized implementation. On the contrary, our Tiled MAC is consistently more power efficient relative to the DRMAC.

These results highlight an important trade-off and impact that the differences between the two designs create. The Tiled MAC has demonstrated through the results to be approximately 10% more energy efficient relative to the DRMAC, whereas the DRMAC has shown to be approximately 28% faster relative to the Tiled MAC.

We were not able to test our MAC units with the BOBBER system due to time constraints on the designing of the MAC units and preparing it to be loaded into the system.

7.0 Expected Performance on Other Systems

We know that the static power is proportional to V^2, and dynamic power is proportional to V^2f, where f is the frequency. With this, we can theoretically calculate the MAC units' performance on various boards. We'll be referencing our single-cycle DRMAC with performance synthesization and performance implementation to calculate the theoretical performance, with a frequency of 262 MHz, 0.064W dynamic power, and 0.105W static power.

7.1 Xilinx Zynq-7020 SoC

The system that we analyzed these MAC units on is the Xilinx Zynq-7020 SoC (our lab FPGA). First, some relevant specs that the board has for our MACs:

- Theoretical maximum implemented frequency of 667 MHz, given Vivado uses the -1 speed setting for our board [11].
- Realistic maximum implemented frequency of ~450 MHz (from scrubbing forums), but generally around 250-300 MHz.
- The board's PIO blocks support voltages from 1.2 to 3.3V [11].

7.2 IGLOO nano AGLN250

Looking at another board, the IGLOO nano AGLN250, which is also used on BOBBER [7], we derive these specs:

- Supports 1.2 to 1.5V.
- Up to 250 MHz input/output [5].

Running the MAC unit at 1.5V and 250 MHz, we have a static power of 21.7 mW and dynamic power of 1.33 mW. At 1.2V and 250 MHz, we have a static power of 13.9 mW and dynamic power of 0.85 mW.

7.3 Lattice Avant-E Evaluation

Looking at one last board, the Avant-E Evaluation from Lattice, the self-proclaimed low power programmable leader, we get the following specs:

- Up to 350 MHz for the FPGA fabric.
- Runs at 0.82V in their core voltages [6].

Running the MAC unit at 0.82V and the same frequency, we have a static power metric of 6.48 mW and 3.95 mW of dynamic power:

These results prove that more power benefits can be achieved through dynamic voltage and frequency scaling (DVFS). It's worth noting we haven't fully explored the potential and realistic constraints of these boards, but we believe that even greater benefits can be achieved.

8.0 Documentation of Code Deliverables

8.1 The Fun Bits

Inside of our top-level project directory, you will find two sub-directories, this is where the fun (and only) stuff lies. The first subdirectory is "mac1", which contains our initial idea premise alongside both the single-cycle and pipelined Tiled MAC implementations. Either one of these single-cycle and pipelined implementations follows the regular structure as seen in our labs, with the hdl subdirectory containing our VHDL files, and the regular Vivado project files being found in the 'vivado' and 'xgui' sub-directories. Our design runs and everything involved can be found within 'vivado'.

8.2 Synthesizing & Implementing The MACs

To synthesize and run implementation on the MAC units, start by opening the project in Vivado. Once you have the project open, you should be able to view the detailed statistics of the multiple synthesization and implementation runs we've performed on our MAC units. In order to see the power metrics, you will need to re-run the synthesization and implementations. Make sure to select all of the synthesis and implementation runs and then press the green play button as shown in Figure 7 below.

Figure 7: Vivado Toolbar.

9.0 Challenges

9.1 Debugging

I love debugging, said no-one ever. We ran into quite a few problems with implementing our designs, which is a big part of why we ended up having to scrap a few of our ideas (2.4.2 and 2.4.3). After about 10 hours working on the shared aspects of design between our DRMAC and Tiled MAC, we found out that our multiplier/MUXiplier (mult_4x4.vhd and mult_8x8.vhd) – and subsequently the rest of our top-level design – needed complete reworking. After spending a few hours testing how to make our general approach for multipliers work, we finally found some behavioral VHDL statements for our partial products that we could modify to get the correct results.

```
P1 <= signed(A_high) * signed(B_high);  -- High part * High part
P2 <= signed(A_high) * signed('0' & B_low) when B_low(3) = '0' else
      -signed((-signed(A_high) * signed('0' & B_low)));   -- High part * Low part
P3 <= signed('0' & A_low) * signed(B_high) when A_low(3) = '0' else
      -signed((signed('0' & A_low) * (-signed(B_high))));   -- Low part * High part
P4 <= signed(A_low) * signed(B_low) when low_Or = '0' else
      signed((unsigned(A_low) * unsigned(B_low)));    -- Low part * Low part
```

With these modifications in mind, we created a few new MUXipliers, mux_4x4_lh.vhd and mux_4x4_uns.vhd, and implemented them in our designs.

9.2 Time

Another large source of challenges that we encountered for this project was personal scheduling. Given the great disparity between our class and personal schedules, finding times to meet and properly work on the project were few and far between. Regardless, we decided we would do everything in our power to make it work and managed to squeeze out a few hours in some successive days.

References

[1] Sim, J.; Park, J.-S.; Kim, M.; Bae, D.; Choi, Y.; Kim, L.-S. A 1.42TOPS/W deep convolution neural network recognition processor for intelligent IoE systems. In Proceedings of the 2016 IEEE International Solid-State Circuits Conference, San Francisco, CA, USA, 31 January–4 February 2016, doi: 10.1109/ISSCC.2016.7418008.

[2] Chen, Y.-H.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE J. Solid State Circuits 2017, 52, 262–263, doi: 10.1109/JSSC.2016.2616357.

[3] Du, L.; Du, Y.; Li, Y.; Su, J.; Kua, Y.-C.; Liu, C.-C.; Chang, M.C.F. A Reconfigurable Streaming Deep Convolutional Neural Network Accelerator for Internet of Things. IEEE Trans. Circuits Syst. 2018, 65, 198–208, doi: 10.1109/TCSI.2017.2735490.

[4] Lin, W.; Adetomi, A.; Arslan, T. Low-Power Ultra-Small Edge AI Accelerators for Image Recognition with Convolution Neural Networks: Analysis and Future Directions. Electronics 2021, 10, 2048. doi: 10.3390/electronics10172048

[5] Microsemi 2020. IGLOO nano Low Power Flash FPGAs Datasheet. Microsemi.

[6] Lattice 2024. Avant Evaluation Board - FPGA-EB-02057-1-6-Avant-Evaluation-Board. Lattice.

[7] V. Narayanan, R. Sahu, J. Sun, and H. Duwe, “BOBBER A Prototyping Platform for Batteryless Intermittent Accelerators,” in Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2023, pp. 221–228. doi: 10.1145/3543622.3573046.

[8] A. Hartstein and T. R. Puzak, “Optimum power/performance pipeline depth,” in Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 36, (Washington, DC, USA), pp. 117–, IEEE Computer Society, 2003, doi: 10.1145/1044823.1044824

[9] T. S. U. Suriya and A. A. Rani, "Low power analysis of MAC using modified booth algorithm," 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, India, 2013, pp. 1-5, doi: 10.1109/ICCCNT.2013.6726527.

[10] M. Masadeh, O. Hasan and S. Tahar, "Input-Conscious Approximate Multiply-Accumulate (MAC) Unit for Energy-Efficiency," in IEEE Access, vol. 7, pp. 147129-147142, 2019, doi: 10.1109/ACCESS.2019.2946513.

[11] Xilinx 2018. Zynq-7000 SoC Data Sheet: Overview. Xilinx