Team Members: Connor Kuczynski, Cory Huynh, Grace Jin, Mudi Huang

ECE 260B Final Project Report - Winter 2024

Team: CoMingtOShort

Deliverables Paths

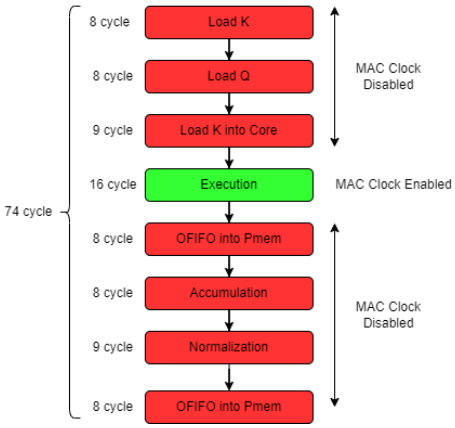
Step	Path
1	/home/linux/ieng6/ee260bwi24/hojin/ECE260B_FinalProject/Grace/step1/deliverable
2	/home/linux/ieng6/ee260bwi24/hojin/ECE260B_FinalProject/Grace/step2/deliverable
3	/home/linux/ieng6/ee260bwi24/hojin/ECE260B_FinalProject/Grace/step3/deliverable
4	/home/linux/ieng6/ee260bwi24/hojin/ECE260B_FinalProject/Grace/step4/deliverable
5	/home/linux/ieng6/ee260bwi24/hojin/ECE260B_FinalProject/Grace/step5/deliverable

Introduction

In the context of expanding demands for machine learning capabilities, enhancing the adaptability, performance, and energy consumption of ML accelerators is essential. The need for higher performance per watt becomes important for scalability. To address this, our designed accelerator aims to achieve higher throughput and energy efficiency compared to traditional scalar processing architectures. Our design process started with a single-core design. After we verified the functionality and efficacy of the architecture, we moved onto a dual-core design for larger computations. Finally, we analyzed the power, timing and area reports of this design and implemented multiple optimization techniques in order to improve the frequency, minimize the power consumption, and minimize the area of our design.

Alpha1 - Clock Gating

To optimize power consumption, clock gating techniques were strategically implemented in our design. Analysis of the testbench revealed that the MAC Array submodule is actively utilized solely during the execution stage. Therefore, we implemented clock gating logic for the MAC Array, effectively disabling the clock signal to this submodule when its functionality is not required. This approach ensures that the MAC Array remains in an non-switching state and does not consume dynamic power during the remaining operational stages. As a result, clock gating ensures power efficiency without compromising functional correctness.



To quantify the impact of clock gating on power optimization, we conducted a comparative analysis across three design configurations: the synthesis power, the pnr power, and the power with clock-gated vcd. As we can see from the below table, both synthesis and pnr power are both relatively high at above 600mW. However, a significant improvement was achieved after implementing clock gated VCD, which demonstrated a remarkable 73.97% reduction in total power consumption, consuming only 200.45 mW during the measured operational stages. **As Professor Mingu mentioned during our poster presentation, the VCD power calculation excluded the data loading stages to reflect realistic power consumption.**

Units (mW)	Internal Power	Switching Power	Leakage Power	Total Power	Power reduction
Synthesis	96.9033	671.0560	2.0138	769.9726	N/A
PNR	353.80515673	247.43905255	3.44975380	604.69396316	21.5%
Clk-Gated VCD	155.92022002	40.99203742	3.53522366	200.44747981	73.97%

Table 2 - Power consumption table (excluding loading phases)

Alpha2 - Dual-Port Psum Memory

Observing the operation workflow, several opportunities to improve our throughput were identified. Originally, during the division process of normalization, psum needed to be read from pmem (Psum Memory) and then written back to ofifo after normalization. Another stage is required to move the result from ofifo to pmem. To improve the throughput and increase the parallelism, we decided to enhance the psum memory architecture by converting it to a dual-port design, enabling write and read operations at the same time. This is done by providing two separate address pointers as inputs for these two operations. Now during normalization, data is read and the result is written to psum memory in the same cycles. As a result, the total clock cycle needed for the two operations was reduced by 8 cycles. After implementing both the changes, we verified the functionality of the design and proved that this improved throughput by over 10%. The throughput data was measured by counting the total cycles needed for the K*Q operation (including normalization).

Stage	Cycles
load K	8
load Q	8
load k into core	9
Execution	16
Move OFIFO to Pmem	8
Accumulation	8
Normalization	9
Move OFIFO to Pmem	8→0

Team Members: Connor Kuczynski, Cory Huynh, Grace Jin, Mudi Huang

Alpha3 - Multi-Cycle Path

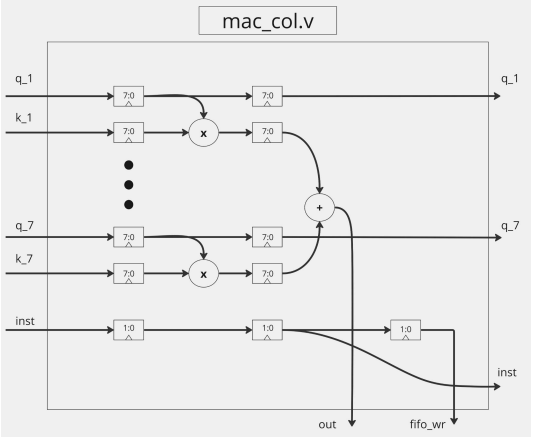
In order to improve the frequency performance, based on observation of the timing report, it is suggested that the time critical path is in the division stage during the normalization process. However, we cannot apply pipelining to it since it is a single operation. Thus we applied a three-cycle multicycle path to the division paths in order to achieve a 1GHz clock frequency.

WNS before Multi-Cycle Path	WNS after Multi-Cycle Path
-1.716 ns	-0.025 ns

As we can see in the above table, after the Multi-Cycle path is applied, we obtained a worst negative slack of -0.025ns, compared to -1.716ns before. Most importantly, the new critical path is no longer the division path. That means we can further improve our operation frequency by applying other techniques in other areas of the chip. Therefore, this technique not only improved our performance, but also enabled us for further optimization opportunities.

Alpha4 - Pipelining

After applying the multi cycle pathing to the division operation in the normalization process. The critical path change to the MAC performed for each column. The primary reason for applying pipelining to the design is reducing the propagation delay from the registers through both the multiplier and addition. The pipelining occurs between the multiplication and addition stages while also pipelining the q and instruction values to prevent the values from being passed to the next column early. We saw after applying the pipelining that the critical path goes back to the divider so we decided not to further pipeline the additional stages as those are now off the critical path. We successfully verified with the testbench the functionality of the design.



WNS before Pipelining (with Multi-Cycle Path)	WNS after Pipelining (with Multi-Cycle Path)
-0.025ns	0 ns

Alpha5 - Flattened pnr

To further optimize our design's area cost, we decided to flatten the pnr and achieved a significant area saving.

Chip area using hierarchical pnr	Chip area using flattened pnr	Area reduction
1856400.00 μm^2	699731.04 μm^2	62.3 %

Alpha6 - DRC check

We have used eco techniques to achieve 0 DRC violations in our designs. Please see the right screenshot from final design pnr.

```
VERIFY DRC ..... Sub-Area : 25 of 25
VERIFY DRC ..... Sub-Area : 25 complete 0 Viols.

Verification Complete : 0 Viols.

*** End Verify DRC (CPU: 0:00:54.6  ELAPSED TIME: 55.00  MEM: 74.6M) ***
```

Conclusion

As discussed in the above sections, multiple design optimization techniques were implemented in the design. We have explored many techniques to improve various aspects of the design metrics from improving the throughput, power consumption, to frequency performance. During the project, we learned to balance the trade-off between latency and throughput, power, area and performance. However, by combining multiple different techniques, we are able to systematically improve both performance and reduce power in our final iteration. At the end, we finally met the 1GHz clock frequency with significant power reduction and throughput enhancement. As a team, we learned a valuable lesson regarding the VLSI design flow from RTL to GDS. Future improvements to the design include changing the architecture to a 2D systolic array where the data K and Q values are loaded simultaneously. Additionally, using pipelining to couple stages together execution and memory loading operations can also help performance.

Appendix

Step1. Single core RTL, Synthesis & PNR

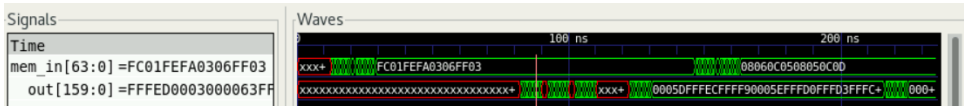
- **Waveform vcd from gate-level sim**

```
##### Estimated multiplication result #####
prd @cycle 0: 00006fffa3000640000cfffa0fffe0fffea00003
prd @cycle 1: 0002300001000023fffe0ffffc00002ffff00004b
prd @cycle 2: fffc6fffa50005400030ffffde0000a0002cffff0
prd @cycle 3: 00005fffd0000050fffe7ffffabfffaffffeafffbd
prd @cycle 4: 00028ffff80004b00038ffffd6ffffdffffd20000b
prd @cycle 5: 000420004 0ffffd0002e000250005f00024ffff6
prd @cycle 6: 0006100075fffd60001d000260002bfffffffd3
prd @cycle 7: 00004fffef0001a0000b0000000008ffff40001e
##### Qmem writing  #####
##### Kmem writing #####
##### K data loading to processor #####
##### execute #####
##### move ofifo to pmem #####
###### read final output from pmem #######
Memory read from PSUM mem address:           0 data: 00006fffa3000640000cfffa0fffe0fffea00003
Memory read from PSUM mem address:           1 data: 0002300001000023fffe0ffffc00002ffff00004b
Memory read from PSUM mem address:           2 data: fffc6fffa50005400030ffffde0000a0002cffff0
Memory read from PSUM mem address:           3 data: 00005fffd0000050fffe7ffffabfffaffffeafffbd
Memory read from PSUM mem address:           4 data: 00028ffff80004b00038ffffd6ffffdffffd20000b
Memory read from PSUM mem address:           5 data: 000420004 0ffffd0002e000250005f00024ffff6
Memory read from PSUM mem address:           6 data: 0006100075fffd60001d000260002bfffffffd3
Memory read from PSUM mem address:           7 data: 00004fffef0001a0000b0000000008ffff40001e
Simulation complete via $finish(1) at time 100500 PS + 0
./netlist/fullchip_tb.v:370    #10 $finish;
```

- **Gate-level sim console output**

Step2. Output normalization

- **Waveform vcd from behavioral sim snapshot**



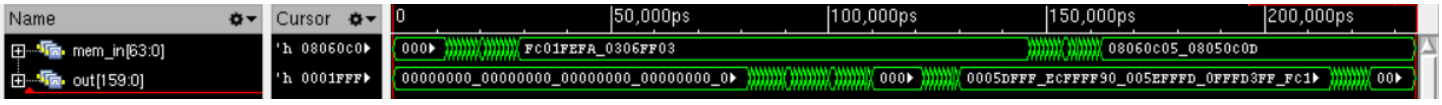
- **Testbench output**

```
###### read normalized for Q*K output from pmem #######
Result Matched! Memory read from PSUM mem address:  0 data: 0001fffffaffffe0001fffff0ffff1fffebfffe4, expect: 0001fffffaffffe0001fffff0ffff1fffebfffe4
Result Matched! Memory read from PSUM mem address:  1 data: 0001affff6fffdc0000cffffefffff0fffecffff5, expect: 0001affff6fffdc0000cffffefffff0fffecffff5
Result Matched! Memory read from PSUM mem address:  2 data: 000130000d000190001fffff20000affffffffe5, expect: 000130000d000190001fffff20000affffffffe5
Result Matched! Memory read from PSUM mem address:  3 data: 00010fffdb000080003effffeffff8ffff3fffe7, expect: 00010fffdb000080003effffeffff8ffff3fffe7
Result Matched! Memory read from PSUM mem address:  4 data: 0001afffffdfffd00004fffd2ffff4fffe2fffee, expect: 0001afffffdfffd00004fffd2ffff4fffe2fffee
Result Matched! Memory read from PSUM mem address:  5 data: 0002a000010001a000440000700007ffff8fffd3, expect: 0002a000010001a000440000700007ffff8fffd3
Result Matched! Memory read from PSUM mem address:  6 data: ffff70001800031ffffd000000000b00017ffff6, expect: ffff70001800031ffffd000000000b00017ffff6
Result Matched! Memory read from PSUM mem address:  7 data: fffeb0000d0002affff1000000001a000180000c, expect: fffeb0000d0002affff1000000001a000180000c

###### read V*N output from pmem #######
Result Matched! Memory read from PSUM mem address:  0
data:   0000b0003e0003c0006e00066fff7dfffc800056       expect: 0000b0003e0003c0006e00066fff7dfffc800056
Result Matched! Memory read from PSUM mem address:  1
data:   000450002b000530000100013005d000840001b       expect: 000450002b000530000100013005d000840001b
Result Matched! Memory read from PSUM mem address:  2
data:   fff1ffffe1fffd1fffb9fffdffff9dfff73fffaa       expect: fff1ffffe1fffd1fffb9fffdffff9dfff73fffaa
Result Matched! Memory read from PSUM mem address:  3
data:   fff32fff36ffeebfff7dfffa0fff50fff3dfff51       expect: fff32fff36ffeebfff7dfffa0fff50fff3dfff51
Result Matched! Memory read from PSUM mem address:  4
data:   fffa5fffadfffd5fffa7fffb00001dfffeffffc5       expect: fffa5fffadfffd5fffa7fffb00001dfffeffffc5
Result Matched! Memory read from PSUM mem address:  5
data:   ffee1fff49ffeecfff3dfff4afff0bfff2afff2d       expect: ffee1fff49ffeecfff3dfff4afff0bfff2afff2d
Result Matched! Memory read from PSUM mem address:  6
data:   0001bfffbbfffd2fffedffff50008300052fffee       expect: 0001bfffbbfffd2fffedffff50008300052fffee
Result Matched! Memory read from PSUM mem address:  7
data:   fff9ffffdcfff84fffd1fff90fff61fff42fffa7       expect: fff9ffffdcfff84fffd1fff90fff61fff42fffa7
```

Step3. Hierarchical Synthesis of core

- **Waveform vcd from gate-level sim snapshot**



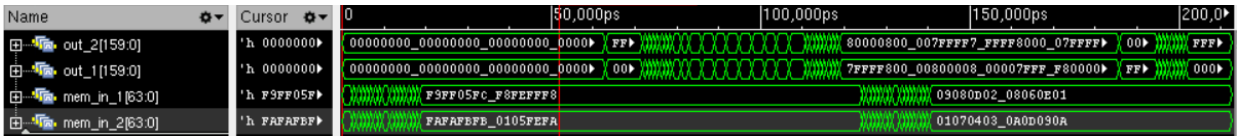
- **Console output**

```
######read normalized for Q*K output from pmem #######
Result Matched! Memory read from PSUM mem address:  0 data: 0001fffffaffffe0001fffff0ffff1fffebfffe4, expect: 0001fffffaffffe0001fffff0ffff1fffebfffe4
Result Matched! Memory read from PSUM mem address:  1 data: 0001affff6fffdc0000cffffefffff0fffecffff5, expect: 0001affff6fffdc0000cffffefffff0fffecffff5
Result Matched! Memory read from PSUM mem address:  2 data: 000130000d000190001fffff20000affffffffe5, expect: 000130000d000190001fffff20000affffffffe5
Result Matched! Memory read from PSUM mem address:  3 data: 00010fffdb000080003effffeffff8ffff3fffe7, expect: 00010fffdb000080003effffeffff8ffff3fffe7
Result Matched! Memory read from PSUM mem address:  4 data: 0001afffffdfffd00004fffd2ffff4fffe2fffee, expect: 0001afffffdfffd00004fffd2ffff4fffe2fffee
Result Matched! Memory read from PSUM mem address:  5 data: 0002a000010001a000440000700007ffff8fffd3, expect: 0002a000010001a000440000700007ffff8fffd3
Result Matched! Memory read from PSUM mem address:  6 data: ffff70001800031ffffd000000000b00017ffff6, expect: ffff70001800031ffffd000000000b00017ffff6
Result Matched! Memory read from PSUM mem address:  7 data: fffeb0000d0002affff1000000001a000180000c, expect: fffeb0000d0002affff1000000001a000180000c
##################No error. Good Job!#####################

###### read V*N output from pmem #######
Result Matched! Memory read from PSUM mem address:  0
data:   0000b0003e0003c0006e00066fff7dfffc800056       expect: 0000b0003e0003c0006e00066fff7dfffc800056
Result Matched! Memory read from PSUM mem address:  1
data:   000450002b000530000100013005d000840001b       expect: 000450002b000530000100013005d000840001b
Result Matched! Memory read from PSUM mem address:  2
data:   fff1ffffe1fffd1fffb9fffdffff9dfff73fffaa       expect: fff1ffffe1fffd1fffb9fffdffff9dfff73fffaa
Result Matched! Memory read from PSUM mem address:  3
data:   fff32fff36ffeebfff7dfffa0fff50fff3dfff51       expect: fff32fff36ffeebfff7dfffa0fff50fff3dfff51
Result Matched! Memory read from PSUM mem address:  4
data:   fffa5fffadfffd5fffa7fffb00001dfffeffffc5       expect: fffa5fffadfffd5fffa7fffb00001dfffeffffc5
Result Matched! Memory read from PSUM mem address:  5
data:   ffee1fff49ffeecfff3dfff4afff0bfff2afff2d       expect: ffee1fff49ffeecfff3dfff4afff0bfff2afff2d
Result Matched! Memory read from PSUM mem address:  6
data:   0001bfffbbfffd2fffedffff50008300052fffee       expect: 0001bfffbbfffd2fffedffff50008300052fffee
Result Matched! Memory read from PSUM mem address:  7
data:   fff9ffffdcfff84fffd1fff90fff61fff42fffa7       expect: fff9ffffdcfff84fffd1fff90fff61fff42fffa7
##################No error. Good Job!#####################
```

Step4. Hierarchical Synthesis of dual core

- **Waveform vcd from gate-level sim snapshot**



- **Console output**

```
###### read normalized output from pmem #######
Result Matched! Memory read from PSUM mem address:  0
data:   00002fffe7ffff9fffee00002ffff4000100001dfffffffffa0000700005ffffd0000b00007ffff2
expect: 00002fffe7ffff9fffee00002ffff4000100001dfffffffffa0000700005ffffd0000b00007ffff2


Result Matched! Memory read from PSUM mem address:  1
data:   0000ffffe8ffff0ffffc0000afffff30000d0001efffff800007000050000f000010000000003000000
expect: 0000ffffe8ffff0ffffc0000afffff30000d0001efffff800007000050000f000010000000003000000


Result Matched! Memory read from PSUM mem address:  2
data:   ffffdfffe8ffff7ffff500001ffff6000060002000000ffff7ffff800002ffff80001200007ffff5
expect: ffffdfffe8ffff7ffff500001ffff6000060002000000ffff7ffff800002ffff80001200007ffff5


Result Matched! Memory read from PSUM mem address:  3
data:   ffff900009ffff8fffc9fffffe0001100005ffffb00006fffe500007ffff700001ffffd00006ffff2
expect: ffff900009ffff8fffc9fffffe0001100005ffffb00006fffe500007ffff700001ffffd00006ffff2


Result Matched! Memory read from PSUM mem address:  4
data:   ffffaffff000002ffff2fffdd00003000002ffffd00005fffe8ffff9fffe800003ffff90001000003
expect: ffffaffff000002ffff2fffdd00003000002ffffd00005fffe8ffff9fffe800003ffff90001000003


Result Matched! Memory read from PSUM mem address:  5
data:   00007ffff7fffffffffeb00013ffff5000070002700008ffff4ffffb00009ffffc0002500009ffff2
expect: 00007ffff7fffffffffeb00013ffff5000070002700008ffff4ffffb00009ffffc0002500009ffff2


Result Matched! Memory read from PSUM mem address:  6
data:   ffffb000140001500005ffffdffffbfffffafffee00008ffff2ffffaffffeffffffc00006ffffcffffb
expect: ffffb000140001500005ffffdffffbfffffafffee00008ffff2ffffaffffeffffffc00006ffffcffffb


Result Matched! Memory read from PSUM mem address:  7
data:   fffe6000030000300017ffff800009ffffedfffecfffffe00006ffffaffff7ffff600001000006ffff6
expect: fffe6000030000300017ffff800009ffffedfffecfffffe00006ffffaffff7ffff600001000006ffff6


################No error on kq test. Good Job!####################
```

```
###### read final output from pmem #######
Data Matched! Cycle:  0
Data:ffff900023fff6afffc900021fff80fff4c000520001fffffffffc80001cffff8fffd4fffedfffe0
Expected:ffff900023fff6afffc900021fff80fff4c000520001fffffffffc80001cffff8fffd4fffedfffe0
Data Matched! Cycle:  1
Data:00026000680006b00038000500008500050000280000c0009700056ffffb000550002100046000 0f
Expected:00026000680006b00038000500008500050000280000c0009700056ffffb000550002100046000 0f
Data Matched! Cycle:  2
Data:fff85fffa1fff59fffaffff95fff0dfff4100017fffabfffa2fff88fff39fffacfffa4fff3ffffb5
Expected:fff85fffa1fff59fffaffff95fff0dfff4100017fffabfffa2fff88fff39fffacfffa4fff3ffffb5
Data Matched! Cycle:  3
Data:fff72fff53ffef4fff3ffff2ffffef6fff12fff9bfff86ffef2ffee7fff65fff76fffb0fff49fff5e
Expected:fff72fff53ffef4fff3ffff2ffffef6fff12fff9bfff86ffef2ffee7fff65fff76fffb0fff49fff5e
Data Matched! Cycle:  4
Data:fffc4fffd3fffa3fffaffffcfffffc5fffff70000600012fff9dfffddfffd8fffa3fffc2ffffa00002
Expected:fffc4fffd3fffa3fffaffffcfffffc5fffff70000600012fff9dfffddfffd8fffa3fffc2ffffa00002
Data Matched! Cycle:  5
Data:ffee5ffef4ffeecfff07fff09ffefcffefbfff48fff7ffffee6ffef7fff4ffff19fff55fff28fff57
Expected:ffee5ffef4ffeecfff07fff09ffefcffefbfff48fff7ffffee6ffef7fff4ffff19fff55fff28fff57
Data Matched! Cycle:  6
Data:0003a0004e0000d000240002700036000 4c000240001700027ffff7ffffee0003b0003e00033ffffc
Expected:0003a0004e0000d000240002700036000 4c000240001700027ffff7ffffee0003b0003e00033ffffc
Data Matched! Cycle:  7
Data:fff94fff1bfffc1fffb2fff4cfffbbfffb2fff4ffffc2fff16fffc100026fff40fffb5fffb60000d
Expected:fff94fff1bfffc1fffb2fff4cfffbbfffb2fff4ffffc2fff16fffc100026fff40fffb5fffb60000d


################No error on N*V test.  Good Job!####################
```

Step5. Optimization & PNR

- **Waveform vcd from gate-level sim snapshot**

- **Console output**

```
###### read normalized output from pmem #######
Result Matched! Memory read from PSUM mem address:  0
data:   00002fffe7ffff9fffee00002ffff4000100001dfffffffffa0000700005ffffd0000b00007ffff2        expect: 00002fffe7ffff9fffee00002ffff4000100001dfffffffffa0000700005ffffd0000b00007ffff2
Result Matched! Memory read from PSUM mem address:  1
data:   0000ffffe8ffff0ffffc0000afffff30000d0001efffff800007000050000f000010000000003000000  expect: 0000ffffe8ffff0ffffc0000afffff30000d0001efffff800007000050000f000010000000003000000
Result Matched! Memory read from PSUM mem address:  2
data:   ffffdfffe8ffff7ffff500001ffff6000060002000000ffff7ffff800002ffff80001200007ffff5  expect: ffffdfffe8ffff7ffff500001ffff6000060002000000ffff7ffff800002ffff80001200007ffff5
Result Matched! Memory read from PSUM mem address:  3
data:   ffff900009ffff8fffc9fffffe0001100005ffffb00006fffe500007ffff700001ffffd00006ffff2  expect: ffff900009ffff8fffc9fffffe0001100005ffffb00006fffe500007ffff700001ffffd00006ffff2
Result Matched! Memory read from PSUM mem address:  4
data:   ffffaffff000002ffff2fffdd00003000002ffffd00005fffe8ffff9fffe800003ffff90001000003  expect: ffffaffff000002ffff2fffdd00003000002ffffd00005fffe8ffff9fffe800003ffff90001000003
Result Matched! Memory read from PSUM mem address:  5
data:   00007ffff7fffffffffeb00013ffff5000070002700008ffff4ffffb00009ffffc0002500009ffff2  expect: 00007ffff7fffffffffeb00013ffff5000070002700008ffff4ffffb00009ffffc0002500009ffff2
Result Matched! Memory read from PSUM mem address:  6
data:   ffffb000140001500005ffffdffffbfffffafffee00008ffff2ffffaffffeffffffc00006ffffcffffb  expect: ffffb000140001500005ffffdffffbfffffafffee00008ffff2ffffaffffeffffffc00006ffffcffffb
Result Matched! Memory read from PSUM mem address:  7
data:   fffe6000030000300017ffff800009ffffedfffecfffffe00006ffffaffff7ffff600001000006ffff6  expect: fffe6000030000300017ffff800009ffffedfffecfffffe00006ffffaffff7ffff600001000006ffff6
################No error on kq test. Good Job!####################
###### read final output from pmem #######
Data Matched! Cycle:  0
Data:ffff900023fff6afffc900021fff80fff4c000520001fffffffffc80001cffff8fffd4fffedfffe0
Expected:ffff900023fff6afffc900021fff80fff4c000520001fffffffffc80001cffff8fffd4fffedfffe0
Data Matched! Cycle:  1
Data:00026000680006b00038000500008500050000280000c0009700056ffffb000550002100046000 0f
Expected:00026000680006b00038000500008500050000280000c0009700056ffffb000550002100046000 0f
Data Matched! Cycle:  2
Data:fff85fffa1fff59fffaffff95fff0dfff4100017fffabfffa2fff88fff39fffacfffa4fff3ffffb5
Expected:fff85fffa1fff59fffaffff95fff0dfff4100017fffabfffa2fff88fff39fffacfffa4fff3ffffb5
Data Matched! Cycle:  3
Data:fff72fff53ffef4fff3ffff2ffffef6fff12fff9bfff86ffef2ffee7fff65fff76fffb0fff49fff5e
Expected:fff72fff53ffef4fff3ffff2ffffef6fff12fff9bfff86ffef2ffee7fff65fff76fffb0fff49fff5e
Data Matched! Cycle:  4
Data:fffc4fffd3fffa3fffaffffcfffffc5fffff70000600012fff9dfffddfffd8fffa3fffc2ffffa00002
Expected:fffc4fffd3fffa3fffaffffcfffffc5fffff70000600012fff9dfffddfffd8fffa3fffc2ffffa00002
Data Matched! Cycle:  5
Data:ffee5ffef4ffeecfff07fff09ffefcffefbfff48fff7ffffee6ffef7fff4ffff19fff55fff28fff57
Expected:ffee5ffef4ffeecfff07fff09ffefcffefbfff48fff7ffffee6ffef7fff4ffff19fff55fff28fff57
Data Matched! Cycle:  6
Data:0003a0004e0000d000240002700036000 4c000240001700027ffff7ffffee0003b0003e00033ffffc
Expected:0003a0004e0000d000240002700036000 4c000240001700027ffff7ffffee0003b0003e00033ffffc
Data Matched! Cycle:  7
Data:fff94fff1bfffc1fffb2fff4cfffbbfffb2fff4ffffc2fff16fffc100026fff40fffb5fffb60000d
Expected:fff94fff1bfffc1fffb2fff4cfffbbfffb2fff4ffffc2fff16fffc100026fff40fffb5fffb60000d


################No error on N*V test.  Good Job!####################
```

Team Members: Connor Kuczynski, Cory Huynh, Grace Jin, Mudi Huang