REPORT

- Tanmay Aron (2023JVL2240)

Primitive Mapping Implementation:

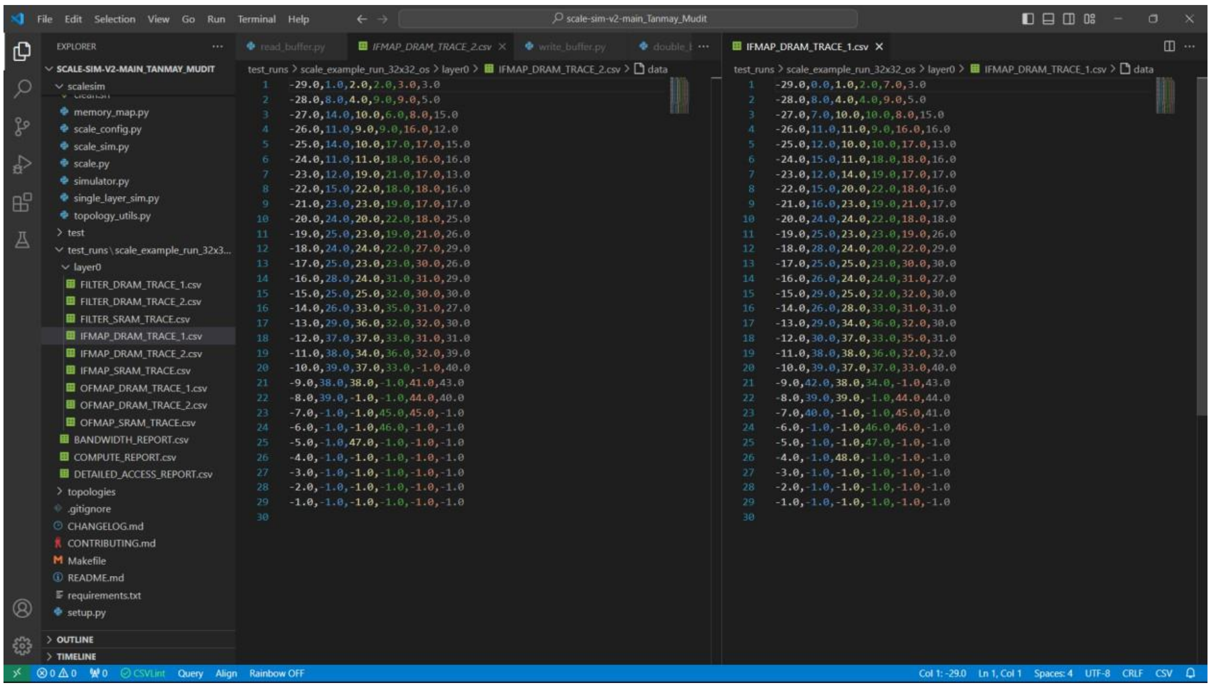
Exploring multiple memory bank modes in Scale- sim-v2. Data from local memory is distributed across different off-chip DRAMs. As a primitive mapping technique, data is cyclically distributed into 2 banks.

Read Buffer: -

- In this implementation, I've sent a row of requests into 2 DRAMs.

For each row, I am dividing addresses alternatively into both DRAMs.

Both the DRAMs are accessing elements parallelly.



IFMAP_DRAM_TRACE_2.csv

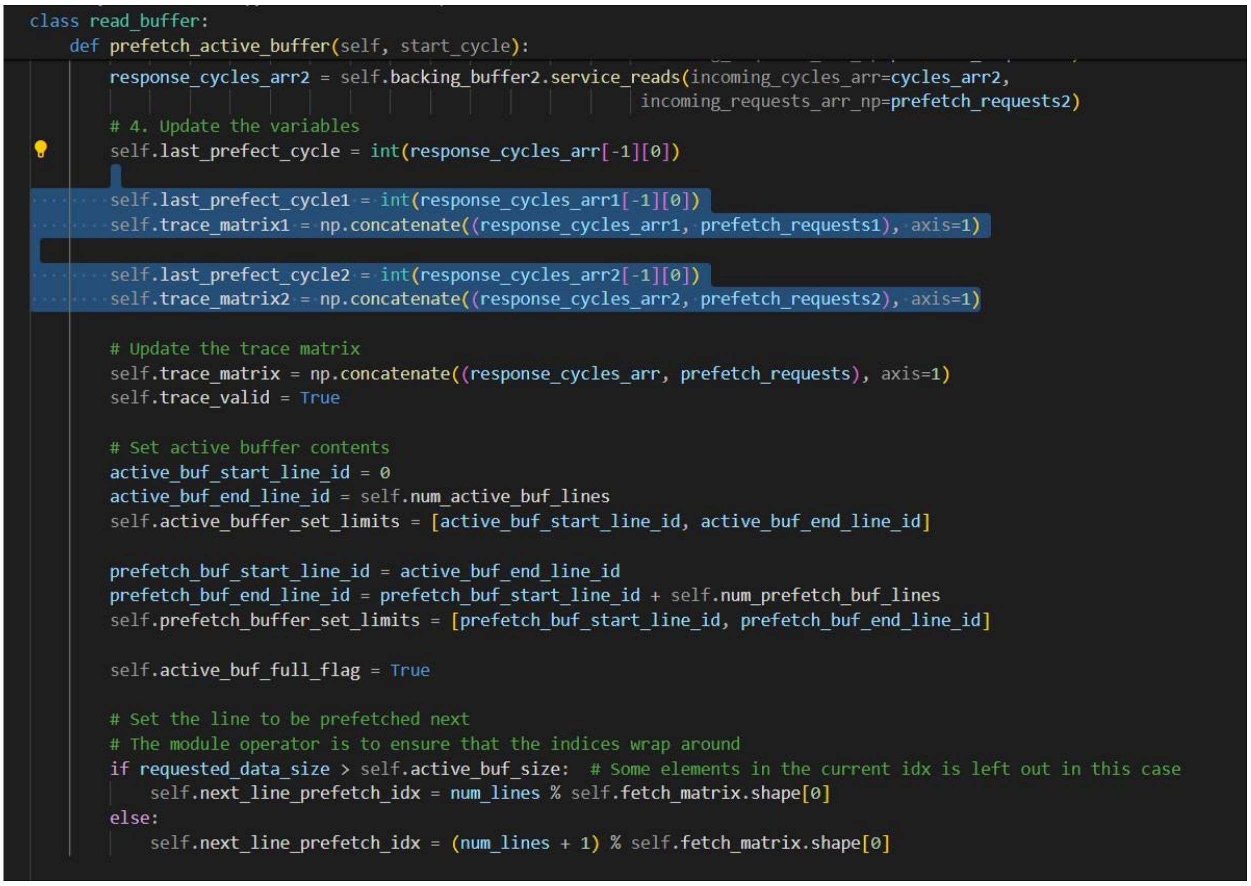
```
-29.0,1.0,2.0,2.0,3.0,3.0
-28.0,8.0,4.0,9.0,9.0,5.0
-27.0,14.0,10.0,6.0,8.0,15.0
-26.0,11.0,9.0,9.0,16.0,12.0
-25.0,14.0,10.0,17.0,17.0,15.0
-24.0,11.0,11.0,18.0,16.0,16.0
-23.0,12.0,19.0,21.0,17.0,13.0
-22.0,15.0,22.0,18.0,18.0,16.0
-21.0,23.0,23.0,19.0,17.0,17.0
-20.0,24.0,20.0,22.0,18.0,25.0
-19.0,25.0,23.0,19.0,21.0,26.0
-18.0,24.0,24.0,22.0,27.0,29.0
-17.0,25.0,23.0,23.0,30.0,26.0
-16.0,28.0,24.0,31.0,31.0,29.0
-15.0,25.0,25.0,32.0,30.0,30.0
-14.0,26.0,33.0,35.0,31.0,27.0
-13.0,29.0,36.0,32.0,32.0,30.0
-12.0,37.0,37.0,33.0,31.0,31.0
-11.0,38.0,34.0,36.0,32.0,39.0
-10.0,39.0,37.0,33.0,-1.0,40.0
-9.0,38.0,38.0,-1.0,41.0,43.0
-8.0,39.0,-1.0,-1.0,44.0,40.0
-7.0,-1.0,-1.0,45.0,45.0,-1.0
-6.0,-1.0,-1.0,46.0,-1.0,-1.0
-5.0,-1.0,47.0,-1.0,-1.0,-1.0
-4.0,-1.0,-1.0,-1.0,-1.0,-1.0
-3.0,-1.0,-1.0,-1.0,-1.0,-1.0
-2.0,-1.0,-1.0,-1.0,-1.0,-1.0
-1.0,-1.0,-1.0,-1.0,-1.0,-1.0
```

IFMAP_DRAM_TRACE_1.csv

```
-29.0,0.0,1.0,2.0,7.0,3.0
-28.0,8.0,4.0,4.0,9.0,5.0
-27.0,7.0,10.0,10.0,8.0,15.0
-26.0,11.0,11.0,9.0,16.0,16.0
-25.0,12.0,10.0,10.0,17.0,13.0
-24.0,15.0,11.0,18.0,18.0,16.0
-23.0,12.0,14.0,19.0,17.0,17.0
-22.0,15.0,20.0,22.0,18.0,16.0
-21.0,16.0,23.0,19.0,21.0,17.0
-20.0,24.0,24.0,22.0,18.0,18.0
-19.0,25.0,23.0,23.0,19.0,26.0
-18.0,28.0,24.0,20.0,22.0,29.0
-17.0,25.0,25.0,23.0,30.0,30.0
-16.0,26.0,24.0,24.0,31.0,27.0
-15.0,29.0,25.0,32.0,32.0,30.0
-14.0,26.0,28.0,33.0,31.0,31.0
-13.0,29.0,34.0,36.0,32.0,30.0
-12.0,30.0,37.0,33.0,35.0,31.0
-11.0,38.0,38.0,36.0,32.0,32.0
-10.0,39.0,37.0,37.0,33.0,40.0
-9.0,42.0,38.0,34.0,-1.0,43.0
-8.0,39.0,39.0,-1.0,44.0,44.0
-7.0,40.0,-1.0,-1.0,45.0,41.0
-6.0,-1.0,-1.0,46.0,46.0,-1.0
-5.0,-1.0,-1.0,47.0,-1.0,-1.0
-4.0,-1.0,48.0,-1.0,-1.0,-1.0
-3.0,-1.0,-1.0,-1.0,-1.0,-1.0
-2.0,-1.0,-1.0,-1.0,-1.0,-1.0
-1.0,-1.0,-1.0,-1.0,-1.0,-1.0
```

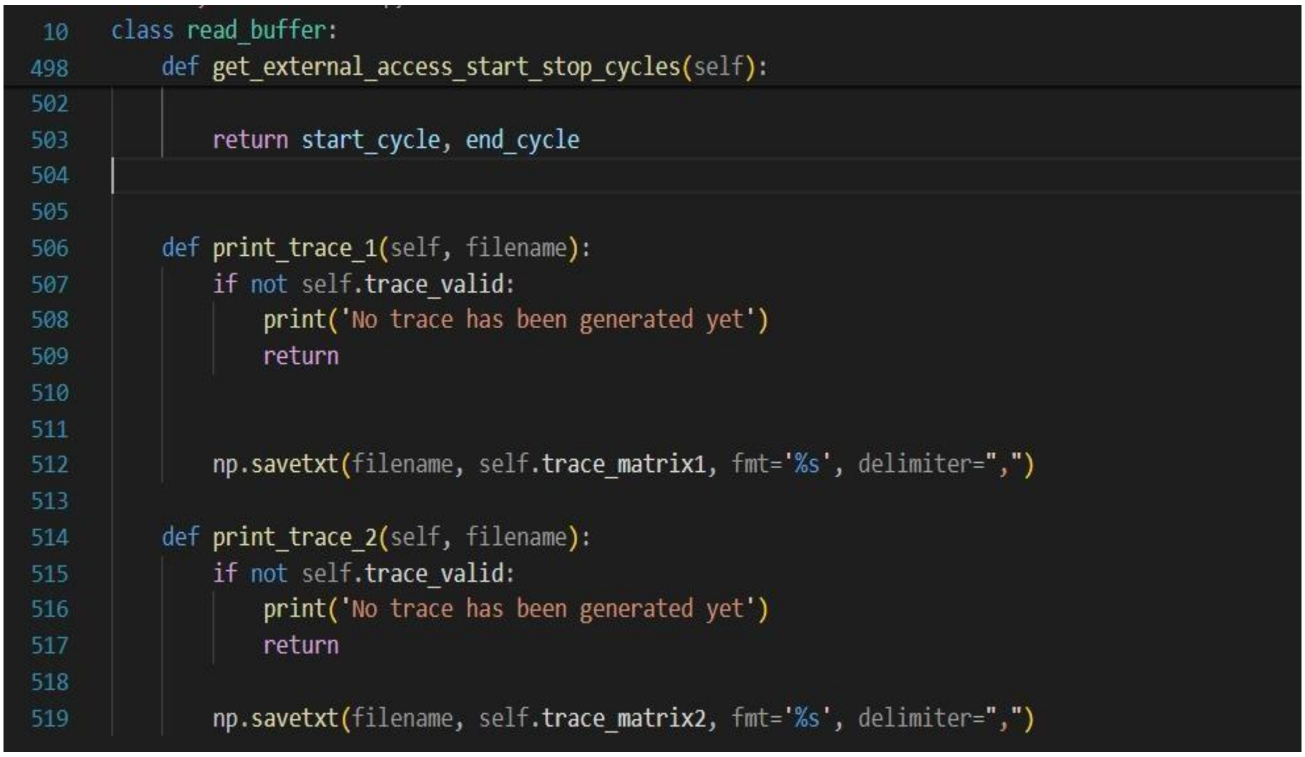
- In the below function prefetch_active_buffer, separate prefetch requests and response cycle arrays for both DRAMS are created.

They are used to fill addresses and cycles in the respective trace matrices



```python
class read_buffer:
    def prefetch_active_buffer(self, start_cycle):
        response_cycles_arr2 = self.backing_buffer2.service_reads(incoming_cycles_arr=cycles_arr2,
                                                                   incoming_requests_arr_np=prefetch_requests2)
        # 4. Update the variables
        self.last_prefect_cycle = int(response_cycles_arr[-1][0])

        self.last_prefect_cycle1 = int(response_cycles_arr1[-1][0])
        self.trace_matrix1 = np.concatenate((response_cycles_arr1, prefetch_requests1), axis=1)

        self.last_prefect_cycle2 = int(response_cycles_arr2[-1][0])
        self.trace_matrix2 = np.concatenate((response_cycles_arr2, prefetch_requests2), axis=1)

        # Update the trace matrix
        self.trace_matrix = np.concatenate((response_cycles_arr, prefetch_requests), axis=1)
        self.trace_valid = True

        # Set active buffer contents
        active_buf_start_line_id = 0
        active_buf_end_line_id = self.num_active_buf_lines
        self.active_buffer_set_limits = [active_buf_start_line_id, active_buf_end_line_id]

        prefetch_buf_start_line_id = active_buf_end_line_id
        prefetch_buf_end_line_id = prefetch_buf_start_line_id + self.num_prefetch_buf_lines
        self.prefetch_buffer_set_limits = [prefetch_buf_start_line_id, prefetch_buf_end_line_id]

        self.active_buf_full_flag = True

        # Set the line to be prefetched next
        # The module operator is to ensure that the indices wrap around
        if requested_data_size > self.active_buf_size:  # Some elements in the current idx is left out in this case
            self.next_line_prefetch_idx = num_lines % self.fetch_matrix.shape[0]
        else:
            self.next_line_prefetch_idx = (num_lines + 1) % self.fetch_matrix.shape[0]
```

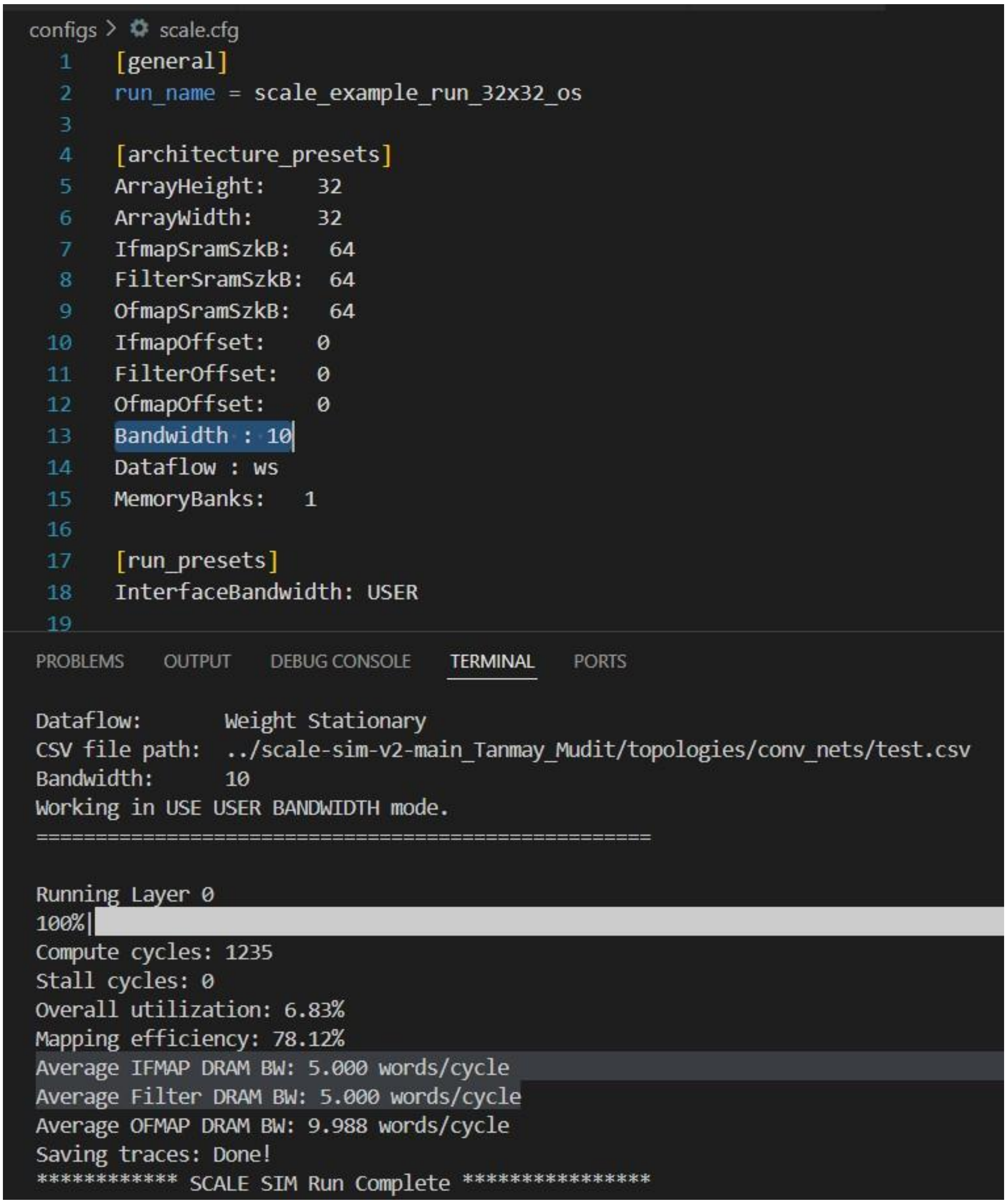
- Below are given functions created in read_buffer.py to read both DRAM's trace files, These functions are called from double_buffered_scratchpad_mem.py:



```python
class read_buffer:
    def get_external_access_start_stop_cycles(self):

        return start_cycle, end_cycle


    def print_trace_1(self, filename):
        if not self.trace_valid:
            print('No trace has been generated yet')
            return


        np.savetxt(filename, self.trace_matrix1, fmt='%s', delimiter=",")

    def print_trace_2(self, filename):
        if not self.trace_valid:
            print('No trace has been generated yet')
            return

        np.savetxt(filename, self.trace_matrix2, fmt='%s', delimiter=",")
```

So, if I've set bandwidth=10 (in config file), then 5 addresses are read from 1st DRAM and other 5 addresses are read from 2nd DRAM.

This helps decrease bandwidth requirements per DRAM ({Bandwidth/n} for n DRAMs).

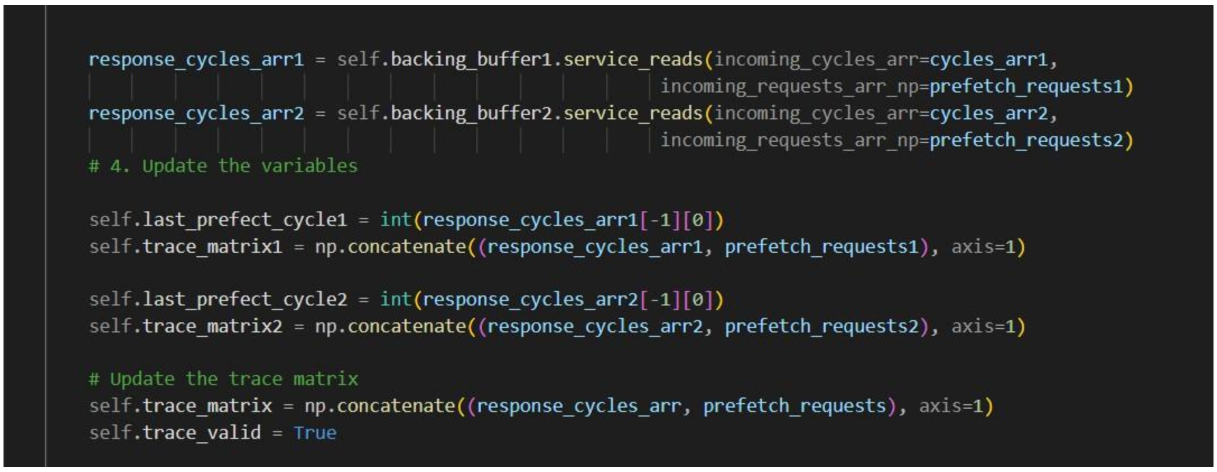


configs > scale.cfg

```
[general]
run_name = scale_example_run_32x32_os

[architecture_presets]
ArrayHeight:    32
ArrayWidth:     32
IfmapSramSzkB:   64
FilterSramSzkB:  64
OfmapSramSzkB:   64
IfmapOffset:    0
FilterOffset:   0
OfmapOffset:    0
Bandwidth : 10
Dataflow : ws
MemoryBanks:   1

[run_presets]
InterfaceBandwidth: USER

```

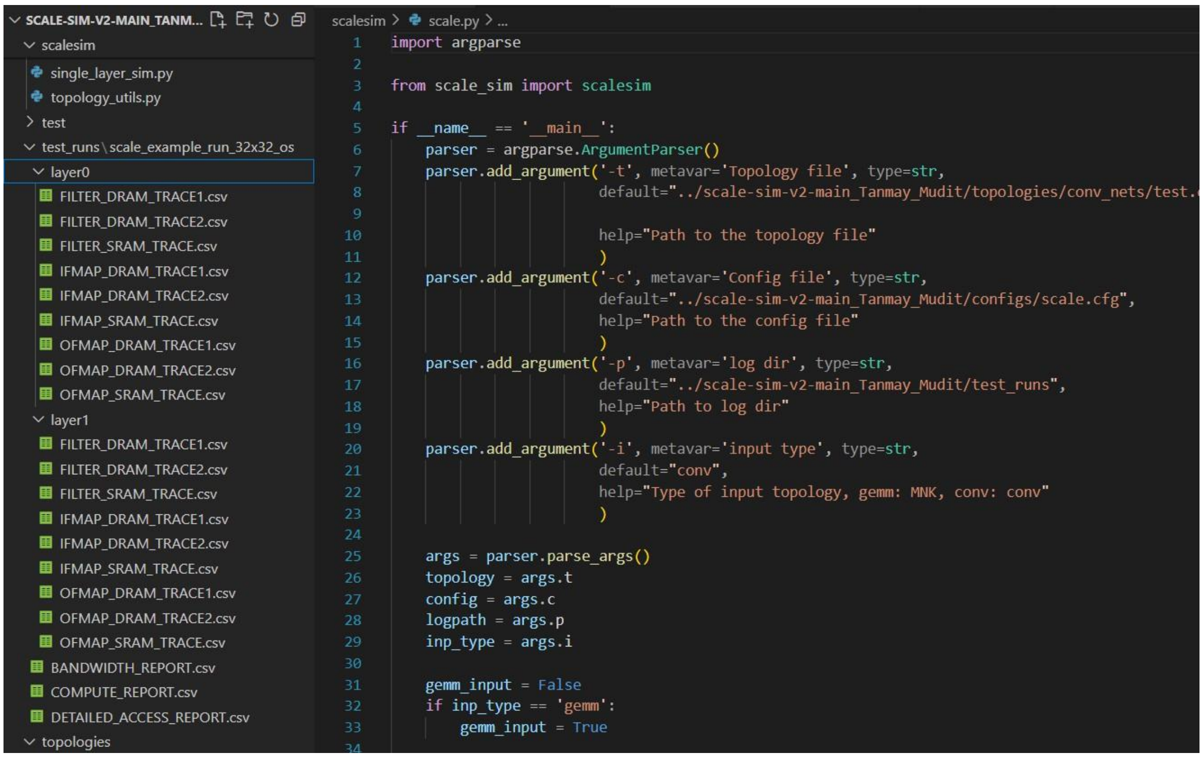
```
Dataflow:       Weight Stationary
CSV file path:  ../scale-sim-v2-main_Tanmay_Mudit/topologies/conv_nets/test.csv
Bandwidth:      10
Working in USE USER BANDWIDTH mode.
====================================================

Running Layer 0
100%|
Compute cycles: 1235
Stall cycles: 0
Overall utilization: 6.83%
Mapping efficiency: 78.12%
Average IFMAP DRAM BW: 5.000 words/cycle
Average Filter DRAM BW: 5.000 words/cycle
Average OFMAP DRAM BW: 9.988 words/cycle
Saving traces: Done!
************ SCALE SIM Run Complete ****************
```

As both the DRAMs are fetching data parallelly, the number of accesses is halved.



```
response_cycles_arr1 = self.backing_buffer1.service_reads(incoming_cycles_arr=cycles_arr1,
                                                           incoming_requests_arr_np=prefetch_requests1)
response_cycles_arr2 = self.backing_buffer2.service_reads(incoming_cycles_arr=cycles_arr2,
                                                           incoming_requests_arr_np=prefetch_requests2)
# 4. Update the variables

self.last_prefect_cycle1 = int(response_cycles_arr1[-1][0])
self.trace_matrix1 = np.concatenate((response_cycles_arr1, prefetch_requests1), axis=1)

self.last_prefect_cycle2 = int(response_cycles_arr2[-1][0])
self.trace_matrix2 = np.concatenate((response_cycles_arr2, prefetch_requests2), axis=1)

# Update the trace matrix
self.trace_matrix = np.concatenate((response_cycles_arr, prefetch_requests), axis=1)
self.trace_valid = True
```

- To make them parallel, I've created 2 read ports connected by 2 backing buffers. Now, both of the DRAMs are accessed independently in the same cycle.



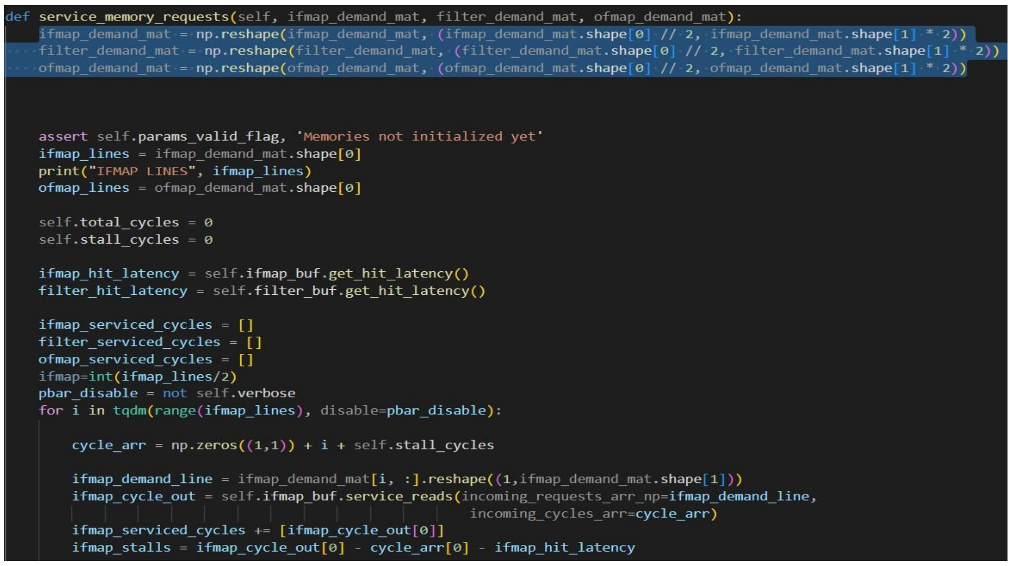
```
SCALE-SIM-V2-MAIN_TANM...
  scalesim
    single_layer_sim.py
    topology_utils.py
  test
  test_runs\scale_example_run_32x32_os
    layer0
      FILTER_DRAM_TRACE1.csv
      FILTER_DRAM_TRACE2.csv
      FILTER_SRAM_TRACE.csv
      IFMAP_DRAM_TRACE1.csv
      IFMAP_DRAM_TRACE2.csv
      IFMAP_SRAM_TRACE.csv
      OFMAP_DRAM_TRACE1.csv
      OFMAP_DRAM_TRACE2.csv
      OFMAP_SRAM_TRACE.csv
    layer1
      FILTER_DRAM_TRACE1.csv
      FILTER_DRAM_TRACE2.csv
      FILTER_SRAM_TRACE.csv
      IFMAP_DRAM_TRACE1.csv
      IFMAP_DRAM_TRACE2.csv
      IFMAP_SRAM_TRACE.csv
      OFMAP_DRAM_TRACE1.csv
      OFMAP_DRAM_TRACE2.csv
      OFMAP_SRAM_TRACE.csv
    BANDWIDTH_REPORT.csv
    COMPUTE_REPORT.csv
    DETAILED_ACCESS_REPORT.csv
  topologies
```

scalesim > scale.py > ...

```
import argparse

from scale_sim import scalesim

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-t', metavar='Topology file', type=str,
                        default="../scale-sim-v2-main_Tanmay_Mudit/topologies/conv_nets/test.

                        help="Path to the topology file"
                        )
    parser.add_argument('-c', metavar='Config file', type=str,
                        default="../scale-sim-v2-main_Tanmay_Mudit/configs/scale.cfg",
                        help="Path to the config file"
                        )
    parser.add_argument('-p', metavar='log dir', type=str,
                        default="../scale-sim-v2-main_Tanmay_Mudit/test_runs",
                        help="Path to log dir"
                        )
    parser.add_argument('-i', metavar='input type', type=str,
                        default="conv",
                        help="Type of input topology, gemm: MNK, conv: conv"
                        )

    args = parser.parse_args()
    topology = args.t
    config = args.c
    logpath = args.p
    inp_type = args.i

    gemm_input = False
    if inp_type == 'gemm':
        gemm_input = True
```

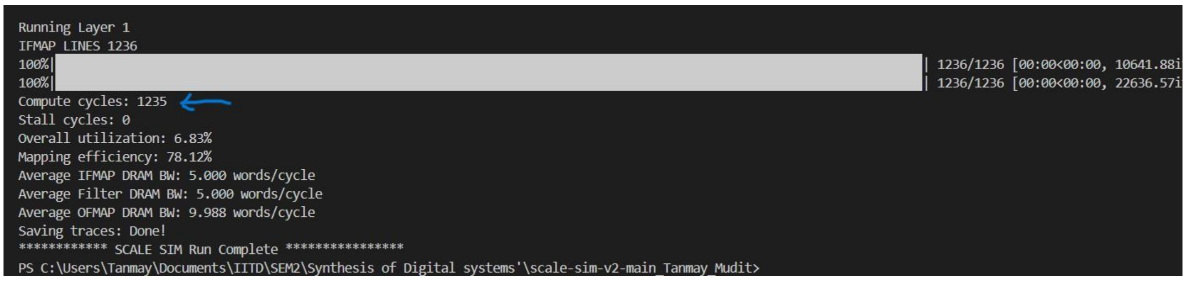
- I've made changes in multiple files to print both the trace matrices of IFMAP, FILTER and OFMAP for every layer.

- In double_buffered_mem_scratchpad.py, we can reshape the matrices to double the number of columns and reduce the number of rows to half before calling service_read function.



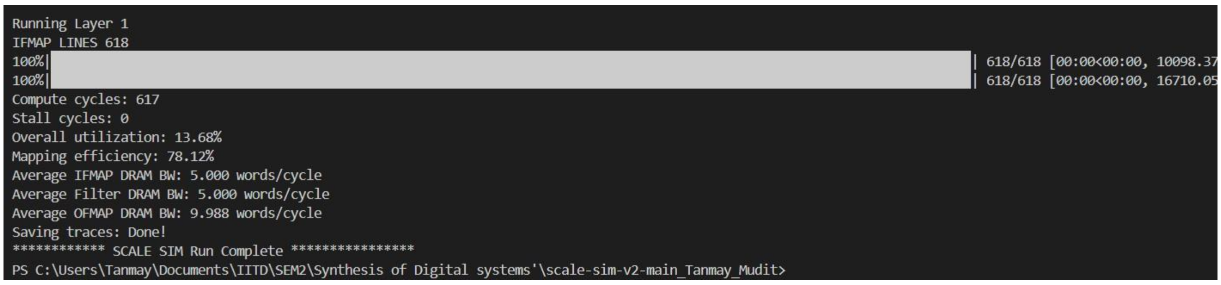
```python
def service_memory_requests(self, ifmap_demand_mat, filter_demand_mat, ofmap_demand_mat):
    ifmap_demand_mat = np.reshape(ifmap_demand_mat, (ifmap_demand_mat.shape[0] // 2, ifmap_demand_mat.shape[1] * 2))
    filter_demand_mat = np.reshape(filter_demand_mat, (filter_demand_mat.shape[0] // 2, filter_demand_mat.shape[1] * 2))
    ofmap_demand_mat = np.reshape(ofmap_demand_mat, (ofmap_demand_mat.shape[0] // 2, ofmap_demand_mat.shape[1] * 2))


    assert self.params_valid_flag, 'Memories not initialized yet'
    ifmap_lines = ifmap_demand_mat.shape[0]
    print("IFMAP LINES", ifmap_lines)
    ofmap_lines = ofmap_demand_mat.shape[0]

    self.total_cycles = 0
    self.stall_cycles = 0

    ifmap_hit_latency = self.ifmap_buf.get_hit_latency()
    filter_hit_latency = self.filter_buf.get_hit_latency()

    ifmap_serviced_cycles = []
    filter_serviced_cycles = []
    ofmap_serviced_cycles = []
    ifmap=int(ifmap_lines/2)
    pbar_disable = not self.verbose
    for i in tqdm(range(ifmap_lines), disable=pbar_disable):

        cycle_arr = np.zeros((1,1)) + i + self.stall_cycles

        ifmap_demand_line = ifmap_demand_mat[i, :].reshape((1,ifmap_demand_mat.shape[1]))
        ifmap_cycle_out = self.ifmap_buf.service_reads(incoming_requests_arr_np=ifmap_demand_line,
                                                        incoming_cycles_arr=cycle_arr)
        ifmap_serviced_cycles += [ifmap_cycle_out[0]]
        ifmap_stalls = ifmap_cycle_out[0] - cycle_arr[0] - ifmap_hit_latency
```

This will reduce the compute cycles to half of the original as more data is sent in a single cycle.



```
Running Layer 1
IFMAP LINES 1236
100%|                                                                                | 1236/1236 [00:00<00:00, 10641.88i
100%|                                                                                | 1236/1236 [00:00<00:00, 22636.57i
Compute cycles: 1235 ⟵
Stall cycles: 0
Overall utilization: 6.83%
Mapping efficiency: 78.12%
Average IFMAP DRAM BW: 5.000 words/cycle
Average Filter DRAM BW: 5.000 words/cycle
Average OFMAP DRAM BW: 9.988 words/cycle
Saving traces: Done!
************ SCALE SIM Run Complete ****************
PS C:\Users\Tanmay\Documents\IITD\SEM2\Synthesis of Digital systems'\scale-sim-v2-main_Tanmay_Mudit>
```

Figure 1 Compute cycles before making the changes

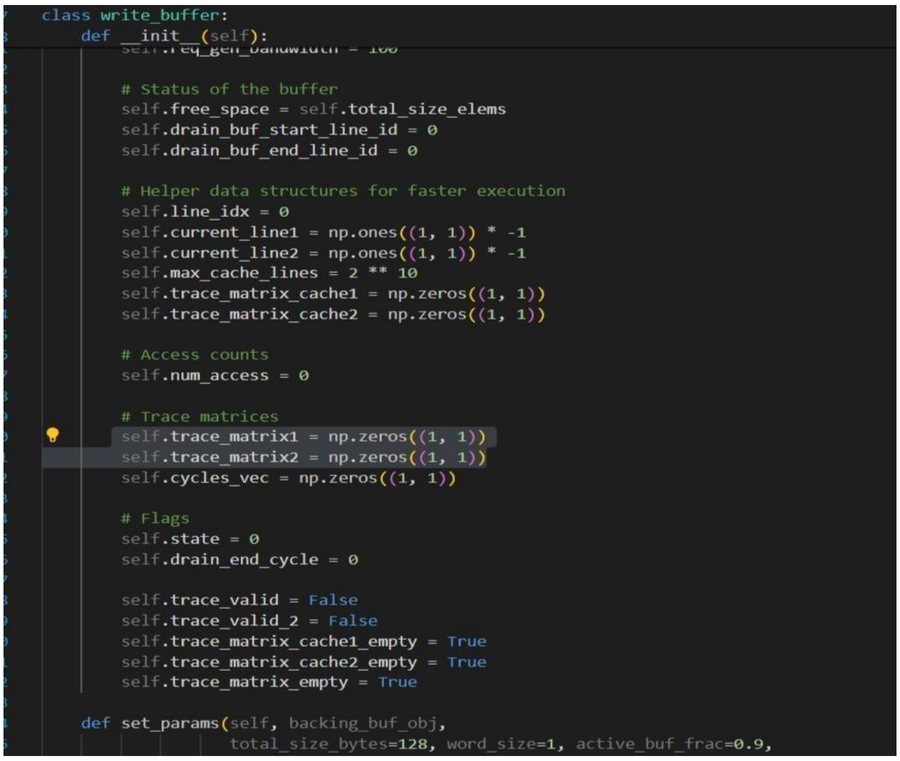


```
Running Layer 1
IFMAP LINES 618
100%|                                                                                | 618/618 [00:00<00:00, 10098.37
100%|                                                                                | 618/618 [00:00<00:00, 16710.05
Compute cycles: 617
Stall cycles: 0
Overall utilization: 13.68%
Mapping efficiency: 78.12%
Average IFMAP DRAM BW: 5.000 words/cycle
Average Filter DRAM BW: 5.000 words/cycle
Average OFMAP DRAM BW: 9.988 words/cycle
Saving traces: Done!
************ SCALE SIM Run Complete ****************
PS C:\Users\Tanmay\Documents\IITD\SEM2\Synthesis of Digital systems'\scale-sim-v2-main_Tanmay_Mudit>
```

Figure 2 Compute cycles after making the changes

Write Buffer:

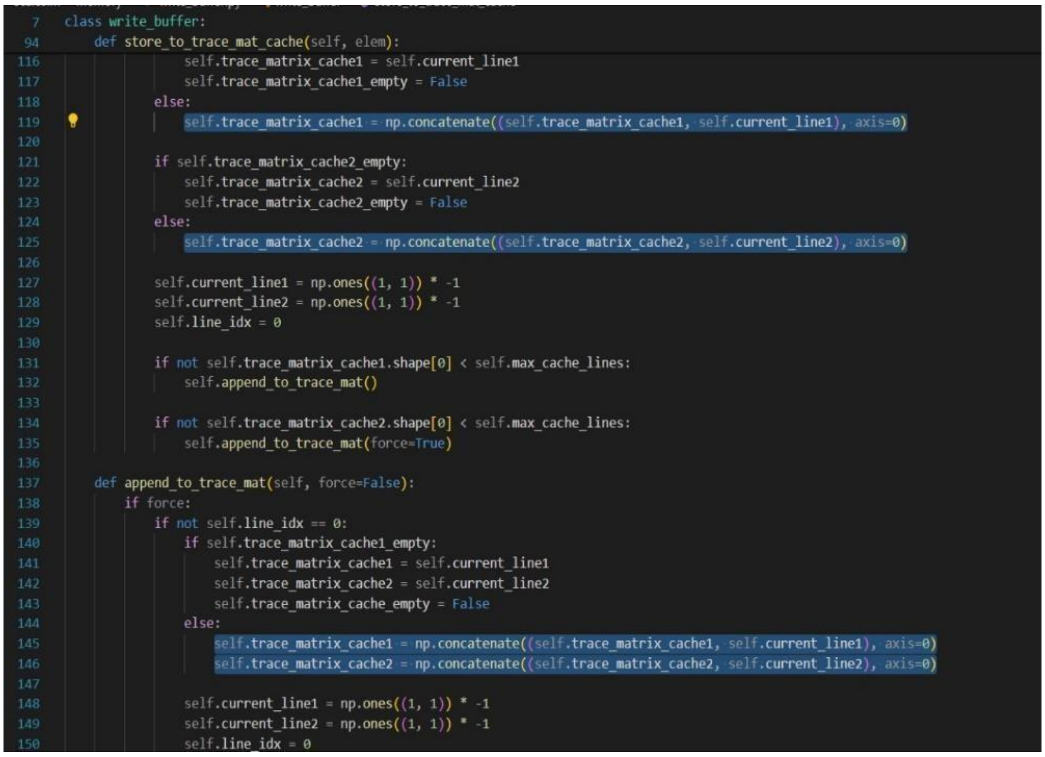
- Creating 2 tracematrices forthewritebuffer:



```python
class write_buffer:
    def __init__(self):
        self.req_gen_bandwidth = 100

        # Status of the buffer
        self.free_space = self.total_size_elems
        self.drain_buf_start_line_id = 0
        self.drain_buf_end_line_id = 0

        # Helper data structures for faster execution
        self.line_idx = 0
        self.current_line1 = np.ones((1, 1)) * -1
        self.current_line2 = np.ones((1, 1)) * -1
        self.max_cache_lines = 2 ** 10
        self.trace_matrix_cache1 = np.zeros((1, 1))
        self.trace_matrix_cache2 = np.zeros((1, 1))

        # Access counts
        self.num_access = 0

        # Trace matrices
        self.trace_matrix1 = np.zeros((1, 1))
        self.trace_matrix2 = np.zeros((1, 1))
        self.cycles_vec = np.zeros((1, 1))

        # Flags
        self.state = 0
        self.drain_end_cycle = 0

        self.trace_valid = False
        self.trace_valid_2 = False
        self.trace_matrix_cache1_empty = True
        self.trace_matrix_cache2_empty = True
        self.trace_matrix_empty = True

    def set_params(self, backing_buf_obj,
                   total_size_bytes=128, word_size=1, active_buf_frac=0.9,
```

Figure 3. Creating 2 trace matrices for write buffer

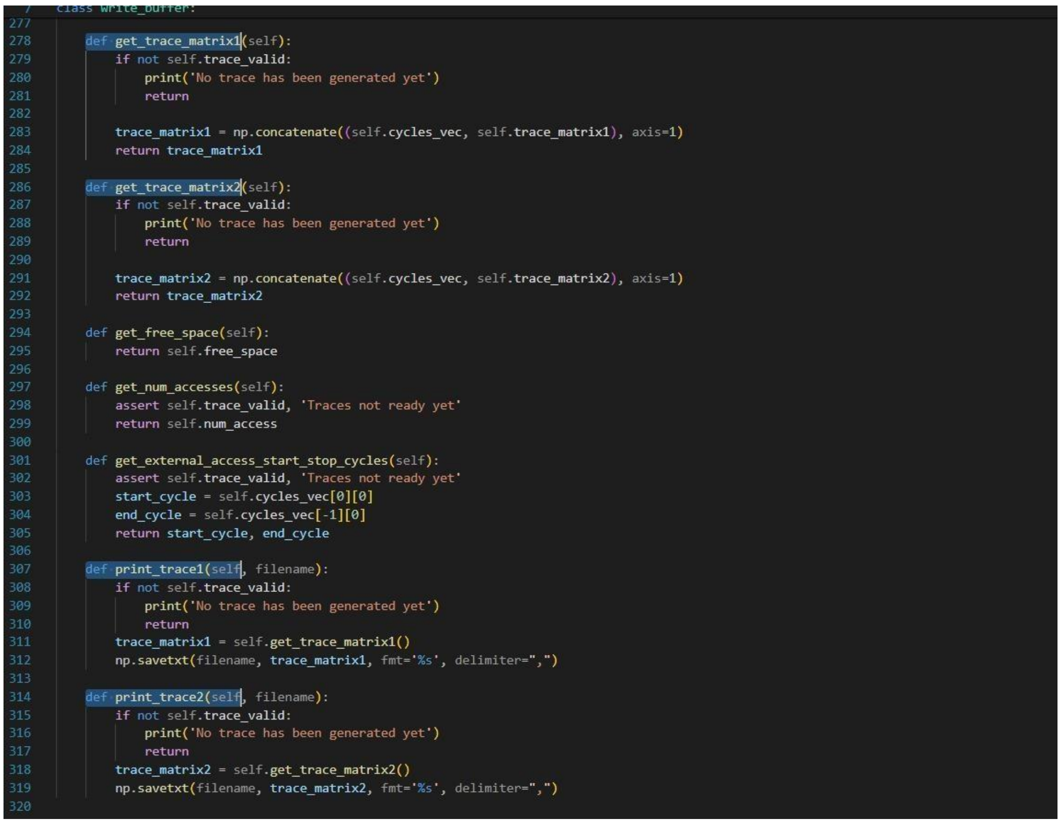
- Both trace matrices are filled in the functions store_to_trace_mat_cache and append_to_trace_mat



```python
class write_buffer:
    def store_to_trace_mat_cache(self, elem):
                self.trace_matrix_cache1 = self.current_line1
                self.trace_matrix_cache1_empty = False
            else:
                self.trace_matrix_cache1 = np.concatenate((self.trace_matrix_cache1, self.current_line1), axis=0)

            if self.trace_matrix_cache2_empty:
                self.trace_matrix_cache2 = self.current_line2
                self.trace_matrix_cache2_empty = False
            else:
                self.trace_matrix_cache2 = np.concatenate((self.trace_matrix_cache2, self.current_line2), axis=0)

            self.current_line1 = np.ones((1, 1)) * -1
            self.current_line2 = np.ones((1, 1)) * -1
            self.line_idx = 0

            if not self.trace_matrix_cache1.shape[0] < self.max_cache_lines:
                self.append_to_trace_mat()

            if not self.trace_matrix_cache2.shape[0] < self.max_cache_lines:
                self.append_to_trace_mat(force=True)

    def append_to_trace_mat(self, force=False):
        if force:
            if not self.line_idx == 0:
                if self.trace_matrix_cache1_empty:
                    self.trace_matrix_cache1 = self.current_line1
                    self.trace_matrix_cache2 = self.current_line2
                    self.trace_matrix_cache_empty = False
                else:
                    self.trace_matrix_cache1 = np.concatenate((self.trace_matrix_cache1, self.current_line1), axis=0)
                    self.trace_matrix_cache2 = np.concatenate((self.trace_matrix_cache2, self.current_line2), axis=0)

                self.current_line1 = np.ones((1, 1)) * -1
                self.current_line2 = np.ones((1, 1)) * -1
                self.line_idx = 0
```

Figure 4. Both trace matrices are filled in the functions store_to_trace_mat_cache and append_to_trace_mat

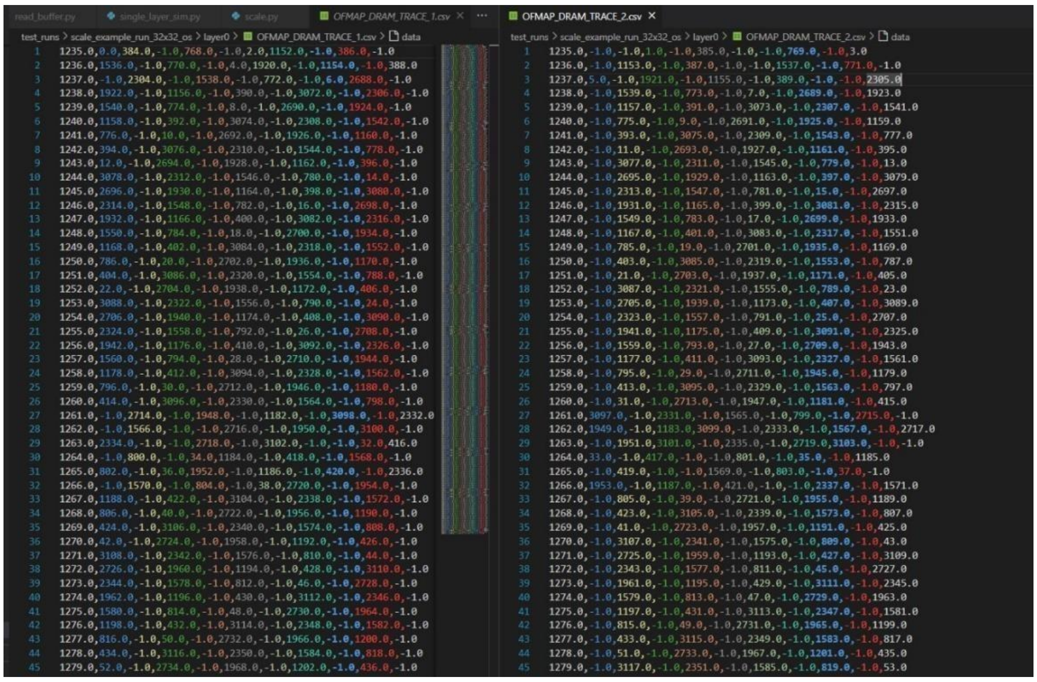
- These trace matrices are filled and printed using these functions:



```python
class write_buffer:

    def get_trace_matrix1(self):
        if not self.trace_valid:
            print('No trace has been generated yet')
            return

        trace_matrix1 = np.concatenate((self.cycles_vec, self.trace_matrix1), axis=1)
        return trace_matrix1

    def get_trace_matrix2(self):
        if not self.trace_valid:
            print('No trace has been generated yet')
            return

        trace_matrix2 = np.concatenate((self.cycles_vec, self.trace_matrix2), axis=1)
        return trace_matrix2

    def get_free_space(self):
        return self.free_space

    def get_num_accesses(self):
        assert self.trace_valid, 'Traces not ready yet'
        return self.num_access

    def get_external_access_start_stop_cycles(self):
        assert self.trace_valid, 'Traces not ready yet'
        start_cycle = self.cycles_vec[0][0]
        end_cycle = self.cycles_vec[-1][0]
        return start_cycle, end_cycle

    def print_trace1(self, filename):
        if not self.trace_valid:
            print('No trace has been generated yet')
            return
        trace_matrix1 = self.get_trace_matrix1()
        np.savetxt(filename, trace_matrix1, fmt='%s', delimiter=",")

    def print_trace2(self, filename):
        if not self.trace_valid:
            print('No trace has been generated yet')
            return
        trace_matrix2 = self.get_trace_matrix2()
        np.savetxt(filename, trace_matrix2, fmt='%s', delimiter=",")
```

Figure 5. The trace matrices are filled and printed using these functions

These trace matrices are generated in write_buffer.py



read_buffer.py | single_layer_sim.py | scale.py | OFMAP_DRAM_TRACE_1.csv

test_runs > scale_example_run_32x32_os > layer0 > OFMAP_DRAM_TRACE_1.csv > data

```
1235.0,0.0,384.0,-1.0,768.0,-1.0,2.0,1152.0,-1.0,386.0,-1.0
1236.0,1536.0,-1.0,770.0,-1.0,4.0,1920.0,-1.0,1154.0,-1.0,388.0
1237.0,-1.0,2304.0,-1.0,1538.0,-1.0,772.0,-1.0,6.0,2688.0,-1.0
1238.0,1922.0,-1.0,1156.0,-1.0,390.0,-1.0,3072.0,-1.0,2306.0,-1.0
1239.0,1540.0,-1.0,774.0,-1.0,8.0,-1.0,2690.0,-1.0,1924.0,-1.0
1240.0,1158.0,-1.0,392.0,-1.0,3074.0,-1.0,2308.0,-1.0,1542.0,-1.0
1241.0,776.0,-1.0,10.0,-1.0,2692.0,-1.0,1926.0,-1.0,1160.0,-1.0
1242.0,394.0,-1.0,3076.0,-1.0,2310.0,-1.0,1544.0,-1.0,778.0,-1.0
1243.0,12.0,-1.0,2694.0,-1.0,1928.0,-1.0,1162.0,-1.0,396.0,-1.0
1244.0,3078.0,-1.0,2312.0,-1.0,1546.0,-1.0,780.0,-1.0,14.0,-1.0
1245.0,2696.0,-1.0,1930.0,-1.0,1164.0,-1.0,398.0,-1.0,3080.0,-1.0
1246.0,2314.0,-1.0,1548.0,-1.0,782.0,-1.0,16.0,-1.0,2698.0,-1.0
1247.0,1932.0,-1.0,1166.0,-1.0,400.0,-1.0,3082.0,-1.0,2316.0,-1.0
1248.0,1550.0,-1.0,784.0,-1.0,18.0,-1.0,2700.0,-1.0,1934.0,-1.0
1249.0,1168.0,-1.0,402.0,-1.0,3084.0,-1.0,2318.0,-1.0,1552.0,-1.0
1250.0,786.0,-1.0,20.0,-1.0,2702.0,-1.0,1936.0,-1.0,1170.0,-1.0
1251.0,404.0,-1.0,3086.0,-1.0,2320.0,-1.0,1554.0,-1.0,788.0,-1.0
1252.0,22.0,-1.0,2704.0,-1.0,1938.0,-1.0,1172.0,-1.0,406.0,-1.0
1253.0,3088.0,-1.0,2322.0,-1.0,1556.0,-1.0,790.0,-1.0,24.0,-1.0
1254.0,2706.0,-1.0,1940.0,-1.0,1174.0,-1.0,408.0,-1.0,3090.0,-1.0
1255.0,2324.0,-1.0,1558.0,-1.0,792.0,-1.0,26.0,-1.0,2708.0,-1.0
1256.0,1942.0,-1.0,1176.0,-1.0,410.0,-1.0,3092.0,-1.0,2326.0,-1.0
1257.0,1560.0,-1.0,794.0,-1.0,28.0,-1.0,2710.0,-1.0,1944.0,-1.0
1258.0,1178.0,-1.0,412.0,-1.0,3094.0,-1.0,2328.0,-1.0,1562.0,-1.0
1259.0,796.0,-1.0,30.0,-1.0,2712.0,-1.0,1946.0,-1.0,1180.0,-1.0
1260.0,414.0,-1.0,3096.0,-1.0,2330.0,-1.0,1564.0,-1.0,798.0,-1.0
1261.0,-1.0,2714.0,-1.0,1948.0,-1.0,1182.0,-1.0,3098.0,-1.0,2332.0
1262.0,-1.0,1566.0,-1.0,-1.0,2716.0,-1.0,1950.0,-1.0,3100.0,-1.0
1263.0,2334.0,-1.0,-1.0,2718.0,-1.0,3102.0,-1.0,-1.0,32.0,416.0
1264.0,-1.0,800.0,-1.0,34.0,1184.0,-1.0,418.0,-1.0,1568.0,-1.0
1265.0,802.0,-1.0,36.0,1952.0,-1.0,1186.0,-1.0,420.0,-1.0,2336.0
1266.0,-1.0,1570.0,-1.0,804.0,-1.0,38.0,2720.0,-1.0,1954.0,-1.0
1267.0,1188.0,-1.0,422.0,-1.0,3104.0,-1.0,2338.0,-1.0,1572.0,-1.0
1268.0,806.0,-1.0,40.0,-1.0,2722.0,-1.0,1956.0,-1.0,1190.0,-1.0
1269.0,424.0,-1.0,3106.0,-1.0,2340.0,-1.0,1574.0,-1.0,808.0,-1.0
1270.0,42.0,-1.0,2724.0,-1.0,1958.0,-1.0,1192.0,-1.0,426.0,-1.0
1271.0,3108.0,-1.0,2342.0,-1.0,1576.0,-1.0,810.0,-1.0,44.0,-1.0
1272.0,2726.0,-1.0,1960.0,-1.0,1194.0,-1.0,428.0,-1.0,3110.0,-1.0
1273.0,2344.0,-1.0,1578.0,-1.0,812.0,-1.0,46.0,-1.0,2728.0,-1.0
1274.0,1962.0,-1.0,1196.0,-1.0,430.0,-1.0,3112.0,-1.0,2346.0,-1.0
1275.0,1580.0,-1.0,814.0,-1.0,48.0,-1.0,2730.0,-1.0,1964.0,-1.0
1276.0,1198.0,-1.0,432.0,-1.0,3114.0,-1.0,2348.0,-1.0,1582.0,-1.0
1277.0,816.0,-1.0,50.0,-1.0,2732.0,-1.0,1966.0,-1.0,1200.0,-1.0
1278.0,434.0,-1.0,3116.0,-1.0,2350.0,-1.0,1584.0,-1.0,818.0,-1.0
1279.0,52.0,-1.0,2734.0,-1.0,1968.0,-1.0,1202.0,-1.0,436.0,-1.0
```

OFMAP_DRAM_TRACE_2.csv

test_runs > scale_example_run_32x32_os > layer0 > OFMAP_DRAM_TRACE_2.csv > data

```
1235.0,-1.0,-1.0,1.0,-1.0,385.0,-1.0,-1.0,769.0,-1.0,3.0
1236.0,-1.0,1153.0,-1.0,387.0,-1.0,-1.0,1537.0,-1.0,771.0,-1.0
1237.0,5.0,-1.0,1921.0,-1.0,1155.0,-1.0,389.0,-1.0,-1.0,2305.0
1238.0,-1.0,1539.0,-1.0,773.0,-1.0,7.0,-1.0,2689.0,-1.0,1923.0
1239.0,-1.0,1157.0,-1.0,391.0,-1.0,3073.0,-1.0,2307.0,-1.0,1541.0
1240.0,-1.0,775.0,-1.0,9.0,-1.0,2691.0,-1.0,1925.0,-1.0,1159.0
1241.0,-1.0,393.0,-1.0,3075.0,-1.0,2309.0,-1.0,1543.0,-1.0,777.0
1242.0,-1.0,11.0,-1.0,2693.0,-1.0,1927.0,-1.0,1161.0,-1.0,395.0
1243.0,-1.0,3077.0,-1.0,2311.0,-1.0,1545.0,-1.0,779.0,-1.0,13.0
1244.0,-1.0,2695.0,-1.0,1929.0,-1.0,1163.0,-1.0,397.0,-1.0,3079.0
1245.0,-1.0,2313.0,-1.0,1547.0,-1.0,781.0,-1.0,15.0,-1.0,2697.0
1246.0,-1.0,1931.0,-1.0,1165.0,-1.0,399.0,-1.0,3081.0,-1.0,2315.0
1247.0,-1.0,1549.0,-1.0,783.0,-1.0,17.0,-1.0,2699.0,-1.0,1933.0
1248.0,-1.0,1167.0,-1.0,401.0,-1.0,3083.0,-1.0,2317.0,-1.0,1551.0
1249.0,-1.0,785.0,-1.0,19.0,-1.0,2701.0,-1.0,1935.0,-1.0,1169.0
1250.0,-1.0,403.0,-1.0,3085.0,-1.0,2319.0,-1.0,1553.0,-1.0,787.0
1251.0,-1.0,21.0,-1.0,2703.0,-1.0,1937.0,-1.0,1171.0,-1.0,405.0
1252.0,-1.0,3087.0,-1.0,2321.0,-1.0,1555.0,-1.0,789.0,-1.0,23.0
1253.0,-1.0,2705.0,-1.0,1939.0,-1.0,1173.0,-1.0,407.0,-1.0,3089.0
1254.0,-1.0,2323.0,-1.0,1557.0,-1.0,791.0,-1.0,25.0,-1.0,2707.0
1255.0,-1.0,1941.0,-1.0,1175.0,-1.0,409.0,-1.0,3091.0,-1.0,2325.0
1256.0,-1.0,1559.0,-1.0,793.0,-1.0,27.0,-1.0,2709.0,-1.0,1943.0
1257.0,-1.0,1177.0,-1.0,411.0,-1.0,3093.0,-1.0,2327.0,-1.0,1561.0
1258.0,-1.0,795.0,-1.0,29.0,-1.0,2711.0,-1.0,1945.0,-1.0,1179.0
1259.0,-1.0,413.0,-1.0,3095.0,-1.0,2329.0,-1.0,1563.0,-1.0,797.0
1260.0,-1.0,31.0,-1.0,2713.0,-1.0,1947.0,-1.0,1181.0,-1.0,415.0
1261.0,3097.0,-1.0,2331.0,-1.0,1565.0,-1.0,799.0,-1.0,2715.0,-1.0
1262.0,1949.0,-1.0,1183.0,3099.0,-1.0,2333.0,-1.0,1567.0,-1.0,2717.0
1263.0,-1.0,1951.0,3101.0,-1.0,2335.0,-1.0,2719.0,3103.0,-1.0,-1.0
1264.0,33.0,-1.0,417.0,-1.0,-1.0,801.0,-1.0,35.0,-1.0,1185.0
1265.0,-1.0,419.0,-1.0,-1.0,1569.0,-1.0,803.0,-1.0,37.0,-1.0
1266.0,1953.0,-1.0,1187.0,-1.0,421.0,-1.0,-1.0,2337.0,-1.0,1571.0
1267.0,-1.0,805.0,-1.0,39.0,-1.0,2721.0,-1.0,1955.0,-1.0,1189.0
1268.0,-1.0,423.0,-1.0,3105.0,-1.0,2339.0,-1.0,1573.0,-1.0,807.0
1269.0,-1.0,41.0,-1.0,2723.0,-1.0,1957.0,-1.0,1191.0,-1.0,425.0
1270.0,-1.0,3107.0,-1.0,2341.0,-1.0,1575.0,-1.0,809.0,-1.0,43.0
1271.0,-1.0,2725.0,-1.0,1959.0,-1.0,1193.0,-1.0,427.0,-1.0,3109.0
1272.0,-1.0,2343.0,-1.0,1577.0,-1.0,811.0,-1.0,45.0,-1.0,2727.0
1273.0,-1.0,1961.0,-1.0,1195.0,-1.0,429.0,-1.0,3111.0,-1.0,2345.0
1274.0,-1.0,1579.0,-1.0,813.0,-1.0,47.0,-1.0,2729.0,-1.0,1963.0
1275.0,-1.0,1197.0,-1.0,431.0,-1.0,3113.0,-1.0,2347.0,-1.0,1581.0
1276.0,-1.0,815.0,-1.0,49.0,-1.0,2731.0,-1.0,1965.0,-1.0,1199.0
1277.0,-1.0,433.0,-1.0,3115.0,-1.0,2349.0,-1.0,1583.0,-1.0,817.0
1278.0,-1.0,51.0,-1.0,2733.0,-1.0,1967.0,-1.0,1201.0,-1.0,435.0
1279.0,-1.0,3117.0,-1.0,2351.0,-1.0,1585.0,-1.0,819.0,-1.0,53.0
```

Figure c. OFMAP tracematrix

➔Please follow discussions regarding parameters (Avg. Bandwidth, utilization, etc.) in the Novel Implementation report

Results (for read_buffer):-

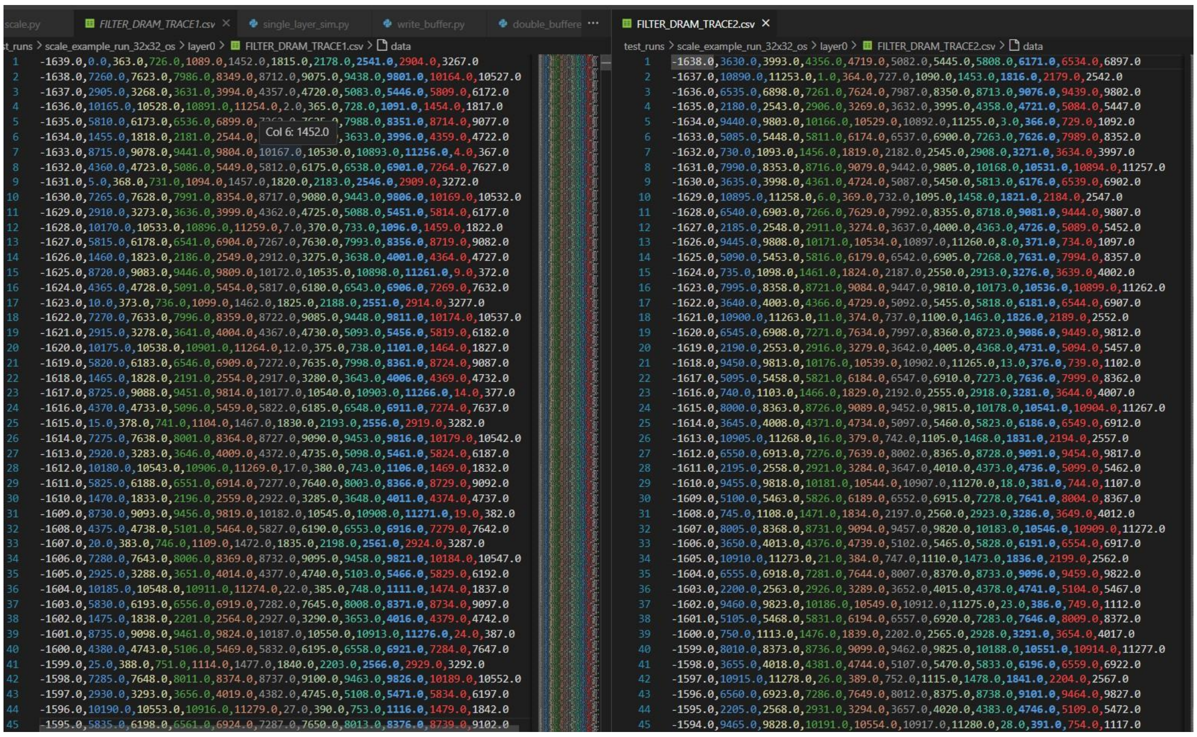
1. Total number of cycles remains the same.
2. The bandwidth per DRAM is halved.

Results (for write_buffer):-

1. Total number of cycles is halved.
2. The bandwidth per DRAM is the same.

Novel Mapping Implementation: -

- In this implementation, I’ve divided the data row-wise alternatively in 2 DRAMs.

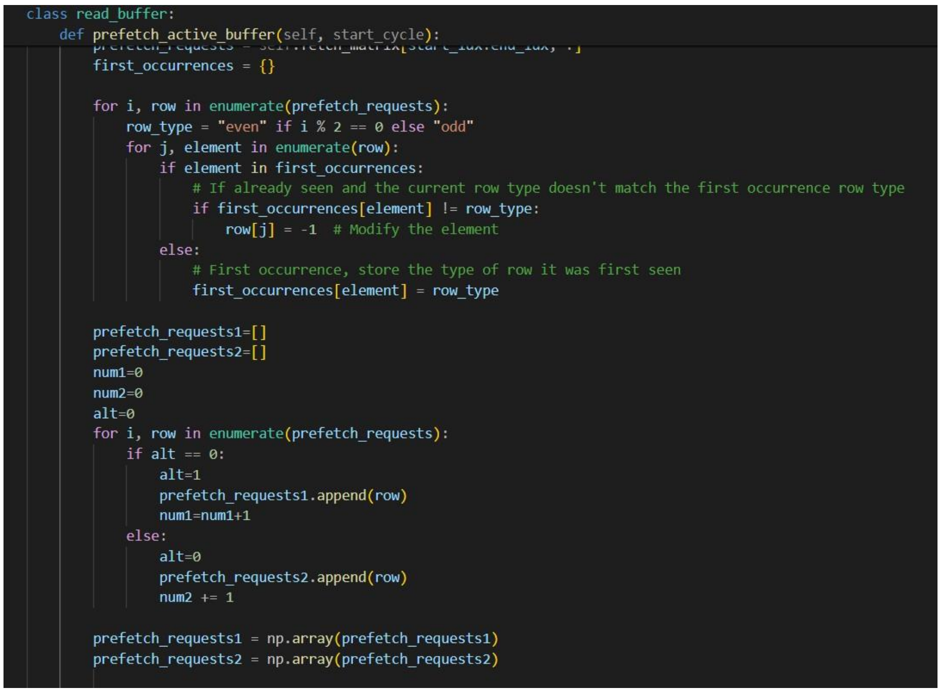


```
t_runs > scale_example_run_32x32_os > layer0 > FILTER_DRAM_TRACE1.csv > data
-1639.0,0.0,363.0,726.0,1089.0,1452.0,1815.0,2178.0,2541.0,2904.0,3267.0
-1638.0,7260.0,7623.0,7986.0,8349.0,8712.0,9075.0,9438.0,9801.0,10164.0,10527.0
-1637.0,2905.0,3268.0,3631.0,3994.0,4357.0,4720.0,5083.0,5446.0,5809.0,6172.0
-1636.0,10165.0,10528.0,10891.0,11254.0,2.0,365.0,728.0,1091.0,1454.0,1817.0
-1635.0,5810.0,6173.0,6536.0,6899.0,[illegible],7988.0,8351.0,8714.0,9077.0
-1634.0,1455.0,1818.0,2181.0,2544.0,[illegible],3633.0,3996.0,4359.0,4722.0
-1633.0,8715.0,9078.0,9441.0,9804.0,10167.0,10530.0,10893.0,11256.0,4.0,367.0
-1632.0,4360.0,4723.0,5086.0,5449.0,5812.0,6175.0,6538.0,6901.0,7264.0,7627.0
-1631.0,5.0,368.0,731.0,1094.0,1457.0,1820.0,2183.0,2546.0,2909.0,3272.0
-1630.0,7265.0,7628.0,7991.0,8354.0,8717.0,9080.0,9443.0,9806.0,10169.0,10532.0
-1629.0,2910.0,3273.0,3636.0,3999.0,4362.0,4725.0,5088.0,5451.0,5814.0,6177.0
-1628.0,10170.0,10533.0,10896.0,11259.0,7.0,370.0,733.0,1096.0,1459.0,1822.0
-1627.0,5815.0,6178.0,6541.0,6904.0,7267.0,7630.0,7993.0,8356.0,8719.0,9082.0
-1626.0,1460.0,1823.0,2186.0,2549.0,2912.0,3275.0,3638.0,4001.0,4364.0,4727.0
-1625.0,8720.0,9083.0,9446.0,9809.0,10172.0,10535.0,10898.0,11261.0,9.0,372.0
-1624.0,4365.0,4728.0,5091.0,5454.0,5817.0,6180.0,6543.0,6906.0,7269.0,7632.0
-1623.0,10.0,373.0,736.0,1099.0,1462.0,1825.0,2188.0,2551.0,2914.0,3277.0
-1622.0,7270.0,7633.0,7996.0,8359.0,8722.0,9085.0,9448.0,9811.0,10174.0,10537.0
-1621.0,2915.0,3278.0,3641.0,4004.0,4367.0,4730.0,5093.0,5456.0,5819.0,6182.0
-1620.0,10175.0,10538.0,10901.0,11264.0,12.0,375.0,738.0,1101.0,1464.0,1827.0
-1619.0,5820.0,6183.0,6546.0,6909.0,7272.0,7635.0,7998.0,8361.0,8724.0,9087.0
-1618.0,1465.0,1828.0,2191.0,2554.0,2917.0,3280.0,3643.0,4006.0,4369.0,4732.0
-1617.0,8725.0,9088.0,9451.0,9814.0,10177.0,10540.0,10903.0,11266.0,14.0,377.0
-1616.0,4370.0,4733.0,5096.0,5459.0,5822.0,6185.0,6548.0,6911.0,7274.0,7637.0
-1615.0,15.0,378.0,741.0,1104.0,1467.0,1830.0,2193.0,2556.0,2919.0,3282.0
-1614.0,7275.0,7638.0,8001.0,8364.0,8727.0,9090.0,9453.0,9816.0,10179.0,10542.0
-1613.0,2920.0,3283.0,3646.0,4009.0,4372.0,4735.0,5098.0,5461.0,5824.0,6187.0
-1612.0,10180.0,10543.0,10906.0,11269.0,17.0,380.0,743.0,1106.0,1469.0,1832.0
-1611.0,5825.0,6188.0,6551.0,6914.0,7277.0,7640.0,8003.0,8366.0,8729.0,9092.0
-1610.0,1470.0,1833.0,2196.0,2559.0,2922.0,3285.0,3648.0,4011.0,4374.0,4737.0
-1609.0,8730.0,9093.0,9456.0,9819.0,10182.0,10545.0,10908.0,11271.0,19.0,382.0
-1608.0,4375.0,4738.0,5101.0,5464.0,5827.0,6190.0,6553.0,6916.0,7279.0,7642.0
-1607.0,20.0,383.0,746.0,1109.0,1472.0,1835.0,2198.0,2561.0,2924.0,3287.0
-1606.0,7280.0,7643.0,8006.0,8369.0,8732.0,9095.0,9458.0,9821.0,10184.0,10547.0
-1605.0,2925.0,3288.0,3651.0,4014.0,4377.0,4740.0,5103.0,5466.0,5829.0,6192.0
-1604.0,10185.0,10548.0,10911.0,11274.0,22.0,385.0,748.0,1111.0,1474.0,1837.0
-1603.0,5830.0,6193.0,6556.0,6919.0,7282.0,7645.0,8008.0,8371.0,8734.0,9097.0
-1602.0,1475.0,1838.0,2201.0,2564.0,2927.0,3290.0,3653.0,4016.0,4379.0,4742.0
-1601.0,8735.0,9098.0,9461.0,9824.0,10187.0,10550.0,10913.0,11276.0,24.0,387.0
-1600.0,4380.0,4743.0,5106.0,5469.0,5832.0,6195.0,6558.0,6921.0,7284.0,7647.0
-1599.0,25.0,388.0,751.0,1114.0,1477.0,1840.0,2203.0,2566.0,2929.0,3292.0
-1598.0,7285.0,7648.0,8011.0,8374.0,8737.0,9100.0,9463.0,9826.0,10189.0,10552.0
-1597.0,2930.0,3293.0,3656.0,4019.0,4382.0,4745.0,5108.0,5471.0,5834.0,6197.0
-1596.0,10190.0,10553.0,10916.0,11279.0,27.0,390.0,753.0,1116.0,1479.0,1842.0
-1595.0,5835.0,6198.0,6561.0,6924.0,7287.0,7650.0,8013.0,8376.0,8739.0,9102.0
```

```
test_runs > scale_example_run_32x32_os > layer0 > FILTER_DRAM_TRACE2.csv > data
-1638.0,3630.0,3993.0,4356.0,4719.0,5082.0,5445.0,5808.0,6171.0,6534.0,6897.0
-1637.0,10890.0,11253.0,1.0,364.0,727.0,1090.0,1453.0,1816.0,2179.0,2542.0
-1636.0,6535.0,6898.0,7261.0,7624.0,7987.0,8350.0,8713.0,9076.0,9439.0,9802.0
-1635.0,2180.0,2543.0,2906.0,3269.0,3632.0,3995.0,4358.0,4721.0,5084.0,5447.0
-1634.0,9440.0,9803.0,10166.0,10529.0,10892.0,11255.0,3.0,366.0,729.0,1092.0
-1633.0,5085.0,5448.0,5811.0,6174.0,6537.0,6900.0,7263.0,7626.0,7989.0,8352.0
-1632.0,730.0,1093.0,1456.0,1819.0,2182.0,2545.0,2908.0,3271.0,3634.0,3997.0
-1631.0,7990.0,8353.0,8716.0,9079.0,9442.0,9805.0,10168.0,10531.0,10894.0,11257.0
-1630.0,3635.0,3998.0,4361.0,4724.0,5087.0,5450.0,5813.0,6176.0,6539.0,6902.0
-1629.0,10895.0,11258.0,6.0,369.0,732.0,1095.0,1458.0,1821.0,2184.0,2547.0
-1628.0,6540.0,6903.0,7266.0,7629.0,7992.0,8355.0,8718.0,9081.0,9444.0,9807.0
-1627.0,2185.0,2548.0,2911.0,3274.0,3637.0,4000.0,4363.0,4726.0,5089.0,5452.0
-1626.0,9445.0,9808.0,10171.0,10534.0,10897.0,11260.0,8.0,371.0,734.0,1097.0
-1625.0,5090.0,5453.0,5816.0,6179.0,6542.0,6905.0,7268.0,7631.0,7994.0,8357.0
-1624.0,735.0,1098.0,1461.0,1824.0,2187.0,2550.0,2913.0,3276.0,3639.0,4002.0
-1623.0,7995.0,8358.0,8721.0,9084.0,9447.0,9810.0,10173.0,10536.0,10899.0,11262.0
-1622.0,3640.0,4003.0,4366.0,4729.0,5092.0,5455.0,5818.0,6181.0,6544.0,6907.0
-1621.0,10900.0,11263.0,11.0,374.0,737.0,1100.0,1463.0,1826.0,2189.0,2552.0
-1620.0,6545.0,6908.0,7271.0,7634.0,7997.0,8360.0,8723.0,9086.0,9449.0,9812.0
-1619.0,2190.0,2553.0,2916.0,3279.0,3642.0,4005.0,4368.0,4731.0,5094.0,5457.0
-1618.0,9450.0,9813.0,10176.0,10539.0,10902.0,11265.0,13.0,376.0,739.0,1102.0
-1617.0,5095.0,5458.0,5821.0,6184.0,6547.0,6910.0,7273.0,7636.0,7999.0,8362.0
-1616.0,740.0,1103.0,1466.0,1829.0,2192.0,2555.0,2918.0,3281.0,3644.0,4007.0
-1615.0,8000.0,8363.0,8726.0,9089.0,9452.0,9815.0,10178.0,10541.0,10904.0,11267.0
-1614.0,3645.0,4008.0,4371.0,4734.0,5097.0,5460.0,5823.0,6186.0,6549.0,6912.0
-1613.0,10905.0,11268.0,16.0,379.0,742.0,1105.0,1468.0,1831.0,2194.0,2557.0
-1612.0,6550.0,6913.0,7276.0,7639.0,8002.0,8365.0,8728.0,9091.0,9454.0,9817.0
-1611.0,2195.0,2558.0,2921.0,3284.0,3647.0,4010.0,4373.0,4736.0,5099.0,5462.0
-1610.0,9455.0,9818.0,10181.0,10544.0,10907.0,11270.0,18.0,381.0,744.0,1107.0
-1609.0,5100.0,5463.0,5826.0,6189.0,6552.0,6915.0,7278.0,7641.0,8004.0,8367.0
-1608.0,745.0,1108.0,1471.0,1834.0,2197.0,2560.0,2923.0,3286.0,3649.0,4012.0
-1607.0,8005.0,8368.0,8731.0,9094.0,9457.0,9820.0,10183.0,10546.0,10909.0,11272.0
-1606.0,3650.0,4013.0,4376.0,4739.0,5102.0,5465.0,5828.0,6191.0,6554.0,6917.0
-1605.0,10910.0,11273.0,21.0,384.0,747.0,1110.0,1473.0,1836.0,2199.0,2562.0
-1604.0,6555.0,6918.0,7281.0,7644.0,8007.0,8370.0,8733.0,9096.0,9459.0,9822.0
-1603.0,2200.0,2563.0,2926.0,3289.0,3652.0,4015.0,4378.0,4741.0,5104.0,5467.0
-1602.0,9460.0,9823.0,10186.0,10549.0,10912.0,11275.0,23.0,386.0,749.0,1112.0
-1601.0,5105.0,5468.0,5831.0,6194.0,6557.0,6920.0,7283.0,7646.0,8009.0,8372.0
-1600.0,750.0,1113.0,1476.0,1839.0,2202.0,2565.0,2928.0,3291.0,3654.0,4017.0
-1599.0,8010.0,8373.0,8736.0,9099.0,9462.0,9825.0,10188.0,10551.0,10914.0,11277.0
-1598.0,3655.0,4018.0,4381.0,4744.0,5107.0,5470.0,5833.0,6196.0,6559.0,6922.0
-1597.0,10915.0,11278.0,26.0,389.0,752.0,1115.0,1478.0,1841.0,2204.0,2567.0
-1596.0,6560.0,6923.0,7286.0,7649.0,8012.0,8375.0,8738.0,9101.0,9464.0,9827.0
-1595.0,2205.0,2568.0,2931.0,3294.0,3657.0,4020.0,4383.0,4746.0,5109.0,5472.0
-1594.0,9465.0,9828.0,10191.0,10554.0,10917.0,11280.0,28.0,391.0,754.0,1117.0
```

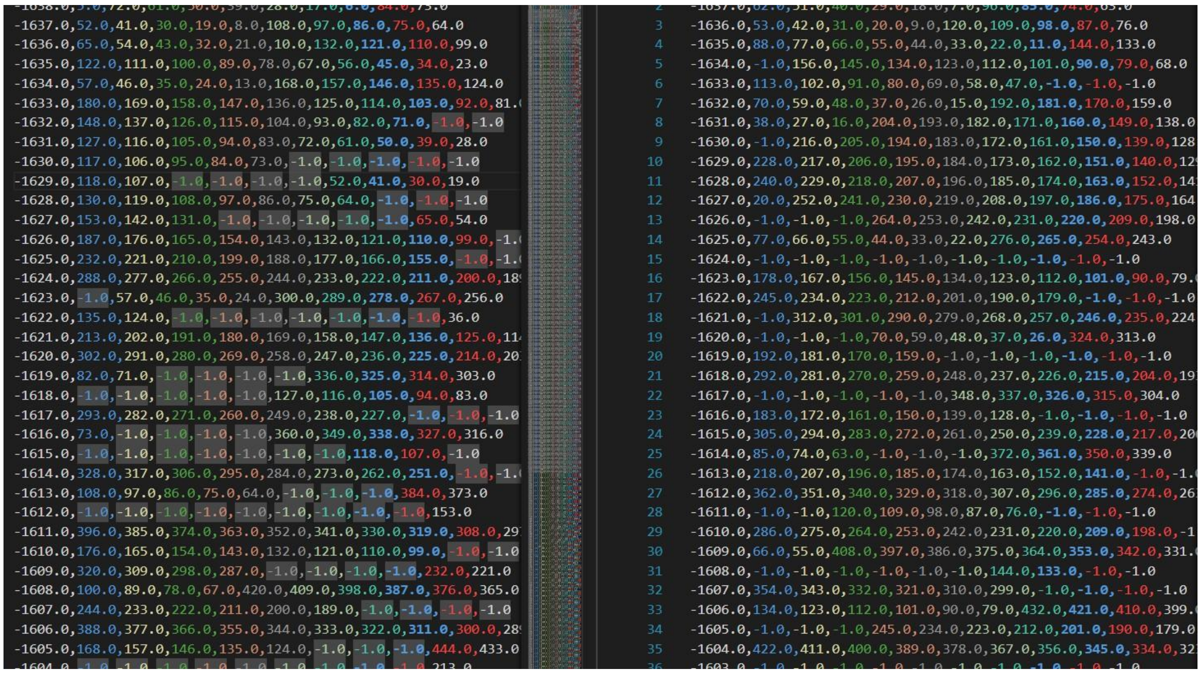
Figure 6 Trace Matrices

- During memory mapping, if a data block is previously fetched from DRAM i, and if it is now requested from DRAM j, then the trace matrix of DRAM j will show ‘-1’. And, data block is accessed from DRAM i.



```python
class read_buffer:
    def prefetch_active_buffer(self, start_cycle):
        prefetch_requests = self.fetch_matrix[start_idx:end_idx, :]
        first_occurrences = {}

        for i, row in enumerate(prefetch_requests):
            row_type = "even" if i % 2 == 0 else "odd"
            for j, element in enumerate(row):
                if element in first_occurrences:
                    # If already seen and the current row type doesn't match the first occurrence row type
                    if first_occurrences[element] != row_type:
                        row[j] = -1  # Modify the element
                else:
                    # First occurrence, store the type of row it was first seen
                    first_occurrences[element] = row_type

        prefetch_requests1=[]
        prefetch_requests2=[]
        num1=0
        num2=0
        alt=0
        for i, row in enumerate(prefetch_requests):
            if alt == 0:
                alt=1
                prefetch_requests1.append(row)
                num1=num1+1
            else:
                alt=0
                prefetch_requests2.append(row)
                num2 += 1

        prefetch_requests1 = np.array(prefetch_requests1)
        prefetch_requests2 = np.array(prefetch_requests2)
```

Figure 7. Memory mapping and division of rows



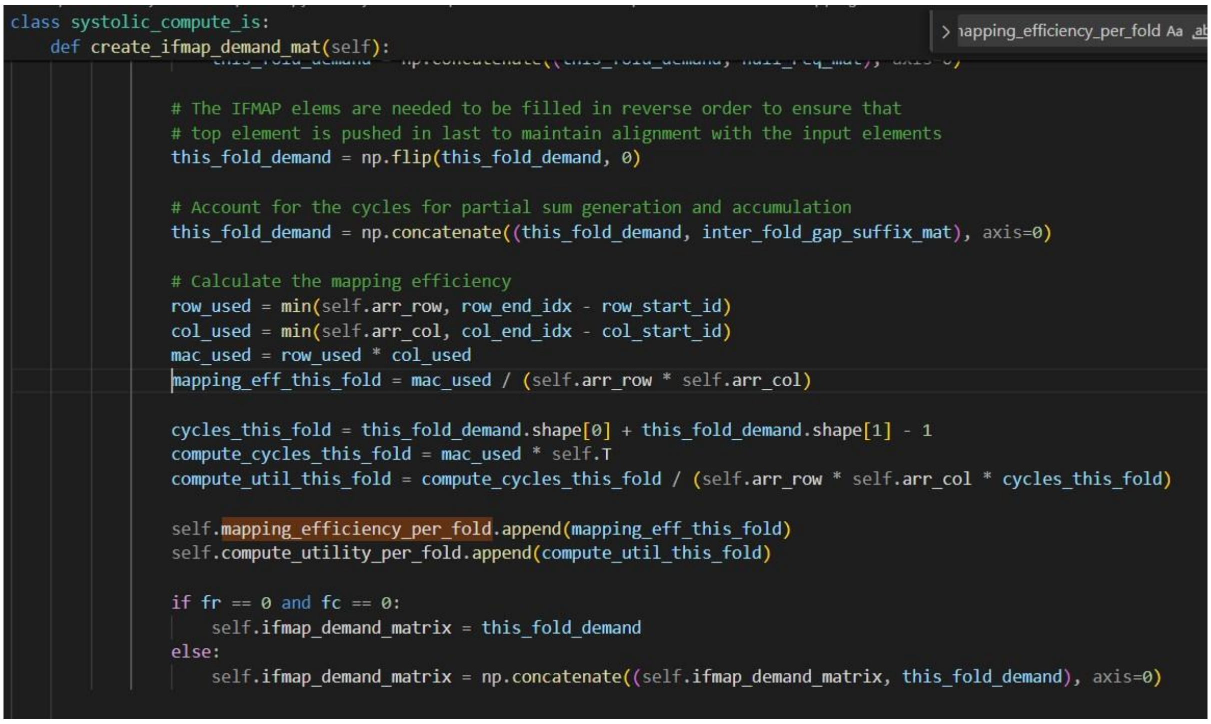
```
-1638.0,5.0,72.0,61.0,50.0,39.0,28.0,17.0,6.0,84.0,73.0
-1637.0,52.0,41.0,30.0,19.0,8.0,108.0,97.0,86.0,75.0,64.0
-1636.0,65.0,54.0,43.0,32.0,21.0,10.0,132.0,121.0,110.0,99.0
-1635.0,122.0,111.0,100.0,89.0,78.0,67.0,56.0,45.0,34.0,23.0
-1634.0,57.0,46.0,35.0,24.0,13.0,168.0,157.0,146.0,135.0,124.0
-1633.0,180.0,169.0,158.0,147.0,136.0,125.0,114.0,103.0,92.0,81.
-1632.0,148.0,137.0,126.0,115.0,104.0,93.0,82.0,71.0,-1.0,-1.0
-1631.0,127.0,116.0,105.0,94.0,83.0,72.0,61.0,50.0,39.0,28.0
-1630.0,117.0,106.0,95.0,84.0,73.0,-1.0,-1.0,-1.0,-1.0,-1.0
-1629.0,118.0,107.0,-1.0,-1.0,-1.0,-1.0,52.0,41.0,30.0,19.0
-1628.0,130.0,119.0,108.0,97.0,86.0,75.0,64.0,-1.0,-1.0,-1.0
-1627.0,153.0,142.0,131.0,-1.0,-1.0,-1.0,-1.0,-1.0,65.0,54.0
-1626.0,187.0,176.0,165.0,154.0,143.0,132.0,121.0,110.0,99.0,-1.
-1625.0,232.0,221.0,210.0,199.0,188.0,177.0,166.0,155.0,-1.0,-1.
-1624.0,288.0,277.0,266.0,255.0,244.0,233.0,222.0,211.0,200.0,18
-1623.0,-1.0,57.0,46.0,35.0,24.0,300.0,289.0,278.0,267.0,256.0
-1622.0,135.0,124.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,36.0
-1621.0,213.0,202.0,191.0,180.0,169.0,158.0,147.0,136.0,125.0,11
-1620.0,302.0,291.0,280.0,269.0,258.0,247.0,236.0,225.0,214.0,20
-1619.0,82.0,71.0,-1.0,-1.0,-1.0,-1.0,336.0,325.0,314.0,303.0
-1618.0,-1.0,-1.0,-1.0,-1.0,-1.0,127.0,116.0,105.0,94.0,83.0
-1617.0,293.0,282.0,271.0,260.0,249.0,238.0,227.0,-1.0,-1.0,-1.0
-1616.0,73.0,-1.0,-1.0,-1.0,-1.0,360.0,349.0,338.0,327.0,316.0
-1615.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,118.0,107.0,-1.0
-1614.0,328.0,317.0,306.0,295.0,284.0,273.0,262.0,251.0,-1.0,-1.
-1613.0,108.0,97.0,86.0,75.0,64.0,-1.0,-1.0,-1.0,384.0,373.0
-1612.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,153.0
-1611.0,396.0,385.0,374.0,363.0,352.0,341.0,330.0,319.0,308.0,29
-1610.0,176.0,165.0,154.0,143.0,132.0,121.0,110.0,99.0,-1.0,-1.0
-1609.0,320.0,309.0,298.0,287.0,-1.0,-1.0,-1.0,-1.0,232.0,221.0
-1608.0,100.0,89.0,78.0,67.0,420.0,409.0,398.0,387.0,376.0,365.0
-1607.0,244.0,233.0,222.0,211.0,200.0,189.0,-1.0,-1.0,-1.0,-1.0
-1606.0,388.0,377.0,366.0,355.0,344.0,333.0,322.0,311.0,300.0,28
-1605.0,168.0,157.0,146.0,135.0,124.0,-1.0,-1.0,-1.0,444.0,433.0
-1604.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,213.0
```

```
-1637.0,62.0,51.0,40.0,29.0,18.0,7.0,96.0,85.0,74.0,63.0
-1636.0,53.0,42.0,31.0,20.0,9.0,120.0,109.0,98.0,87.0,76.0
-1635.0,88.0,77.0,66.0,55.0,44.0,33.0,22.0,11.0,144.0,133.0
-1634.0,-1.0,156.0,145.0,134.0,123.0,112.0,101.0,90.0,79.0,68.0
-1633.0,113.0,102.0,91.0,80.0,69.0,58.0,47.0,-1.0,-1.0,-1.0
-1632.0,70.0,59.0,48.0,37.0,26.0,15.0,192.0,181.0,170.0,159.0
-1631.0,38.0,27.0,16.0,204.0,193.0,182.0,171.0,160.0,149.0,138.0
-1630.0,-1.0,216.0,205.0,194.0,183.0,172.0,161.0,150.0,139.0,128
-1629.0,228.0,217.0,206.0,195.0,184.0,173.0,162.0,151.0,140.0,12
-1628.0,240.0,229.0,218.0,207.0,196.0,185.0,174.0,163.0,152.0,14
-1627.0,20.0,252.0,241.0,230.0,219.0,208.0,197.0,186.0,175.0,164
-1626.0,-1.0,-1.0,-1.0,264.0,253.0,242.0,231.0,220.0,209.0,198.0
-1625.0,77.0,66.0,55.0,44.0,33.0,22.0,276.0,265.0,254.0,243.0
-1624.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0
-1623.0,178.0,167.0,156.0,145.0,134.0,123.0,112.0,101.0,90.0,79.
-1622.0,245.0,234.0,223.0,212.0,201.0,190.0,179.0,-1.0,-1.0,-1.0
-1621.0,-1.0,312.0,301.0,290.0,279.0,268.0,257.0,246.0,235.0,224
-1620.0,-1.0,-1.0,-1.0,70.0,59.0,48.0,37.0,26.0,324.0,313.0
-1619.0,192.0,181.0,170.0,159.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0
-1618.0,292.0,281.0,270.0,259.0,248.0,237.0,226.0,215.0,204.0,19
-1617.0,-1.0,-1.0,-1.0,-1.0,-1.0,348.0,337.0,326.0,315.0,304.0
-1616.0,183.0,172.0,161.0,150.0,139.0,128.0,-1.0,-1.0,-1.0,-1.0
-1615.0,305.0,294.0,283.0,272.0,261.0,250.0,239.0,228.0,217.0,20
-1614.0,85.0,74.0,63.0,-1.0,-1.0,-1.0,372.0,361.0,350.0,339.0
-1613.0,218.0,207.0,196.0,185.0,174.0,163.0,152.0,141.0,-1.0,-1.
-1612.0,362.0,351.0,340.0,329.0,318.0,307.0,296.0,285.0,274.0,26
-1611.0,-1.0,-1.0,120.0,109.0,98.0,87.0,76.0,-1.0,-1.0,-1.0
-1610.0,286.0,275.0,264.0,253.0,242.0,231.0,220.0,209.0,198.0,-1
-1609.0,66.0,55.0,408.0,397.0,386.0,375.0,364.0,353.0,342.0,331.
-1608.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,144.0,133.0,-1.0,-1.0
-1607.0,354.0,343.0,332.0,321.0,310.0,299.0,-1.0,-1.0,-1.0,-1.0
-1606.0,134.0,123.0,112.0,101.0,90.0,79.0,432.0,421.0,410.0,399.
-1605.0,-1.0,-1.0,-1.0,245.0,234.0,223.0,212.0,201.0,190.0,179.0
-1604.0,422.0,411.0,400.0,389.0,378.0,367.0,356.0,345.0,334.0,32
-1603.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0
```

Figure 8. '-1's are put if the blocks are previously fetched from different DRAM

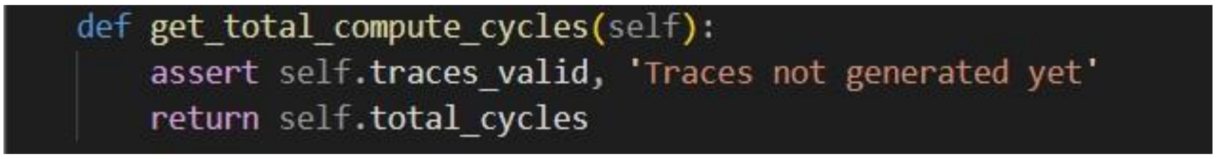
Results and Observations: -

1. Mapping efficiency- Depends on number of MACs, array height and width of systolic arrays. This can't be improved by modifying memory mapping.
   Although, by altering row and column size we can increase mapping efficiency.



```
class systolic_compute_is:
    def create_ifmap_demand_mat(self):

                # The IFMAP elems are needed to be filled in reverse order to ensure that
                # top element is pushed in last to maintain alignment with the input elements
                this_fold_demand = np.flip(this_fold_demand, 0)

                # Account for the cycles for partial sum generation and accumulation
                this_fold_demand = np.concatenate((this_fold_demand, inter_fold_gap_suffix_mat), axis=0)

                # Calculate the mapping efficiency
                row_used = min(self.arr_row, row_end_idx - row_start_id)
                col_used = min(self.arr_col, col_end_idx - col_start_id)
                mac_used = row_used * col_used
                mapping_eff_this_fold = mac_used / (self.arr_row * self.arr_col)

                cycles_this_fold = this_fold_demand.shape[0] + this_fold_demand.shape[1] - 1
                compute_cycles_this_fold = mac_used * self.T
                compute_util_this_fold = compute_cycles_this_fold / (self.arr_row * self.arr_col * cycles_this_fold)

                self.mapping_efficiency_per_fold.append(mapping_eff_this_fold)
                self.compute_utility_per_fold.append(compute_util_this_fold)

                if fr == 0 and fc == 0:
                    self.ifmap_demand_matrix = this_fold_demand
                else:
                    self.ifmap_demand_matrix = np.concatenate((self.ifmap_demand_matrix, this_fold_demand), axis=0)
```

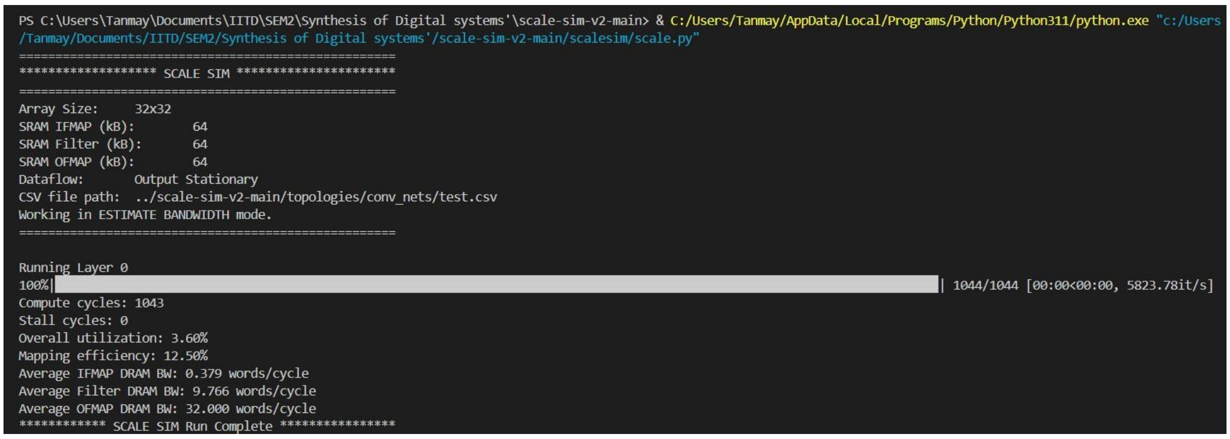
2. Compute cycles- Compute cycles are total cycles after writing the output feature map values.



```
def get_total_compute_cycles(self):
    assert self.traces_valid, 'Traces not generated yet'
    return self.total_cycles
```

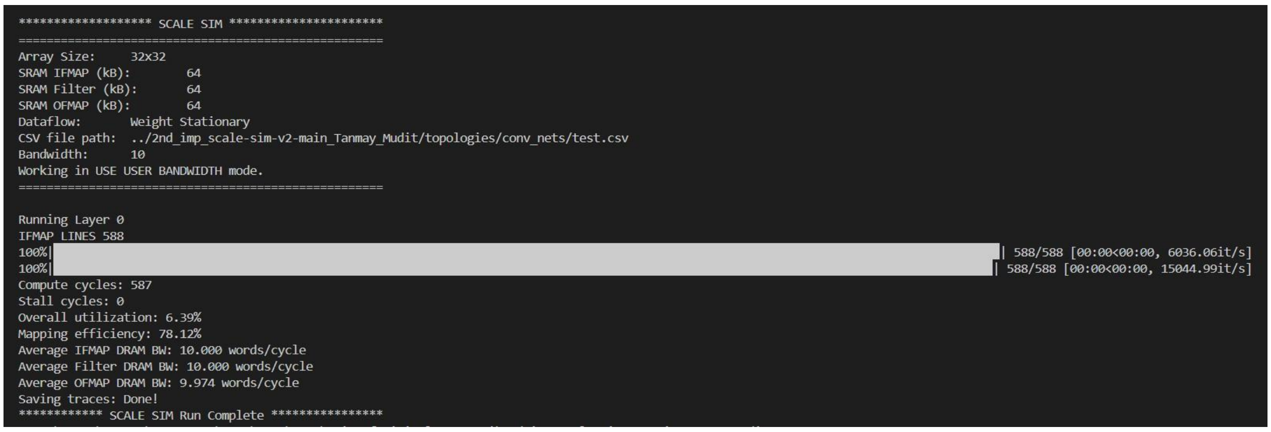
   Ways compute cycles are decreased: -

   a) Increasing row size sent per cycle.
   b) Decreasing cycles during Read and Write:
      In our implementation, total cycles by half by fetching 2 rows in a single cycle through 2 DRAMs.



```
PS C:\Users\Tanmay\Documents\IITD\SEM2\Synthesis of Digital systems'\scale-sim-v2-main> & C:/Users/Tanmay/AppData/Local/Programs/Python/Python311/python.exe "c:/Users
/Tanmay/Documents/IITD/SEM2/Synthesis of Digital systems'/scale-sim-v2-main/scalesim/scale.py"
====================================================
****************** SCALE SIM *********************
====================================================
Array Size:     32x32
SRAM IFMAP (kB):        64
SRAM Filter (kB):       64
SRAM OFMAP (kB):        64
Dataflow:       Output Stationary
CSV file path:  ../scale-sim-v2-main/topologies/conv_nets/test.csv
Working in ESTIMATE BANDWIDTH mode.
====================================================

Running Layer 0
100%|                                                                  | 1044/1044 [00:00<00:00, 5823.78it/s]
Compute cycles: 1043
Stall cycles: 0
Overall utilization: 3.60%
Mapping efficiency: 12.50%
Average IFMAP DRAM BW: 0.379 words/cycle
Average Filter DRAM BW: 9.766 words/cycle
Average OFMAP DRAM BW: 32.000 words/cycle
************ SCALE SIM Run Complete ****************
```

Figure 9. Original Scale-sim results



```
****************** SCALE SIM *********************
====================================================
Array Size:     32x32
SRAM IFMAP (kB):        64
SRAM Filter (kB):       64
SRAM OFMAP (kB):        64
Dataflow:       Weight Stationary
CSV file path:  ../2nd_imp_scale-sim-v2-main_Tanmay_Mudit/topologies/conv_nets/test.csv
Bandwidth:      10
Working in USE USER BANDWIDTH mode.
====================================================

Running Layer 0
IFMAP LINES 588
100%|                                                                  | 588/588 [00:00<00:00, 6036.06it/s]
100%|                                                                 | 588/588 [00:00<00:00, 15044.99it/s]
Compute cycles: 587
Stall cycles: 0
Overall utilization: 6.39%
Mapping efficiency: 78.12%
Average IFMAP DRAM BW: 10.000 words/cycle
Average Filter DRAM BW: 10.000 words/cycle
Average OFMAP DRAM BW: 9.974 words/cycle
Saving traces: Done!
************ SCALE SIM Run Complete ****************
```

Figure 10. New Scale-sim results

3. Overall Utilization-

```
self.overall_util = (self.num_compute * 100) / (self.total_cycles * self.num_mac_unit)
```

This formula computes what portion of the maximum possible computational capacity is actually being utilized. This can provide insights into how efficiently the computational resources are being used.

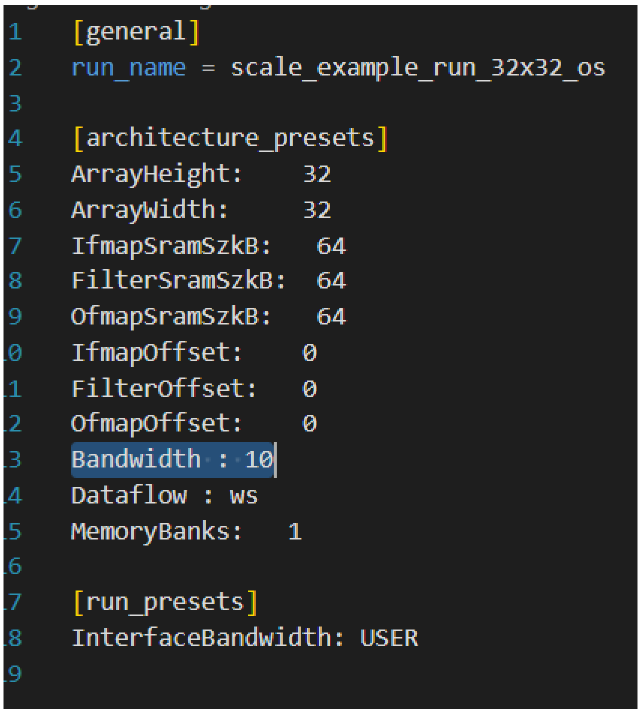
➔As shown in the above figures, there's about twice the increase in overall utilization.

4. Average DRAM BW:-

In the config file, we have input the bandwidth of DRAMs.

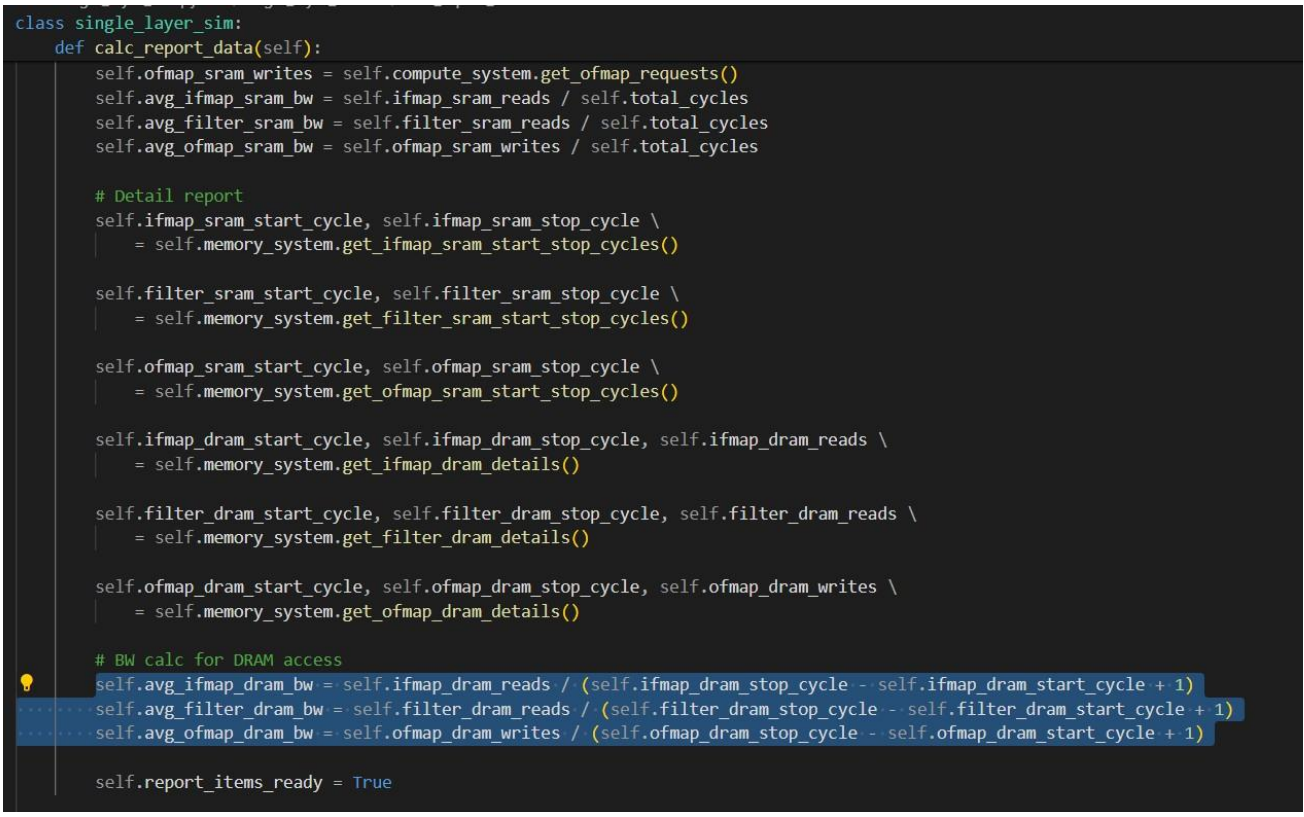
So, the average DRAM BW should not be more than this value. Whereas, the total DRAM bandwidth will be:-

Total DRAM BW= Avg. DRAM BW * No. of DRAMs



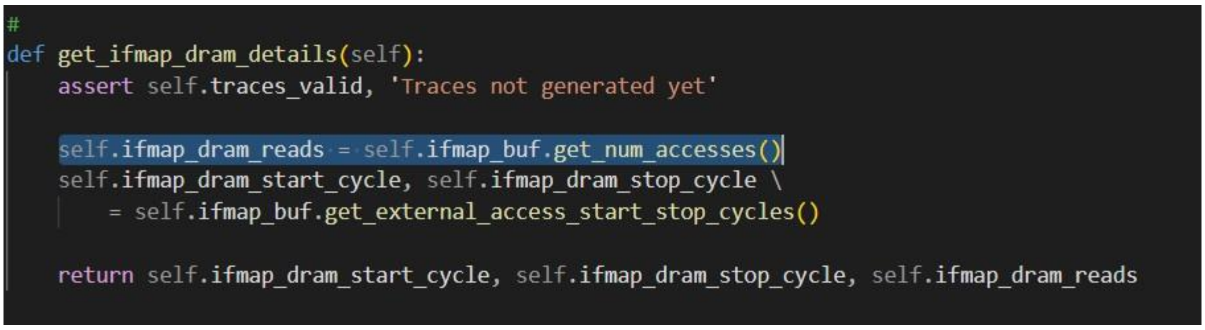
```
[general]
run_name = scale_example_run_32x32_os

[architecture_presets]
ArrayHeight:    32
ArrayWidth:     32
IfmapSramSzkB:   64
FilterSramSzkB:  64
OfmapSramSzkB:   64
IfmapOffset:    0
FilterOffset:   0
OfmapOffset:    0
Bandwidth : 10
Dataflow : ws
MemoryBanks:   1

[run_presets]
InterfaceBandwidth: USER
```

Bandwidth is dependent on reads/writes and start, stop cycles.



```
class single_layer_sim:
    def calc_report_data(self):
        self.ofmap_sram_writes = self.compute_system.get_ofmap_requests()
        self.avg_ifmap_sram_bw = self.ifmap_sram_reads / self.total_cycles
        self.avg_filter_sram_bw = self.filter_sram_reads / self.total_cycles
        self.avg_ofmap_sram_bw = self.ofmap_sram_writes / self.total_cycles

        # Detail report
        self.ifmap_sram_start_cycle, self.ifmap_sram_stop_cycle \
            = self.memory_system.get_ifmap_sram_start_stop_cycles()

        self.filter_sram_start_cycle, self.filter_sram_stop_cycle \
            = self.memory_system.get_filter_sram_start_stop_cycles()

        self.ofmap_sram_start_cycle, self.ofmap_sram_stop_cycle \
            = self.memory_system.get_ofmap_sram_start_stop_cycles()

        self.ifmap_dram_start_cycle, self.ifmap_dram_stop_cycle, self.ifmap_dram_reads \
            = self.memory_system.get_ifmap_dram_details()

        self.filter_dram_start_cycle, self.filter_dram_stop_cycle, self.filter_dram_reads \
            = self.memory_system.get_filter_dram_details()

        self.ofmap_dram_start_cycle, self.ofmap_dram_stop_cycle, self.ofmap_dram_writes \
            = self.memory_system.get_ofmap_dram_details()

        # BW calc for DRAM access
        self.avg_ifmap_dram_bw = self.ifmap_dram_reads / (self.ifmap_dram_stop_cycle - self.ifmap_dram_start_cycle + 1)
        self.avg_filter_dram_bw = self.filter_dram_reads / (self.filter_dram_stop_cycle - self.filter_dram_start_cycle + 1)
        self.avg_ofmap_dram_bw = self.ofmap_dram_writes / (self.ofmap_dram_stop_cycle - self.ofmap_dram_start_cycle + 1)

        self.report_items_ready = True
```

Reads/Writes are computed using number of accesses either for fetching or writing on addresses.



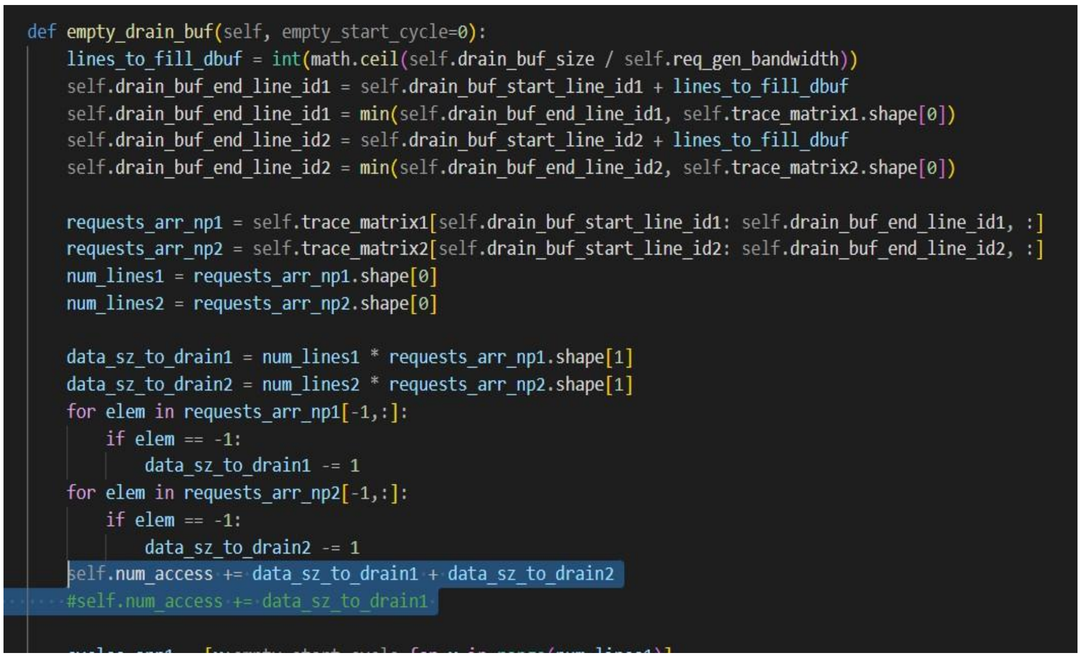
```
#
def get_ifmap_dram_details(self):
    assert self.traces_valid, 'Traces not generated yet'

    self.ifmap_dram_reads = self.ifmap_buf.get_num_accesses()
    self.ifmap_dram_start_cycle, self.ifmap_dram_stop_cycle \
        = self.ifmap_buf.get_external_access_start_stop_cycles()

    return self.ifmap_dram_start_cycle, self.ifmap_dram_stop_cycle, self.ifmap_dram_reads
```

num_accesses are calculated using addition of number of data blocks.

- This is a simulation of CNN accelerator. We are performing DRAM accesses parallelly (by creating different read ports for different DRAMs).

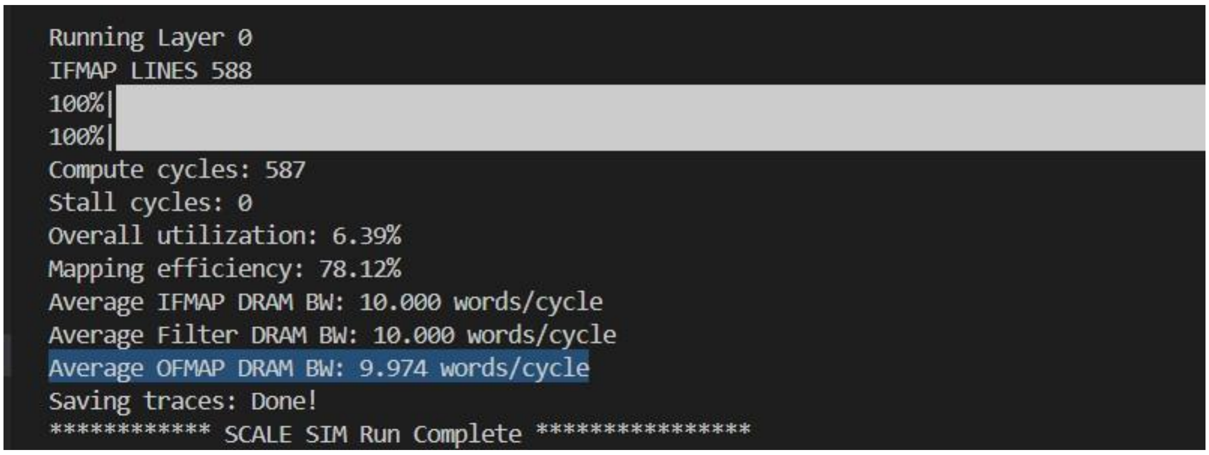
So, I’ve modified num_access to the total number of data blocks *accessed by 1 DRAM* as other DRAMs are accessed in parallel.

➔For demonstration purposes, let’s change num_access to total number of data blocks fetched for writing in OFMAP.



```
def empty_drain_buf(self, empty_start_cycle=0):
    lines_to_fill_dbuf = int(math.ceil(self.drain_buf_size / self.req_gen_bandwidth))
    self.drain_buf_end_line_id1 = self.drain_buf_start_line_id1 + lines_to_fill_dbuf
    self.drain_buf_end_line_id1 = min(self.drain_buf_end_line_id1, self.trace_matrix1.shape[0])
    self.drain_buf_end_line_id2 = self.drain_buf_start_line_id2 + lines_to_fill_dbuf
    self.drain_buf_end_line_id2 = min(self.drain_buf_end_line_id2, self.trace_matrix2.shape[0])

    requests_arr_np1 = self.trace_matrix1[self.drain_buf_start_line_id1: self.drain_buf_end_line_id1, :]
    requests_arr_np2 = self.trace_matrix2[self.drain_buf_start_line_id2: self.drain_buf_end_line_id2, :]
    num_lines1 = requests_arr_np1.shape[0]
    num_lines2 = requests_arr_np2.shape[0]

    data_sz_to_drain1 = num_lines1 * requests_arr_np1.shape[1]
    data_sz_to_drain2 = num_lines2 * requests_arr_np2.shape[1]
    for elem in requests_arr_np1[-1,:]:
        if elem == -1:
            data_sz_to_drain1 -= 1
    for elem in requests_arr_np2[-1,:]:
        if elem == -1:
            data_sz_to_drain2 -= 1
    self.num_access += data_sz_to_drain1 + data_sz_to_drain2
    #self.num_access += data_sz_to_drain1
```

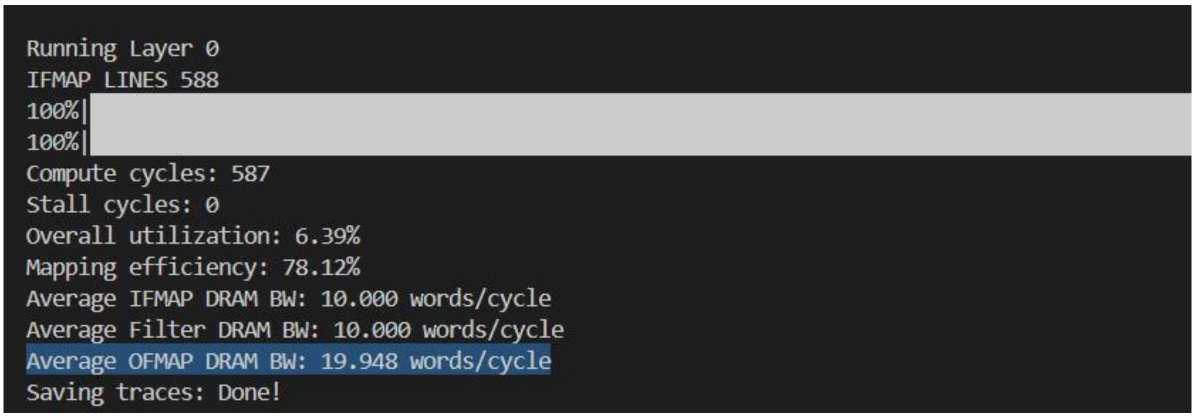
Avg DRAM BW when accessed parallelly: -



```
Running Layer 0
IFMAP LINES 588
100%|
100%|
Compute cycles: 587
Stall cycles: 0
Overall utilization: 6.39%
Mapping efficiency: 78.12%
Average IFMAP DRAM BW: 10.000 words/cycle
Average Filter DRAM BW: 10.000 words/cycle
Average OFMAP DRAM BW: 9.974 words/cycle
Saving traces: Done!
************ SCALE SIM Run Complete ****************
```

Figure 11. Correct Avg DRAM BW

Avg DRAM BW when data blocks are accessed sequentially (Incorrect) : -



```
Running Layer 0
IFMAP LINES 588
100%|
100%|
Compute cycles: 587
Stall cycles: 0
Overall utilization: 6.39%
Mapping efficiency: 78.12%
Average IFMAP DRAM BW: 10.000 words/cycle
Average Filter DRAM BW: 10.000 words/cycle
Average OFMAP DRAM BW: 19.948 words/cycle
Saving traces: Done!
```