PART 1 Questions and Answers

a. How many arithmetic operations are required to multiply a 4 by 4 matrix with a vector of length 4? Then, generalize this: how many arithmetic operations are required to multiply a k by k with a vector of length k?

If we have 4 by 4 matrix with a vector with 4 x values in it, then

- With the first row of the w matrix and the first x value of a vector, there will be 4 multiplication and 3 additions, meaning that it needs 7 arithmetic operations to deal with each row of *w* and each *x* value of the vector. In other words, the first output, y_0, would be $y_0 = w_{0,0}x_0 + w_{0,1}x_1 + w_{0,2}x_2 + w_{0,3}x_3$.
- It will require repeating the same procedure for the other remaining 3 outputs, y_1,y_2 and y_3. To make these outputs, 21 arithmetic operations will need to take place.
- In total, **28** arithmetic operations are required to do 4 by 4 matrix-vector multiplication.

To generalize, for each row we do k multiplications and (k-1) addition to get the final result. With k rows, we do a total of k(k+k-1) operations.

Using inductive reasoning,

2 by 2 matrix vector multiplication needs 6 operations

2(2+2-1) = 6

3 by 3 matrix vector multiplication needs 15 operations

3(3+3-1) = 15

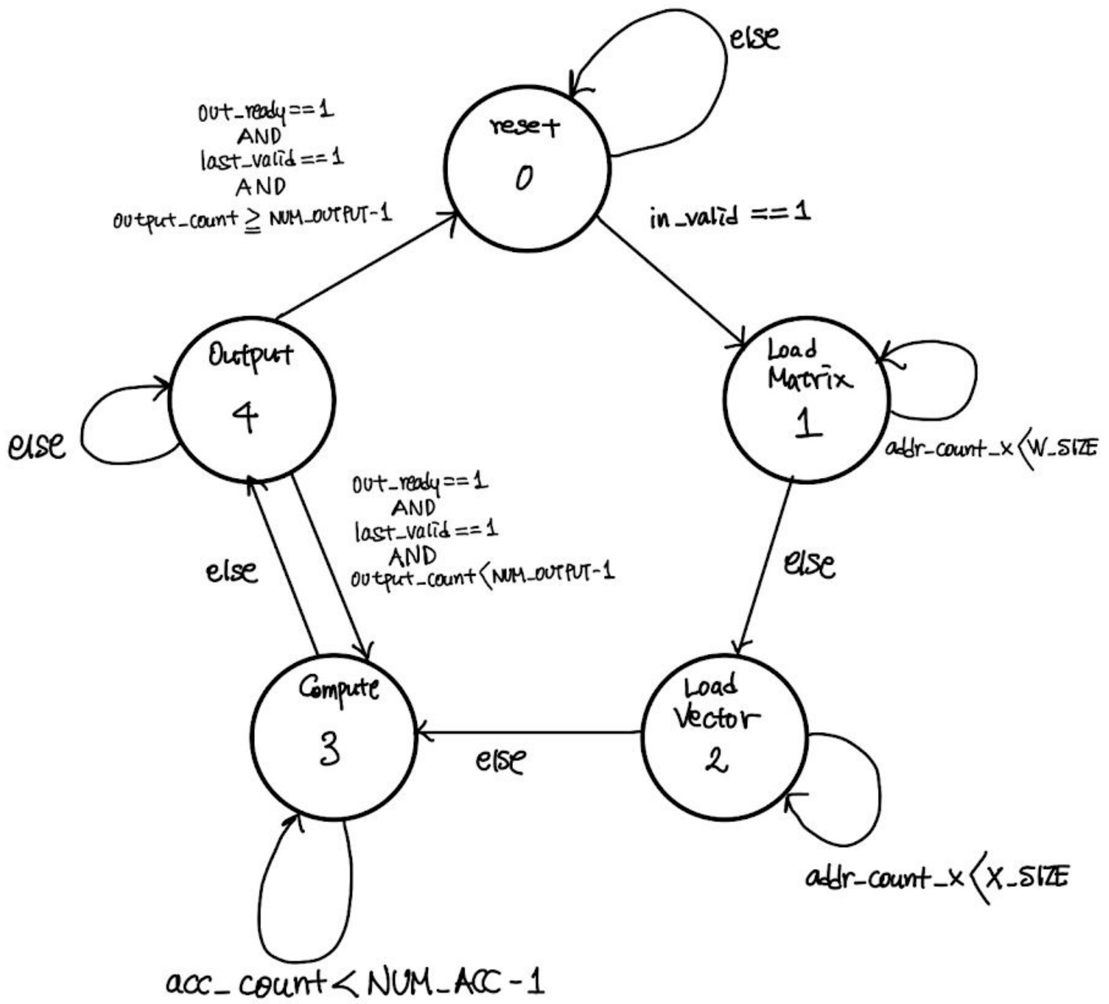
4 by 4 matrix vector multiplication needs 28 operations

4(4+4-1) = 28

So, if we have k by k matrix then we would need $k^2+(k-1)k$ or $\mathbf{2k^2-k}$ operations for k by k matrix-vector multiplication.

b. Explain how your control module works.What are the steps it goes through? How does it keep track of its place in execution?

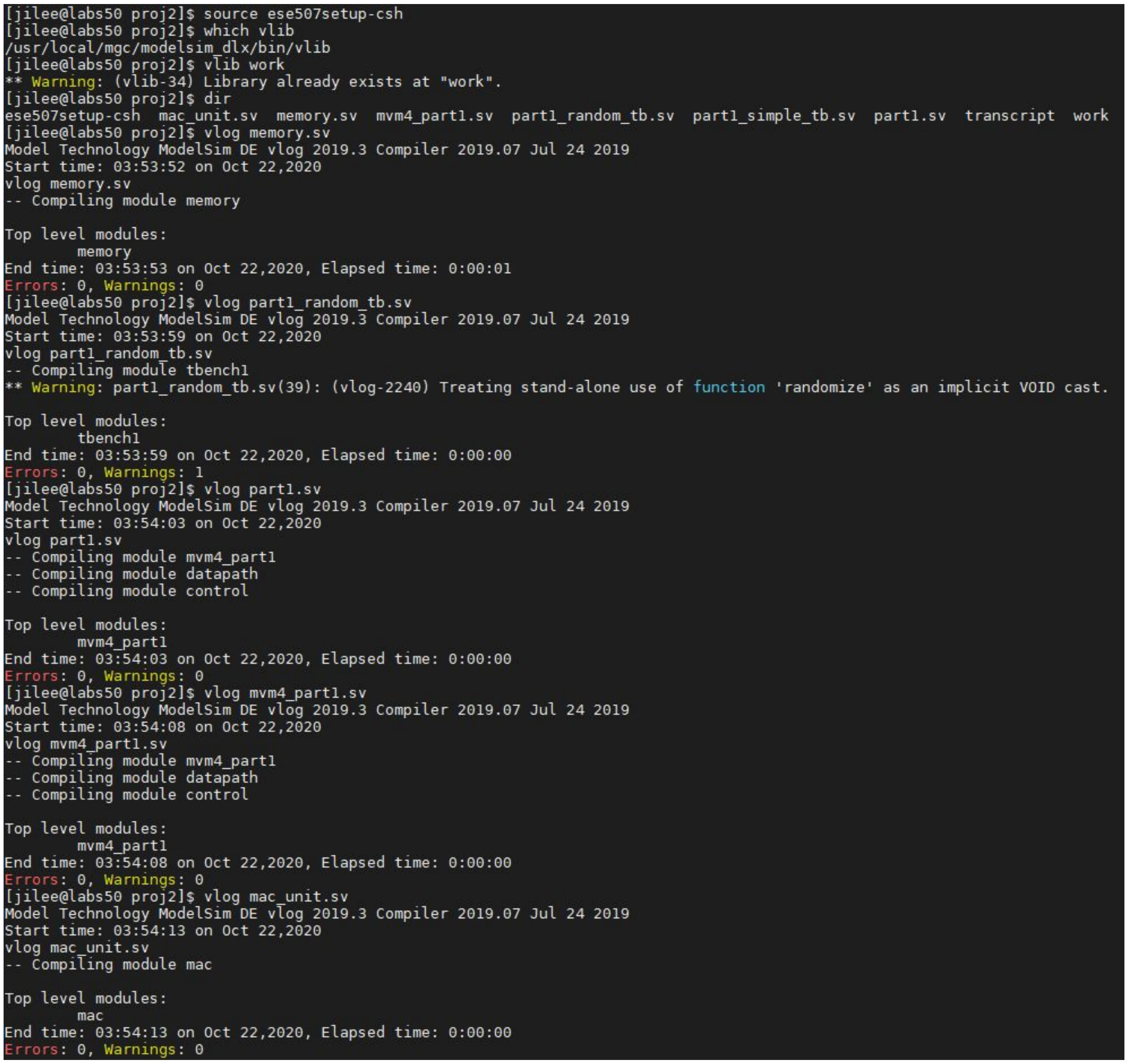
Our Control Module consists of 5 different FSM states:
State 0 is Reset State,
State 1 is Load Matrix,
State 2 is Load Vector,
State 3 is Compute,
State 4 is Output.
The conditions and directions to travel through the states are shown on the diagram below.
At positive clk edge, when reset is asserted state is 0, otherwise state becomes next_state.



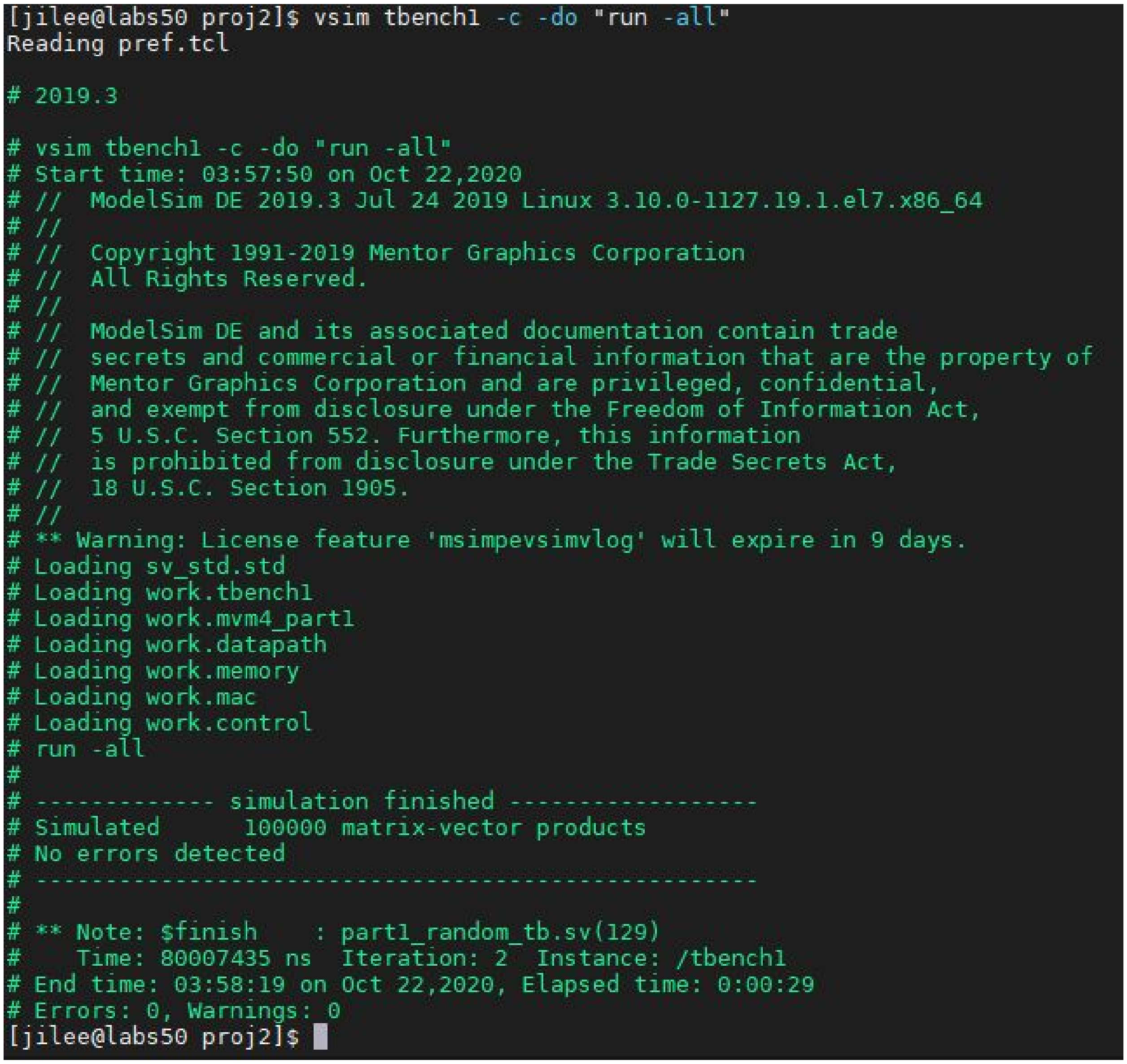
c. Explain how you verified your system. What problems or bugs did you encounter? Did you write any other testbenches besides those provided to you? (If you did, please include them with your submission.)

-Our simulation went smoothly with the provided testbenches except a few syntax errors which were alerted during the compile time, for example 'end module', which is not right, instead of 'endmodule'. At the end we could simulate our part1.sv without an error using the provided random testbench for part 1 following the vlog and vsim procedures shown below with various seeds.

The post-synthesis simulation generated some errors we weren't able to explain and solve, but these are likely due to non-design problems so we decided to proceed.



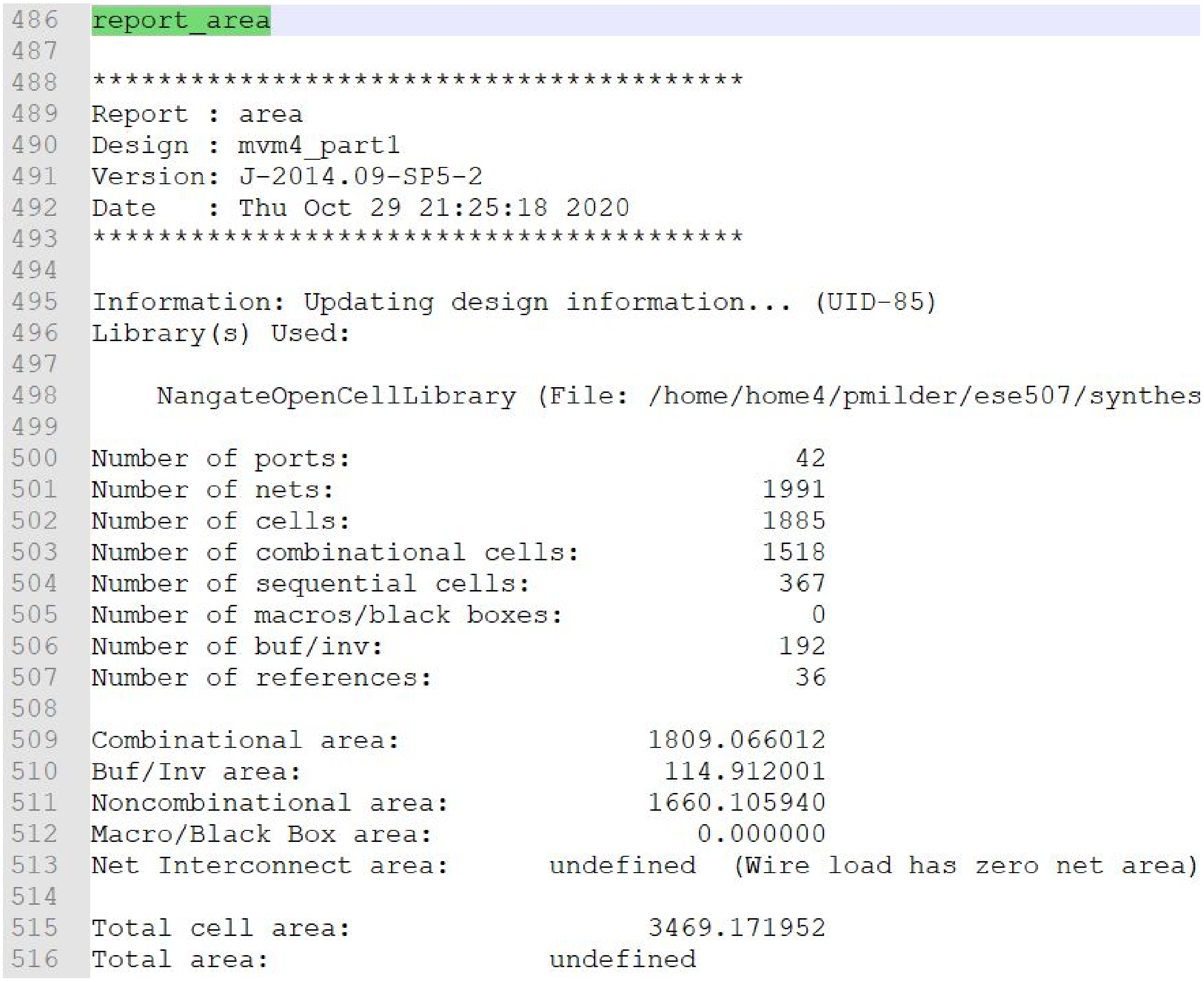
```
[jilee@labs50 proj2]$ source ese507setup-csh
[jilee@labs50 proj2]$ which vlib
/usr/local/mgc/modelsim_dlx/bin/vlib
[jilee@labs50 proj2]$ vlib work
** Warning: (vlib-34) Library already exists at "work".
[jilee@labs50 proj2]$ dir
ese507setup-csh  mac_unit.sv  memory.sv  mvm4_part1.sv  part1_random_tb.sv  part1_simple_tb.sv  part1.sv  transcript  work
[jilee@labs50 proj2]$ vlog memory.sv
Model Technology ModelSim DE vlog 2019.3 Compiler 2019.07 Jul 24 2019
Start time: 03:53:52 on Oct 22,2020
vlog memory.sv
-- Compiling module memory

Top level modules:
        memory
End time: 03:53:53 on Oct 22,2020, Elapsed time: 0:00:01
Errors: 0, Warnings: 0
[jilee@labs50 proj2]$ vlog part1_random_tb.sv
Model Technology ModelSim DE vlog 2019.3 Compiler 2019.07 Jul 24 2019
Start time: 03:53:59 on Oct 22,2020
vlog part1_random_tb.sv
-- Compiling module tbench1
** Warning: part1_random_tb.sv(39): (vlog-2240) Treating stand-alone use of function 'randomize' as an implicit VOID cast.

Top level modules:
        tbench1
End time: 03:53:59 on Oct 22,2020, Elapsed time: 0:00:00
Errors: 0, Warnings: 1
[jilee@labs50 proj2]$ vlog part1.sv
Model Technology ModelSim DE vlog 2019.3 Compiler 2019.07 Jul 24 2019
Start time: 03:54:03 on Oct 22,2020
vlog part1.sv
-- Compiling module mvm4_part1
-- Compiling module datapath
-- Compiling module control

Top level modules:
        mvm4_part1
End time: 03:54:03 on Oct 22,2020, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
[jilee@labs50 proj2]$ vlog mvm4_part1.sv
Model Technology ModelSim DE vlog 2019.3 Compiler 2019.07 Jul 24 2019
Start time: 03:54:08 on Oct 22,2020
vlog mvm4_part1.sv
-- Compiling module mvm4_part1
-- Compiling module datapath
-- Compiling module control

Top level modules:
        mvm4_part1
End time: 03:54:08 on Oct 22,2020, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
[jilee@labs50 proj2]$ vlog mac_unit.sv
Model Technology ModelSim DE vlog 2019.3 Compiler 2019.07 Jul 24 2019
Start time: 03:54:13 on Oct 22,2020
vlog mac_unit.sv
-- Compiling module mac

Top level modules:
        mac
End time: 03:54:13 on Oct 22,2020, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
```



```
[jilee@labs50 proj2]$ vsim tbench1 -c -do "run -all"
Reading pref.tcl

# 2019.3

# vsim tbench1 -c -do "run -all"
# Start time: 03:57:50 on Oct 22,2020
# //  ModelSim DE 2019.3 Jul 24 2019 Linux 3.10.0-1127.19.1.el7.x86_64
# //
# //  Copyright 1991-2019 Mentor Graphics Corporation
# //  All Rights Reserved.
# //
# //  ModelSim DE and its associated documentation contain trade
# //  secrets and commercial or financial information that are the property of
# //  Mentor Graphics Corporation and are privileged, confidential,
# //  and exempt from disclosure under the Freedom of Information Act,
# //  5 U.S.C. Section 552. Furthermore, this information
# //  is prohibited from disclosure under the Trade Secrets Act,
# //  18 U.S.C. Section 1905.
# //
# ** Warning: License feature 'msimpevsimvlog' will expire in 9 days.
# Loading sv_std.std
# Loading work.tbench1
# Loading work.mvm4_part1
# Loading work.datapath
# Loading work.memory
# Loading work.mac
# Loading work.control
# run -all
#
# ------------- simulation finished ------------------
# Simulated      100000 matrix-vector products
# No errors detected
# ----------------------------------------------------
#
# ** Note: $finish    : part1_random_tb.sv(129)
#    Time: 80007435 ns  Iteration: 2  Instance: /tbench1
# End time: 03:58:19 on Oct 22,2020, Elapsed time: 0:00:29
# Errors: 0, Warnings: 0
[jilee@labs50 proj2]$ 
```

d. Report the area, power, frequency , and critical path location you determined from your synthesis report. Explain the critical path location descriptively; in other words, explain where the path flows through your design and why it makes sense that it is the critical path.

Area of Combinational (logic gates) cells is 1809.066012 µm²
Non-combinational (flip flops and registers) is 1660.105940 µm²
Total Cell Area of estimate, which is the sum of combinational and non-combinational, is **3469.171952 µm²**

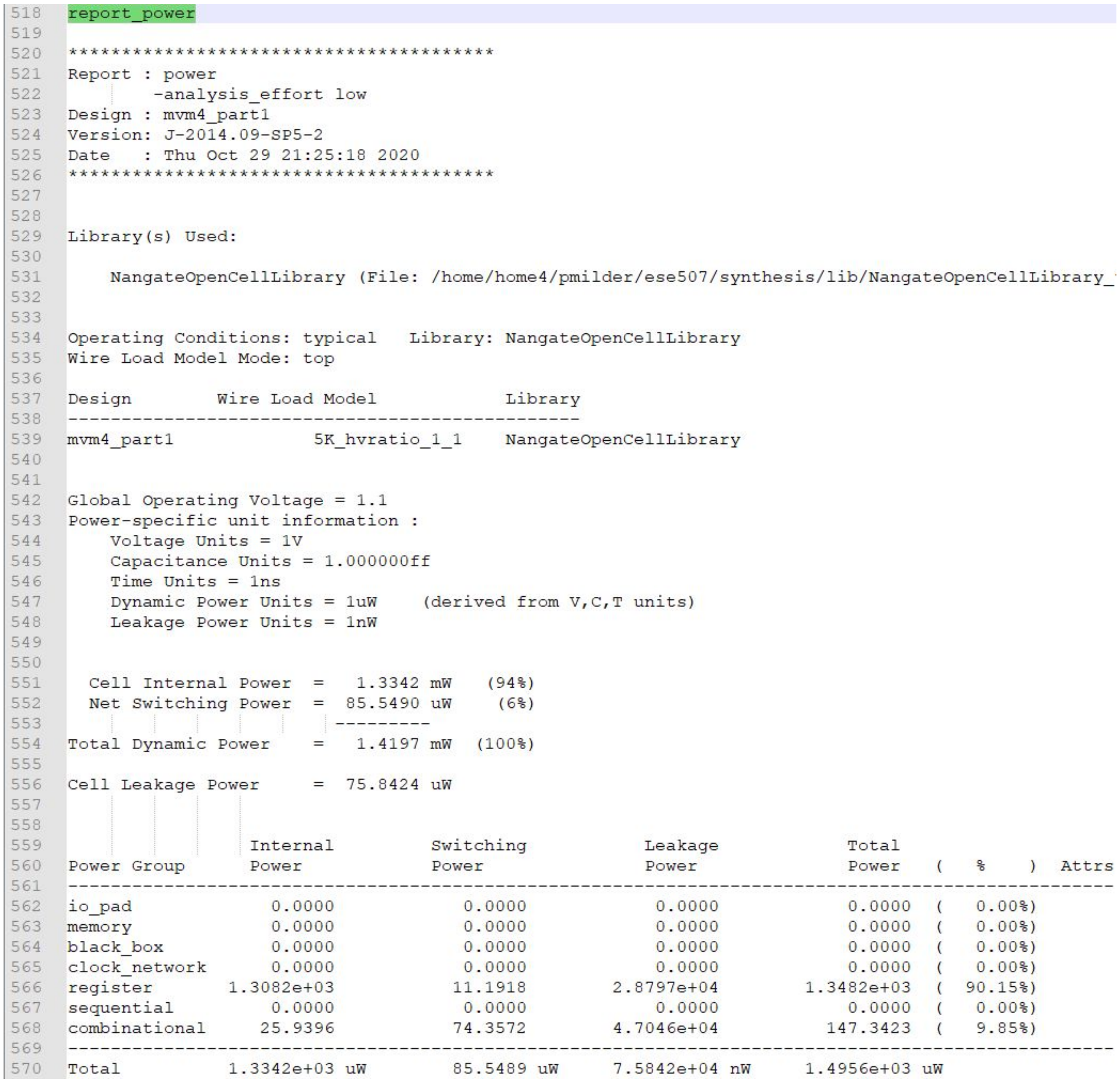


```
report_area

****************************************
Report : area
Design : mvm4_part1
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:25:18 2020
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    NangateOpenCellLibrary (File: /home/home4/pmilder/ese507/synthes

Number of ports:                           42
Number of nets:                          1991
Number of cells:                         1885
Number of combinational cells:           1518
Number of sequential cells:               367
Number of macros/black boxes:               0
Number of buf/inv:                        192
Number of references:                      36

Combinational area:               1809.066012
Buf/Inv area:                      114.912001
Noncombinational area:            1660.105940
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  3469.171952
Total area:                 undefined
```

The **Total Dynamic power is 1.4197 mW** and it is the sum of Cell Internal(logic) and Net Switching Power(wires). Total dynamic power is almost linearly correlated with the clock frequency. Static power does not correlate with the clock frequency and it is affected by the area proportionally instead.

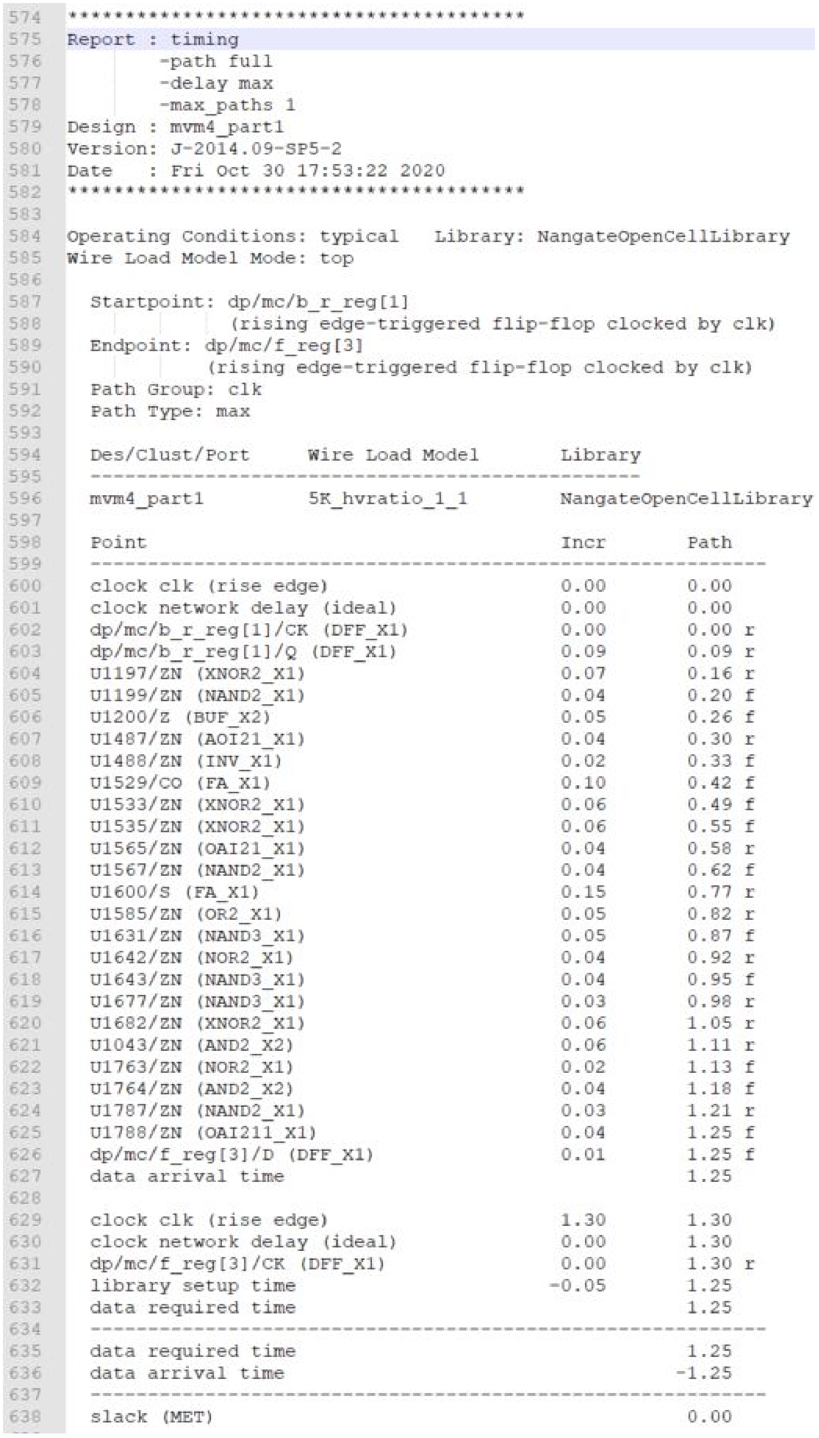
Total Power is Total Dynamic Power plus Cell Leakage Power:

1.4197 mW + 75.8424 µW = **1.496 mW**



```
report_power

****************************************
Report : power
        -analysis_effort low
Design : mvm4_part1
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:25:18 2020
****************************************


Library(s) Used:

    NangateOpenCellLibrary (File: /home/home4/pmilder/ese507/synthesis/lib/NangateOpenCellLibrary_


Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
mvm4_part1             5K_hvratio_1_1    NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =   1.3342 mW   (94%)
  Net Switching Power  =  85.5490 uW    (6%)
                         ---------
Total Dynamic Power    =   1.4197 mW  (100%)

Cell Leakage Power     =  75.8424 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad              0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory              0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box           0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network       0.0000           0.0000            0.0000            0.0000  (   0.00%)
register        1.3082e+03          11.1918        2.8797e+04        1.3482e+03  (  90.15%)
sequential          0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational      25.9396          74.3572        4.7046e+04          147.3423  (   9.85%)
--------------------------------------------------------------------------------------------------
Total           1.3342e+03 uW       85.5489 uW     7.5842e+04 nW     1.4956e+03 uW
```

Critical Path: from the input register (b_reg) to the output register(f_reg) of our MAC unit, same as in Project 1. We were able to modify our MAC unit since we received feedback on Project 1 before the due date but it still contains the critical path spanning through the multiply and add logic.
Our requested clock period was **1.3 ns**, which is about **769MHz in frequency.**
FF setup time was 0.05ns. So, data is required to arrive in less than or equal to 1.25 ns. Data arrives right on time at 1.25 ns. This makes 0 ns of slack which meets the timing requirement. The critical path goes from data_out_reg, gates to testf_reg.



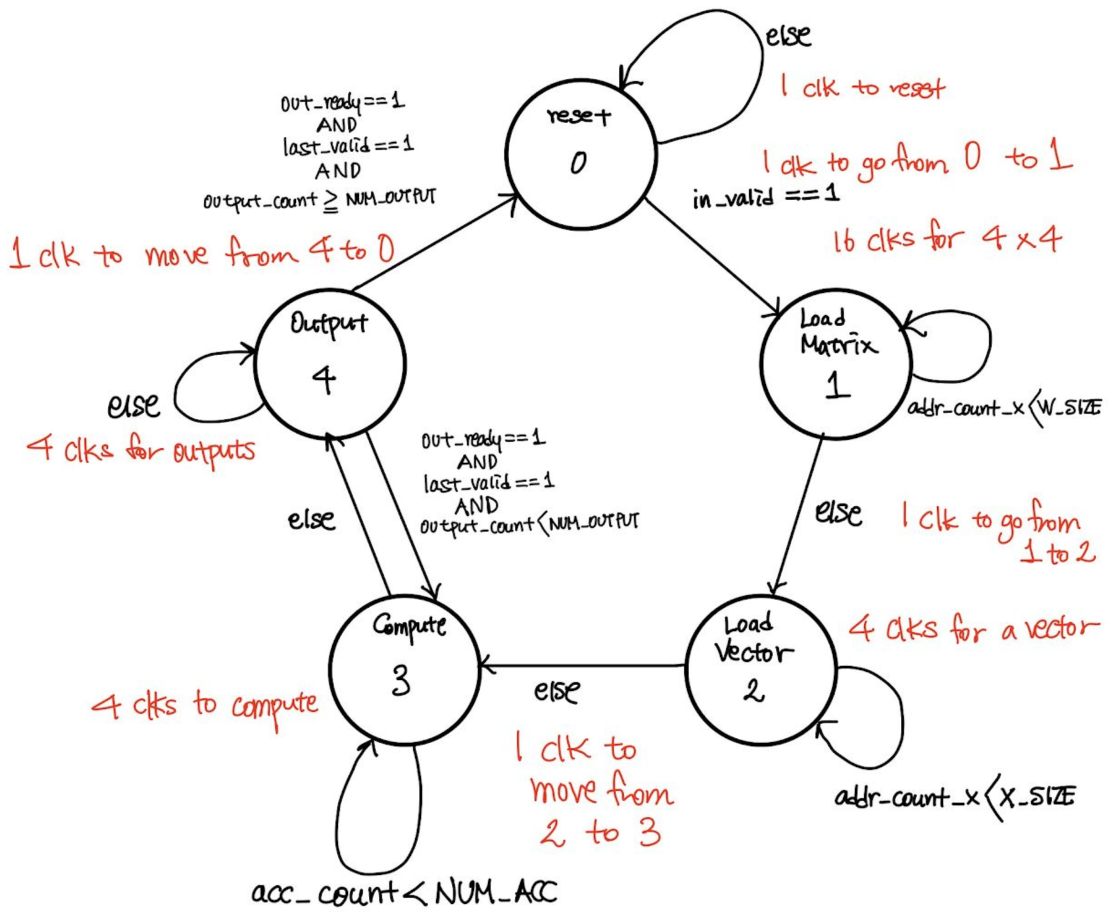
```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : mvm4_part1
Version: J-2014.09-SP5-2
Date   : Fri Oct 30 17:53:22 2020
****************************************

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: dp/mc/b_r_reg[1]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: dp/mc/f_reg[3]
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  mvm4_part1         5K_hvratio_1_1        NangateOpenCellLibrary

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  dp/mc/b_r_reg[1]/CK (DFF_X1)             0.00       0.00 r
  dp/mc/b_r_reg[1]/Q (DFF_X1)              0.09       0.09 r
  U1197/ZN (XNOR2_X1)                      0.07       0.16 r
  U1199/ZN (NAND2_X1)                      0.04       0.20 f
  U1200/Z (BUF_X2)                         0.05       0.26 f
  U1487/ZN (AOI21_X1)                      0.04       0.30 r
  U1488/ZN (INV_X1)                        0.02       0.33 f
  U1529/CO (FA_X1)                         0.10       0.42 f
  U1533/ZN (XNOR2_X1)                      0.06       0.49 f
  U1535/ZN (XNOR2_X1)                      0.06       0.55 f
  U1565/ZN (OAI21_X1)                      0.04       0.58 r
  U1567/ZN (NAND2_X1)                      0.04       0.62 f
  U1600/S (FA_X1)                          0.15       0.77 r
  U1585/ZN (OR2_X1)                        0.05       0.82 r
  U1631/ZN (NAND3_X1)                      0.05       0.87 f
  U1642/ZN (NOR2_X1)                       0.04       0.92 r
  U1643/ZN (NAND3_X1)                      0.04       0.95 f
  U1677/ZN (NAND3_X1)                      0.03       0.98 r
  U1682/ZN (XNOR2_X1)                      0.06       1.05 r
  U1043/ZN (AND2_X2)                       0.06       1.11 r
  U1763/ZN (NOR2_X1)                       0.02       1.13 f
  U1764/ZN (AND2_X2)                       0.04       1.18 f
  U1787/ZN (NAND2_X1)                      0.03       1.21 r
  U1788/ZN (OAI211_X1)                     0.04       1.25 f
  dp/mc/f_reg[3]/D (DFF_X1)                0.01       1.25 f
  data arrival time                                   1.25

  clock clk (rise edge)                    1.30       1.30
  clock network delay (ideal)              0.00       1.30
  dp/mc/f_reg[3]/CK (DFF_X1)               0.00       1.30 r
  library setup time                      -0.05       1.25
  data required time                                  1.25
  -----------------------------------------------------------
  data required time                                  1.25
  data arrival time                                  -1.25
  -----------------------------------------------------------
  slack (MET)                                         0.00
```

e. From your simulation (and your understanding of your design), determine how many clock cycles the system takes to load one set of inputs, compute one matrix-vector product of size k=4 (that is, a 4x4 matrix), and output the result, assuming that the testbench does not stall your design (that is, the testbench ensures that input_valid and output_ready are always asserted). Count from the first cycle of loading the input until the last cycle of outputting the result. Then, multiply this by the fastest clock period you could reach to find the minimum delay of the system (in nanoseconds).

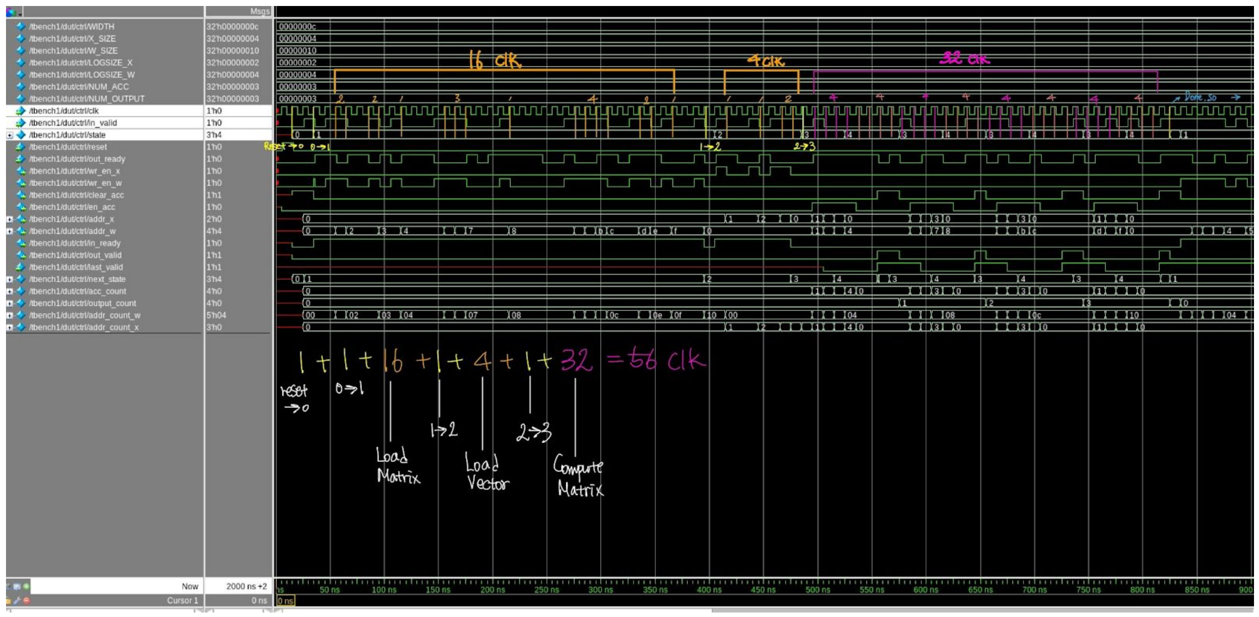
The fastest clock period we found after several synthesis was 1.3 ns. As shown below, we determined that it takes 56 clock cycles in total to load, compute and output the results without stall. Then,

1.3 ns $\times$ 56 clks = 72.8 ns

Meaning that 72.8 ns is required to process a 4 by 4 matrix vector multiplication.



1clk(reset->state0) +
1clk(state0->state1) +
16clks(state1) +
1clk(state1->state2) +
 4clks(state2) +
1clk(state2->state3) +
(4clks(state3) + 4clks(state4)) $\times$ 4(iterations)
= **56 clocks in total**


/tbench1/dut/ctrl/WIDTH
/tbench1/dut/ctrl/X_SIZE
/tbench1/dut/ctrl/W_SIZE
/tbench1/dut/ctrl/LOGSIZE_X
/tbench1/dut/ctrl/LOGSIZE_W
/tbench1/dut/ctrl/NUM_ACC
/tbench1/dut/ctrl/NUM_OUTPUT
/tbench1/dut/ctrl/clk
/tbench1/dut/ctrl/in_valid
/tbench1/dut/ctrl/state
/tbench1/dut/ctrl/reset
/tbench1/dut/ctrl/out_ready
/tbench1/dut/ctrl/wr_en_x
/tbench1/dut/ctrl/wr_en_w
/tbench1/dut/ctrl/clear_acc
/tbench1/dut/ctrl/en_acc
/tbench1/dut/ctrl/addr_x
/tbench1/dut/ctrl/addr_w
/tbench1/dut/ctrl/in_ready
/tbench1/dut/ctrl/out_valid
/tbench1/dut/ctrl/last_valid
/tbench1/dut/ctrl/next_state
/tbench1/dut/ctrl/acc_count
/tbench1/dut/ctrl/output_count
/tbench1/dut/ctrl/addr_count_w
/tbench1/dut/ctrl/addr_count_x
16 clk
4 clk
32 clk
Reset→0 0→1
1→2
2→3
Done, 50
1 + 1 + 16 + 1 + 4 + 1 + 32 = 56 clk
reset →0
0→1
Load Matrix
1→2
Load Vector
2→3
Compute Matrix
Now 2000 ns
Cursor 1 0 ns

As shown in the annotated waveform above, manually counting clock cycles by hands also results in 56 clocks.

f. A joint metric that combines the effects of area and speed in a single value is the area-delay product. The area-delay product is found by multiplying the area of the system times its delay. (Since these are both metrics that we want to minimize, lower area-delay products are better than higher ones.) Calculate the area-delay product of your system.

Total Cell Area is 3469.171952 µm².

The delay from the critical path reported is 1.25 ns. Then,

$$3469.171952\ \mu m^2 \times 1.25\ ns = 4.336 \times 10^{-18} m^2 s$$

would be the area-delay product of our system.

g. Based on the synthesis power estimate, how much energy is consumed by your system while computing one matrix-vector product (using the delay you found in part e)?

For 1ns the total power required is **1.4956 mW**.

The time required to process one 4by4 matrix-vector product is 67.2ns. Then,

$$1.49\ mW \times\ 72.8\ ns = 1.08472 \times\ 10^{-10}\ joules$$

h. We can define the arithmetic operation count as the number of useful additions and multiplications required to perform one matrix-vector product. What is the operation count for your system? How much energy does your system consume per arithmetic operation?

4 by 4 matrix vector multiplication needs 28 arithmetic operations. joules is consumed to do one 4by4 matrix-vector multiplication. Dividing it by 28 arithmetic operations can be calculated as following:

$$\frac{1.08472 \times\ 10^{-10}\text{joules}}{28\ arithemetic\ operations} = 3.874 \times 10^{-12}\text{joules/arithmetic operation}$$

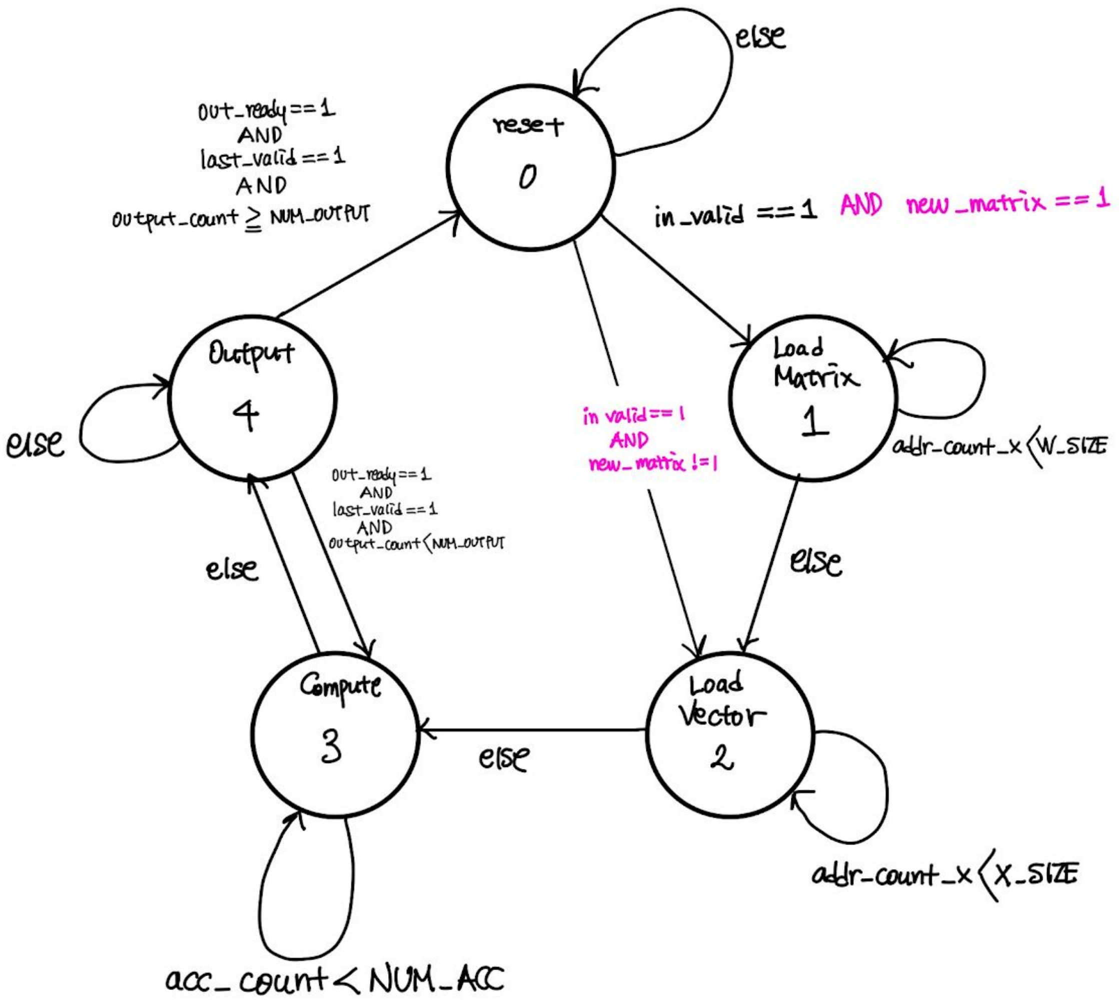
PART 2 Questions and Answers

a. What changes did you make to your prior design to complete this task? How did your control module change?

- Got rid of parameter WIDTH = 12; i nmvm4_part1 module and changed

signed [WIDTH-1:0]   input_data;
output signed [WIDTH*2-1:0] output_data;
to
input signed [11:0]   input_data;
output signed [23:0]   output_data;

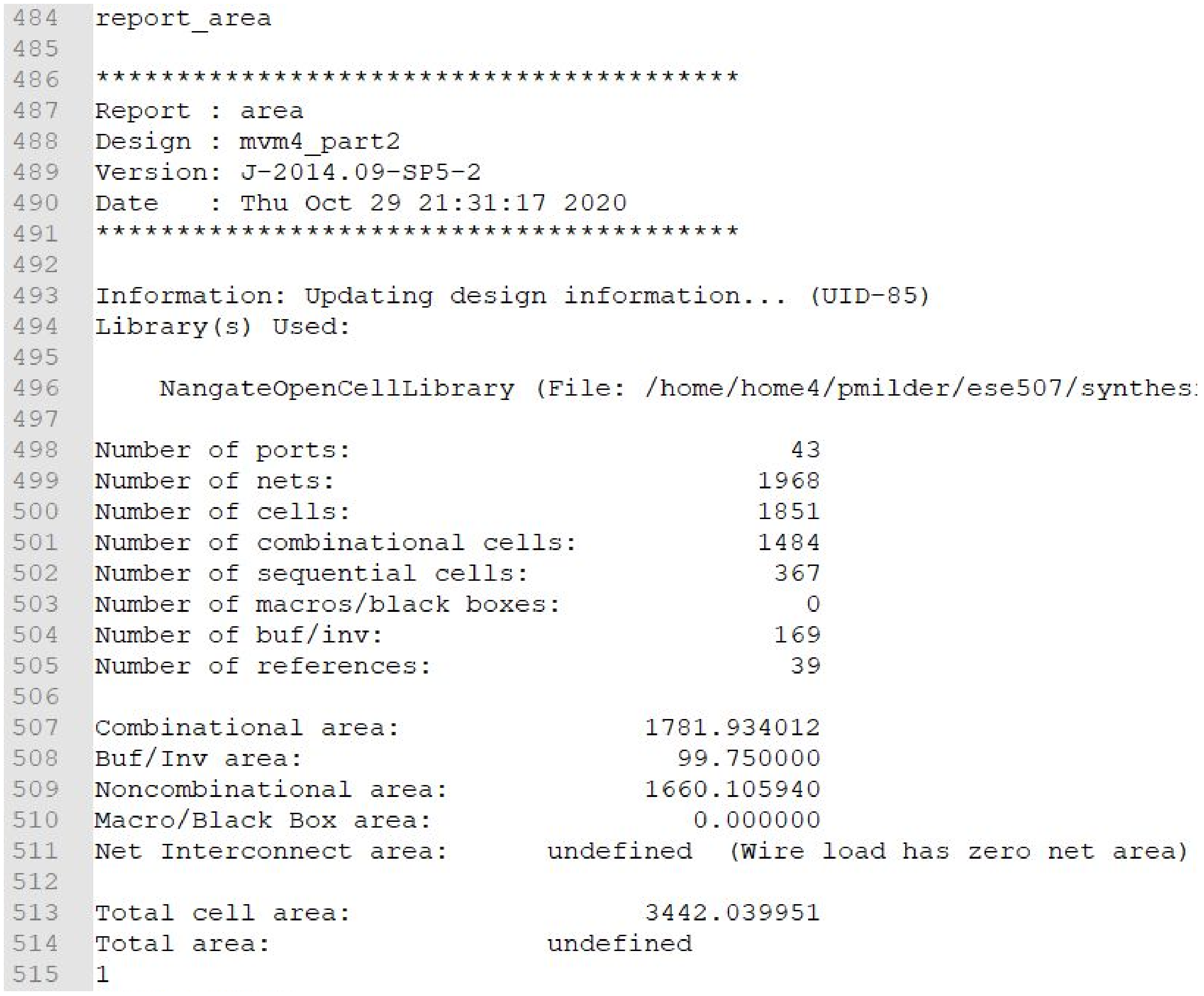
- Added .new_matrix(new_matrix) to the inputs of control instantiation.
- Added an input port for new_matrix in control module.
- Changed addr_count_x from 3 bit binary to 4 bit binary in Vector Addressor
- Changed state 0 to go to state 1 if in valid == 1 AND new_matrix == 1(if new matrix needs to be loaded we go to state 1)
- Made it go from state 0 from state 2 if in_valid == 1 and new_matrix !=0. (if not, it goes to state 2 to load new vector values) The FSM for part 2 is shown as following:



else
reset 0
out_ready == 1 AND last_valid == 1 AND output_count ≥ NUM_OUTPUT
in_valid == 1 AND new_matrix == 1
Output 4
else
Load Matrix 1
addr_count_x < W_SIZE
in valid == 1 AND new_matrix != 1
out_ready == 1 AND last_valid == 1 AND output_count < NUM_OUTPUT
else
else
Compute 3
else
Load Vector 2
addr_count_x < X_SIZE
acc_count < NUM_ACC

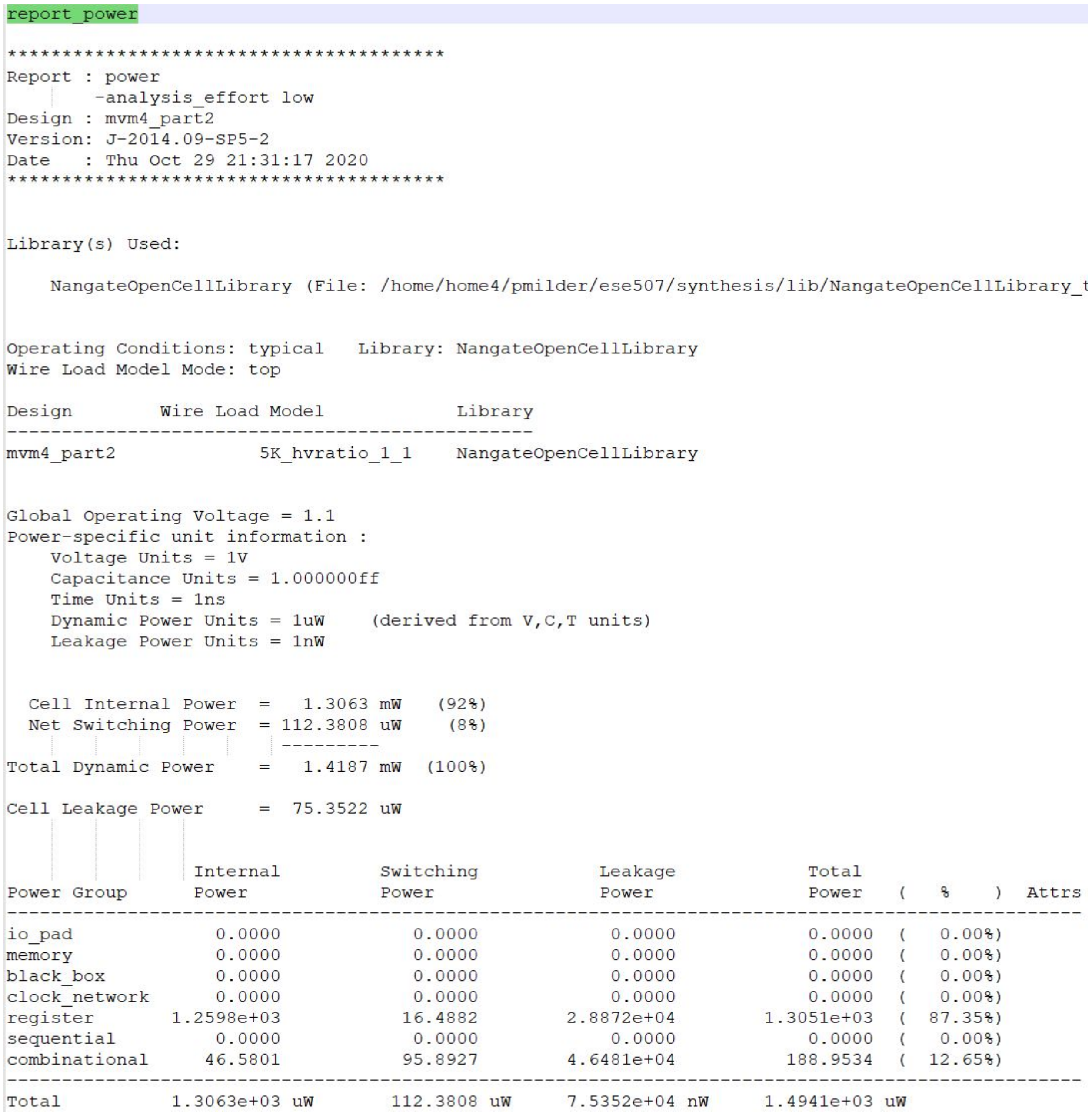
b. Report the area, power, frequency, and critical path location you determined from your synthesis report. Explain the critical path location descriptively; in other words, explain where the path flows through your design and why it makes sense that it is the critical path.

For Part 2, Area of Combinational (logic gates) cells is 1781.934012 µm²
Non-combinational (flip flops and registers) is 1660.105940 µm²
Total Cell Area is **3442.039951 µm²**



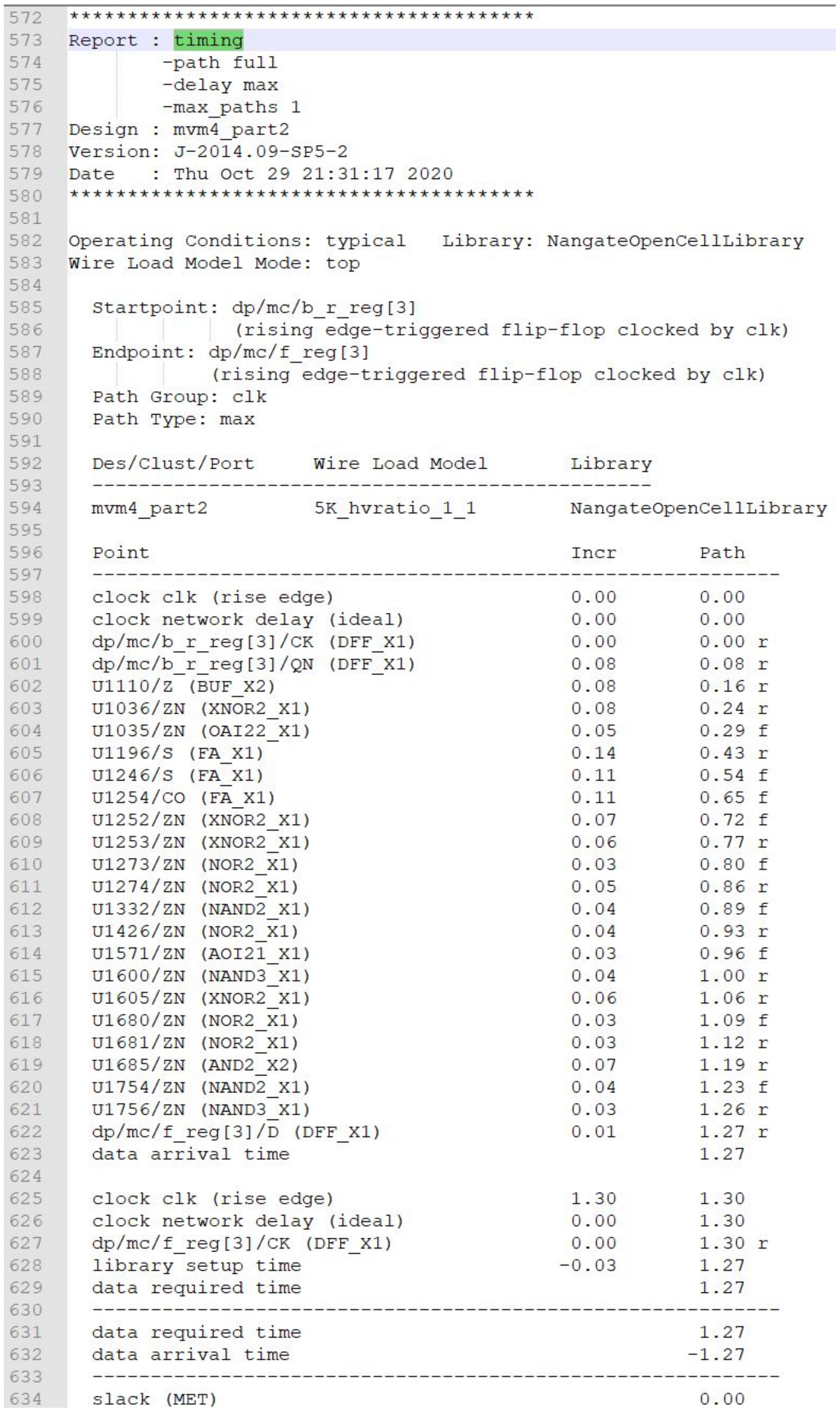
```
report_area

****************************************
Report : area
Design : mvm4_part2
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:31:17 2020
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    NangateOpenCellLibrary (File: /home/home4/pmilder/ese507/synthes

Number of ports:                           43
Number of nets:                          1968
Number of cells:                         1851
Number of combinational cells:           1484
Number of sequential cells:               367
Number of macros/black boxes:               0
Number of buf/inv:                        169
Number of references:                      39

Combinational area:               1781.934012
Buf/Inv area:                       99.750000
Noncombinational area:            1660.105940
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  3442.039951
Total area:                 undefined
1
```

For Part 2, the **Total Dynamic power is 1.4187 mW.**
Total Power is 1.4941 mW
1.4187 mW (Total Dynamic power) + 75.3522 µW (Cell Leakage Power) = **1.4941 mW (Total Power)**



```
report_power

****************************************
Report : power
        -analysis_effort low
Design : mvm4_part2
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:31:17 2020
****************************************


Library(s) Used:

    NangateOpenCellLibrary (File: /home/home4/pmilder/ese507/synthesis/lib/NangateOpenCellLibrary_t


Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
mvm4_part2             5K_hvratio_1_1    NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =   1.3063 mW   (92%)
  Net Switching Power  = 112.3808 uW    (8%)
                         ---------
Total Dynamic Power    =   1.4187 mW  (100%)

Cell Leakage Power     =  75.3522 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000            0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000            0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000            0.0000            0.0000            0.0000  (   0.00%)
clock_network      0.0000            0.0000            0.0000            0.0000  (   0.00%)
register       1.2598e+03           16.4882        2.8872e+04        1.3051e+03  (  87.35%)
sequential         0.0000            0.0000            0.0000            0.0000  (   0.00%)
combinational     46.5801           95.8927        4.6481e+04          188.9534  (  12.65%)
--------------------------------------------------------------------------------------------------
Total          1.3063e+03 uW      112.3808 uW     7.5352e+04 nW     1.4941e+03 uW
```

For part 2, our requested clock period was 1.3 ns, which is **769 MHz in frequency.**



```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : mvm4_part2
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:31:17 2020
****************************************

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: dp/mc/b_r_reg[3]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: dp/mc/f_reg[3]
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  mvm4_part2         5K_hvratio_1_1        NangateOpenCellLibrary

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  dp/mc/b_r_reg[3]/CK (DFF_X1)             0.00       0.00 r
  dp/mc/b_r_reg[3]/QN (DFF_X1)             0.08       0.08 r
  U1110/Z (BUF_X2)                         0.08       0.16 r
  U1036/ZN (XNOR2_X1)                      0.08       0.24 r
  U1035/ZN (OAI22_X1)                      0.05       0.29 f
  U1196/S (FA_X1)                          0.14       0.43 r
  U1246/S (FA_X1)                          0.11       0.54 f
  U1254/CO (FA_X1)                         0.11       0.65 f
  U1252/ZN (XNOR2_X1)                      0.07       0.72 f
  U1253/ZN (XNOR2_X1)                      0.06       0.77 r
  U1273/ZN (NOR2_X1)                       0.03       0.80 f
  U1274/ZN (NOR2_X1)                       0.05       0.86 r
  U1332/ZN (NAND2_X1)                      0.04       0.89 f
  U1426/ZN (NOR2_X1)                       0.04       0.93 r
  U1571/ZN (AOI21_X1)                      0.03       0.96 f
  U1600/ZN (NAND3_X1)                      0.04       1.00 r
  U1605/ZN (XNOR2_X1)                      0.06       1.06 r
  U1680/ZN (NOR2_X1)                       0.03       1.09 f
  U1681/ZN (NOR2_X1)                       0.03       1.12 r
  U1685/ZN (AND2_X2)                       0.07       1.19 r
  U1754/ZN (NAND2_X1)                      0.04       1.23 f
  U1756/ZN (NAND3_X1)                      0.03       1.26 r
  dp/mc/f_reg[3]/D (DFF_X1)                0.01       1.27 r
  data arrival time                                   1.27

  clock clk (rise edge)                    1.30       1.30
  clock network delay (ideal)              0.00       1.30
  dp/mc/f_reg[3]/CK (DFF_X1)               0.00       1.30 r
  library setup time                      -0.03       1.27
  data required time                                  1.27
  -----------------------------------------------------------
  data required time                                  1.27
  data arrival time                                  -1.27
  -----------------------------------------------------------
  slack (MET)                                         0.00
```

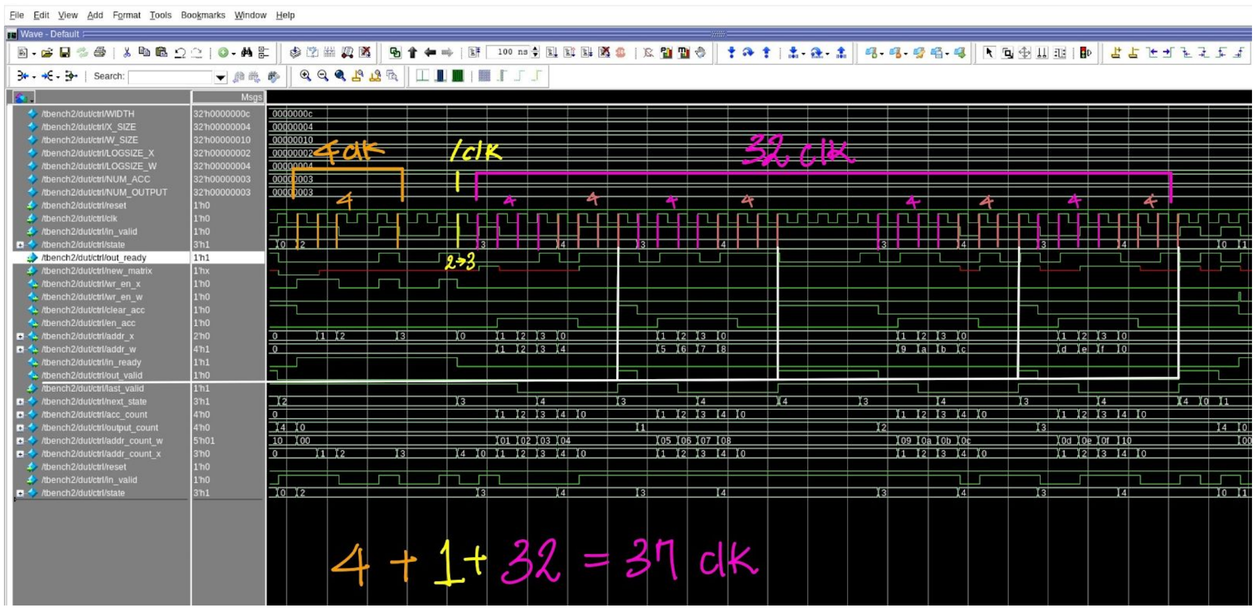
The critical path again is in the MAC unit going from input register b, through the multiplier and adder, finally to the output register f_reg.

c. Did you observe any meaningful difference in the critical path (relative to Part 1)? If yes, explain why the change makes sense. If there was not a meaningful change, explain why the modification you made for Part 2 shouldn't change the critical path.

Critical path reported for Part 2 is again from input register (b_r_reg) to output register (f_reg). It should not change the critical path because our modification added to the FSM to account for the "new matrix" situation and the MAC unit was the same as before. The longest path is still from the input registers of MAC, through the multiplier and adder, finally to the output register. Unless we add some more complicated combinatorial logic that has longer delay, the MAC unit will still contain the critical path of the design.

d. In Part 1 (Question 4.e), you computed the delay of your system, including the time it takes to load a matrix. Now, your system can operate in two modes: one where it must load a matrix and one where it uses the old matrix. Determine the delay of your system when it does not need to load a new matrix (again assuming the testbench does not stall the system). Count from the first cycle of loading the input vector until the last cycle of outputting the result. Compare the result with the delay you found for your Part 1 design.

To do process the mode without loading a new matrix, 4 clocks for loading a vector, 1 clock to transit from state 2 to 3 and 32 clocks to do a matrix-multiplication are needed. So, 37 clocks in total are needed assuming no stall(once out_valid = 1, it is done for state 4) as expected as shown in the calculation and the annotated waveform below:
4 clks (state 2 to load a vector) +
1clks (state 2 -> state 3) +
(4clks(state3) + 4clks(state4)) × 4(iterations)
= 37 clocks in total
So, the the dealy taken when it does not load a new matrix is then,
1.3 ns × 37 = 48.1ns



```
/tbench2/dut/ctrl/WIDTH          32'h0000000c
/tbench2/dut/ctrl/X_SIZE         32'h00000004
/tbench2/dut/ctrl/W_SIZE         32'h00000010
/tbench2/dut/ctrl/LOGSIZE_X      32'h00000002
/tbench2/dut/ctrl/LOGSIZE_W      32'h00000004
/tbench2/dut/ctrl/NUM_ACC        32'h00000003
/tbench2/dut/ctrl/NUM_OUTPUT     32'h00000003
/tbench2/dut/ctrl/reset          1'h0
/tbench2/dut/ctrl/clk            1'h0
/tbench2/dut/ctrl/in_valid       1'h0
/tbench2/dut/ctrl/state          3'h1
/tbench2/dut/ctrl/out_ready      1'h1
/tbench2/dut/ctrl/new_matrix     1'hx
/tbench2/dut/ctrl/wr_en_x        1'h0
/tbench2/dut/ctrl/wr_en_w        1'h0
/tbench2/dut/ctrl/clear_acc      1'h0
/tbench2/dut/ctrl/en_acc         1'h0
/tbench2/dut/ctrl/addr_x         2'h0
/tbench2/dut/ctrl/addr_w         4'h1
/tbench2/dut/ctrl/in_ready       1'h1
/tbench2/dut/ctrl/out_valid      1'h0
/tbench2/dut/ctrl/last_valid     1'h1
/tbench2/dut/ctrl/next_state     3'h1
/tbench2/dut/ctrl/acc_count      4'h0
/tbench2/dut/ctrl/output_count   4'h0
/tbench2/dut/ctrl/addr_count_w   5'h01
/tbench2/dut/ctrl/addr_count_x   3'h0
/tbench2/dut/ctrl/reset          1'h0
/tbench2/dut/ctrl/in_valid       1'h0
/tbench2/dut/ctrl/state          3'h1
```

We could compare and verify that this no new matrix loading mode requires less clocks (37 clocks) than the Part 1 mode, which always loads a new matrix, does (56 clocks). Therefore, not loading a new matrix requires less delay.

e. Compute the system's energy consumption and area-delay product as in Part 1 (Question 4.f and g), now assuming that you are not loading a new matrix. (That is, new_matrix == 0.)

Total Cell Area is 3469.171952 μm².
The delay from the critical path is 1.27 ns. Then,

$$3469.171952\ \mu m^2 \times 1.27\ ns = 4.406 \times 10^{-18} m^{2s}$$

is the area-delay product for Part2 of our system.

In Part 2, for 1 ns, the total power required is **1.4941 mW**
The time required to process a 4 by 4 matrix-vector multiplication without loading a new matrix is **48.1ns**. Then,

$$1.4941\ \text{mW} \times 48.1\text{ns} = 7.186621 \times 10^{-11}\ \text{joules}$$

PART 3 Questions and Answers

1. How did your design change? What changes did you have to make?
From Part 2, to accommodate 8 by 8 matrix vector multiplication,

- The address bits, addr_x and addr_w, had to be modified
  from
  logic [1:0] addr_x;
   logic [3:0] addr_w;
  to
    logic [2:0] addr_x;
     logic [5:0] addr_w;
- The parameters were changed
  from
   datapath #(12, 4, 16) dp (...);
  control #(12, 4, 16) ctrl (...);
  to
  datapath #(12, 8, 64) dp (...);
  control #(12, 8, 64) ctrl (...);

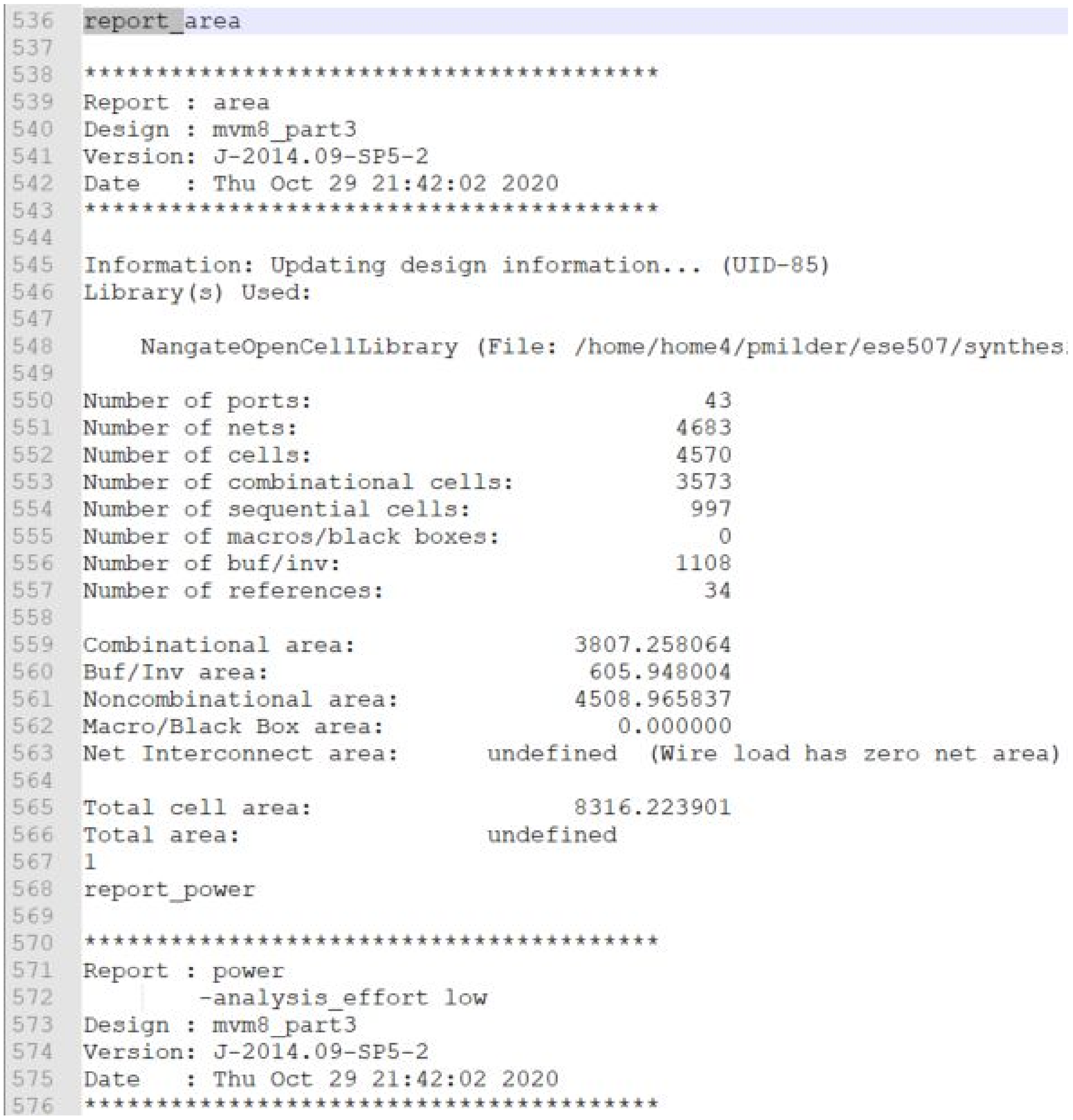
2. Are values in this system more likely to saturate than in your Part 2 design? Why or why not? What could you do to reduce the likelihood of saturation? What could you do to the system to guarantee that saturation can never happen? (Be specific)
The values in this part are more likely to saturate since we are accumulating more values for each valid result (8 instead of previous 4). One way to prevent saturation is to not take any more inputs that will cause a saturation. For example if currently the output value is 6 and saturation value is 8, only inputs <2 will be accepted. However in reality this will probably mess up the design functionality, so a more straightforward way would be to analyze the system expected input values and choose a big enough register size to prevent overflow/underflow of a value.

3. If you wanted to build a design for much larger values of k, how would you do so? Would it be easy or difficult to change? In your report, explain exactly how your design would change as k. Be specific.
Since our design specifies k as a parameter when instantiating modules, it should be fairly easy to modify our design to account for different sizes of k.
For example, to instantiate a design with k = 80, data width = 32 bits,
We could do :

```
datapath #(32, 80, 6400) dp(...)
control #(32, 80, 6400) ctrl(...)
```

4. Repeat steps 2b–2e from Part 2 on your new design. When you find the delay, compute it for both possible use-cases (new_matrix == 0 and new_matrix == 1). This will mean you will have two different measurements for delay, area-delay product, and energy consumption. Compare all of your results to the measurements from Parts 1 and 2.

For Part 3, Area of Combinational is 3807.258064 µm²
Non-combinational is 4508.965837 µm²
Total Cell Area is 8316.223901 µm²

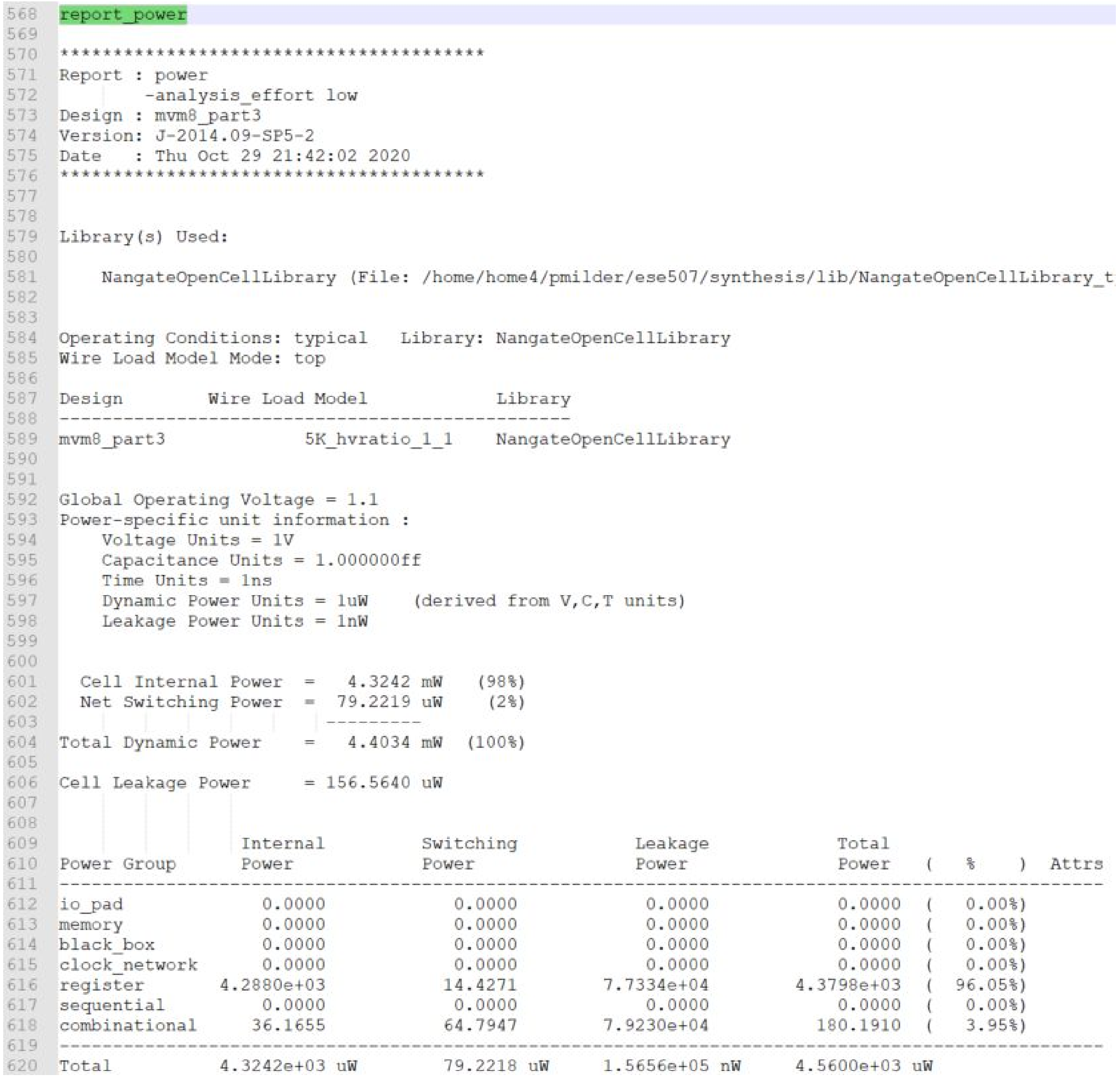


```
report_area

****************************************
Report : area
Design : mvm8_part3
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:42:02 2020
****************************************

Information: Updating design information... (UID-85)
Library(s) Used:

    NangateOpenCellLibrary (File: /home/home4/pmilder/ese507/synthes

Number of ports:                           43
Number of nets:                          4683
Number of cells:                         4570
Number of combinational cells:           3573
Number of sequential cells:               997
Number of macros/black boxes:               0
Number of buf/inv:                       1108
Number of references:                      34

Combinational area:               3807.258064
Buf/Inv area:                      605.948004
Noncombinational area:            4508.965837
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  8316.223901
Total area:                 undefined
1
report_power

****************************************
Report : power
        -analysis_effort low
Design : mvm8_part3
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:42:02 2020
****************************************
```

The **Total Dynamic power is 4.4034 mW.**

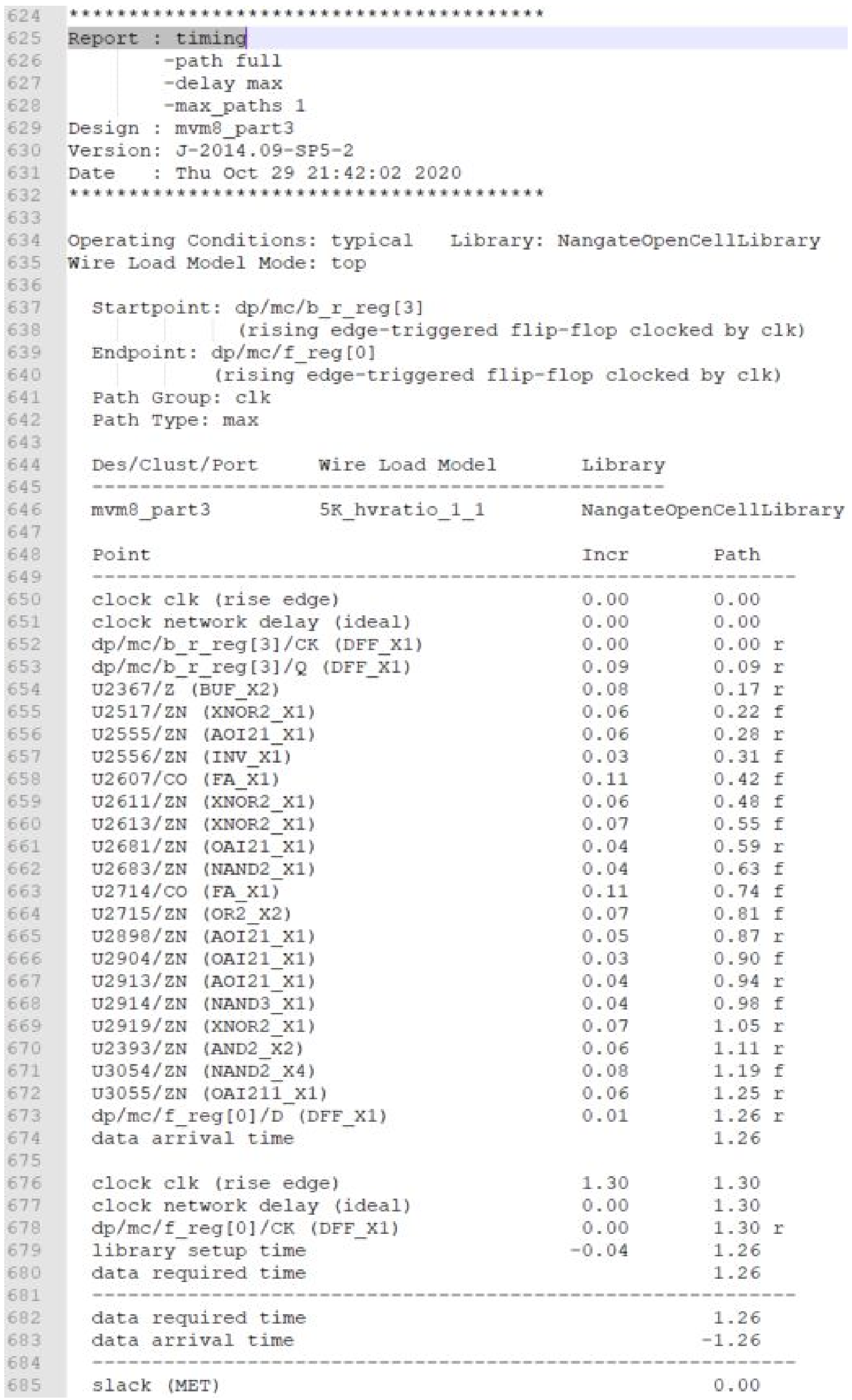
Total Power is 4.56 mW

4.4034 mW (Total Dynamic power) + 156.5640 µW (Cell Leakage Power) = **4.56 mW (Total Power)**



```
report_power

****************************************
Report : power
        -analysis_effort low
Design : mvm8_part3
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:42:02 2020
****************************************


Library(s) Used:

    NangateOpenCellLibrary (File: /home/home4/pmilder/ese507/synthesis/lib/NangateOpenCellLibrary_t


Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
mvm8_part3             5K_hvratio_1_1    NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =   4.3242 mW   (98%)
  Net Switching Power  =  79.2219 uW    (2%)
                         ---------
Total Dynamic Power    =   4.4034 mW  (100%)

Cell Leakage Power     = 156.5640 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power    (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000             0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000             0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000             0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000             0.0000             0.0000  (   0.00%)
register       4.2880e+03          14.4271         7.7334e+04         4.3798e+03  (  96.05%)
sequential         0.0000           0.0000             0.0000             0.0000  (   0.00%)
combinational     36.1655          64.7947         7.9230e+04           180.1910  (   3.95%)
--------------------------------------------------------------------------------------------------
Total          4.3242e+03 uW        79.2218 uW     1.5656e+05 nW      4.5600e+03 uW
```

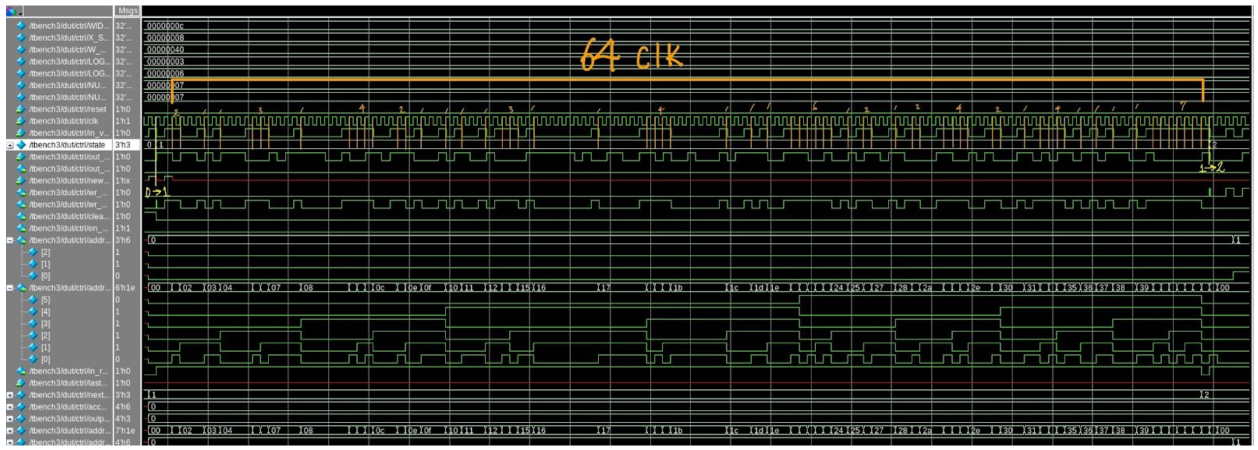
For Part 3, our requested clock period was 1.3 ns, which is 769 MHz in frequency.

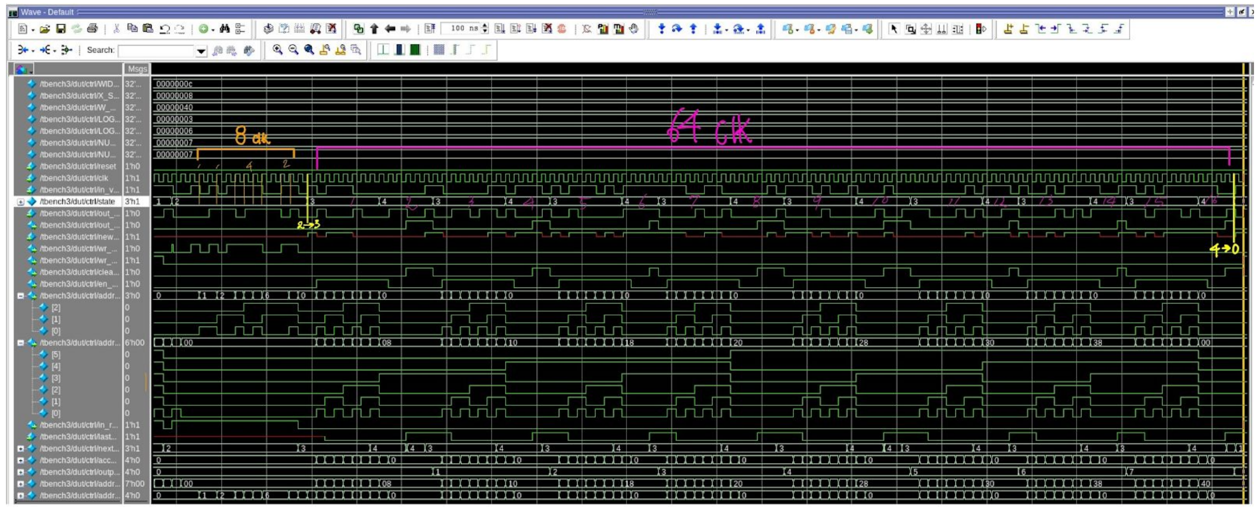


```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : mvm8_part3
Version: J-2014.09-SP5-2
Date   : Thu Oct 29 21:42:02 2020
****************************************

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: dp/mc/b_r_reg[3]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: dp/mc/f_reg[0]
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  mvm8_part3         5K_hvratio_1_1        NangateOpenCellLibrary

  Point                                    Incr       Path
  ---------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  dp/mc/b_r_reg[3]/CK (DFF_X1)             0.00       0.00 r
  dp/mc/b_r_reg[3]/Q (DFF_X1)              0.09       0.09 r
  U2367/Z (BUF_X2)                         0.08       0.17 r
  U2517/ZN (XNOR2_X1)                      0.06       0.22 f
  U2555/ZN (AOI21_X1)                      0.06       0.28 r
  U2556/ZN (INV_X1)                        0.03       0.31 f
  U2607/CO (FA_X1)                         0.11       0.42 f
  U2611/ZN (XNOR2_X1)                      0.06       0.48 f
  U2613/ZN (XNOR2_X1)                      0.07       0.55 f
  U2681/ZN (OAI21_X1)                      0.04       0.59 r
  U2683/ZN (NAND2_X1)                      0.04       0.63 f
  U2714/CO (FA_X1)                         0.11       0.74 f
  U2715/ZN (OR2_X2)                        0.07       0.81 f
  U2898/ZN (AOI21_X1)                      0.05       0.87 r
  U2904/ZN (OAI21_X1)                      0.03       0.90 f
  U2913/ZN (AOI21_X1)                      0.04       0.94 r
  U2914/ZN (NAND3_X1)                      0.04       0.98 f
  U2919/ZN (XNOR2_X1)                      0.07       1.05 r
  U2393/ZN (AND2_X2)                       0.06       1.11 r
  U3054/ZN (NAND2_X4)                      0.08       1.19 f
  U3055/ZN (OAI211_X1)                     0.06       1.25 r
  dp/mc/f_reg[0]/D (DFF_X1)                0.01       1.26 r
  data arrival time                                   1.26

  clock clk (rise edge)                    1.30       1.30
  clock network delay (ideal)              0.00       1.30
  dp/mc/f_reg[0]/CK (DFF_X1)               0.00       1.30 r
  library setup time                      -0.04       1.26
  data required time                                  1.26
  ---------------------------------------------------------
  data required time                                  1.26
  data arrival time                                  -1.26
  ---------------------------------------------------------
  slack (MET)                                         0.00
```

The critical path for Part 3 again starts from input register (b_r_reg) to output register (f_reg). The reason is because the MAC unit has been the same as before and not necessary to be modified to accomplish 8by8 matrix vector multiplication, meaning that the longest path goes through in the order of the input registers of MAC, the multiplier, the adder and the output register.

To carry out an 8 by 8 matrix multiplication **with a new matrix loading in**, we expected that
1clk(reset->state0) +
1clk(state0->state1) +
64clks(state1) +
1clk(state1->state2) +
8clks(state2) +
1clk(state2->state3) +
(4clks(state3) + 4clks(state4)) × 8(iterations)
= **140 clocks in total**
We manually annotated and counted the clocks like following and the number total clocks counted by hand was 140 clocks as well. The delay for this case is then, 140 × 1.3 ns = **182 ns.**





To carry out an 8 by 8 matrix multiplication **without loading a new matrix**, counting from the first cycle of loading the input vector until the last cycle of outputting the result, we expected that,
8clks(state2) +
1clk(state2->state3) +
(4clks(state3) + 4clks(state4)) × 8(iterations)
= **73 clocks in total.**
We could confirm that 73 clocks need to take place to calculate an 8 by 8 vector multiplication without a new matrix. The delay without loading a new matrix is then, 73 × 1.3 ns **94.9 ns.**

For Part 3, the system's energy consumption and area-delay product assuming no loading a new matrix can be calculated as following:
Total Cell Area is 8316.223901 μm²
The delay from the critical path is **1.26 ns.** Then,

$$8316.223901\,\mu m^2 \times 1.26\,ns = 1.048 \times 10^{-17}\,m^2 s$$

is the area-delay product for Part3.

In Part 3, per 1 ns, the total power required is **4.56 mW.**
The time required to process a 8 by 8 matrix-vector multiplication with loading a new matrix is **182 ns.**
And, without loading a new matrix, it is **94.9 ns.** Then,

$$4.56\,mW \times 182\,ns = 8.2992 \times 10^{-10}\,joules$$

of energy is consumed for the mode calculating with a new 8 by 8 matrix loading in.

$$4.56\,mW \times 94.9\,ns = 4.32744 \times 10^{-10}\,joules$$

of energy is consumed for the mode calculating without a new 8 by 8 loading in.

The Collection all the results from Part 1,2 and 3 is shown in the table below.

	Part 1	**Part2**	**Part 3**
Mode	Loading a new 4 by 4 matrix	No loading a new 4 by 4 matrix	Sometimes loading a new 8 by 8 matrix
Total Cell Area	3469.171952 μm²	3442.039951 μm²	8316.223901 μm²
Total Power	1.496 mW	1.4941 mW	4.56 mW
Area-delay Product	4.336×10^{-18} m^2 s	4.406×10^{-18} m^{2}s	1.048×10^{-17} m^2 s
Energy	1.08472×10^{-10} joules	7.186621×10^{-11} joules	8.2992×10^{-10} joules, when loaded. 4.32744×10^{-10} joules, when not loaded.
Frequency	769MHz	769MHz	769MHz

- Total Cell Area, Total Power, and Area-delay product significantly increase when the size of the matrix and vector increases from 4 to 8.
- When a new matrix is not loaded, the energy consumed significantly decreases compared to when a new matrix is loaded due to the difference in their time delays that are necessary to carry out the longer or the shorter version of matrix vector multiplication.
- The frequency stays the same as the critical path stays the same as we have not modified the MAC.

PART 4 Questions and Answers

a. How did you change your design to reduce its delay? If you tried multiple things, explain what they were and whether or not they helped.

For part4 we tried multiple approaches suggested in the project description. The first thing we did was parallelizing the compute stage by adding more MAC units and memories (our final system has 8 MAC units to compute outputs at the same time). By parallelizing, we were able to eliminate the need to go back and forth between the output state and the compute state since now all 8 outputs are generated concurrently. This change increased our design area greatly as there are additional logic and parts and reduced the computation time needed to generate all outputs to 1/8 of that of Part 3. We then realized that since the system must stall during both input and output states, these states are actually consuming most of the processing time, so we decided to overlap the input and output states. By doing this, we completely got rid of the output state in the FSM and instead we load all the outputs to an output vector (since they are generated at the same time this was fairly easy to do) when they are ready, and output them one by one while we load the inputs for the next iteration. With every change the power and area of our design also increased, but we then realized that the delay of the system is constrained by the input and output ports. In order to achieve a significant boost, we need to be able to run at a might higher frequency. So our last modification was adapting a 3-stage pipelined multiplier from DesignWare into our MAC unit. By doing this, our system's new minimum period reached to about 0.8ns, which gives us a 1.25GHz frequency. However, this step increased the power of our system from about 800uW to almost 1.8mW.

b. Collect the information requested in steps 2b–2e from Part 2 for your new design and include them in your report. When you find the delay, compute it for both possible use-cases (new_matrix == 0 and new_matrix == 1). This will mean you will have two different measurements for delay, area-delay product, and energy consumption. Compare all results to your measurements from Part 3.

For the energy consumed in Part 2 of when loading a new matrix with 56 clocks

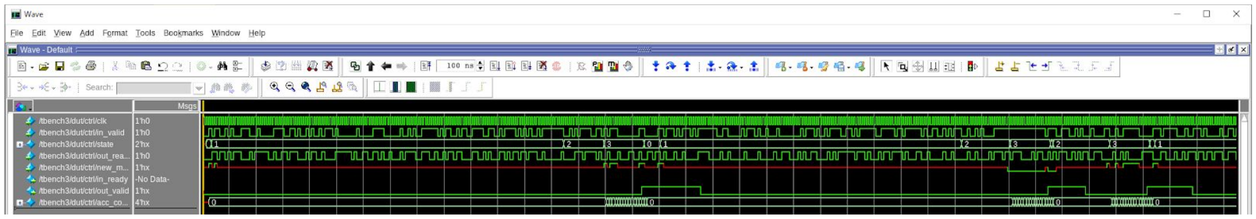
- 1.3 ns × 56 clocks = 72.8 ns
- (72.8 ns) × 1.4941 mW= **$1.0877048 \times 10^{-10}$ joules**

To calculate the area-delay product for Part 4:

- data required time reported is 0.77. Total Cell Area is 20548.7657µm²
- 20548.7657 $\mu m^2 \times 0.77$ ns = 1.581×10^{-17} m^2s

We counted the clock cycles needed for each state.

To count clocks from the waveform, we had to take several screenshots.



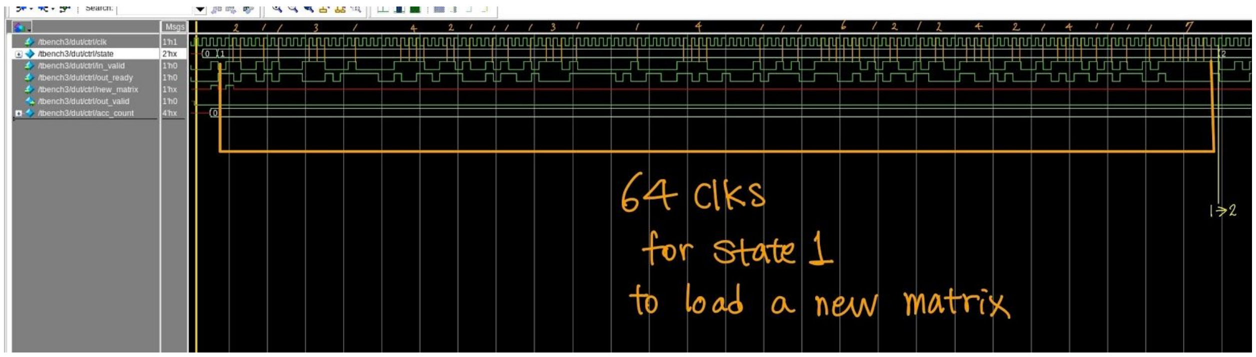
```
/tbench3/dut/ctrl/clk         1'h0
/tbench3/dut/ctrl/in_valid    1'h0
/tbench3/dut/ctrl/state       2'hx
/tbench3/dut/ctrl/out_rea...  1'h0
/tbench3/dut/ctrl/new_m...    1'hx
/tbench3/dut/ctrl/in_ready    -No Data-
/tbench3/dut/ctrl/out_valid   1'hx
/tbench3/dut/ctrl/acc_co...   4'hx
```

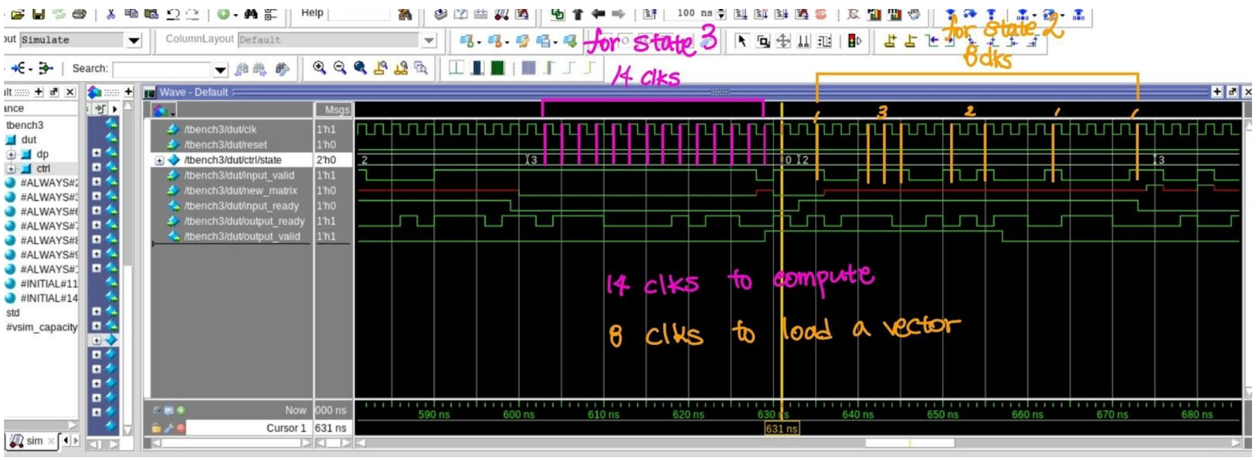
This waveform above shows the two modes:

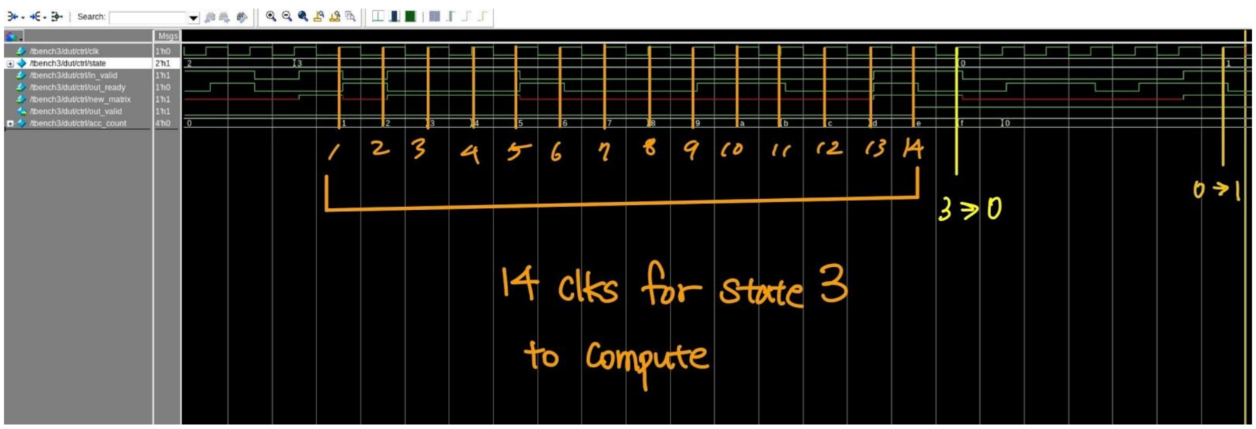
State 0->State 1->State 2->State 3->State 0 when a new matrix is loaded

or

State 0->State 2->State 3->State 0 when a new matrix is not loaded.







As shown in the zoomed-in screenshots with annotations above we could verify that
In State 1, it needs 64 clocks to load a new 8 by 8 matrix
In State 2, it need 8 clocks to load a vector
In State 3, it need 14 clocks to compute
Then,
1 clk (from state 0 to state 1) + 64 clks (state 1 to load a matrix)
+ 1clk(from state 1 to state 2) + 8 clks (state 2 to load a vector)
+ 1 clk (from state 2 to state 3) +14 clks (state 3 to compute)
= **89 clks in total**, when a new 8 by 8 matrix is loaded
The **delay** with loading a new matrix is and the energy consumed with loading a new matrix is then,
0.8 ns × 89 clks = **71.2ns**
71.2 ns × 18.63 mW = **1.326456 × 10^{-9} joules**

Also, counting from the first cycle of loading the input vector until the last cycle of outputting the result,
8 clks (state 2 to load a vector)
1 clk (from state 2 to state 3) +14 clks (state 3 to compute)
= **23 clks in total**, when a new matrix is not loaded
The **delay** without loading a new matrix is and the energy consumed without loading a new matrix is then,
0.8ns × 23 clks = **18.4 ns**
18.4 ns × 18.63 mW = 3.42792×10^{-10} joules

	Part 2		Part 3		Part 4	
Mode	No loading a new 4 by 4 matrix	Yes loading a new 4 by 4 matrix	No loading a new 8 by 8 matrix	Yes loading a new 8 by 8 matrix	No loading a new 8 by 8 matrix	Yes loading a new 8 by 8 matrix
Total Cell Area	3442.0399 μm²	3442.0399 μm²	8316.2239 μm²	8316.2239 μm²	20548.7657μm²	20548.7657μm²
Total Power	1.49 mW	1.49 mW	4.56 mW	4.56 mW	18.63 mW	18.63 mW
Area-delay Product	4.406×10^{-18} m²s	4.406×10^{-18} m²s	1.048×10^{-17} m² s	1.048×10^{-17} m² s	1.581×10^{-17} m²s	1.581×10^{-17} m²s
Energy	7.1866×10^{-11} joules	1.0877×10^{-10} joules	4.3274×10^{-10} joules	8.2992×10^{-10} joules	3.4279×10^{-10} joules	1.3264×10^{-9} joules
Frequency	769MHz	769MHz	769MHz	769MHz	1.25 GHz	1.25 GHz

- The total cell area of Part 4 is greater than those of Part 2 and Part 3.
- The total power (18.63 mW) and area-delay product (1.581×10^{-17} m2s) of Part 4 are greater than those of Part 2 and Part3.
- The energy consumed without loading a new matrix in Part 4 (3.4279×10^{-10} joules) is smaller than that of Part 3(4.3274×10^{-10} joules) but bigger than that of part 2 (7.1866×10^{-11} joules).

c. Your new design performs the same computation as your design in Part 3, but it should be faster, larger, and consume higher power. In step b you compared your new design's area-delay product and energy consumption with your Part 3 design. Based on these metrics, is your delay-optimized design better or worse than your previous design?

The Area-Delay product of Part 4 (1.581×10^{-17}m²s) is bigger than that of Part 3 (1.048×10^{-17} m²s). However, when a new matrix is not loaded, the energy consumed for Part 4 is 3.4279 × 10-10 joules and it is smaller than that of Part 3, 4.3274 × 10-10 joules. So, Part 4 is better than Part 3 when a new matrix is not loaded. In contrast, when a new matrix is loaded, Part 4 consumes more energy (1.3264×10^{-9} joules) than that of Part 3(8.2992×10^{-10} joules), so Part 4 was worse when a new matrix had to be loaded, in terms of energy consumption.

d. If instead of optimizing for delay, what if your goal was to optimize for the overall lowest energy-per-operation? How would you build a matrix-vector multiplication system to minimize energy?

One way is to run at a slower frequency, since power is affected by switching activity. In previous syntheses we noticed great variance in factors when running our design at its maximum frequency (1.3ns period) and a much slower one (5ns period). Another way is to use simpler logic while retaining the desired functionality. Reducing on-chip activity will also reduce dynamic power, which usually contributes to a great portion of the power consumption.

e. Because you are constrained by the number of input and output ports, the maximum speed (minimum delay) of your design here is limited. What could you do to make a design even faster, if you were allowed to change the input/output timing and ports? Estimate (quantitatively) how fast such a system could be.

Because the input and output ports are streaming data, the time it takes to load all data is proportional to the number of data we need to load. For the system in part4, if we could modify the input and output ports, the fastest way would be to parallelize it with multiple input and output signals. For instance, send 8 inputs at a time (either a row of matrix or the vector) indicated by one in_valid signal. Since our part4 design is parallelized, we could also modify the output port to accept all 8 outputs at the same time. This would reduce the input and output time by an order of magnitude depending on the number of signals. In this case, if we are able to transfer 8 data values each time, input would take ⅛ of the previous time, though this may vary due to the control signals *in_valid* and *out_ready*. This approach would reduce input and output latency but to run the whole system even faster, we would want to shorten the critical path.