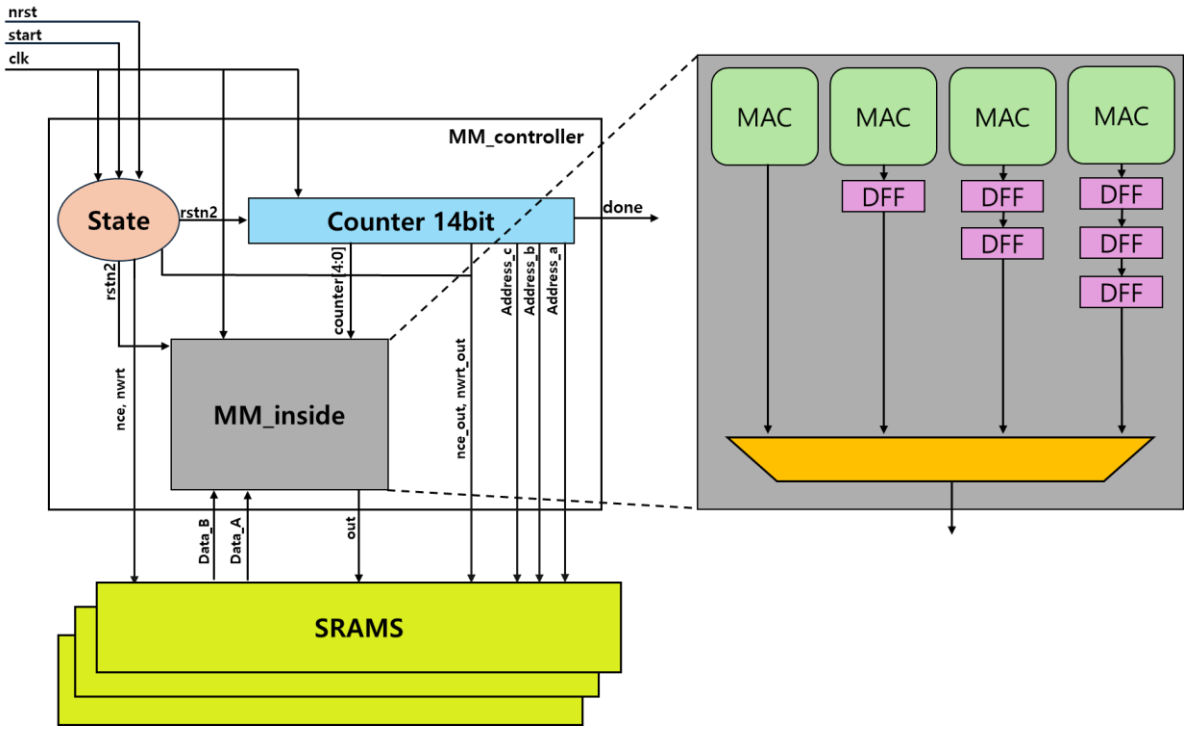
VLSI_Project01_Report

Name: 김준엽

Student ID: 2020171018

Structure Explanation

1. Block Diagram



2. Explanation

2.1. **State**: 외부의 input 신호로부터 matrix multiplication controller의 동작 여부를 결정해주는 reg형 변수이다. 이 값은 오직 nrst와 start에 의해서만 변화한다. State가 0일 때는 idle state를 나타내고, 1일 때는 running 상태임을 알려준다. 이 state에 따라서 sram들의 동작 여부를 결정하는 NCE, NWRT 값과 counter와 연산을 진행하는 MM_inside의 reset을 담당하는 rstn2의 값도 결정한다.

2.2. **Counter 14bit**: 대체로 Sram에서 읽어오거나 저장할 cell의 주소들을 결정해준다. 추가적으로, MAC 연산 후에 어떤 값을 저장할지도 결정해야 하기 때문에 MM_inside로 counter값 일부가 들어간다.

2.2.1. Address A: A의 주소는 row-wise로 저장한 32개의 연속된 값을 B의 column의 끝까지 계속 반복해서 읽어야 한다. B는 한 번에 4개의 값을 읽기 때문에 총 8번 column을 반복한다. 따라서 32개의 값을 8번씩 반복하

기 때문에 counter의 제일 아래 5bit와, 중간 값을 생략한 나머지 5bit를 합하면 된다. **A = {count[12:8] , count[4:0] }**

2.2.2. Address B: A와 B는 모두 row-wise로 저장되어 있기 때문에, 뒤 행렬인 B는 column 기준으로 읽어야 하므로 주소를 32씩 증가시켜야 한다. 하지만, B는 조건 상으로 4개의 값이 한꺼번에 한 Cell에 존재하므로 8만큼씩 주소를 증가시킨다. 또한 각 column은 모두 32개 이므로 이를 통해 cycle에서 몇 번째 값인지 알 수 있다. 만약 현재 x cycle이라면 x를 32로 나눈 값이 column의 순서일 것이고, 그 나머지가 하나의 column에서 row의 위치를 알 수 있다. 따라서, 각 column 마다 8씩 증가하므로 shift left를 3번 해주면 된다. **B = count[7:5] + (count[4:0] <<< 3)**

2.2.3. Address C: C는 A, B와 다른 점이 있다면 바로 매번 sram에 write하면 안된다는 점이다. 32개의 값을 전부 더한 뒤에 저장해야 맞는 값이기 때문에 언제 값을 저장할지 구분해야 한다. Matrix의 각 index마다 계산이 끝나는데 32 cycle이 걸리기 때문에, 바로 다음 cycle부터인 33,34,35,36이 4개의 값을 한꺼번에 저장한다. 따라서 32의 배수마다 그 다음 cycle부터 4개씩 저장하는 규칙을 가지기에 sram C의 nce와 nwrt는 counter에서 32와 4 사이의 bit 값인 count[4], count[3], count[2]가 모두 0일 때 0이 되어야 한다. 따라서 OR 연산을 통해 값을 결정하고, C의 주소는 cycle이 33일 때 주소가 0이어야 하므로 **C = ((count[12:5] – 1) <<< 2) + (count[2:0] – 1).**

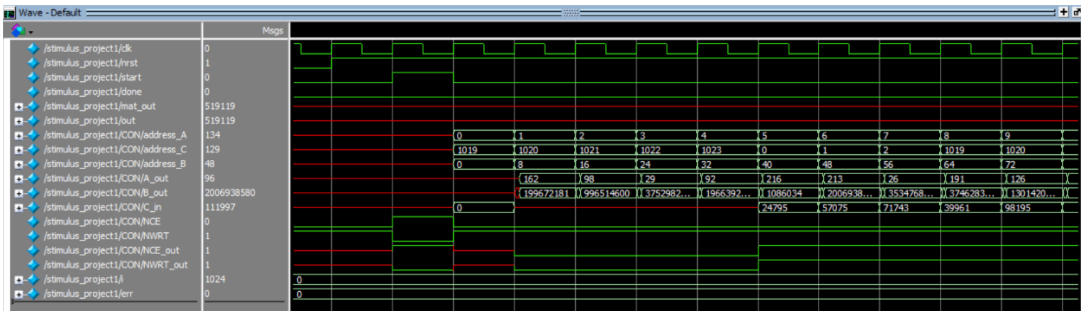
2.2.4. MM_inside: sram: 각 MAC 구조에서 연산할 때, mux를 통해 0또는 이전 단계에서 계산한 값을 결정하여 더해주어야 한다. 추가적으로, 연산 결과가 총 4개가 나타나는데 SRAM의 특성 상 한 번에 4개의 값을 저장할 수 없기에 어떤 값을 언제 저장해야 하는지 counter 값을 통해 결정한다.

2.3. **MM_inside**: 8bit x 8bit 곱셈기와 21bit 덧셈기, 그리고 이 값들을 매번 저장해주고 더해주는 register와 mux까지 합한 하나를 MAC으로 구성했다. MAC 내부에 존재하는 2 to 1 mux는 0 또는 이전 덧셈 결과 값 중 하나를 선택하는데, matrix에서 하나의 값을 계산할 때 제일 처음 0을 더해야 하기 때문에 cycle이 32의 배수 + 1마다 selection bit이 0이 되도록 하여 mux에서 0이 나타나도록 한다. 제시된 조건에 따라서, matrix multiplication 연산의 throughput이 4배가 되어야 하므로 MAC의 개수를 총 4개 사용한다. 각 연산 결과는 SRAM에 한꺼번에 저장할 수 없기에 DFF으로 나머지 3개의 결과를 다음 cycle에 연속해서 저장하도록 한다. Cycle이 32의 배수 +1, +2, +3, +4마다 각각 값을 저장하기 때문에 MM_inside 내부에 추가로 4 to 1 mux를 연결해 count[1:0]으로 output을 결정한다.

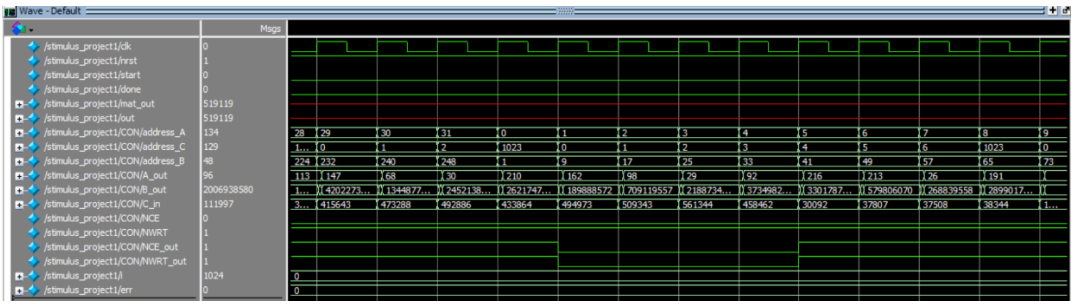
Experiment Results

1. Modelsim wave Results

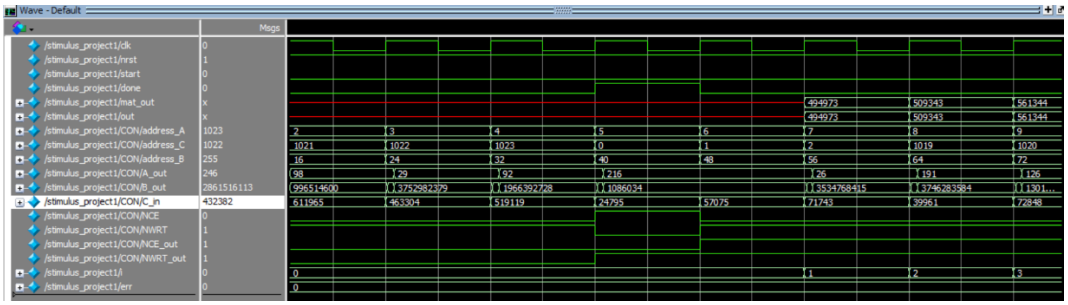
1.1. Initial wave with start signal



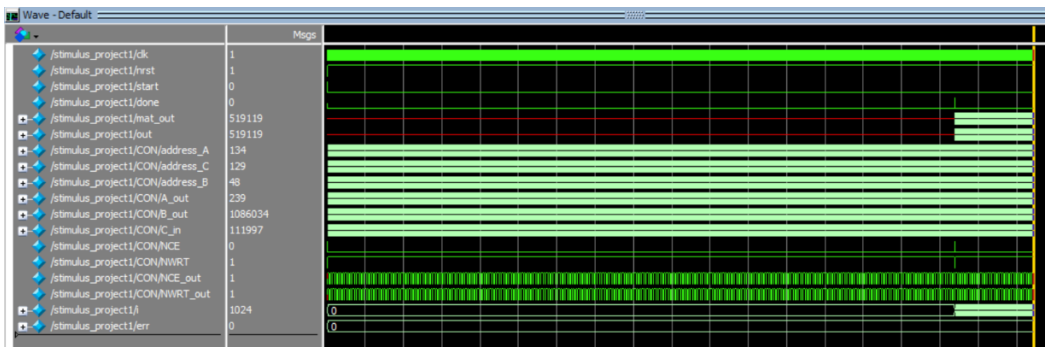
1.2. First 4 results writing into C



1.3. Done signal after calculation and writing



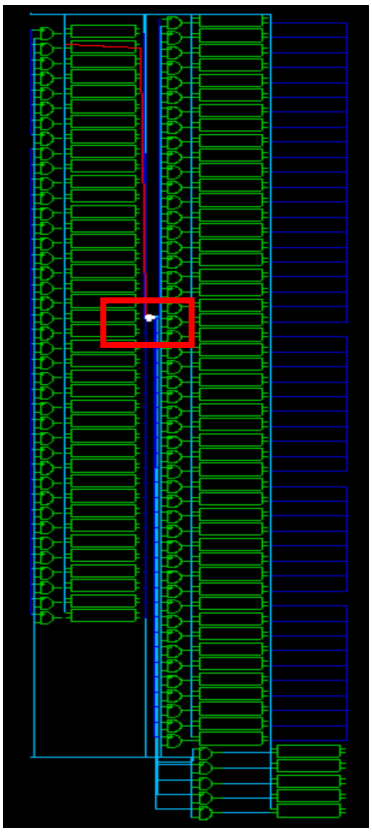
1.4. Full wave (err = 0)



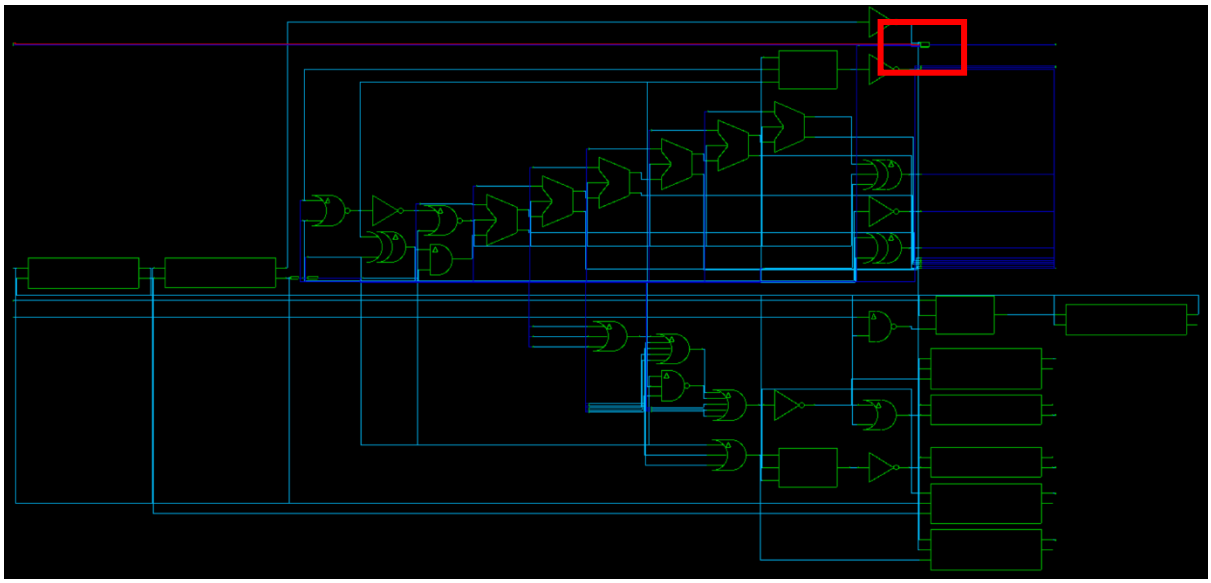
2. Synthesis Results

2.1. Schematics with critical path

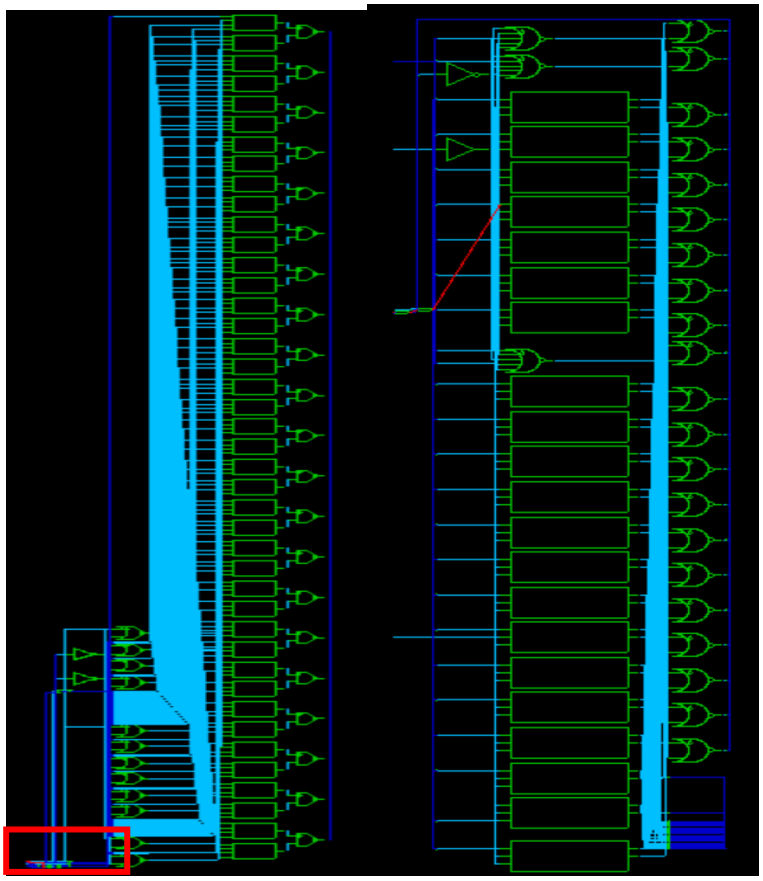
2.1.1. Full design



2.1.2. MM_controller



2.1.3. MM_inside and Inside one MAC that critical path passes



2.2. Timing Report

2.2.1. Violated when clock period is 5.9ns

```
clock clk (rise edge)                                   5.90       5.90
clock network delay (ideal)                             0.00       5.90
ctrl/MM0/M2/accumulate_reg[17]/CLK (dffcs2)             0.00       5.90 r
library setup time                                     -0.48       5.42
data required time                                                 5.42
--------------------------------------------------------------------------
data required time                                                 5.42
data arrival time                                                 -5.62
--------------------------------------------------------------------------
slack (VIOLATED)                                                  -0.19
```

2.2.2. Slack MET when clock period is 6.0ns

Clock Frequency = 166.667MHz

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : mm_controller_synthesis
Version: Z-2007.03-SP4
Date   : Mon May  5 04:03:30 2025
****************************************

Operating Conditions: nom_pvt   Library: lec25dscc25_SS
Wire Load Model Mode: top

  Startpoint: A_out_q_reg[1]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: ctrl/MM0/M2/accumulate_reg[18]
              (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                                   Incr       Path
  ------------------------------------------------------------------------
  clock clk (rise edge)                                   0.00       0.00
  clock network delay (ideal)                             0.00       0.00
  A_out_q_reg[1]/CLK (dffs2)                              0.00       0.00 r
  A_out_q_reg[1]/Q (dffs2)                                0.35       0.35 r
  ctrl/A_out[1] (mm_controller)                           0.00       0.35 r
  ctrl/MM0/A[1] (matmul_controller)                       0.00       0.35 r
  ctrl/MM0/M2/a[1] (MAC_ONE_2)                            0.00       0.35 r
  ctrl/MM0/M2/mult_12/a[1] (MAC_ONE_2_DW_mult_uns_1)      0.00       0.35 r
  ctrl/MM0/M2/mult_12/U590/Q (and2s1)                     0.38       0.73 r
  ctrl/MM0/M2/mult_12/U516/OUTS (hadd1s3)                 0.54       1.27 f
  ctrl/MM0/M2/mult_12/U139/OUTS (fadd1s3)                 0.85       2.12 r
  ctrl/MM0/M2/mult_12/U337/Q (xor2s2)                     0.30       2.41 r
  ctrl/MM0/M2/mult_12/U334/Q (xor2s3)                     0.34       2.75 r
  ctrl/MM0/M2/mult_12/U356/Q (nor2s1)                     0.19       2.94 f
  ctrl/MM0/M2/mult_12/U463/Q (oai21s2)                    0.26       3.20 r
  ctrl/MM0/M2/mult_12/U496/Q (aoi21s3)                    0.26       3.46 f
  ctrl/MM0/M2/mult_12/U302/Q (i1s1)                       0.17       3.62 r
  ctrl/MM0/M2/mult_12/U490/Q (nnd2s2)                     0.11       3.73 f
  ctrl/MM0/M2/mult_12/U400/Q (nnd2s2)                     0.13       3.86 r
  ctrl/MM0/M2/mult_12/U402/Q (xnr2s3)                     0.35       4.21 f
  ctrl/MM0/M2/mult_12/product[14] (MAC_ONE_2_DW_mult_uns_1)
                                                          0.00       4.21 f
  ctrl/MM0/M2/add_13/A[14] (MAC_ONE_2_DW01_add_1)         0.00       4.21 f
  ctrl/MM0/M2/add_13/U165/Q (nor2s3)                      0.15       4.35 r
  ctrl/MM0/M2/add_13/U164/Q (nor2s3)                      0.14       4.49 f
  ctrl/MM0/M2/add_13/U265/Q (nnd2s2)                      0.16       4.66 r
  ctrl/MM0/M2/add_13/U228/Q (i1s3)                        0.09       4.75 f
  ctrl/MM0/M2/add_13/U230/Q (nnd2s1)                      0.15       4.90 r
  ctrl/MM0/M2/add_13/U220/Q (nnd2s2)                      0.13       5.04 f
  ctrl/MM0/M2/add_13/U221/Q (nnd2s2)                      0.12       5.16 r
  ctrl/MM0/M2/add_13/U176/Q (xor2s1)                      0.36       5.52 r
  ctrl/MM0/M2/add_13/SUM[18] (MAC_ONE_2_DW01_add_1)       0.00       5.52 r
```

```
ctrl/MM0/M2/add_13/SUM[18] (MAC_ONE_2_DW01_add_1)       0.00       5.52 r
ctrl/MM0/M2/accumulate_reg[18]/CLRB (dffcs1)            0.00       5.52 r
data arrival time                                                  5.52

clock clk (rise edge)                                   6.00       6.00
clock network delay (ideal)                             0.00       6.00
ctrl/MM0/M2/accumulate_reg[18]/CLK (dffcs1)             0.00       6.00 r
library setup time                                     -0.48       5.52
data required time                                                 5.52
------------------------------------------------------------------------
data required time                                                 5.52
data arrival time                                                 -5.52
------------------------------------------------------------------------
slack (MET)                                                        0.00
```

2.3. Area Report

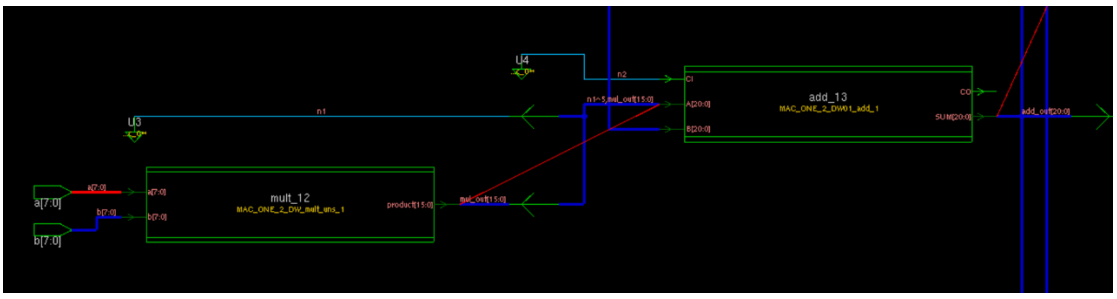
```
****************************************
Report : area
Design : mm_controller_synthesis
Version: Z-2007.03-SP4
Date   : Mon May  5 04:15:14 2025
****************************************

Library(s) Used:

    lec25dscc25_SS (File: /home/admin/lib/lec25/lec25dscc25_SS.db)

Number of ports:                97
Number of nets:                285
Number of cells:               189
Number of references:            4

Combinational area:       141884.034779
Noncombinational area:     53507.202148
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:          195391.234375
Total area:                 undefined
1
design_vision-xg-t>
```

2.4. Power Report

```
****************************************
Report : power
        -analysis_effort low
Design : mm_controller_synthesis
Version: Z-2007.03-SP4
Date   : Mon May  5 04:15:54 2025
****************************************


Library(s) Used:

    lec25dscc25_SS (File: /home/admin/lib/lec25/lec25dscc25_SS.db)


Operating Conditions: nom_pvt   Library: lec25dscc25_SS
Wire Load Model Mode: top


Global Operating Voltage = 2.25
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =  17.2215 mW   (74%)
  Net Switching Power  =   5.9327 mW   (26%)
---------
Total Dynamic Power    =  23.1542 mW  (100%)

Cell Leakage Power     = 904.3980 uW
```

Conclusion and Discussion

1. SRAM에서 두개의 값을 읽어온 다음, 한 cycle 내에 multiplication과 addition을 수행하므로, 일반적으로 생각할 때는, MAC 구조 안에서 계산하는 과정이 제일 오래 걸릴 것으로 예상된다. Schematic의 critical path를 살펴본 결과, 예상과 동일했다. 아래의 사진에서 알 수 있듯이, 하나의 MAC 내부에서 계산하는 multiplication과 addition 후, 임시로 값을 저장해 두는 register까지의 경로가 critical path임을 보여주고 있다. 추가적으로 Area를 보면 combinational 부분이 sequential 부분보다 훨씬 많은데, 이는 counter의 값을 통한 주소 연산과, matrix에서 각 요소의 곱셈과 덧셈이 이 controller의 주요 과정이기 때문인 것으로 보인다. Power 또한 Dynamic과 leakage의 차이가 크게 나타났다.



2. Counter를 설계할 때, 기존에는 13bit로 설계하려 하였으나, SRAM의 성질 및 done signal의 필요성으로 14bit로 설계했다. SRAM의 특징 상, 현재 cycle에서 counter의 값이 나타내는 주소의 값을 읽으면, 바로 다음 cycle에서 data가 전송되어 값이 계산되는 것을 몇 번의 시행착오를 통해 알게 되었다. 마찬가지로 write도 결국에는 cycle의 값이 계산이 완료되는 32의 배수가 아닌, 32의 배수 + 1부터 write를 진행해야 한다. 따라서 총 계산 횟수인 32 x 32 x 8의 과정 후에, 마지막 4개에 대한 값을 저장하고, 또 저장 후에 done signal을 보내기 위해서 14bit로 counter를 만들었다.
3. SRAM C에 값을 넣어주기 위해서 C에 해당하는 nce_out, nwrt_out을 counter에 따라 바뀌도록 결정해 주었는데, stimulus에서 C의 값을 nce와 nwrt를 이용해 읽는 것이 아니라 array로 바로 값을 읽는 것을 알게 되었다. 결국에는 C에서 read 자체를 하지 않기 때문에 nwrt_out의 값을 변화시키지 않아도 된다는 사실을 뒤늦게 알게 되었다. 이러한 nwrt_out의 신호 결정 방식을 조금 더 간소화했다면, maximum clock frequency와 area 등이 더욱 빠르고 작게 나왔을 것 같다는 아쉬움이 있다.