Project 2
Matrix Multiplication Unit

8조
조원: 김지원(2020105400)
인선우(2020105420)

Agenda

chapter 1 **Overview**

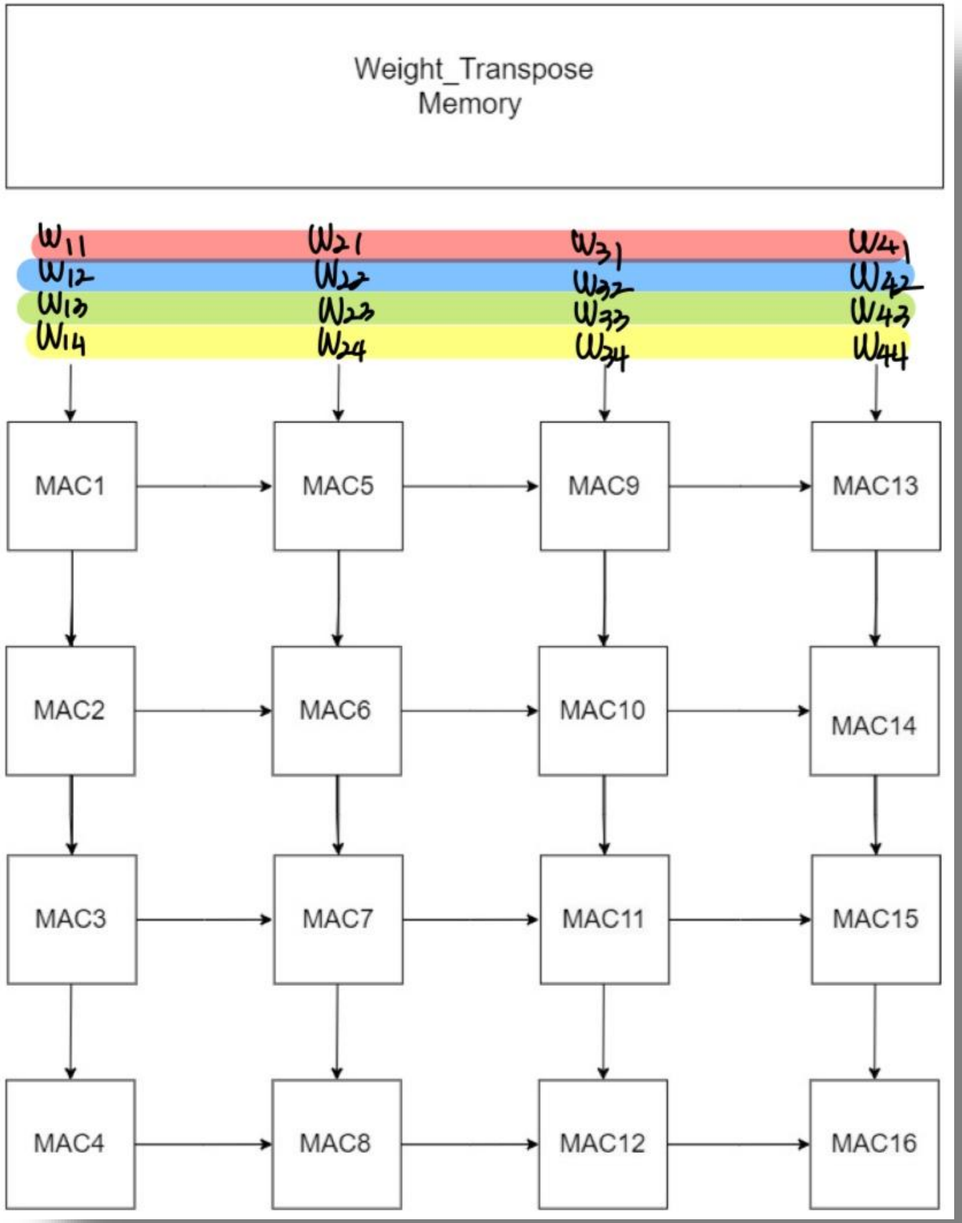
chapter 2 **State machine Design**

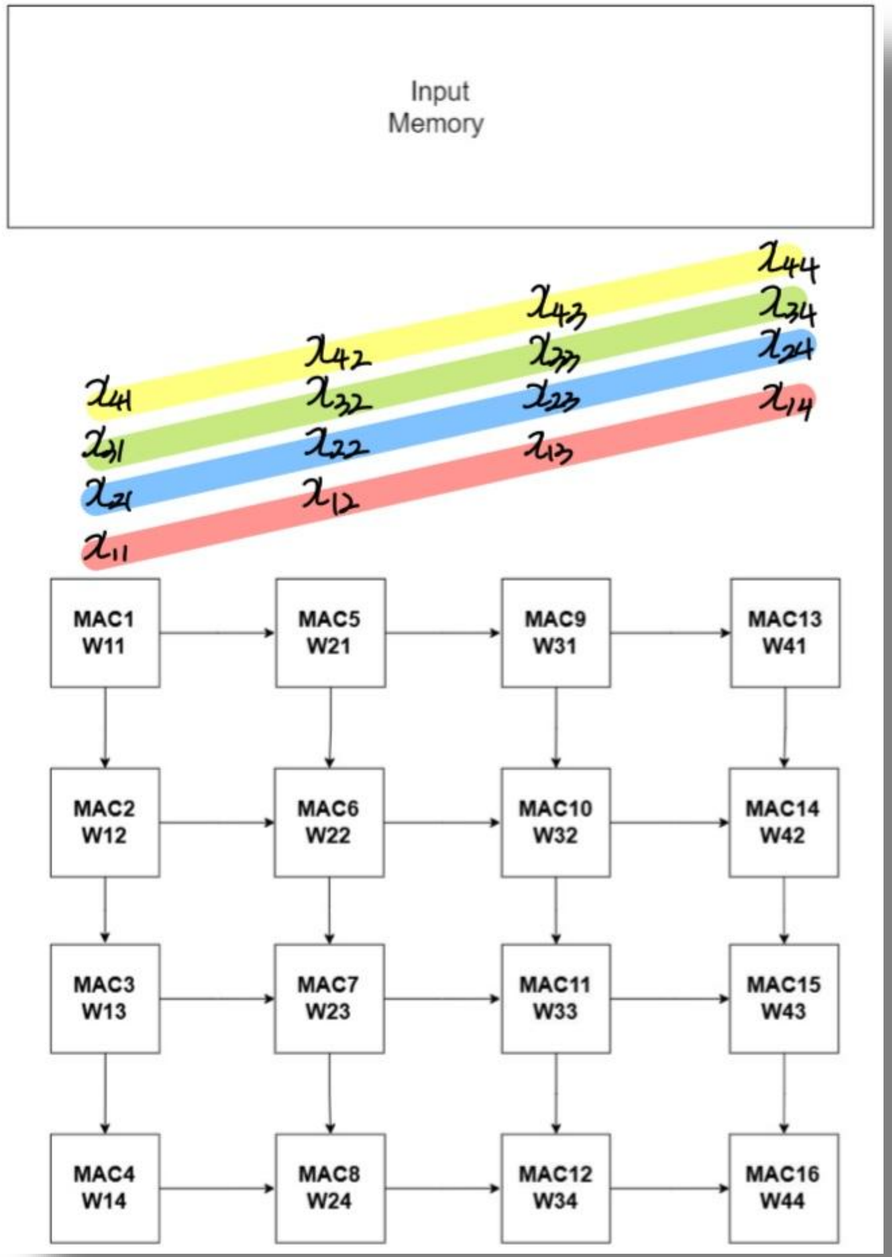
chapter 3 **Control Unit Design**

chapter 1

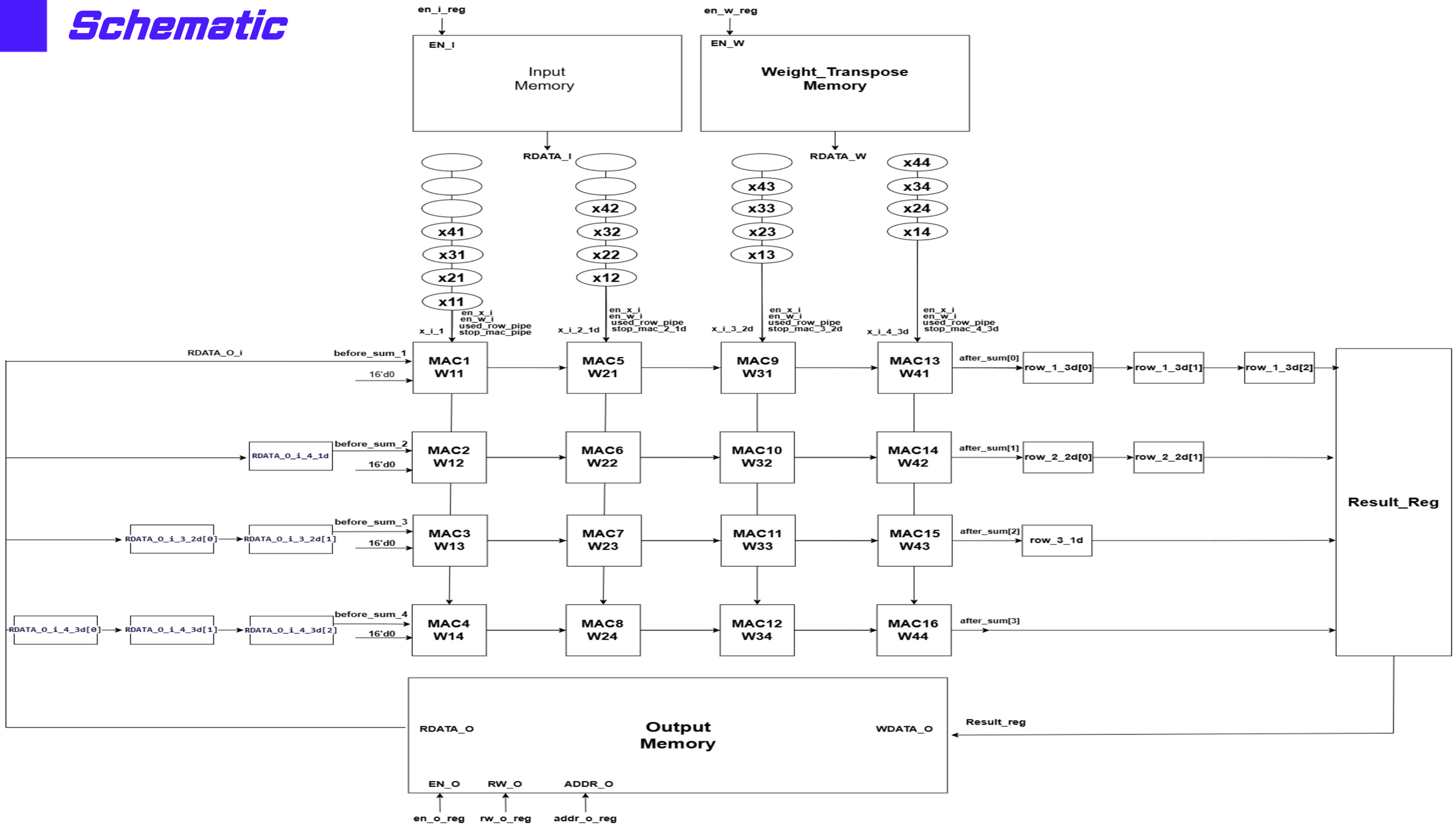
Overview

Datapath overview





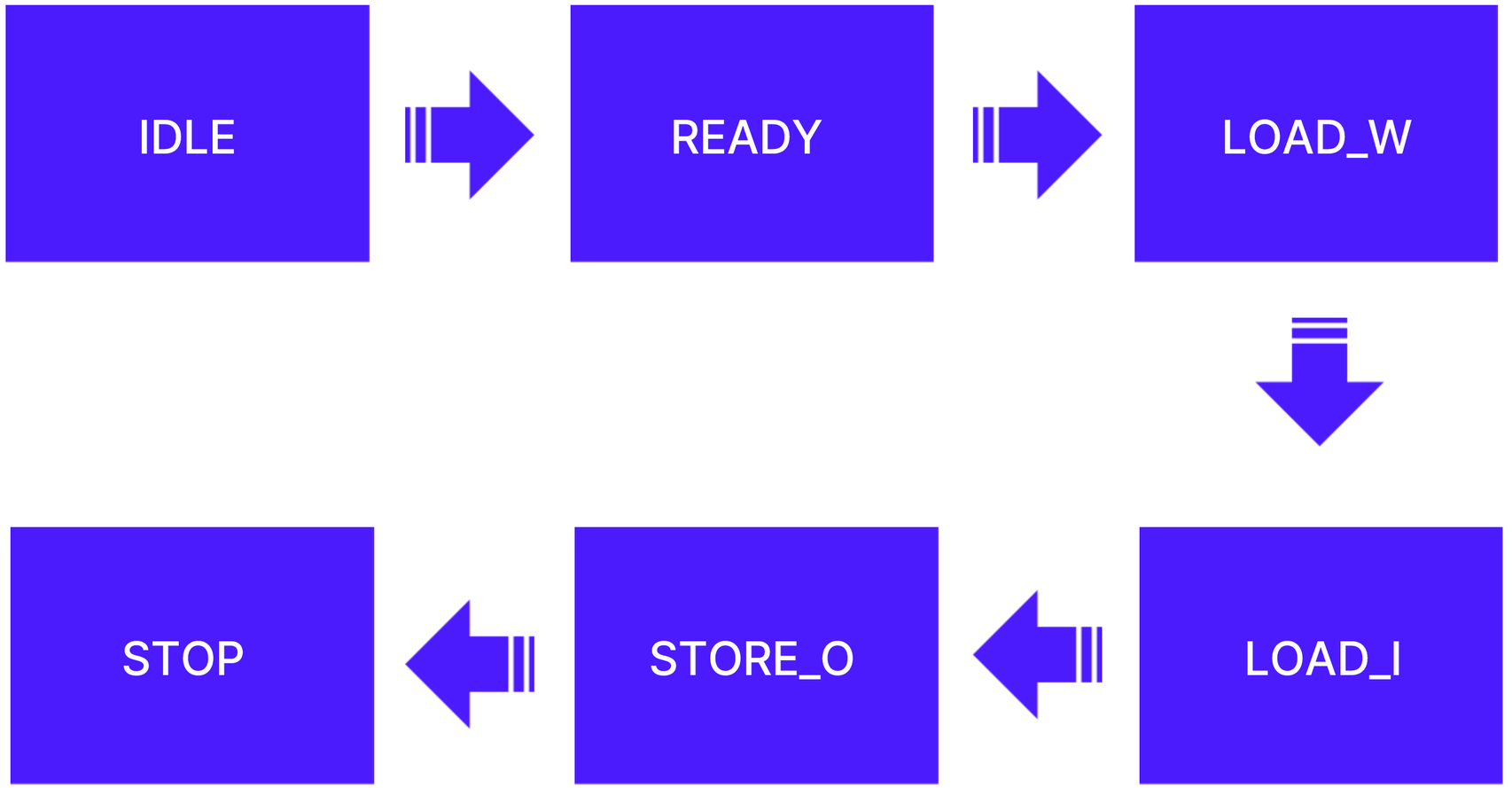
Schematic


en_i_reg
en_w_reg
EN_I
Input Memory
EN_W
Weight_Transpose Memory
RDATA_I
RDATA_W
x11
x21
x31
x41
x12
x22
x32
x42
x13
x23
x33
x43
x14
x24
x34
x44
en_x_i
en_w_i
used_row_pipe
stop_mac_pipe
x_i_1
x_i_2_1d
stop_mac_2_1d
x_i_3_2d
stop_mac_3_2d
x_i_4_3d
stop_mac_4_3d
RDATA_O_i
before_sum_1
16'd0
MAC1 W11
MAC5 W21
MAC9 W31
MAC13 W41
after_sum[0]
row_1_3d[0]
row_1_3d[1]
row_1_3d[2]
RDATA_O_i_4_1d
before_sum_2
MAC2 W12
MAC6 W22
MAC10 W32
MAC14 W42
after_sum[1]
row_2_2d[0]
row_2_2d[1]
RDATA_O_i_3_2d[0]
RDATA_O_i_3_2d[1]
before_sum_3
MAC3 W13
MAC7 W23
MAC11 W33
MAC15 W43
after_sum[2]
row_3_1d
RDATA_O_i_4_3d[0]
RDATA_O_i_4_3d[1]
RDATA_O_i_4_3d[2]
before_sum_4
MAC4 W14
MAC8 W24
MAC12 W34
MAC16 W44
after_sum[3]
Result_Reg
RDATA_O
Output Memory
WDATA_O
Result_reg
EN_O
RW_O
ADDR_O
en_o_reg
rw_o_reg
addr_o_reg

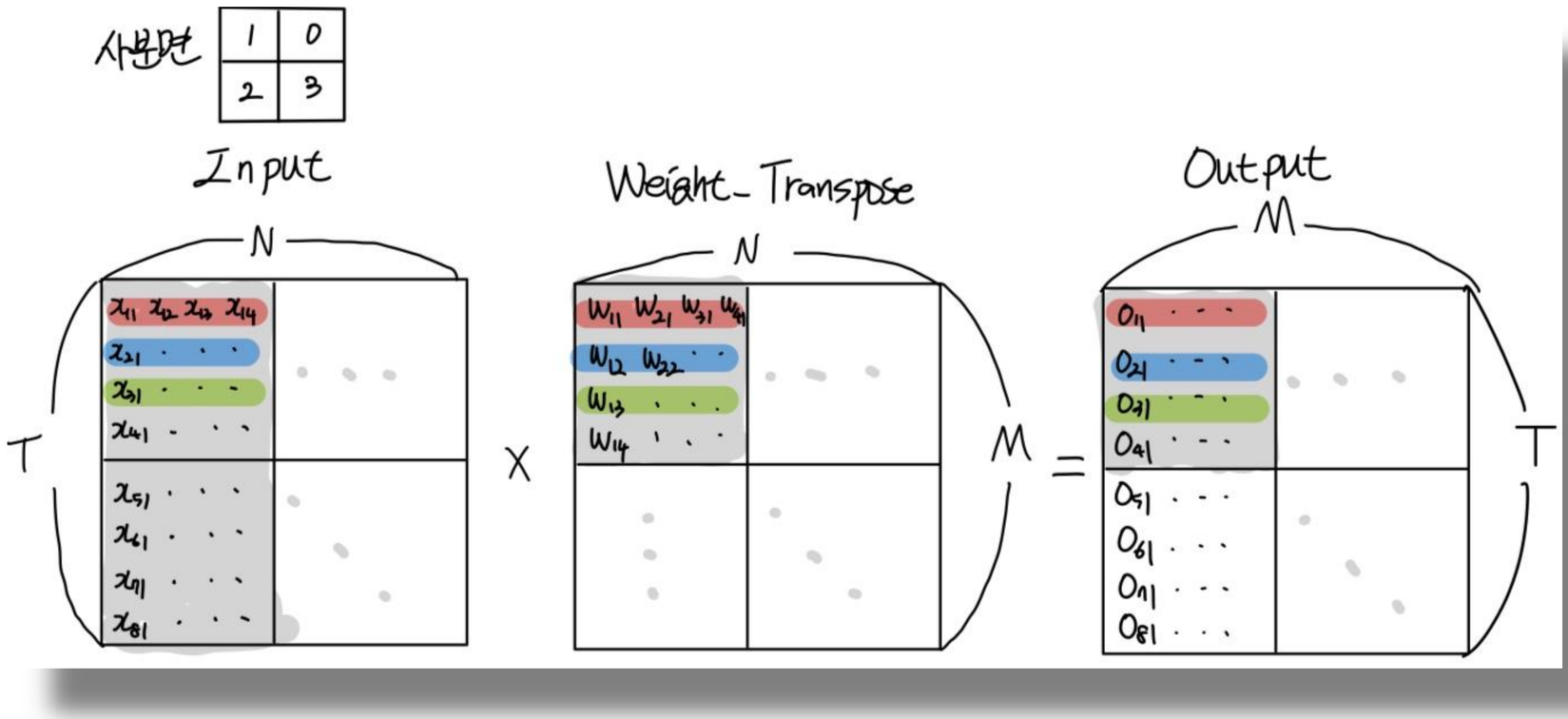
chapter 2

State machine design

State machine design



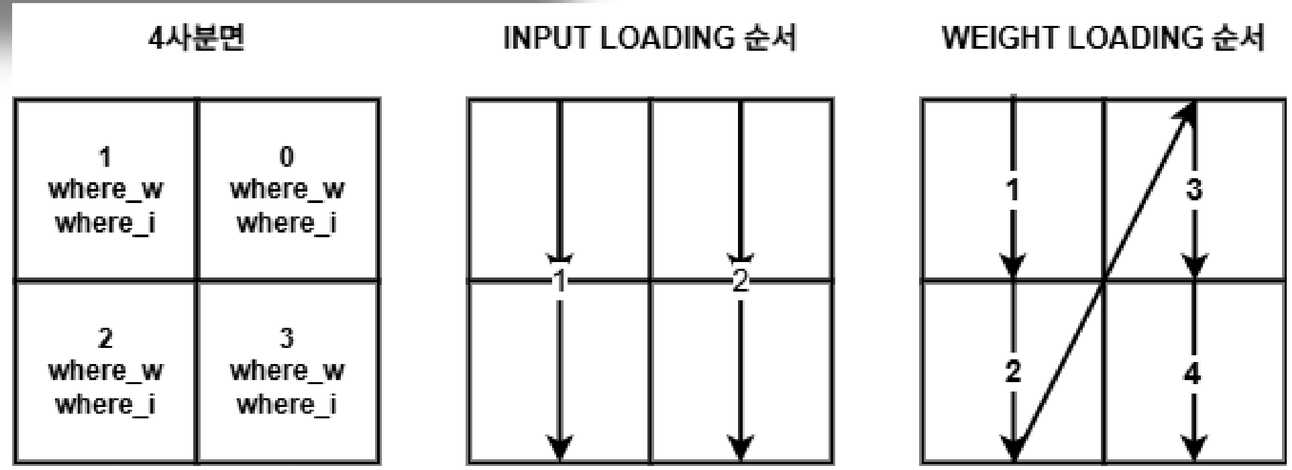
Stage (사분면)



- Input과 Weight_Transpose는 4분면을 기준으로 좌측끼리, 우측끼리만 곱해짐.
- MNT 값을 통해 stage를 구성하면 State를 stage에 맞게 결정 가능

Stage

```
wire        [2:0]   stage;
assign  stage   =   {(M>4'd4),(N>4'd4),(T>4'd4)};
/*                                                  | 사분면 (input, weight_transpoe)
  stage 0 | 3'b000 | M≤4, N≤4, T≤4일 때 1CYCLE       | (1,1)

        1 | 3'b001 | M≤4, N≤4, T>4일 때 1CYCLE       | (1,1), (2,1)

        2 | 3'b010 | M≤4, N>4, T≤4일 때 1CYCLE + LOAD_W  | (1,1), (0,0)

        3 | 3'b011 | M≤4, N>4, T>4일 때 1CYCLE + LOAD_W  | (1,1), (2,1), (1,0), (2,0)


        4 | 3'b100 | M>4, N≤4, T≤4일 때 1CYCLE + LOAD_W  | (1,1), (1,2)

        5 | 3'b101 | M>4, N≤4, T>4일 때 1CYCLE + LOAD_W  | (1,1), (2,1), (1,2), (2,2)


        6 | 3'b110 | M>4, N>4, T≤4일 때 1CYCLE + LOAD_W*3  | (1,1), (1,2), 0(,0), (0,3)

        7 | 3'b111 | M>4, N>4, T>4일 때 1CYCLE + LOAD_W*3  | (1,1), (2,1), (1,2), (2,2), (0,0), (3,0), (0,3), (3,3)
```



State (IDLE, READY, LOAD_W, LOAD_I, STOP)

```
reg     [2:0]   load_w_times, load_i_times;
reg     [3:0]   compute_cycle;
always @(*) begin
    if((N>=4)) begin
        if(where_w_reg==2'd0||where_w_reg==2'd3) begin
            compute_cycle   =   (N-4'd4) + 4'd5;
        end else begin
            compute_cycle   =   4'd9;
        end
    end else begin
        compute_cycle   =   N + 4'd5;
    end
end
always @(*) begin
    case (state)
        IDLE:       next_state  =   (START)                     ? READY     :   IDLE;
        STOP:       next_state  =   (START && ~prev_START)      ? READY     :   STOP;
        READY:      next_state  =   (START)                     ? LOAD_W    :   READY;
        LOAD_W:     next_state  =   (addr_w_reg[1:0]==2'b00)    ? LOAD_I    :   LOAD_W;
        LOAD_I:     next_state  =   (cycle_count==compute_cycle)         ? STORE_O   :   LOAD_I;
```

State(STORE_O)

- load_w_times 값을 비교하여 Cycle을 몇 번 돌았는지 확인하여 다음 state를 결정하고 where_reg에 다음 연산 시 사용할 Weight, Input 위치 저장

```
3'd7: begin
    if(store_count==T) begin
        if(load_w_times==1) begin
            next_state  =   READY;
        end else if(load_w_times==2) begin
            next_state  =   READY;
        end else if(load_w_times==3) begin
            next_state  =   READY;
        end else begin
            next_state  =   STOP;
        end
    end else begin
        next_state      =   STORE_O;
    end
end
default:    begin
    next_state  =   IDLE;
```

```
3'd7: begin
    if(store_count==T) begin
        if(load_w_times==1) begin
            where_w_reg     <=   2'd2;
            where_i_reg     <=   2'd1;
        end else if(load_w_times==2) begin
            where_w_reg     <=   2'd0;
            where_i_reg     <=   2'd0;
        end else if(load_w_times==3) begin
            where_w_reg     <=   2'd3;
            where_i_reg     <=   2'd0;
        end else begin
            where_w_reg     <=   2'd1;
            where_i_reg     <=   2'd1;
        end
    end else begin
        where_w_reg     <=   where_w_reg;
        where_i_reg     <=   where_i_reg;
    end
```

chapter 3

Control Unit Design

Key considerations

1. **MNT에 따른 Input, Weight 고려**

2. **Memory 접근 최소화**

3. **Register 최소화**

1. Control Signal (LOAD_W)

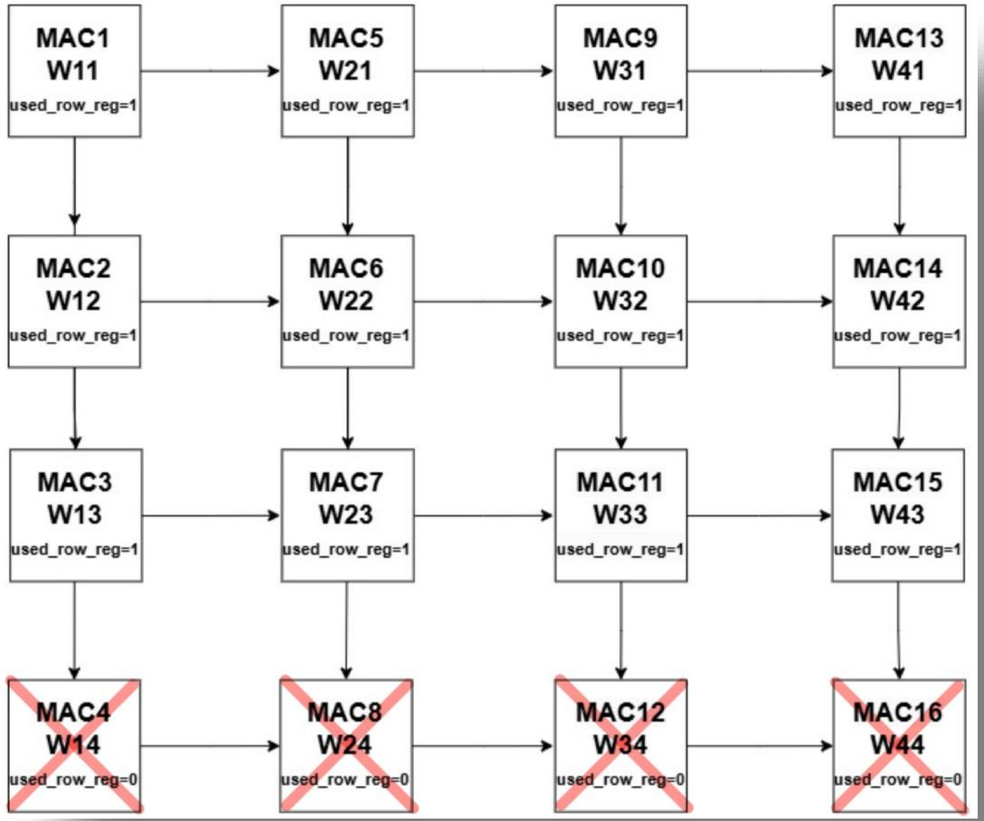
1) used_row_reg

MAC ARRAY에 고정되는 Weight와 같이 저장되어 만약 used_row_reg가 0이면 해당 MAC에서 연산이 이루어지지 않으므로, SUM=0

```
if(M>4) begin
    used_row_reg    <=  (addr_w_reg[1:0] <= M[1:0]-2'd1) ? 1'b1 : ~addr_w_reg[2];
end else begin
    used_row_reg    <=  (addr_w_reg[1:0] <= M[1:0]-2'd1) ? 1'b1 : 1'b0;
end
```

4<M<8 인 경우에도 작동

ex)M=3



1. Control Signal (LOAD_W)

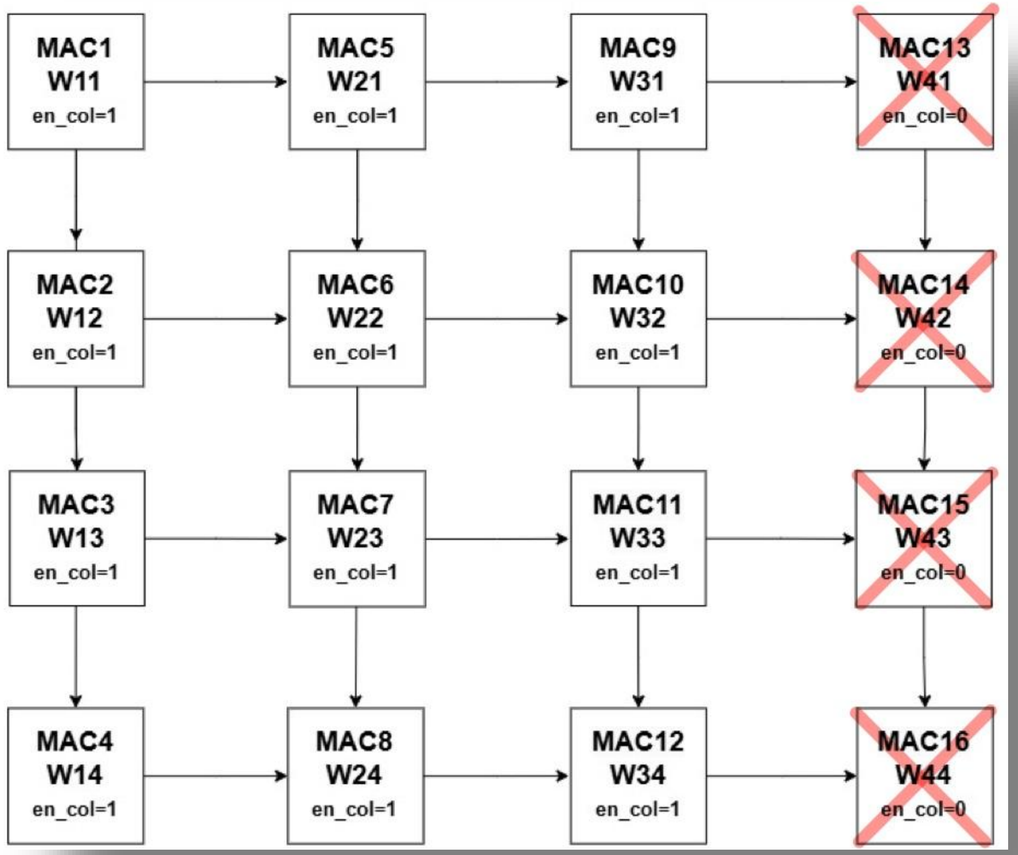
2) en_col

N<4 이거나 4<N<8일 경우 사용하지 않는 Column이 존재하므로 en_col을 통해 연산을 제어함.

```
reg     [3:0]   en_col;
wire    [3:0]   shift;
assign  shift =  (where_w_reg==2'd0||where_w_reg==2'd3) ? (4'd8 - N) : 4'd0;
always @(*) begin
    if(EN_W) begin
        if(N>4) begin
            en_col  =   4'b1111 << shift;
        end else begin
            en_col  =   4'b1111 << (4'd4-N);
        end
    end else begin
        en_col = 4'd0;
    end
end
```

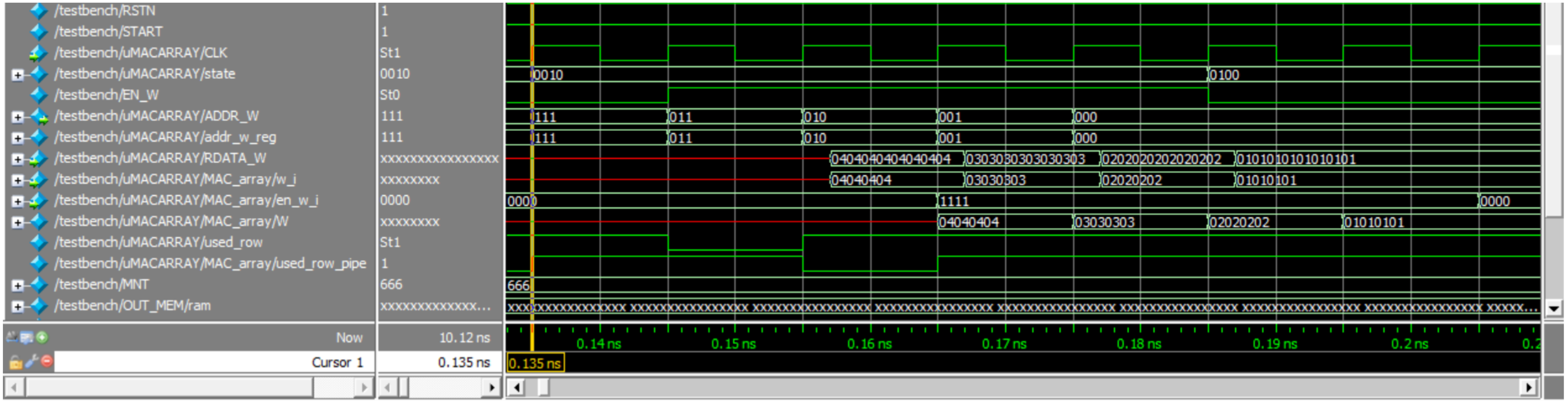
4<N<8일 경우 오른쪽 사분면 연산시 shift를 통해 en_col 할당

ex)N=3



1. (LOAD_W) Waveform

Ex)MNT=666



2. Control Signal (LOAD_I)

1) compute_cycle

N이 4보다 작거나 크면 사용하지 않는 MAC 열이 존재하므로 RESULT를 빨리 저장할 수 있음

2) cycle_count

LOAD_I state가 유지될 때 마다 증가하여 T값과 비교하여 state 갱신

```
reg     [3:0]   compute_cycle;
always @(*) begin
    if((N>=4)) begin
        if(where_w_reg==2'd0||where_w_reg==2'd3) begin
            compute_cycle   =   (N-4'd4) + 4'd5;
        end else begin
            compute_cycle   =   4'd9;
        end
    end else begin
        compute_cycle       =   N + 4'd5;
    end
end
```

(compute_cycle 조건)

```
end else if (cycle_count==T) begin
    stop_mac_reg    <=  1'b1;
    en_i_reg        <=  1'b0;
    addr_i_reg      <=  3'd0;
    cycle_count     <=  cycle_count + 4'd1;
    en_o_reg        <=  1'b0;
    addr_o_reg      <=  {3'd0,(where_w_reg==2'd2||where_w_reg==2'd3)};
end else if (cycle_count==compute_cycle) begin
    load_i_times    <=  load_i_times + 2'd1;
    stop_mac_reg    <=  1'b1;
    en_i_reg        <=  1'b0;
    addr_i_reg      <=  3'd0;
    cycle_count     <=  4'd0;
```

(cycle_count 비교 조건)

2. Control Signal (LOAD_I)

3) stop_mac_reg

마지막으로 Loding되는 INPUT과 같이 MAC Array에 전달되어 모든 Input이 로딩되고나서 불필요한 연산을 막음
Input과 마찬가지로 delay 적용

4) overwrite_sig

N>4일 경우, 이전에 Output Memory에 저장한 data를 read하여 사용할 때 Flag로 사용.

```
stop_mac_4_3d[0] <=  stop_mac_pipe;
stop_mac_4_3d[1] <=  stop_mac_4_3d[0];
stop_mac_4_3d[2] <=  stop_mac_4_3d[1];

stop_mac_3_2d[0] <=  stop_mac_pipe;
stop_mac_3_2d[1] <=  stop_mac_3_2d[0];

stop_mac_2_1d    <=  stop_mac_pipe;
```

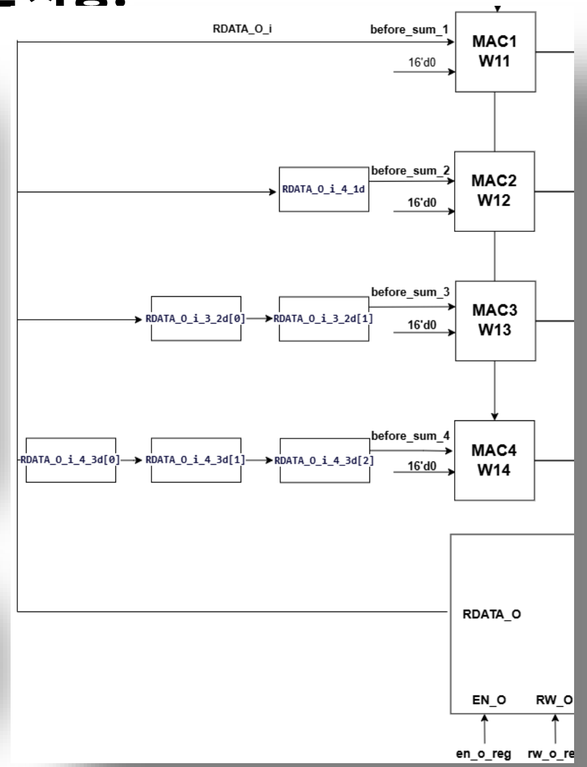
(stop_mac_reg delay)

```
always @(*) begin
    if((stage==3'd2)||(stage==3'd3)) begin
        overwrite_sig   =   (load_w_times>=3'd2);
    end else if((stage==3'd6)||(stage==3'd7)) begin
        overwrite_sig   =   (load_w_times>=3'd3);
    end else begin
        overwrite_sig   =   1'b0;
    end
end
```

(overwrite_sig 조건)

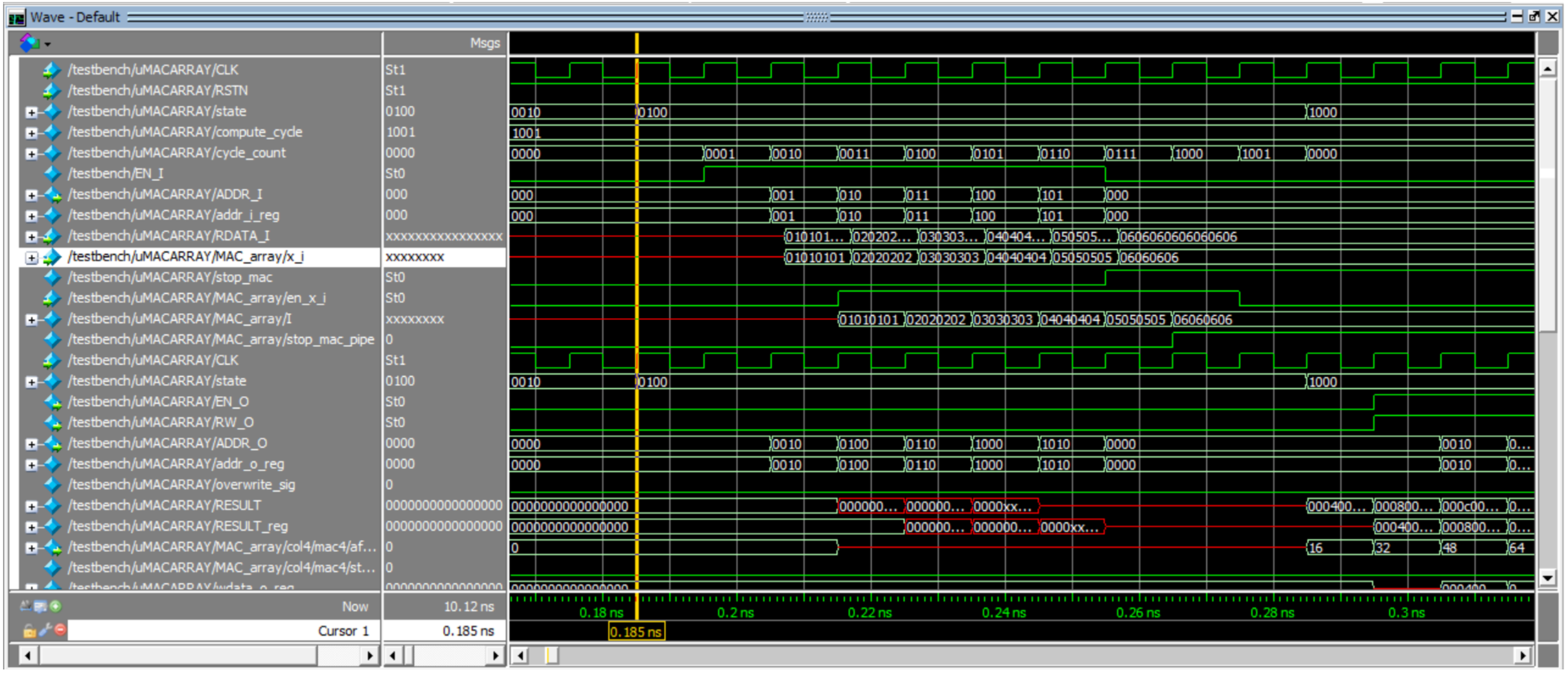
```
en_o_reg        <=  overwrite_sig;
rw_o_reg        <=  1'b0; // Read Mode
addr_o_reg      <=  {3'd0,(where_w_reg==2'd2||where_w_reg==2'd3)};
```

(en_o_reg에 overwrite_sig 할당 및 주소 할당)



2. (LOAD_I) Waveform

Ex)MNT=666



3. Control Signal (STORE_O)

1) where_w_reg, where_i_reg

STORE_O 종료후 READY state로 이동 시, 다음 연산 시 weight와 input 위치 저장

2) store_count

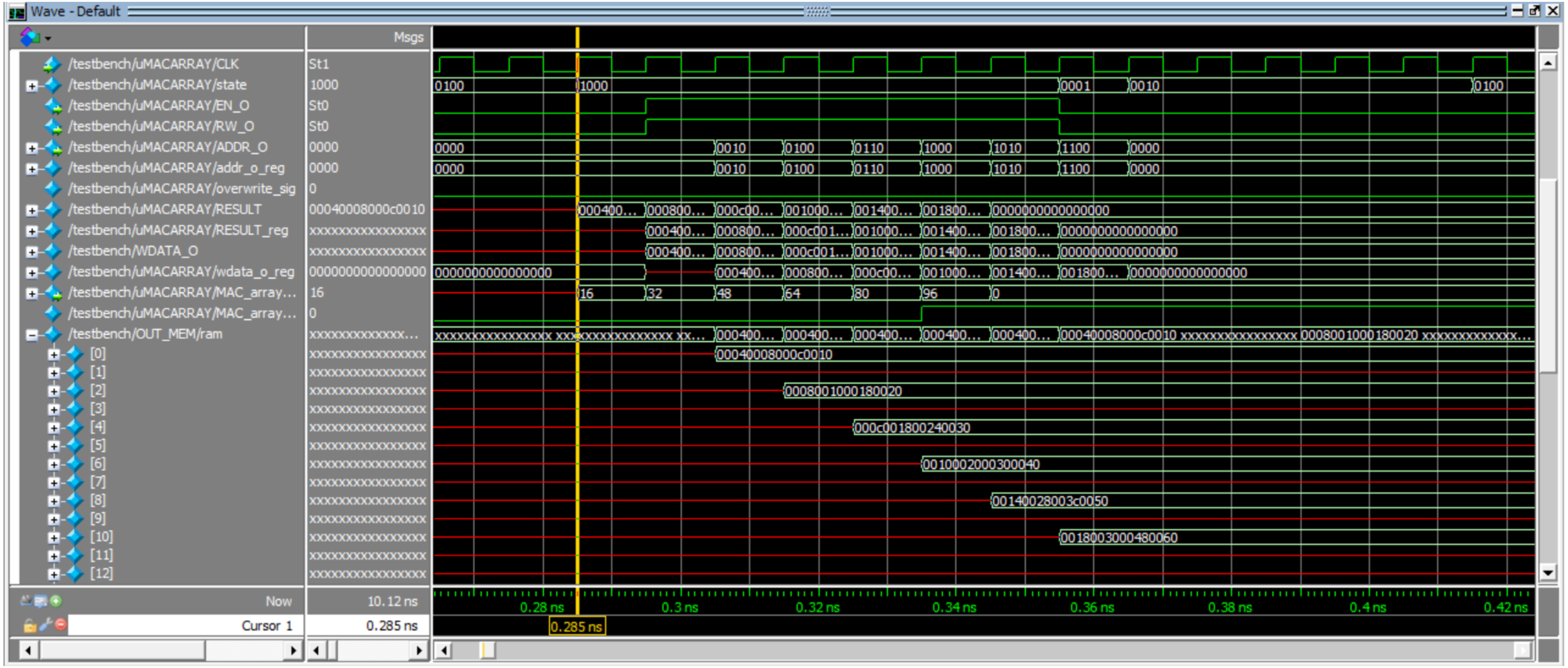
STORE_O state가 유지될 때 마다 증가하고, store_count==T가 되면 저장 종료

```
3'd7: begin
    if(store_count==T) begin
        if(load_w_times==1) begin
            where_w_reg     <=   2'd2;
            where_i_reg     <=   2'd1;
        end else if(load_w_times==2) begin
            where_w_reg     <=   2'd0;
            where_i_reg     <=   2'd0;
        end else if(load_w_times==3) begin
            where_w_reg     <=   2'd3;
            where_i_reg     <=   2'd0;
        end else begin
            where_w_reg     <=   2'd1;
            where_i_reg     <=   2'd1;
        end
    end else begin
        where_w_reg     <=   where_w_reg;
        where_i_reg     <=   where_i_reg;
    end
end
```

```
if(store_count==4'd0) begin
    en_o_reg    <=  1'b1;
    rw_o_reg    <=  1'b1; // Write Mode
    wdata_o_reg <=  RESULT_reg;
    addr_o_reg  <=  {3'd0,(where_w_reg==2'd2||where_w_reg==2'd3)};
    store_count <=  4'd1;
end else if(store_count < T) begin
    addr_o_reg  <=  addr_o_reg + 4'd2;
    store_count <=  store_count + 4'd1;
    wdata_o_reg <=  RESULT_reg;
end else if(store_count == T) begin
    en_o_reg    <=  1'b0;
    rw_o_reg    <=  1'b0; // Write 종료
    addr_o_reg  <=  addr_o_reg + 4'd2;
    store_count <=  4'd0;
    wdata_o_reg <=  RESULT_reg;
end else begin
    store_count <=  4'd0;
    en_o_reg    <=  1'b0;
    rw_o_reg    <=  1'b0;
end
```

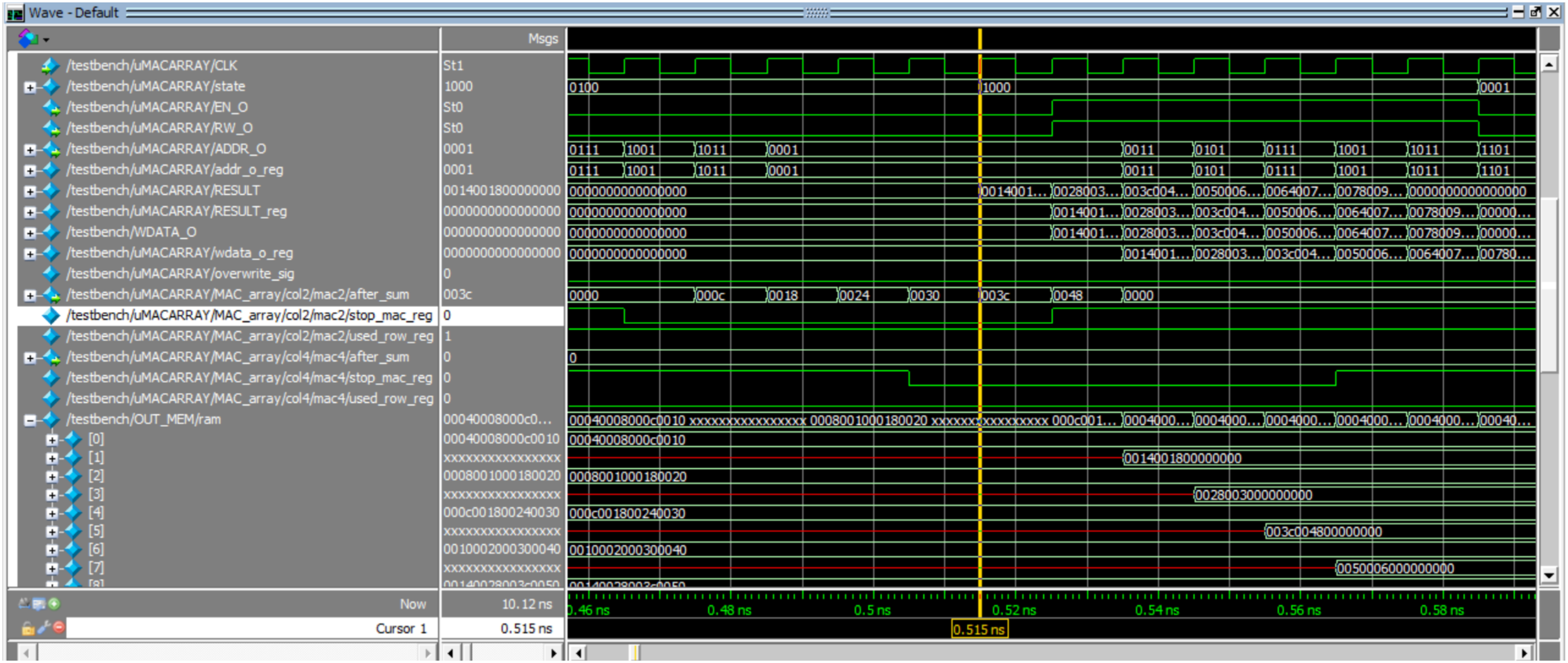
3. (STORE_O) Waveform

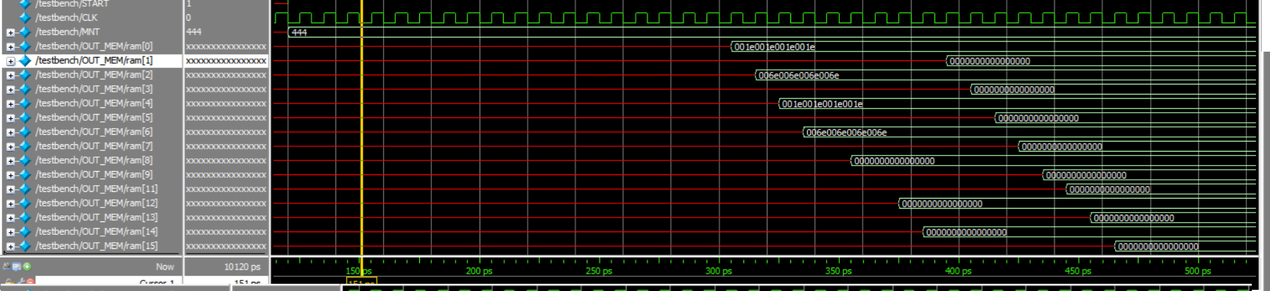
Ex)M<4, N<4

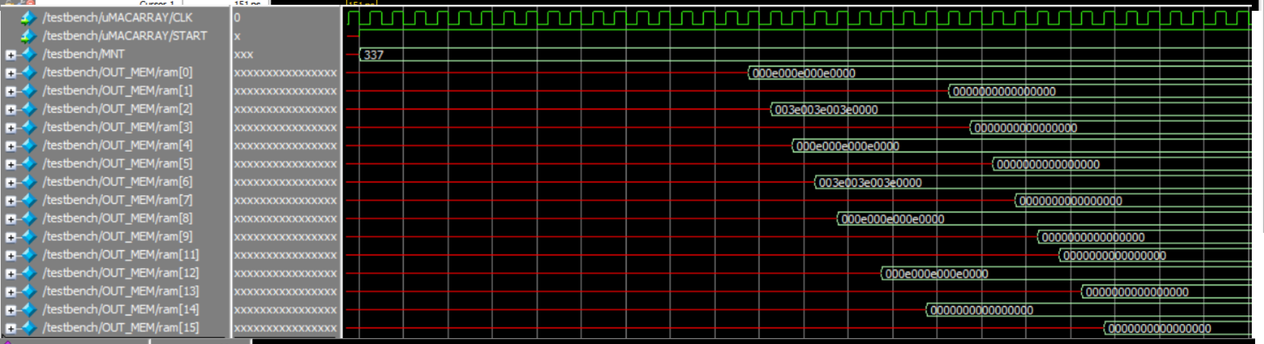


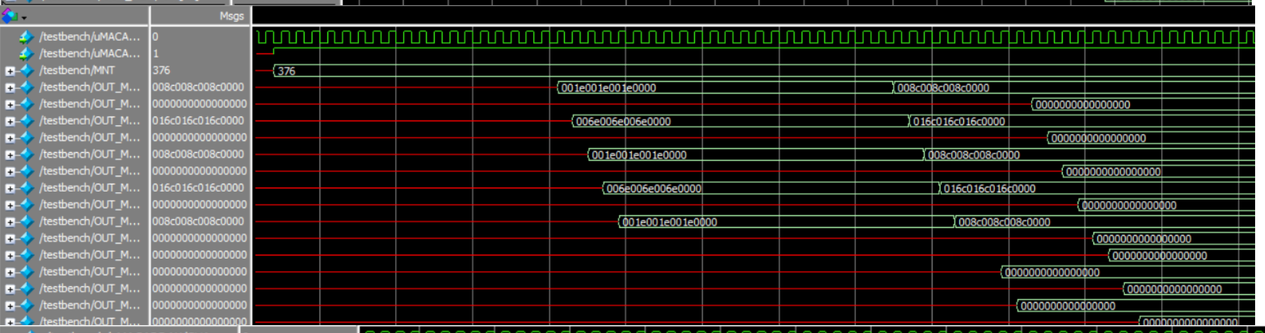
3. (STORE_O) Waveform

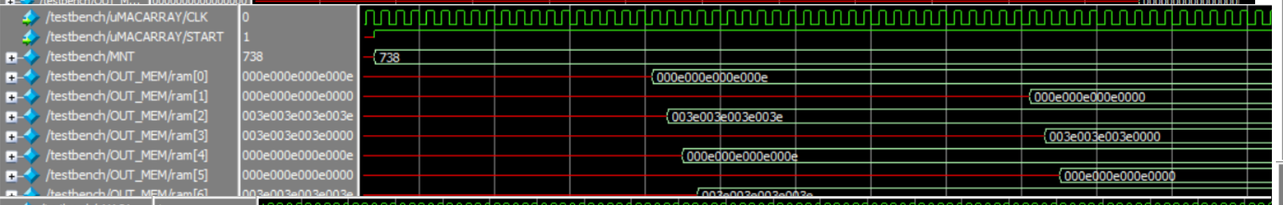
Ex)M>4, N<4

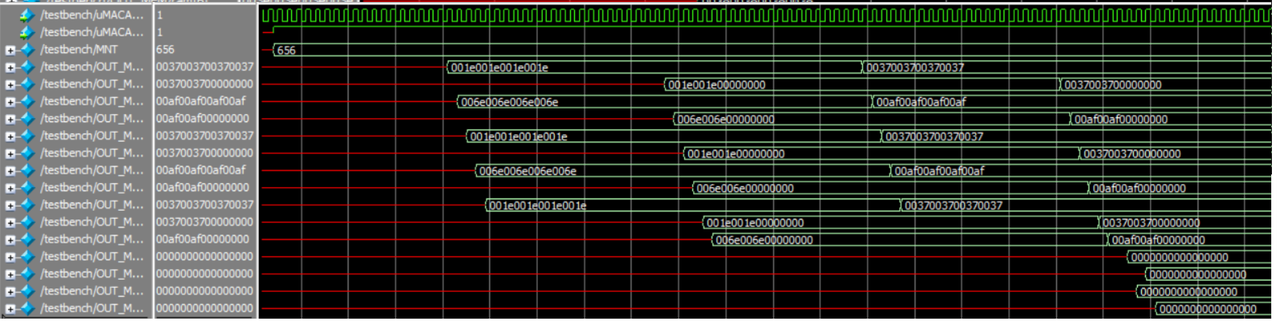


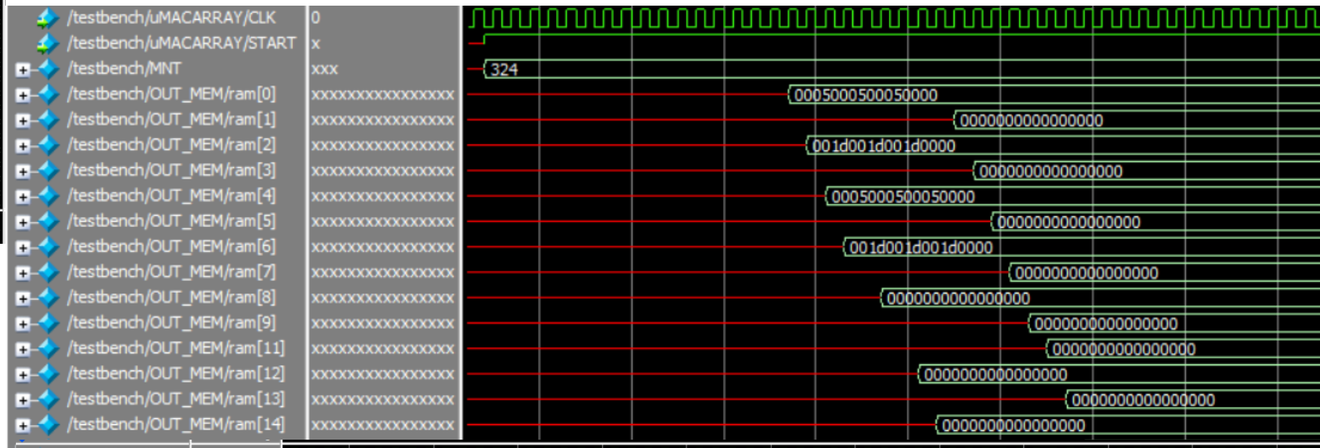


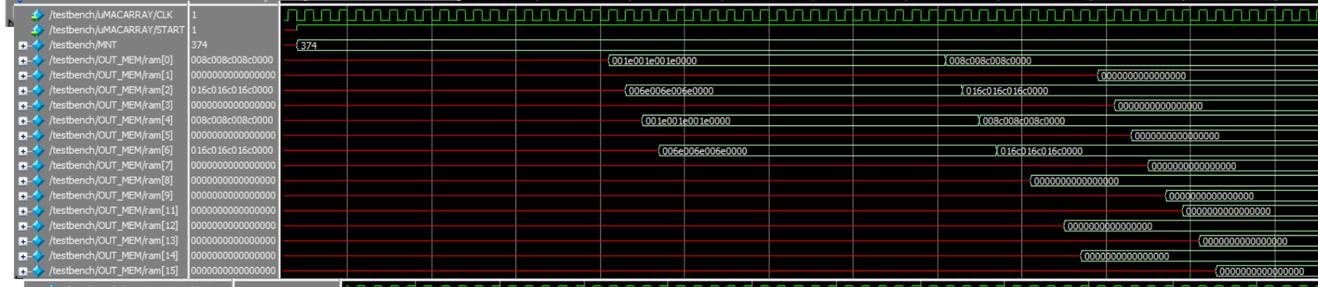


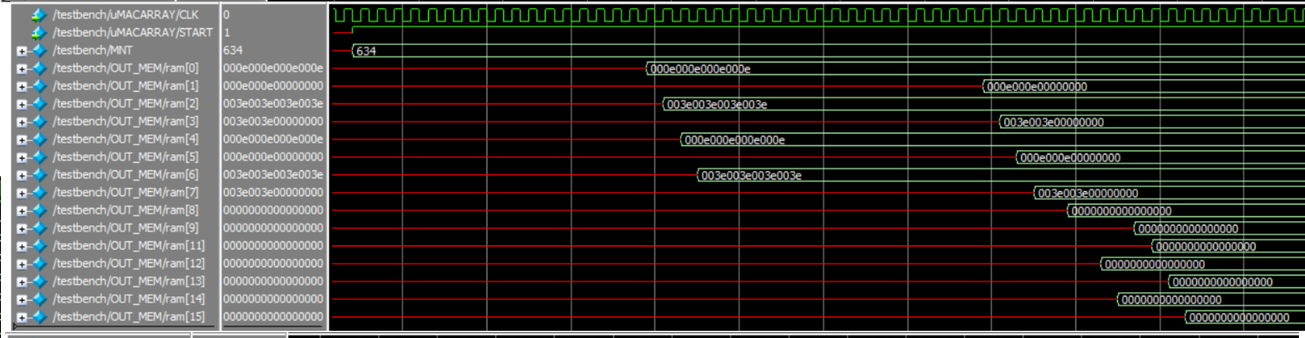


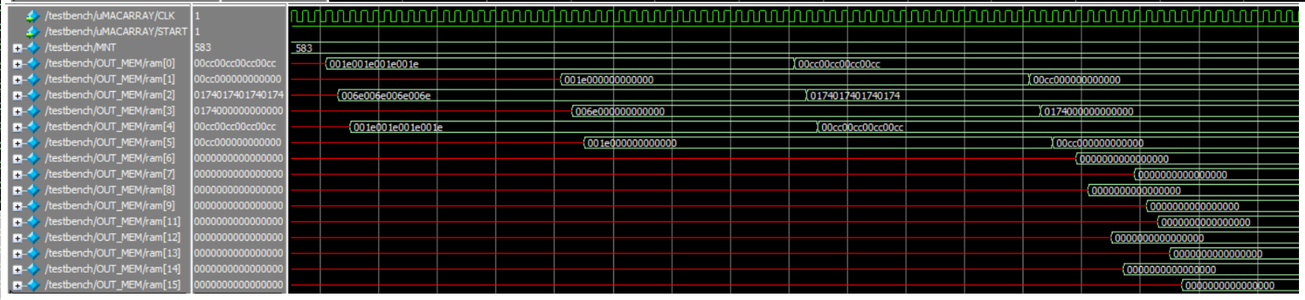












444, 324, 337, 374, 376, 634, 738, 583, 656 검증완료

Thank you