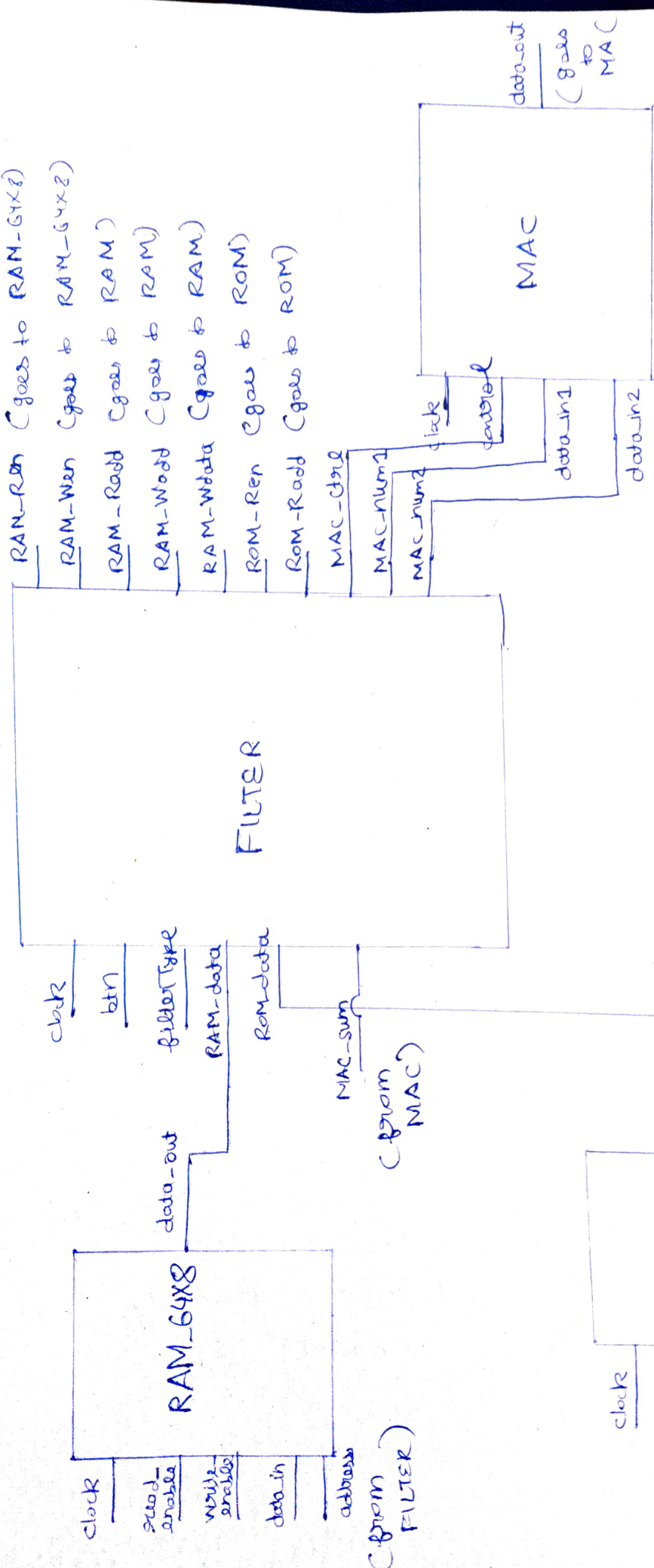# Design Overview

TANISHQ DUBEY
2019CS51077

- A button is used to start filtering the image, few counters as described in the later section perform the sliding window operational filter task and write the data in the memory.

- Pressing the button again after writing data will load to display of image, however this display is not a part of this assignment.

- It is assumed all data is stored grow-wise, both for image matrix and filtering coefficients, See example below –

2D-matrix   RAM/ROM
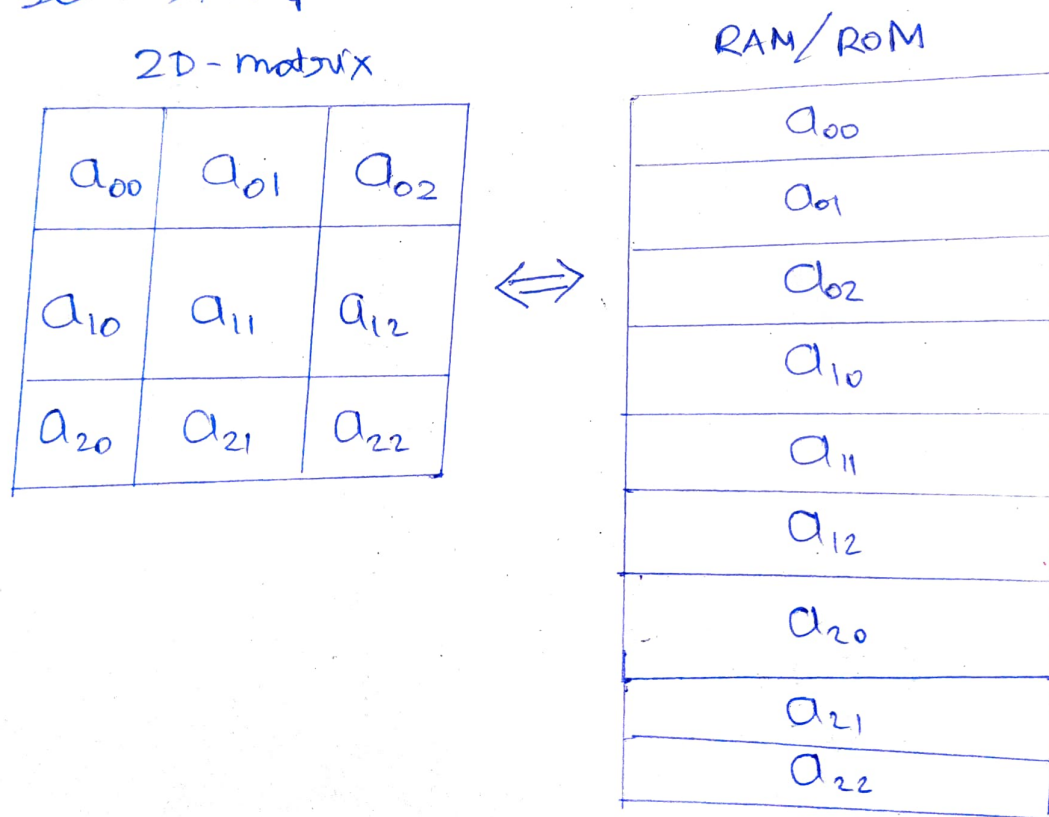
- Design data-path and specifications on the following page, including 4 entities namely RAM_64X8, ROM_32X9, MAC and FILTER.

FILTER (block)

Outputs:
- RAM_Ren (goes to RAM_64x8)
- RAM_Wen (goes to RAM_64x2)
- RAM_Radd (goes to RAM)
- RAM_Wadd (goes to RAM)
- RAM_Wdata (goes to RAM)
- ROM_Ren (goes to ROM)
- ROM_Radd (goes to ROM)
- MAC_ctrl
- MAC_num1
- MAC_num2

Inputs:
- clock
- btn
- filterType
- RAM_data
- ROM_data
- MAC_sum (from MAC)

MAC (block)
- clock
- control
- data_in1
- data_in2
- data_out (goes to MAC)

RAM_64X8 (block)
- clock
- read_enable
- write_enable
- data_in
- address (from FILTER)
- data_out

ROM_32X9 (block)
- clock
- read_enable
- address (from FILTER)
- data_out

Note: common clock and certain connections aren't shown with lines for sake of legibility.

Control Description and ASM chart

```
for (j = 0, j ≤ 118, j++)
{    for (i = 0, i ≤ 158, i++)
   {    MAC and reset
        Write data in RAM
        for (k = 0, k ≤ 2, k++)
      {    for (l = 0, l ≤ 2, l++)
         {    if (smoothen) then
                  RAM read (160(j+k) + i+l)
                  ROM read (3k + l)
                  assign read data to MAC
              elseif (sharpen) then
                  RAM read (160(j+k) + i+l)
                  ROM read (3k + l + 16)
                  assign read data to MAC
         }
      }
   }
}
```
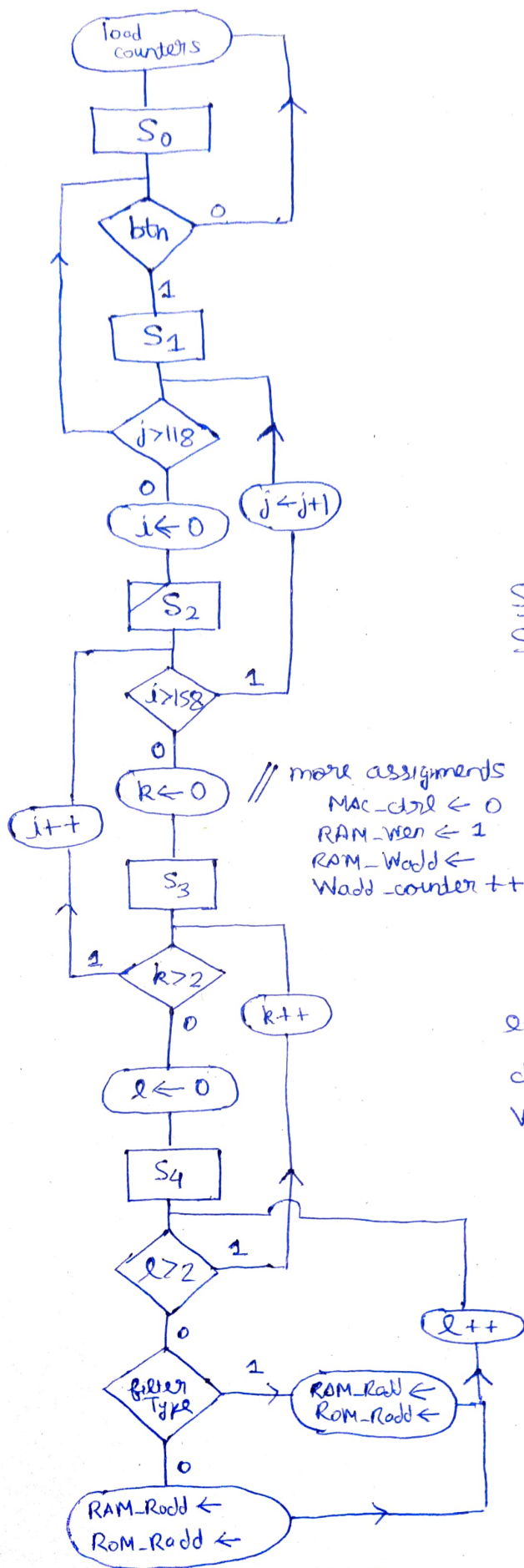
Pseudo-code for sliding window filter operation

ASM-Chart
for window
sliding
Filter

load counters

S_0

btn — 0

1

S_1

$j > 118$ — 1 → $j \leftarrow j+1$

0

$i \leftarrow 0$

S_2

$i > 158$ — 1

0

$R \leftarrow 0$ // more assignments
                        MAC_ctrl $\leftarrow$ 0
$i++$ RAM_wen $\leftarrow$ 1
                        ROM_Wadd $\leftarrow$
S_3 Wadd_counter ++

1     $k > 2$ → $k++$

0

$l \leftarrow 0$

S_4

$l > 2$ — 1

0

filter Type — 1 → ROM_RaN $\leftarrow$
                                    ROM_Radd $\leftarrow$

$l++$

0

RAM_Rodd $\leftarrow$
ROM_Radd $\leftarrow$

// exact details
in pseudo code
and VHDL code

States
S_0, S_1, S_2, S_3, S_4

Note

exact assignment
details in
VHDL code and
pseudo code