The data path and the controller are very similar to the midterm project, but few things have been changed:

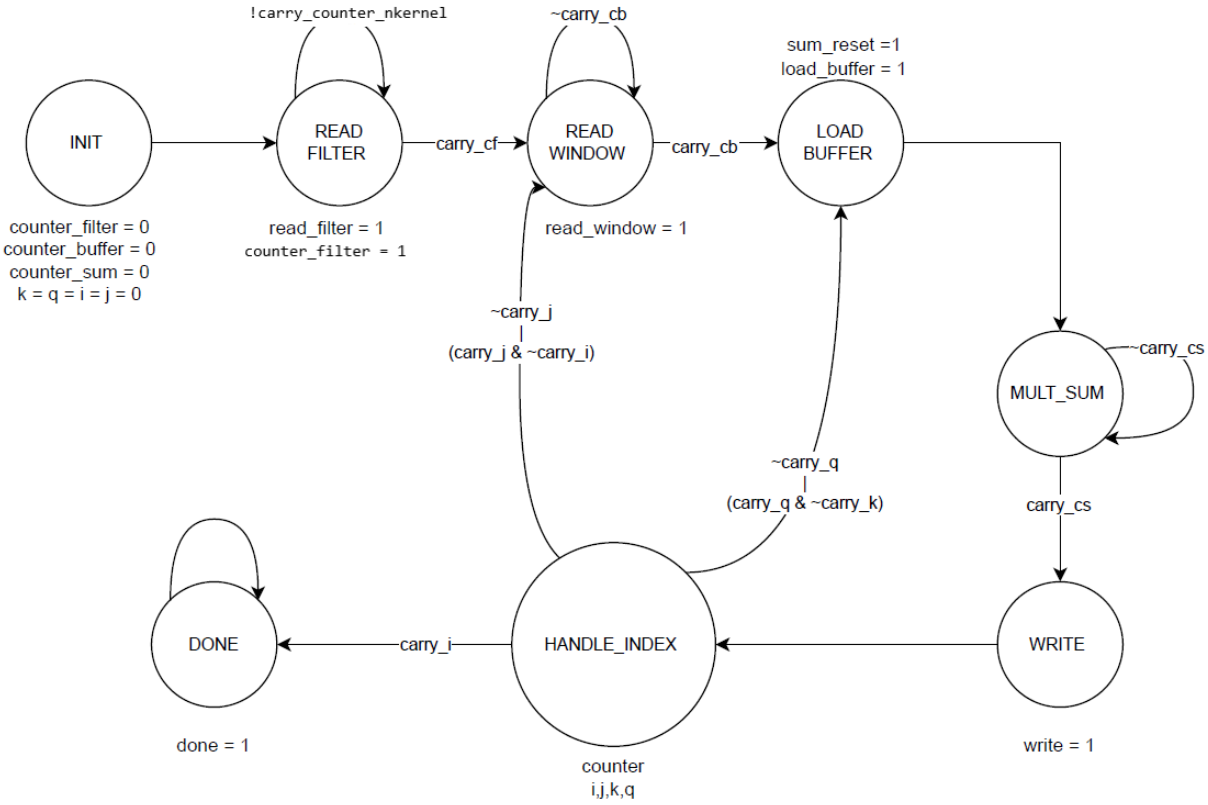
- Instead of 1 kernel, now we have many kernels, in the source code, the parameter ‘nkernel’ is used to define number of kernels.
- Number of clocks to read kernels is expanded from 4 to 4 × nkernel.
- A ‘counter_nkernel’, is added to the controller to show in which kernel we should load the memory.
- The addresses of kernels are considered to be ranged from ‘y’ to ‘4 × nkernel + y’, all kernels are expected to be continuously after each other in the memory.
- Number of MACs expanded to ‘nkernel’.
- The output of i-th MAC will be saved on i-th OFM (output memory) respectively.
- ‘z’ is equal for all OFMs.
- ‘generate’ method is used for instancing multiple modules.

And the most important thing:

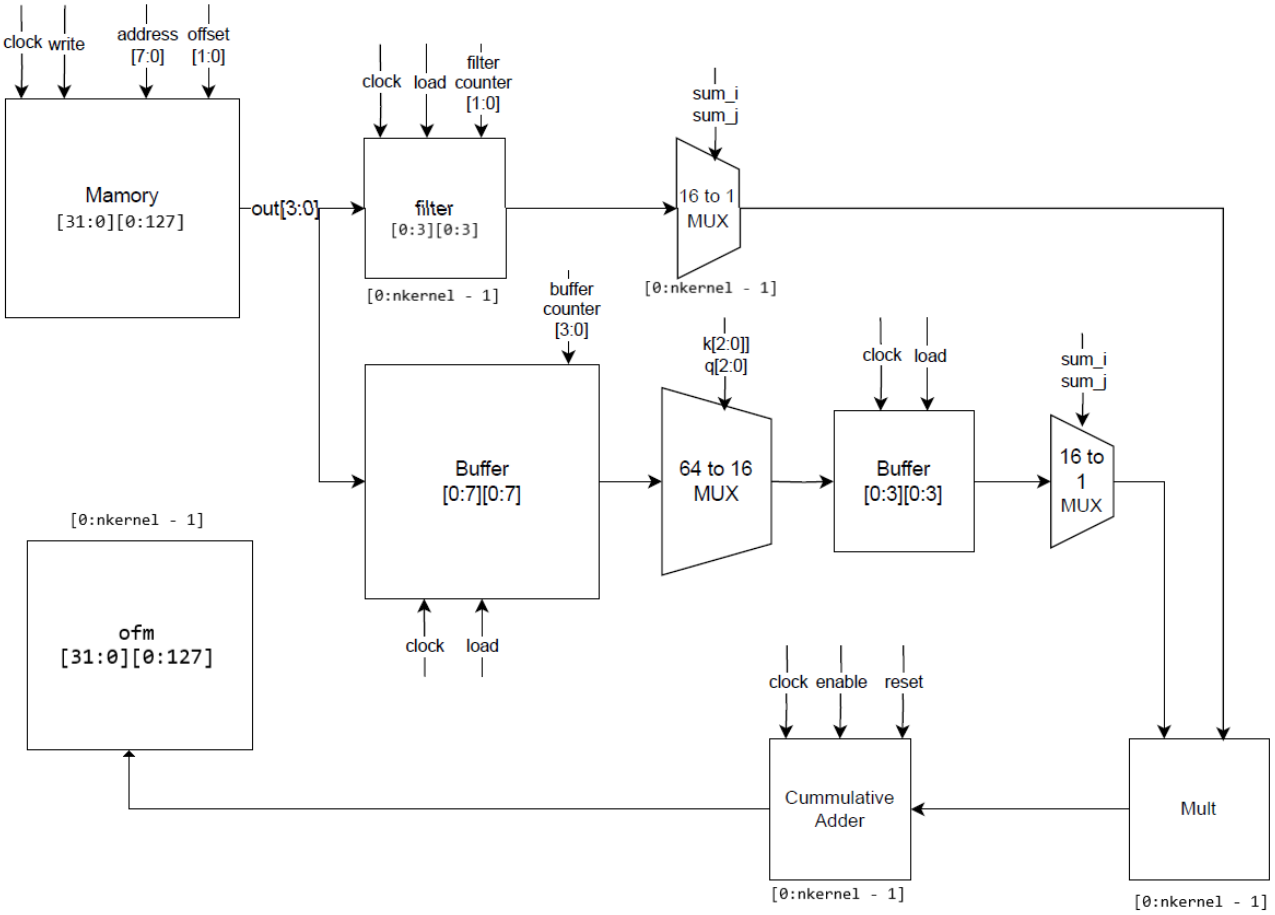
- All calculations are in parallel and at the same time.

The diagrams can be seen in the fallowing pages.

Controller:


!carry_counter_nkernel
~carry_cb
sum_reset =1
load_buffer = 1
INIT
READ FILTER
carry_cf
READ WINDOW
carry_cb
LOAD BUFFER
counter_filter = 0
counter_buffer = 0
counter_sum = 0
k = q = i = j = 0
read_filter = 1
counter_filter = 1
read_window = 1
~carry_j
(carry_j & ~carry_i)
~carry_cs
MULT_SUM
~carry_q
(carry_q & ~carry_k)
carry_cs
DONE
carry_i
HANDLE_INDEX
WRITE
done = 1
counter
i,j,k,q
write = 1

Datapath:


clock
write
address [7:0]
offset [1:0]
Mamory [31:0][0:127]
out[3:0]
clock
load
filter counter [1:0]
filter [0:3][0:3]
[0:nkernel - 1]
sum_i sum_j
16 to 1 MUX
[0:nkernel - 1]
buffer counter [3:0]
Buffer [0:7][0:7]
clock
load
k[2:0]] q[2:0]
64 to 16 MUX
clock
load
Buffer [0:3][0:3]
sum_i sum_j
16 to 1 MUX
[0:nkernel - 1]
ofm [31:0][0:127]
clock
enable
reset
Cummulative Adder
[0:nkernel - 1]
Mult
[0:nkernel - 1]