INPUT SIZE – n x m x ch

KERNEL SIZE – k x k x ch

Input Fetching strategy –

An image representing fetching groups until we reach the last channel

Once we start getting the cylinders, we can start MAC operation with respective kernels

3 pipes for input and calculation: inp_pipe(continuously fetches input), data_pipe(gets the input for dot products), ker_pipe(gets kernel values for dot products)

There will k x k pipes meant for storage of input values from previous groups. Let the storage pipes be named S[k][k]

Each of the storage pipes will be re-used k-1 times, so we need to push back the used values.

```
fetchGroupInitial( r, c, chn):

    S.clear()                              //clears previous storage

    for I in range(k):

        for j in range(k):

            data_pipe.push(inp_pipe) //push 64bits of data or 1 cylinder

            S[j][i].push(inp_pipe)
```

```
fetchGroupLater( r, c, chn, p, q):

    //store the new slice in storage

    //fetch the new group according to p and q
```

```
switchRow(r, c, chn, q) :

    for I in range k:

        S[q][i].clear()

        S[q][i].push(inp_data)             //put the data in qth row
```

fetchKernel() :

```
    for I in range k:
        for j in range k:
            cyl = kern[j][i]
            kern_pipe.push(cyl)      //pushes a cylinder in kernel_pipe
            kern[j][i].push(cyl)     //pushes the cylinder back in kernel
```

colvolution():

```
    for I in $k^2$*8:
        result += kern_pipe*data_pipe    //dot product of data and kernel pixels
    return result
```

Main_func():

```
    For r in range(n/k):
        For c in range(m/k):
            For chn in range(ch/8):
                If (r=0):
                    fetchGroupInitial()      //fetches group for first  row
                else :
                    if (chn = 0):
                        p = (p+1)%k
                        for I in range(k):
                            S[i][p].clear()     //clear the row of pth col
                    If (c=m-k):        //on the last group of row
                        q = (q+1)%k
                        switchRow()
                    fetchGroupLater(p,q)
            fetchKernel()       //fetches kernel values for 8 channels
            out += convolution(data_pipe , kernel_pipe)
output[r][c] = out          //return the output
out = 0
```