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.

fetchGroup(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)
```

fetchSlice(r, c, chn, p):

```
        for I in range(k):
                data_pipe.push(inp_data)
                S[i][p-1].push(inp_pipe)         //push the slice into storage
        For I in range(k-1):
                For j in range(k):
                        Cyl = S[j][(i+p)%k]
                        Data_pipe.push(cyl)      //push a cylinder in data_pipe
                        S[j][(i+p)%k].push(cyl)  //push the cylinder back in storage for further use
```

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%k=0):
                    fetchGroup()    //fetches group for new row
                else :
                    if (chn = 0):
                        for I in range(k):
                            S[i][p-1].clear()  //clear the row of pth col
                    fetchSlice(p)
                fetchKernel()   //fetches kernel values for first 8 channels
                out += convolution(inter_pipe , kernel_pipe)
        p += 1              //incrementing the value of p for next group
        output[r][c] = out      //return the output
        out = 0
```