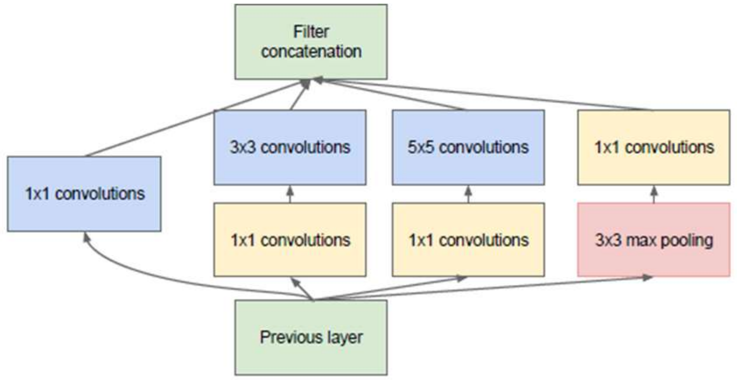
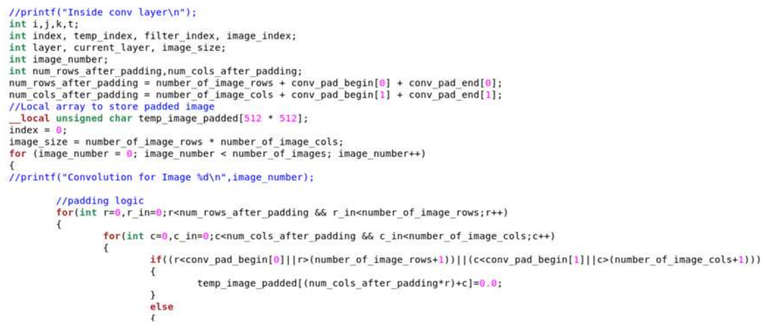
Routing Kernels

Suprajith Suresh Hakathur

Motivation

- Lots of kernels in our design to realise different layers





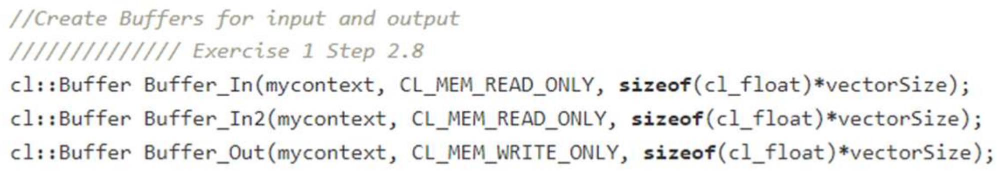
```
//printf("Inside conv layer\n");
int i,j,k,t;
int index, temp_index, filter_index, image_index;
int layer, current_layer, image_size;
int image_number;
int num_rows_after_padding,num_cols_after_padding;
num_rows_after_padding = number_of_image_rows + conv_pad_begin[0] + conv_pad_end[0];
num_cols_after_padding = number_of_image_cols + conv_pad_begin[1] + conv_pad_end[1];
//Local array to store padded image
__local unsigned char temp_image_padded[512 * 512];
index = 0;
image_size = number_of_image_rows * number_of_image_cols;
for (image_number = 0; image_number < number_of_images; image_number++)
{
//printf("Convolution for Image %d\n",image_number);

        //padding logic
        for(int r=0,r_in=0;r<num_rows_after_padding && r_in<number_of_image_rows;r++)
        {
                for(int c=0,c_in=0;c<num_cols_after_padding && c_in<number_of_image_cols;c++)
                {
                        if((r<conv_pad_begin[0]||r>(number_of_image_rows+1))||(c<conv_pad_begin[1]||c>(number_of_image_cols+1)))
                        {
                                temp_image_padded[(num_cols_after_padding*r)+c]=0.0;
                        }
                        else
                        {
```

- Amount of data transferred between these kernels : HUGE !

Motivation

- Can we use global memory for transfers ?

  Extremely inefficient !

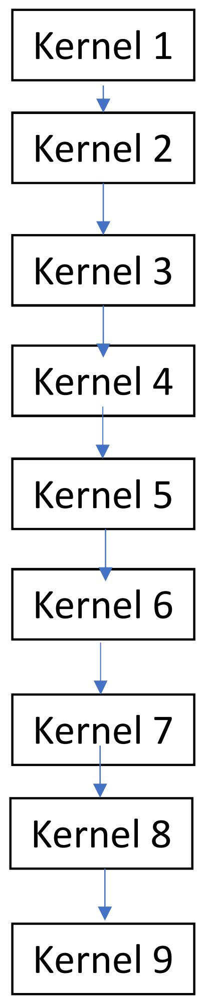


```
//Create Buffers for input and output
////////////// Exercise 1 Step 2.8
cl::Buffer Buffer_In(mycontext, CL_MEM_READ_ONLY, sizeof(cl_float)*vectorSize);
cl::Buffer Buffer_In2(mycontext, CL_MEM_READ_ONLY, sizeof(cl_float)*vectorSize);
cl::Buffer Buffer_Out(mycontext, CL_MEM_WRITE_ONLY, sizeof(cl_float)*vectorSize);
```

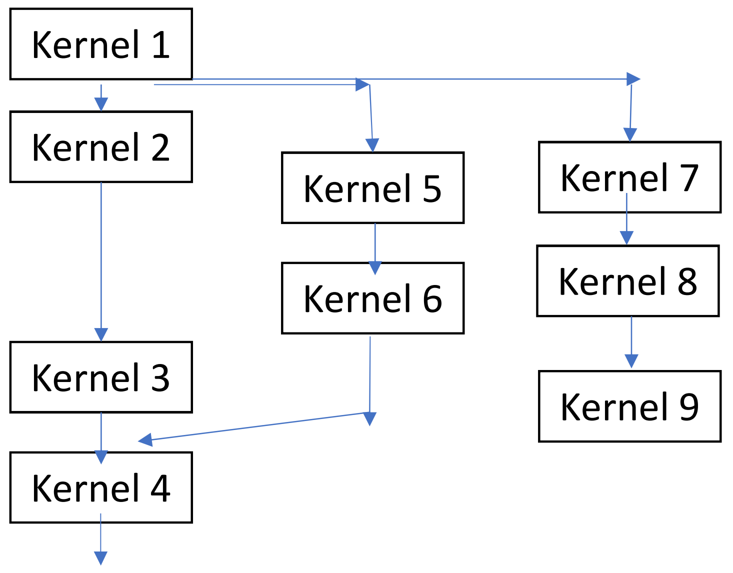
  - Take away : Avoid using Global Memory to transfer data between kernels.

- Can we use dedicated channels ?

  Leads to inflexible design.



Linear Design

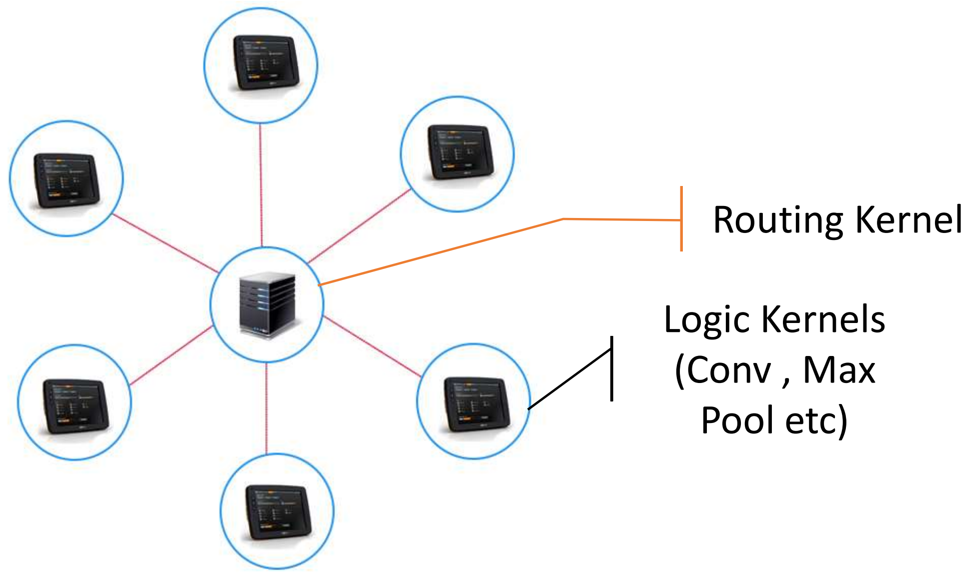


Design closer to our project

Need lots of separate channels running between kernels.
Need to hand code this as we do not have a OpenCL code generator which can do this much of coding for us.

Alternate solution – Routing Kernels

- Write a separate kernel to move data from source to sink



- Star topology

Routing kernel design

- Simple design paradigm :
  - Data : source → router
  - Router looks at the source “address”, sink “address” and amount of data to be “routed”
  - Data : router → sink

- Data is divided into two categories :
  - Data needed by kernels
  - Data needed for “routing”

- Data needed for “routing”
  - Source ID
  - Sink ID
  - Amount of data
- Send Headers to indicate Source ,Sink etc? No need ! We are working at OpenCL level !

OpenCL implementation - 1

- __kernel void RoutingKernel(int source , int dest, int Num_data , int fpga_ext_routing)
  - source : Producer`s unique ID.
  - dest : Consumer`s unique ID.

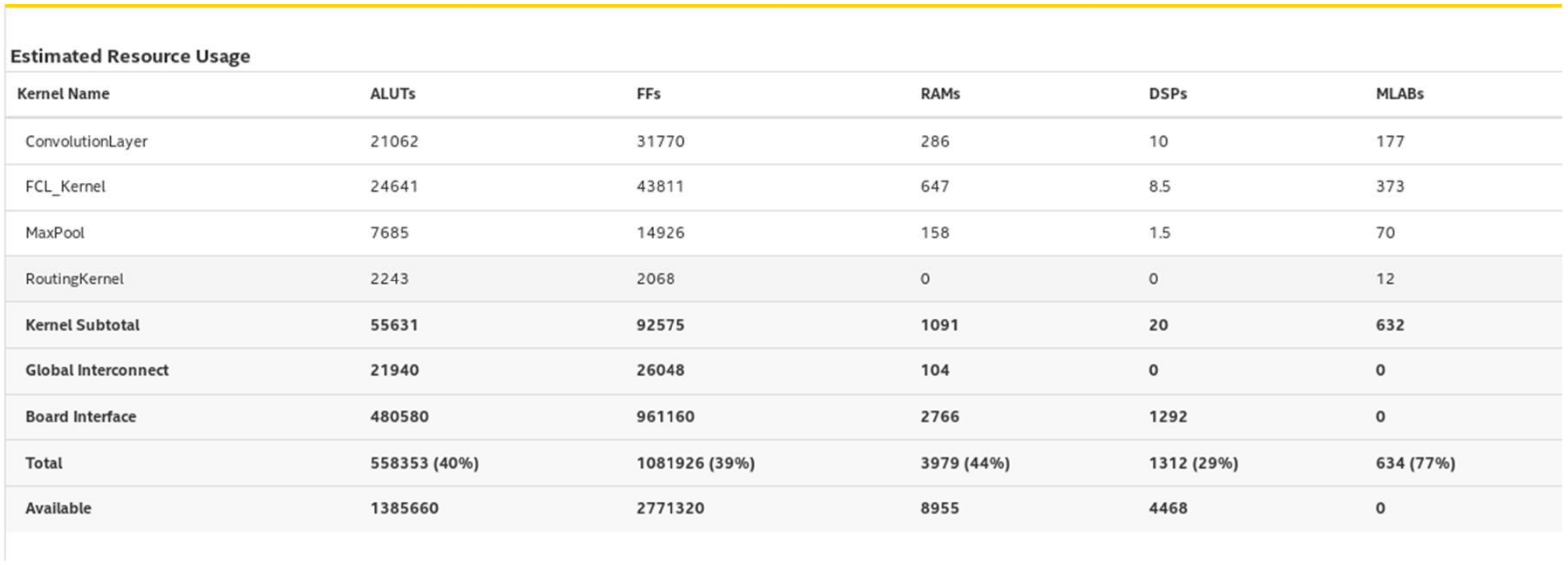
  ❯ Assigned at higher design level
  - Num_data : Number of iterations needed to transfer data (eg: at the end of conv layer →dim_x * dim_y * number_of_images*number_of_filters)
  - Fpga_ext_routing : Do we need to send the data out of the current FPGA ?
    - For scaling purposes
- Need to create two channels between every kernel and the “router”

OpenCL implementation - 2

- Changes in logic kernels
  - Need to replace writes to Global Memory with
    “**write_channel_intel**(<*name of producer to kernel channel*>, <*output*>); “
  - Similarly , replace Global Memory reads with read channel function
    “input = **read_channel_intel**(<*name of router to kernel channel*>);”
- Changes in host code
  - Need to separately launch Routing Kernel as soon as we have launched a logic kernel.

Resource Utilization

- No clear picture yet. Demo kernels are too small.
    - Tested on SimpleCNN using OpenVINO plugin
    - This design : Almost same resource usage.



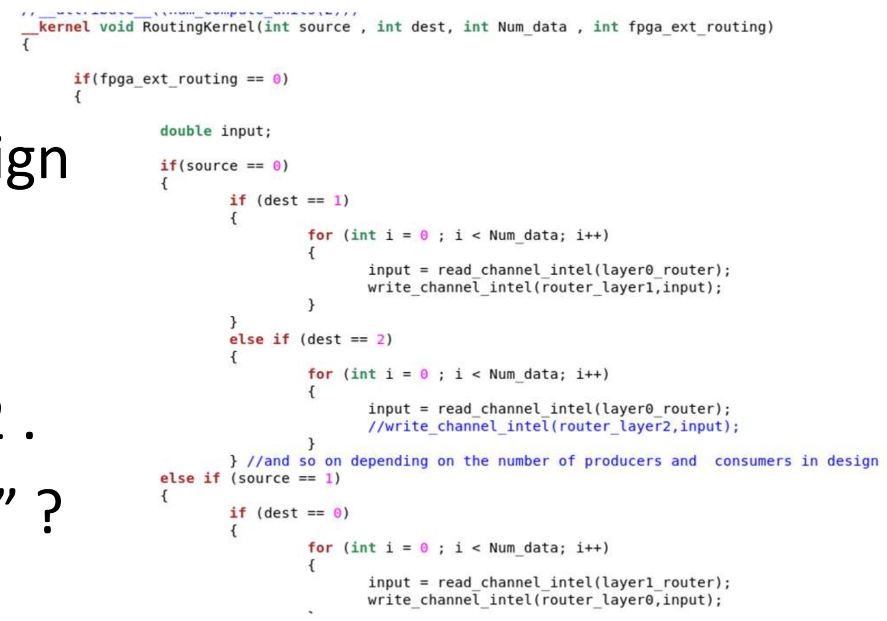
Estimated Resource Usage

Kernel Name	ALUTs	FFs	RAMs	DSPs	MLABs
ConvolutionLayer	21062	31770	286	10	177
FCL_Kernel	24641	43811	647	8.5	373
MaxPool	7685	14926	158	1.5	70
RoutingKernel	2243	2068	0	0	12
Kernel Subtotal	**55631**	**92575**	**1091**	**20**	**632**
Global Interconnect	**21940**	**26048**	**104**	**0**	**0**
Board Interface	**480580**	**961160**	**2766**	**1292**	**0**
Total	**558353 (40%)**	**1081926 (39%)**	**3979 (44%)**	**1312 (29%)**	**634 (77%)**
Available	**1385660**	**2771320**	**8955**	**4468**	**0**

Conclusion

- We need Routing Kernels for flexible design
- Easy to implement at OpenCL level
  - But tedious to code the routing table
- Layer -3 concept but we use it at Layer -2 .

  So better to call them “switching kernels” ?



```
__kernel void RoutingKernel(int source , int dest, int Num_data , int fpga_ext_routing)
{

    if(fpga_ext_routing == 0)
    {

            double input;

            if(source == 0)
            {
                    if (dest == 1)
                    {
                            for (int i = 0 ; i < Num_data; i++)
                            {
                                    input = read_channel_intel(layer0_router);
                                    write_channel_intel(router_layer1,input);
                            }
                    }
                    else if (dest == 2)
                    {
                            for (int i = 0 ; i < Num_data; i++)
                            {
                                    input = read_channel_intel(layer0_router);
                                    //write_channel_intel(router_layer2,input);
                            }
                    } //and so on depending on the number of producers and  consumers in design
            else if (source == 1)
            {
                    if (dest == 0)
                    {
                            for (int i = 0 ; i < Num_data; i++)
                            {
                                    input = read_channel_intel(layer1_router);
                                    write_channel_intel(router_layer0,input);
```