

## GPU - Programming

### Homework - 6

#### Topic - Convolution filter

#### Code -

```
#include <stdio.h>
#include <cuda.h>

#define N          100
#define BLOCKSIZE  32

__global__ void init(int *input) {
    unsigned id = blockDim.x * blockIdx.x + threadIdx.x;
    if (id < N) input[id] = id + 1;
}

__global__ void print(int *output) {
    for (unsigned ii = 0; ii < N; ++ii)
        printf("%d ", output[ii]);
    printf("\n");
}

// defined in class by instructor
__global__ void convolutionV1(int *input, int *filter, int *output, int fsize) {
    unsigned id = blockDim.x * blockIdx.x + threadIdx.x;
    if (id >= N) return;

    //int *filteroutput = (int *)malloc(fsize * sizeof(int));
    int sum = 0;
    int halff = fsize / 2;
    int istart = id - halff, iend = id + halff + 1;
    int fstart = 0, fend = fsize;

    if (istart < 0) {
        fstart -= istart;
        istart = 0;
    }
    if (iend > N) {
        fend -= (iend - N);
        iend = N;
    }
}
```

```

    for (unsigned ii = fstart; ii < fend; ++ii) {
        // filteroutput[ii] = input[id + ii] * filter[ii];
        // sum+=filteroutput[ii];
        sum += input[istart + ii - fstart] * filter[ii];
    }
    output[id] = sum;
}

// defined by me using two loops for convolution and considers starting elements also thus output is not
//the same as version 1.
__global__ void convolutionV2(int *input,int *filter,int *output,int fsize)
{
    int tid = blockDim.x*blockIdx.x+threadIdx.x;
    int step = blockDim.x*gridDim.x;
    int convlength = N+fsize-1,n,k;
    for(n=tid;n<convlength;n+=step)
    {
        int sum = 0;
        int limit = min(N,n);
        for(k=0;k<limit;k++)
        {
            if(k<N && n-k<fsize)
            {
                sum+=input[k]*filter[n-k];
            }
        }
        output[n]=sum;
    }
}

// this version gives the same output as version 1 but provides 2 more outputs at the beginning and end.
__global__ void convolutionV3(int *input,int *filter,int *output,int fsize)
{
    int tid = threadIdx.x;
    int blocksize = blockDim.x;
    int i,j;
    for(i=tid;i<N;i+=blocksize)
    {
        int end,sum=0;
        if(i<fsize)
            end = i+1;
        else

```

```

        end = fsize;
        for(j=0;j<end;j++)
        {
            sum+=input[i-j]*filter[j];
        }
        output[i]=sum;
    }

}

int main() {
    int *input, *filter, *output1,*output2,*output3;
    int hf[] = {3, 4, 5, 4, 3};
    int fsize = sizeof(hf) / sizeof(*hf);

    if (fsize % 2 == 0) {
        printf("Error: Filter size (%d) is even.\n", fsize);
        exit(1);
    }
    cudaMalloc(&input, N * sizeof(int));
    cudaMalloc(&filter, fsize * sizeof(int));
    cudaMalloc(&output1, N * sizeof(int));
    cudaMalloc(&output2, (N+fsize-1)*sizeof(int));
    cudaMalloc(&output3, N*sizeof(int));
    cudaMemcpy(filter, hf, fsize * sizeof(int), cudaMemcpyHostToDevice);

    int nblocks = (N + BLOCKSIZE - 1) / BLOCKSIZE;
    init<<<nblocks, BLOCKSIZE>>>(input);

    printf("First version\n");
    convolutionV1<<<nblocks, BLOCKSIZE>>>(input, filter, output1, fsize);

    print<<<1, 1>>>(output1);
    cudaDeviceSynchronize();
    printf("\n");
    printf("Second Version\n");
    convolutionV2<<<nblocks, BLOCKSIZE>>>(input, filter, output2, fsize);

    print<<<1, 1>>>(output2);
    cudaDeviceSynchronize();
    printf("\n");
    printf("Third Version\n");
    convolutionV3<<<nblocks, BLOCKSIZE>>>(input, filter, output3, fsize);

```

```

    print<<<1, 1>>>(output3);
    cudaDeviceSynchronize();
    printf("\n");

    return 0;
}

```

## Outputs -

```

budhwani1@B05S8-DL02: ~/gpu_programming/homeworks/csl7520
First version
22 38 57 76 95 114 133 152 171 190 209 228 247 266 285 304 323 342 361 380 399 418 437 456 475 494 513 532 551 570 589 608 627 646 665 684 703 722 741 760 779 798 817 836 8
55 874 893 912 931 950 969 988 1007 1026 1045 1064 1083 1102 1121 1140 1159 1178 1197 1216 1235 1254 1273 1292 1311 1330 1349 1368 1387 1406 1425 1444 1463 1482 1501 1520 1
539 1558 1577 1596 1615 1634 1653 1672 1691 1710 1729 1748 1767 1786 1805 1824 1843 1862 1578 1190

Second Version
0 4 13 26 42 58 74 90 106 122 138 154 170 186 202 218 234 250 266 282 298 314 330 346 362 378 394 410 426 442 458 474 490 506 522 538 554 570 586 602 618 634 650 666 682 69
8 714 730 746 762 778 794 810 826 842 858 874 890 906 922 938 954 970 986 1002 1018 1034 1050 1066 1082 1098 1114 1130 1146 1162 1178 1194 1210 1226 1242 1258 1274 1290 130
6 1322 1338 1354 1370 1386 1402 1418 1434 1450 1466 1482 1498 1514 1530 1546 1562

Third Version
3 10 22 38 57 76 95 114 133 152 171 190 209 228 247 266 285 304 323 342 361 380 399 418 437 456 475 494 513 532 551 570 589 608 627 646 665 684 703 722 741 760 779 798 817
836 855 874 893 912 931 950 969 988 1007 1026 1045 1064 1083 1102 1121 1140 1159 1178 1197 1216 1235 1254 1273 1292 1311 1330 1349 1368 1387 1406 1425 1444 1463 1482 1501 1
520 1539 1558 1577 1596 1615 1634 1653 1672 1691 1710 1729 1748 1767 1786 1805 1824 1843 1862

budhwani1@B05S8-DL02:~/gpu_programming/homeworks/csl7520$ ./conv
First version
22 38 57 76 95 114 133 152 171 190 209 228 247 266 285 304 323 342 361 380 399 418 437 456 475 494 513 532 551 570 589 608 627 646 665 684 703 722 741 760 779 798 817 836 8
55 874 893 912 931 950 969 988 1007 1026 1045 1064 1083 1102 1121 1140 1159 1178 1197 1216 1235 1254 1273 1292 1311 1330 1349 1368 1387 1406 1425 1444 1463 1482 1501 1520 1
539 1558 1577 1596 1615 1634 1653 1672 1691 1710 1729 1748 1767 1786 1805 1824 1843 1862 1578 1190

Second Version
0 4 13 26 42 58 74 90 106 122 138 154 170 186 202 218 234 250 266 282 298 314 330 346 362 378 394 410 426 442 458 474 490 506 522 538 554 570 586 602 618 634 650 666 682 69
8 714 730 746 762 778 794 810 826 842 858 874 890 906 922 938 954 970 986 1002 1018 1034 1050 1066 1082 1098 1114 1130 1146 1162 1178 1194 1210 1226 1242 1258 1274 1290 130
6 1322 1338 1354 1370 1386 1402 1418 1434 1450 1466 1482 1498 1514 1530 1546 1562

Third Version
3 10 22 38 57 76 95 114 133 152 171 190 209 228 247 266 285 304 323 342 361 380 399 418 437 456 475 494 513 532 551 570 589 608 627 646 665 684 703 722 741 760 779 798 817
836 855 874 893 912 931 950 969 988 1007 1026 1045 1064 1083 1102 1121 1140 1159 1178 1197 1216 1235 1254 1273 1292 1311 1330 1349 1368 1387 1406 1425 1444 1463 1482 1501 1
520 1539 1558 1577 1596 1615 1634 1653 1672 1691 1710 1729 1748 1767 1786 1805 1824 1843 1862

budhwani1@B05S8-DL02:~/gpu_programming/homeworks/csl7520$ █

```

## Result:

Version 1 and 3 provide the same output of filters just a few elements more, and version 2 gives different outputs.