

Module 11 Assignment: Convolution Modification

Scott Wroten
11/13/2017
Introduction to GPU Programming

1. Modify assignment to take in 7X7 filter. (35%)

7X7 Mask Filter as gradient (100% one unit from center, 75% two units, 50% three units, and 25% four units):

```
cl_float mask[maskWidth][maskHeight] =  
{  
    { 0.00, 0.00, 0.25, 0.50, 0.25, 0.00, 0.00 },  
    { 0.00, 0.25, 0.50, 0.75, 0.50, 0.25, 0.00 },  
    { 0.25, 0.50, 0.75, 1.00, 0.75, 0.50, 0.25 },  
    { 0.50, 0.75, 1.00, 0.00, 1.00, 0.75, 0.50 },  
    { 0.25, 0.50, 0.75, 1.00, 0.75, 0.50, 0.25 },  
    { 0.00, 0.25, 0.50, 0.75, 0.50, 0.25, 0.00 },  
    { 0.00, 0.00, 0.25, 0.50, 0.25, 0.00, 0.00 },  
};
```

49X49 Input Signal:

```
Input Signal:  
1 8 9 8 0 1 2 3 2 0 3 0 2 8 9 7 8 2 7 1 1 1 7 3 6 5 3 1 8 1 6 7 4 1 3 6 5 9 0 4 1 4 5 1 9 3 9 0 7  
7 7 1 1 7 5 2 2 9 6 0 0 3 2 0 4 2 6 3 9 5 6 5 5 6 0 8 5 1 9 6 6 1 1 8 1 7 8 7 8 7 4 8 3 2 2 0 2 7  
4 4 3 8 3 8 9 8 3 7 2 8 7 3 3 8 8 6 9 3 4 8 7 5 6 4 3 7 7 0 0 0 9 7 3 6 1 0 5 4 8 3 7 6 3 1 4 3 1  
0 2 5 2 4 6 4 2 0 5 2 3 3 6 4 1 0 3 8 5 1 8 8 1 2 2 2 3 0 2 2 1 2 5 2 6 4 1 6 3 5 5 8 2 7 1 4 1 3  
9 7 9 6 9 1 1 5 7 0 3 4 1 3 1 4 1 9 0 9 3 2 2 5 9 1 8 3 4 8 8 2 6 9 7 7 7 3 1 9 7 8 9 8 2 0 9 6 0  
4 7 8 6 6 2 9 3 7 0 6 5 5 0 7 3 8 4 0 7 7 7 7 5 0 2 6 4 8 3 9 9 3 1 0 0 6 9 8 2 8 4 7 8 7 8 8 8 7  
8 9 4 0 5 4 6 6 8 4 7 0 7 3 5 5 4 5 8 9 3 7 8 8 4 7 0 2 8 1 8 3 1 8 6 4 8 3 1 4 8 1 7 1 9 1 5 7 0  
8 3 0 3 1 2 9 1 7 5 1 1 8 5 3 5 8 6 8 8 5 1 0 4 9 7 1 2 1 4 9 8 5 1 8 9 2 9 8 9 4 0 1 9 3 8 3 7 3  
2 1 7 8 0 5 8 8 3 4 5 4 5 2 3 2 6 2 0 0 8 2 2 0 3 9 2 0 3 3 9 5 9 7 0 8 3 3 5 0 2 3 6 8 6 7 2 2 6  
4 9 6 0 9 8 2 6 0 9 5 5 8 8 2 6 0 5 5 1 5 6 0 5 1 5 6 1 6 1 9 1 7 8 1 9 0 6 6 5 3 3 5 1 3 2 3 8 3  
5 8 3 1 9 6 5 2 3 5 4 6 3 0 4 8 9 3 1 4 6 0 8 4 1 5 0 2 7 0 4 5 0 6 1 4 2 5 5 7 8 4 0 3 8 8 0 7 1  
5 6 6 2 6 2 5 1 8 6 0 2 2 7 1 9 3 5 4 7 9 4 9 7 9 4 5 7 8 4 2 6 2 6 9 9 7 5 9 4 4 0 1 3 1 9 1 9 9  
1 5 1 5 8 6 4 4 2 4 2 0 4 2 2 2 0 6 3 5 0 9 2 2 9 0 4 7 9 0 0 6 6 9 5 8 8 4 2 7 1 6 3 4 0 2 7 5 5  
7 0 5 0 3 5 9 6 5 8 6 3 1 2 9 9 0 6 6 5 8 4 8 1 5 3 6 0 5 7 8 8 1 9 3 7 8 0 6 6 3 7 2 3 6 3 1 9 6  
1 2 4 9 4 3 1 2 2 0 5 1 5 0 5 8 1 3 5 9 6 2 4 5 4 7 2 5 9 7 8 7 2 6 5 5 5 5 6 0 5 6 9 8 0 9 2 1  
1 8 2 4 8 9 2 9 0 1 6 6 5 9 8 9 7 4 9 0 3 0 0 9 5 1 9 4 5 0 8 9 6 6 3 6 9 0 2 3 4 5 0 2 7 2 8 9 5  
5 6 0 7 4 2 5 1 3 6 1 6 9 1 7 4 6 4 2 9 6 3 5 9 6 2 3 1 6 8 7 8 2 6 1 4 7 6 3 0 1 6 5 7 6 8 8 1 1  
2 0 9 8 9 4 0 5 7 8 1 6 5 0 4 5 3 6 1 2 0 6 5 0 0 2 4 0 6 1 3 7 5 5 5 8 1 5 2 9 7 0 7 2 5 2 3 3 0  
7 2 7 3 9 6 8 0 2 3 1 7 1 8 5 0 5 2 9 8 6 2 2 3 5 9 7 3 3 4 0 0 3 8 3 6 3 5 1 0 7 9 4 1 1 5 2 0 9  
6 4 3 5 2 1 3 3 9 1 7 6 0 5 6 5 6 8 2 9 4 7 0 9 8 8 7 0 0 7 7 4 4 4 8 4 1 2 8 2 8 4 7 2 5 9 8 7  
1 8 7 1 5 8 9 6 7 4 3 7 6 0 8 8 5 8 2 4 7 5 6 3 4 2 8 3 0 7 6 9 0 0 0 7 7 8 2 0 1 3 9 6 9 8 3 1 3  
4 6 2 2 5 2 3 4 5 8 0 7 4 7 0 8 8 2 3 5 8 8 7 5 6 2 1 9 8 9 4 1 7 3 3 8 1 9 8 1 3 5 0 1 5 1 3 9 0  
2 5 6 0 3 1 9 9 3 4 0 1 5 9 7 0 3 5 4 0 9 8 8 0 3 7 6 5 8 5 2 5 5 7 1 0 3 6 8 5 1 7 6 7 6 2 2 2 8  
3 0 0 1 3 1 6 2 4 9 4 9 3 0 7 1 9 5 0 4 4 1 7 7 4 8 3 9 2 3 1 8 5 8 7 9 7 2 8 1 3 9 1 3 1 4 8 0 1  
2 9 1 6 3 9 7 9 3 2 8 1 0 5 4 3 4 1 2 4 1 2 8 6 9 5 8 7 3 8 9 6 3 3 9 5 1 2 7 0 1 0 1 0 5 9 3 2 5  
2 0 6 4 7 7 9 3 2 0 3 6 8 3 3 2 8 0 3 0 5 1 1 7 0 8 7 2 0 2 1 3 1 1 0 6 2 5 4 3 5 4 9 6 4 8 4 1  
1 0 5 0 4 6 9 4 1 7 0 2 9 4 1 2 3 1 1 5 6 4 6 8 5 3 6 0 6 3 6 9 8 1 3 9 7 0 0 9 1 8 0 0 1 8 0 7 7  
6 6 7 6 3 2 7 4 1 0 7 1 1 3 2 9 5 9 1 6 9 0 3 3 9 1 2 5 2 0 4 8 2 5 0 5 5 0 5 5 7 4 2 3 2 2 5 1  
8 1 5 1 8 9 6 0 3 6 2 2 3 0 6 7 7 1 8 1 6 8 9 6 9 4 7 8 0 1 3 5 8 5 2 0 3 2 5 8 0 4 2 5 8 7 0 8 7  
5 3 4 8 0 5 9 5 2 0 9 7 4 4 3 0 6 7 9 5 6 2 5 4 1 4 2 6 9 0 3 3 0 7 9 1 4 5 9 9 5 0 1 6 5 1 1 1 9  
7 8 1 9 8 2 3 9 5 7 0 6 7 8 0 3 4 6 8 9 6 4 8 4 2 6 3 7 7 4 6 8 1 6 1 3 4 2 9 6 1 2 9 0 0 7 2 0 4  
6 9 2 8 8 0 5 6 3 0 3 5 4 5 1 1 5 0 9 0 7 5 9 1 8 9 7 0 2 0 2 2 7 2 2 5 5 0 5 2 2 3 7 7 9 9 1 0 9  
1 3 0 4 0 0 7 3 5 5 8 1 6 9 9 3 8 0 1 8 1 2 0 9 8 8 8 8 2 7 2 3 5 1 6 5 5 6 3 5 2 5 0 2 2 5 0 7 8  
8 4 0 1 6 3 6 4 1 6 8 3 3 2 0 3 6 2 3 3 4 9 0 5 1 5 5 1 8 5 1 9 0 6 1 8 1 7 0 7 1 4 9 0 1 0 7 1 0  
9 4 1 4 8 9 6 2 2 7 5 9 6 5 4 7 1 4 5 8 0 9 0 5 8 6 8 8 2 9 4 4 5 1 7 0 4 2 9 6 5 1 3 9 3 3 6 6 0  
2 6 4 3 1 1 5 8 2 3 7 9 5 4 2 0 5 3 1 7 6 8 7 7 6 6 5 3 9 0 3 8 9 1 0 1 1 8 9 3 5 6 3 5 3 4 6 9 1  
7 0 6 9 9 8 8 8 2 8 7 5 2 1 9 9 5 3 4 2 1 3 2 4 0 0 8 0 4 5 2 5 0 5 1 7 2 0 5 8 3 6 3 4 5 0 8 3 8  
9 6 0 8 8 1 2 7 6 4 0 9 8 3 3 5 1 6 2 6 2 8 5 7 2 7 7 3 4 1 6 1 9 7 5 7 6 0 1 5 8 4 1 3 2 1 0 0 5  
5 6 7 8 9 9 5 5 4 9 8 2 1 9 1 5 9 2 5 9 1 6 5 3 4 2 3 6 8 2 0 3 3 4 2 4 6 6 1 9 9 5 3 2 9 5 5 3 0  
4 1 1 0 7 5 4 8 0 2 9 6 0 5 4 8 6 8 3 3 4 9 2 4 9 7 3 2 2 0 5 2 5 8 2 6 3 8 1 3 2 2 5 5 0 3 8 7 8  
3 8 7 8 2 3 4 1 7 6 3 5 3 8 8 3 4 2 8 8 9 8 1 0 0 2 1 0 7 9 4 8 4 5 9 7 3 3 2 0 8 9 6 6 4 4 8 3 9  
1 4 7 7 2 2 1 2 0 5 0 3 1 2 2 7 5 3 9 1 5 6 4 8 2 6 0 2 2 8 9 1 2 6 0 0 0 0 3 2 4 1 9 0 8 2 9 0 2  
2 9 7 7 8 5 6 5 3 7 7 2 5 7 0 8 4 4 2 0 2 6 8 8 2 4 8 8 5 9 6 6 7 4 5 7 0 6 3 5 2 2 5 8 5 3 5 4 8  
3 6 7 8 2 2 1 7 4 7 4 4 5 9 5 3 2 3 8 3 3 2 6 1 1 7 1 4 8 1 0 4 2 9 1 4 8 4 9 1 7 4 9 8 7 2 7 3 0  
3 3 3 3 8 5 6 4 0 4 7 4 2 6 5 7 5 4 7 3 4 3 4 5 5 1 0 1 4 7 8 6 6 8 8 1 1 3 7 5 5 9 8 1 3 9 3 2 8  
4 7 3 8 9 8 3 4 0 5 8 4 2 1 9 9 3 3 0 8 1 8 6 2 4 3 9 8 7 3 2 7 7 4 3 2 3 7 2 7 8 9 5 3 4 2 3 5  
1 8 5 3 0 6 3 4 5 7 3 2 1 7 3 3 2 6 1 4 5 2 8 3 4 2 2 3 3 3 7 7 2 3 6 1 8 0 3 7 4 3 3 7 0 9 2 1 1  
1 8 9 7 7 7 5 4 3 9 7 7 7 2 9 6 8 4 8 2 6 9 2 4 1 4 7 9 0 4 9 2 2 8 8 1 5 3 9 0 7 9 5 3 2 2 0 4  
3 2 9 1 8 7 5 4 1 8 1 9 4 1 8 1 8 4 8 0 3 2 8 4 2 6 9 5 5 7 0 6 0 5 8 3 6 2 0 4 9 6 5 7 0 4 0 2 7
```

Output Signal

94.75	103.5	96.75	77.5	77.5	80.25	88.75	71	80.25	68.75	76.25	77.5	77.5	80.25	83.75	76.25	91.25	84.25	81.25	100.75	88.75	91.75	83.75	75.75	69.25	77	100.75	81.25	81.75	95.75	95.75	87	80.75	91.75	84.5	69.25	65.5	79.75	68.25	91.75	91.75	96.25		
94.75	103.5	96.75	77.5	77.5	80.25	88.75	71	80.25	68.75	76.25	77.5	77.5	80.25	83.75	76.25	91.25	84.25	81.25	100.75	88.75	91.75	83.75	75.75	69.25	77	100.75	81.25	81.75	95.75	95.75	87	80.75	91.75	84.5	69.25	65.5	79.75	68.25	91.75	91.75	96.25		
94.75	103.5	96.75	77.5	77.5	80.25	88.75	71	80.25	68.75	76.25	77.5	77.5	80.25	83.75	76.25	91.25	84.25	81.25	100.75	88.75	91.75	83.75	75.75	69.25	77	100.75	81.25	81.75	95.75	95.75	87	80.75	91.75	84.5	69.25	65.5	79.75	68.25	91.75	91.75	96.25		
93.25	105	95	108.75	94.75	75	80.25	68.75	76.25	77.5	77.5	80.25	75.75	75.75	83.75	76.25	91.25	84.25	81.25	100.75	88.75	91.75	83.75	75.75	69.25	77	100.75	81.25	81.75	95.75	95.75	87	80.75	91.75	84.5	69.25	65.5	79.75	68.25	91.75	91.75	96.25		
94.25	100	113.75	91.25	81.75	66.25	73.75	83.75	75	80.25	61.25	70.25	75.75	80.25	66.25	68.25	75.75	85.25	95.75	92.75	81.75	77.75	77.25	90.75	74.75	93.75	92.75	80.75	83.75	84.5	84.75	97.75	85.75	78	90.75	80.75	104.5	80.75	94.5	93.25				
94.25	100	113.75	91.25	81.75	66.25	73.75	83.75	75	80.25	61.25	70.25	75.75	80.25	66.25	68.25	75.75	85.25	95.75	92.75	81.75	77.75	77.25	90.75	74.75	93.75	92.75	80.75	83.75	84.5	84.75	97.75	85.75	78	90.75	80.75	104.5	80.75	94.5	93.25				
94.25	100	113.75	91.25	81.75	66.25	73.75	83.75	75	80.25	61.25	70.25	75.75	80.25	66.25	68.25	75.75	85.25	95.75	92.75	81.75	77.75	77.25	90.75	74.75	93.75	92.75	80.75	83.75	84.5	84.75	97.75	85.75	78	90.75	80.75	104.5	80.75	94.5	93.25				
76.75	91.5	97.25	90.75	82.75	75.75	85.75	79.75	87.75	70.75	64.5	80.25	75.75	80.25	75.75	88.5	83.75	93.75	83.75	92.25	100.75	96.25	88.5	81.75	61.75	69.75	78.75	80.25	90.75	84.75	79.25	101.25	90.75	100.25	92.75	88.75	83.75	70.75	71.75	76.75	82.75	93.75	96.25	
90.25	101.5	85.75	92.75	87.75	88.75	90.25	77.75	75.75	75.75	75.75	64.5	80.25	75.75	80.25	75.75	88.5	83.75	93.75	83.75	92.25	100.75	96.25	88.5	81.75	61.75	69.75	78.75	80.25	90.75	84.75	79.25	101.25	90.75	100.25	92.75	88.75	83.75	70.75	71.75	76.75	82.75	93.75	96.25
90.25	101.5	85.75	92.75	87.75	88.75	90.25	77.75	75.75	75.75	75.75	64.5	80.25	75.75	80.25	75.75	88.5	83.75	93.75	83.75	92.25	100.75	96.25	88.5	81.75	61.75	69.75	78.75	80.25	90.75	84.75	79.25	101.25	90.75	100.25	92.75	88.75	83.75						

```
Execution Times:
  Execution Time - Trial 0: 572160 ns
```

2. Modify assignment to take in 49X49 filter. (35%)

49X49 Mask Filter as gradient (100% one unit from center, 75% two units, 50% three units, and 25% four units):

[illegible]

73X73 Input Signal:

```
49x49 Filter - Trial 1:
Input Signal:
7 6 1 3 5 1 6 3 0 2 1 4 9 8 8 2 3 2 7 8 0 6 0 5 0 0 9 1 1 8 8 5 1 7 6 0 7 8 4 8 2 9 9 0 5 7 5 9 8 8 1 1 1 4 0 1 2 8 8 1 6 0 0 0 5 0 3 3 9 4 7 2 2
1 6 0 4 4 6 6 8 8 5 9 1 8 2 2 0 4 3 6 1 9 8 2 4 3 4 1 0 1 1 1 8 5 7 6 8 1 8 9 8 4 0 9 3 7 8 4 5 6 6 7 0 3 2 0 0 7 5 2 9 4 0 5 8 2 7 5 2 6 3 1 9 8
8 4 4 3 2 1 3 7 8 8 0 4 5 3 2 0 5 2 6 9 5 0 5 7 1 4 4 2 3 5 4 7 7 1 1 6 8 3 0 8 2 1 0 6 5 4 4 2 8 5 8 9 8 2 2 3 0 2 1 8 8 8 9 7 4 6 0 1 4 3 6 5
4 3 8 9 7 7 5 4 7 7 3 6 8 3 1 1 3 4 0 2 8 3 6 5 3 3 6 3 0 4 5 1 5 9 8 9 0 1 8 0 9 9 8 5 4 5 3 6 5 5 7 2 3 0 5 0 3 4 2 9 2 3 7 8 1 7 0 9 8 0 3 3 5
9 2 1 2 9 5 6 1 5 0 8 3 9 3 0 7 2 9 6 8 4 5 5 9 9 2 1 9 0 2 8 5 9 7 9 9 6 3 5 1 3 5 1 2 8 3 2 6 1 5 8 4 8 6 7 0 2 2 6 3 4 6 5 6 8 7 6 2 8 3 1 6 7
5 6 7 8 9 1 1 5 6 2 1 4 1 3 4 6 3 3 5 1 0 9 6 6 3 0 5 4 7 2 1 1 0 8 8 3 8 8 7 8 1 4 2 2 0 1 7 0 3 7 9 7 8 9 8 7 6 3 1 6 7 5 9 1 3 8 4 7 9 7 7 6 4
8 9 6 1 6 1 7 4 8 9 6 0 8 8 1 9 6 7 8 6 3 2 4 9 3 0 2 8 5 6 3 0 2 6 5 5 2 1 5 6 7 0 9 7 9 1 9 5 6 7 8 4 9 3 6 4 6 8 8 4 6 3 9 5 2 3 3 7 6 9 4 2 8
3 3 0 6 1 1 9 1 5 8 4 5 0 6 1 3 3 8 3 8 1 0 4 9 1 1 9 6 5 1 4 0 0 0 8 1 3 2 2 6 6 2 2 3 3 6 7 8 5 3 4 5 0 7 3 1 1 4 6 9 7 0 5 6 8 4 1 3 1 5 3 1 4 2
1 2 7 6 8 3 3 0 8 8 1 3 9 2 2 8 2 3 5 3 3 4 7 9 1 4 8 0 6 0 4 1 0 1 1 2 5 2 3 0 0 7 7 9 1 8 8 5 4 9 7 2 9 6 2 0 3 5 9 3 0 0 3 6 1 7 8 3 1 5 5 7
5 1 2 5 6 0 0 9 0 5 0 8 2 1 2 7 7 6 0 4 1 9 4 9 0 0 8 4 3 5 5 7 7 8 5 0 7 2 1 9 0 0 5 9 9 1 1 6 9 4 4 2 8 4 3 1 4 2 6 1 2 1 5 7 5 5 3 9 0 5 1 6 1
0 5 9 9 4 4 8 7 9 6 0 0 6 5 3 8 1 6 5 8 9 5 2 8 4 8 8 5 4 0 7 5 0 6 3 7 3 2 7 0 9 6 7 6 9 7 8 8 1 4 8 1 6 9 2 9 0 0 2 1 5 6 3 6 8 7 1 2 8 6 6 7 4
6 2 9 9 8 8 8 7 1 3 5 9 5 5 7 0 0 9 9 4 7 2 3 3 9 0 6 7 2 2 4 6 3 1 0 6 4 3 8 8 3 5 8 8 3 0 7 4 3 5 3 5 8 1 3 1 6 8 6 0 4 7 9 2 6 3 5 3 1 5 8 8 7 7
0 6 1 8 5 9 0 3 9 1 5 1 9 8 9 2 1 7 6 3 1 7 3 6 9 1 9 1 7 5 3 5 0 5 4 4 1 5 4 1 7 1 7 3 6 9 1 5 5 0 0 9 7 8 7 7 8 1 6 4 3 1 4 3 4 2 7 0
0 3 6 1 7 2 7 8 2 7 3 5 8 1 1 4 7 9 6 7 6 8 2 1 3 9 6 6 9 0 2 2 7 5 0 5 4 1 7 1 4 4 9 4 5 7 6 8 4 0 0 1 5 0 1 8 6 9 0 6 2 1 9 2 8 1 1 7 6 4 0 7 8
7 7 8 5 7 4 0 5 5 8 6 0 5 6 7 7 2 5 7 8 3 4 1 0 3 6 4 2 2 6 5 5 4 5 1 9 8 8 3 3 4 3 9 2 5 1 7 4 9 6 3 4 3 0 4 1 5 4 7 0 4 7 3 5 6 2 4 7 6 6 6 9 3
0 0 1 7 3 7 6 3 0 2 1 3 1 0 2 8 3 7 3 2 9 5 7 5 4 6 9 6 5 6 7 1 1 4 1 7 1 1 3 0 3 0 1 6 1 2 6 6 9 5 3 3 6 5 5 6 8 5 0 5 7 5 6 8 2 3 3 6 5 8 4 4
6 8 9 5 3 0 5 8 4 4 1 1 0 9 5 7 7 9 9 6 5 6 5 4 0 5 8 9 3 1 1 2 4 3 0 7 2 8 9 9 5 3 7 7 3 5 5 9 6 0 6 2 4 8 2 3 3 7 7 4 2 5 6 1 0 3 7 5 9 4 3 9 0
0 9 2 1 0 7 0 5 0 0 1 7 3 6 2 4 5 2 8 3 8 1 0 1 2 3 5 3 9 3 6 1 6 9 0 8 6 6 8 4 1 2 3 7 6 1 2 8 9 8 3 7 1 2 4 9 8 3 0 5 6 1 3 1 5 3 6 1 8 8 8 7 4
0 3 2 0 1 9 1 5 5 0 7 7 1 1 3 6 7 0 2 0 8 5 1 0 2 7 6 9 5 4 1 8 0 6 3 4 6 4 2 6 6 7 4 2 8 0 8 9 0 8 4 0 1 9 0 3 6 5 3 5 1 4 1 8 3 0 3 4 3 2 7 2 7
8 4 4 3 5 7 6 3 1 3 1 3 6 1 7 9 6 2 1 5 5 2 8 9 0 0 0 1 7 1 3 9 7 9 8 4 8 7 1 4 5 0 0 3 0 6 3 1 6 3 7 1 1 6 4 2 2 7 7 1 7 9 3 9 2 3 3 6 1 2 6 6 9
6 7 0 0 1 1 9 0 0 2 8 4 6 2 8 3 6 9 2 5 9 3 3 2 4 6 0 0 9 9 6 3 0 5 6 6 8 9 2 3 8 2 9 8 4 8 4 2 8 8 3 5 0 1 6 2 0 3 5 4 9 5 0 7 3 6 9 8 2 9 1 9 8 7
1 9 0 6 9 8 0 9 7 4 4 7 5 0 7 6 6 1 1 1 0 0 7 8 7 8 4 7 8 5 7 7 8 6 7 6 8 9 7 2 6 3 4 8 6 6 1 7 6 6 6 8 5 4 5 8 8 2 8 0 6 1 7 4 0 6 5 7 6 5 1 8 7
7 5 4 8 8 9 2 2 2 6 0 4 2 7 1 1 0 5 4 7 1 1 9 8 8 8 1 9 3 3 1 6 0 3 3 8 5 7 6 3 6 9 1 6 8 4 6 1 0 2 5 9 6 4 0 7 2 0 2 4 9 8 5 2 6 2 9 1 6 8 8 6 9
5 1 3 7 1 7 1 4 1 9 4 3 7 6 2 6 1 2 8 5 1 2 9 8 5 2 7 1 1 9 4 2 1 4 7 4 9 2 2 1 5 9 5 3 7 6 4 5 0 9 3 4 4 9 0 7 7 6 3 8 5 4 2 3 3 5 1 5 3 5 3 1
7 7 2 6 2 8 9 3 1 9 1 6 1 4 4 1 8 7 6 8 5 1 7 9 3 0 3 2 1 0 9 0 3 1 8 8 5 6 3 8 1 3 0 3 5 8 7 6 3 5 1 5 3 6 5 3 4 0 3 5 7 4 6 1 6 1 9 3 8 5 8 7 9
5 5 5 2 9 2 5 5 2 1 9 3 9 9 0 2 2 3 3 3 5 9 7 5 5 1 1 1 9 0 5 7 1 1 8 8 6 3 7 9 9 8 4 7 0 0 9 9 9 3 7 9 2 2 7 4 3 7 2 9 7 8 9 9 4 8 9 7 3 5 4 8 4
3 4 3 8 9 4 4 8 6 2 2 8 1 7 2 1 1 3 8 4 9 6 2 3 2 6 4 5 4 7 6 8 6 6 1 4 6 8 9 9 5 1 5 6 1 7 4 5 5 0 7 8 7 3 2 1 2 1 4 7 2 0 6 8 1 6 4 7 2 6 4 7 9
6 2 5 3 2 2 5 3 6 2 6 7 7 9 5 8 8 9 4 7 8 4 4 6 7 3 9 2 4 4 0 1 0 4 8 3 3 5 3 6 8 5 8 5 7 6 0 1 0 1 0 1 6 3 0 5 0 0 4 9 3 9 5 7 1 8 3 6 0 5 0
9 5 0 3 4 6 7 3 9 8 6 2 0 0 7 7 6 7 8 3 5 7 7 5 9 4 2 2 5 3 9 9 0 4 7 6 5 1 8 0 0 2 1 1 6 6 1 8 5 8 2 8 2 1 2 6 8 4 6 5 6 3 4 0 6 9 5 6 7 7 4 2 7 3
2 0 5 2 8 3 9 5 9 9 2 2 4 2 9 2 6 0 9 0 0 3 3 6 9 5 8 4 4 6 3 7 0 6 5 6 9 8 3 1 5 9 6 8 5 6 4 3 4 7 3 2 5 0 5 0 5 5 4 5 2 6 2 1 9 7 1 5 6 8 6 7 9 8
1 8 0 7 5 6 1 7 5 2 4 0 8 3 1 5 0 7 9 8 5 3 7 4 6 6 4 5 1 5 7 8 7 0 5 9 7 3 4 9 1 2 5 4 9 9 4 7 8 2 8 7 7 7 0 1 0 4 2 0 6 2 5 7 5 5 3 0 8 6 1 0 1 9
7 7 0 6 4 6 1 2 8 7 2 7 0 3 9 0 1 4 0 1 9 5 1 2 2 0 8 5 9 9 1 3 0 3 1 5 2 7 8 3 6 7 3 2 9 6 5 5 6 4 6 7 2 1 8 8 7 9 9 9 9 7 3 3 8 9 7 9 4 4 6 8 8
5 2 2 9 6 3 7 2 8 0 1 1 0 7 0 7 5 6 1 9 3 3 7 1 5 4 1 4 7 7 2 1 1 0 2 5 2 1 7 0 8 7 3 2 8 3 8 0 5 6 0 5 3 5 2 9 8 3 4 0 9 6 4 5 6 8 9 4 9 5 1 8 6
0 3 1 5 6 4 3 3 9 8 6 2 8 1 3 9 5 0 8 1 0 5 9 3 6 8 2 8 7 3 4 2 0 9 0 6 1 5 4 7 7 5 5 1 3 3 5 8 6 7 6 9 8 6 9 1 3 7 5 2 6 5 2 9 7 5 2 2 5 0 1 8 5
6 4 8 3 0 1 0 3 6 2 8 7 6 7 5 4 8 3 3 3 8 5 4 2 3 5 1 0 8 5 1 3 6 2 5 1 7 8 7 8 8 2 6 9 0 9 2 1 0 1 3 1 2 0 2 0 5 8 9 0 0 9 8 7 2 8 8 8 2 2 9 8 7
5 2 8 7 2 6 3 2 4 4 4 3 9 6 0 2 1 9 2 0 6 0 8 8 4 3 5 7 3 9 6 2 7 6 6 8 9 7 2 7 5 8 5 3 9 2 0 8 8 9 7 5 1 3 5 5 0 0 0 3 8 9 5 6 4 1 9 0 7 8 8 2
1 2 5 2 4 8 0 1 1 4 9 3 1 6 9 3 0 6 2 0 7 3 5 0 6 7 6 4 8 5 9 7 3 4 9 3 4 1 3 8 0 3 3 6 0 9 2 7 6 0 1 9 6 0 7 5 0 0 3 3 6 9 1 9 7 1 6 4 9 6 3 5 0 0
0 3 6 6 8 9 4 6 1 5 9 2 5 0 1 7 9 0 3 5 5 0 3 6 1 1 0 9 7 3 6 3 5 7 0 2 0 3 4 6 9 2 9 7 3 8 9 9 4 7 9 8 6 3 2 1 2 5 8 0 4 7 9 2 9 2 1 2 5 6 9 7 6
8 1 4 8 4 1 9 6 0 9 8 4 8 2 7 7 1 4 6 5 6 9 8 9 0 0 6 9 3 2 0 6 8 4 2 6 6 1 6 6 9 4 7 3 1 1 8 3 0 7 2 8 9 4 7 4 6 7 8 2 5 0 2 8 2 9 9 2 8 7 4 7 1
2 0 3 9 5 5 6 2 8 4 2 4 3 3 2 4 3 9 8 1 8 9 9 0 6 8 8 8 2 4 8 9 0 3 0 1 0 3 5 4 3 2 9 4 4 9 9 5 4 7 0 9 2 2 4 9 9 3 8 2 0 3 1 5 9 6 4 7 7 9 6 8 1
1 2 0 7 3 6 5 9 8 5 3 0 2 4 8 4 3 0 3 6 9 0 1 0 2 6 7 2 6 9 4 7 4 6 8 0 9 2 7 7 6 9 2 9 7 6 7 1 3 6 9 6 8 2 3 2 1 5 0 5 7 1 0 5 6 1 1 5 0 7 9 8 6
7 3 5 7 3 5 6 2 4 5 0 8 9 5 1 8 1 4 0 7 2 4 1 3 0 7 2 4 8 2 7 4 2 8 2 1 4 3 8 8 7 2 4 7 5 5 6 9 1 5 5 9 8 0 2 7 7 7 7 5 9 5 6 6 7 2 0 1 6 4 6 7 0
9 9 2 7 2 3 0 2 4 7 7 0 1 3 7 9 6 5 6 7 6 7 3 2 0 1 6 1 0 9 2 4 0 2 0 0 9 8 9 2 3 8 5 8 5 5 6 1 2 4 4 3 0 4 7 2 1 4 3 1 1 2 3 2 0 6 8 8 9 9 8 6 8
9 6 5 4 3 8 5 4 9 8 3 0 3 5 0 3 0 0 4 4 2 3 4 7 7 8 8 1 9 0 3 0 0 9 3 8 8 0 2 8 3 2 2 0 9 4 9 4 2 6 6 3 0 2 9 8 1 5 8 6 9 4 9 2 0 3 8 2 6 2 9 8 3
5 0 3 1 7 7 4 9 8 3 4 3 4 8 6 3 4 3 1 5 6 9 4 5 4 3 2 3 5 8 4 3 1 2 3 4 5 4 4 0 5 5 4 5 7 2 3 9 8 6 7 2 8 3 6 2 5 3 4 3 4 2 2 3 3 1 0 6 7 1 3 7 4
6 2 8 6 2 2 9 6 1 0 0 0 3 6 2 8 2 2 7 6 9 6 9 5 6 8 8 7 9 9 0 4 2 8 8 5 7 8 1 1 7 9 8 6 3 1 3 5 4 4 9 2 5 7 5 7 7 8 2 8 7 3 1 3 0 3 5 0 6 4 8 0 9 6 9
0 2 7 2 4 0 9 0 6 1 6 8 5 0 8 7 2 5 8 8 9 2 1 3 2 4 9 6 9 4 3 3 7 9 1 7 0 9 9 0 6 3 5 7 4 2 8 1 0 5 1 7 1 3 9 5 9 4 4 3 6 9 4 8 5 3 0 5 4 4 6 5 1
5 8 2 9 1 3 4 7 1 7 2 3 2 8 8 0 6 6 9 9 8 4 7 6 1 5 5 0 9 1 4 9 7 8 9 7 0 1 8 8 5 6 4 9 0 9 8 3 7 4 1 4 8 5 5 2 1 7 3 6 7 9 7 0 0 7 2 1 5 4 2 2 6 7 2
9 4 8 0 1 9 2 0 7 8 2 8 7 2 3 0 2 6 8 2 3 3 7 8 6 1 5 3 3 4 3 4 4 9 8 7 1 7 8 5 6 2 2 8 5 5 1 8 4 7 7 7 1 2 7 3 9 2 6 0 7 4 4 4 6 4 5 2 7 8 5 3 9
5 5 3 7 9 1 5 1 4 9 1 5 3 6 2 1 8 6 6 9 0 6 9 1 6 2 2 8 3 5 1 6 5 9 9 9 8 1 5 3 6 5 2 5 7 1 9 2 1 5 8 4 4 3 6 8 0 9 8 6 1 8 7 4 7 3 6 7 8 0 8 8 6
6 9 1 4 3 1 7 8 8 5 9 9 6 8 1 7 9 5 0 7 4 5 6 8 1 8 1 8 9 8 3 0 3 6 6 7 1 2 8 2 5 1 6 5 4 5 3 1 8 2 5 2 4 6 1 0 1 7 3 3 1 2 0 8 9 1 4 7 4 3 5 7 8
3 7 6 0 8 5 5 6 3 7 6 1 2 2 1 6 8 2 4 2 9 3 7 2 5 8 6 5 8 0 7 0 9 6 0 8 2 9 2 8 3 7 6 9 8 2 9 7 3 0 8 2 4 9 7 0 8 5 1 8 5 5 6 4 4 2 8 2 8 3 2 4
3 5 0 3 5 4 5 8 8 1 2 2 7 7 6 6 6 6 0 3 3 9 6 5 6 2 7 0 6 7 3 2 8 2 2 4 1 6 4 3 7 4 9 9 3 3 1 4 2 4 0 2 8 3 3 7 7 0 6 9 5 2 4 7 6 0 7 3 6 0 0 6 7
8 8 3 8 9 8 4 1 7 5 3 5 3 2 1 5 4 6 9 2 0 6 1 6 9 3 5 0 7 3 4 7 7 6 1 0 2 0 3 3 6 4 5 9 1 2 9 3 2 8 0 9 1 5 4 4 6 1 3 8 7 7 4 4 5 5 3 8 1 8 1 6 6
4 4 9 1 8 5 7 3 3 3 1 5 2 5 4 2 6 4 8 6 0 7 9 9 0 3 8 9 5 6 4 9 8 9 5 6 2 2 7 7 8 3 2 4 4 9 8 0 8 1 0 9 9 7 7 2 5 3 0 3 8 2 5 2 9 0 0 4 1 4 0 5 9
3 5 8 8 8 0 4 6 0 3 0 6 1 0 1 5 8 8 6 3 3 4 5 3 2 0 9 3 0 6 2 5 1 4 0 5 9 1 2 2 5 8 7 1 7 5 2 5 9 0 8 6 6 2 4 6 0 3 1 9 7 9 2 0 4 5 1 9 4 0 5 5 5
3 4 9 1 5 6 6 0 2 6 9 0 0 2 0 1 1 8 0 1 8 6 4 1 9 1 8 0 2 6 6 2 1 2 7 9 7 7 9 3 3 7 3 4 7 9 5 1 5 1 2 2 8 2 5 8 8 9 8 7 7 7 9 6 3 3 8 6 1 2 4 7 6
9 3 9 7 4 2 6 4 2 2 5 5 4 3 7 6 6 2 8 5 4 7 0 3 7 6 4 9 2 8 3 5 1 1 2 8 9 5 3 3 9 4 6 4 0 5 1 1 8 3 7 9 3 9 9 2 6 7 7 8 5 2 8 3 3 9 1 2 3 5 1 4 9
5 5 1 3 7 2 1 8 1 2 8 0 0 1 3 0 2 0 7 4 4 2 9 1 5 8 6 0 3 3 9 7 6 0 3 5 0 0 6 8 2 7 1 8 9 0 1 7 6 4 6 7 3 1 3 3 4 0 2 2 0 4 7 5 4 2 1 3 5 3 1 5 1 9
7 8 6 6 3 4 9 2 6 2 5 7 9 2 3 6 3 4 6 5 1 0 5 4 2 2 9 4 0 7 1 2 4 3 4 2 9 9 8 0 2 1 7 3 1 9 2 9 8 0 0 0 0 5 7 3 4 6 7 1 2 3 8 5 2 3 7 0 4 3 9 2
2 3 9 9 9 7 5 8 3 4 3 5 9 2 3 3 2 6 2 1 8 0 8 1 2 7 9 6 4 0 3 0 8 8 4 7 8 9 8 2 5 7 6 5 1 2 6 2 4 2 1 5 0 9 7 4 1 1 7 4 4 6 3 8 7 7 3 2 6 6 6 0 2
7 0 0 1 4 2 4 9 2 3 7 5 0 1 7 9 3 2 7 6 6 2 8 7 1 8 7 0 9 6 9 4 9 1 6 6 0 6 6 6 4 3 0 5 3 7 4 2 5 0 6 9 6 6 6 9 9 8 3 4 5 4 4 3 2 8 2 9 7 0 6 6 0
4 2 0 9 2 2 7 2 7 7 8 9 0 3 6 9 3 6 4 7 1 6 9 8 8 9 3 1 7 2 3 1 6 2 3 4 8 1 1 1 1 3 5 2 2 1 6 0 2 3 0 1 6 2 5 0 7 1 8 1 0 7 8 7 8 9 5 1 2 3 3 1 7
9 1 9 1 8 2 1 7 4 5 9 1 0 9 6 2 7 8 6 5 4 1 6 5 2 8 6 4 3 2 5 7 1 2 5 6 3 0 6 2 6 4 1 7 0 0 1 9 0 1 3 3 9 8 4 6 4 4 3 8 4 3 8 3 2 4 7 8 9 0 1 2 1
0 9 2 7 1 1 9 6 1 6 3 9 7 6 1 3 8 2 6 6 2 7 5 8 2 0 7 0 6 9 3 0 8 8 0 0 4 7 8 3 1 0 1 2 8 4 2 6 0 0 1 1 3 1 5 2 4 7 1 5 2 8 7 3 5 5 8 1 2 6 8 8 3 9 2
0 8 5 1 2 3 5 1 7 9 7 1 6 2 9 9 2 1 1 9 2 3 6 1 9 3 7 6 8 0 2 1 5 4 3 9 2 6 2 1 3 0 5 0 2 4 6 9 5 0 7 6 5 1 9 7 4 7 5 2 8 7 3 4 1 0 2 8 0 0 4 4 6
1 8 1 4 8 4 6 8 1 7 1 6 6 0 2 7 1 1 8 2 0 4 3 6 7 3 0 1 8 8 2 7 7 5 0 5 5 8 3 2 8 0 0 0 3 2 6 1 0 4 8 9 1 5 8 0 3 7 1 3 1 8 1 5 0 7 8 3 4 9 1 3 5 6
2 1 0 5 9 9 6 1 8 8 7 6 1 3 4 3 7 7 8 5 3 6 9 8 7 5 2 4 5 7 3 9 9 6 3 2 3 2 8 4 8 4 2 8 5 2 7 9 1 1 2 2 6 5 1 9 1 9 5 9 2 1 0 6 2 1 0 2 7 3 9 8 6
7 1 8 4 8 2 4 9 3 0 8 8 1 5 9 7 3 9 6 4 3 0 2 6 9 6 0 3 1 3 1 7 2 5 3 0 7 7 9 5 1 0 5 4 9 1 5 8 0 6 2 5 2 4 5 7 2 9 8 7 4 4 9 2 1 0 6 4 2 8 3 1 0
```


49X49 Output Signal (removed 24 columns and rows from each side of input signal to get output signal that trims resulting elements where filter referenced indices outside of input signal):

mapa.html

Execution Time (ns):

```
49x49 Filter - Execution Times:
  Execution Time - Trial 0: 22700000 ns
```

3. Output timing or other metrics for comparison of different datasets (10%).

Ran 3 trials for 7x7 Filter to get execution time for each Input Signal being filtered to get resulting Output Signal:

```
Execution Times:
  Execution Time - Trial 0: 572160 ns
  Execution Time - Trial 1: 590960 ns
  Execution Time - Trial 2: 582000 ns
```

Ran 3 trials for 49x49 Filter to get execution time for each Input Signal being filtered to get resulting Output Signal:

```
49x49 Filter - Execution Times:
Execution Time - Trial 0: 22700000 ns
Execution Time - Trial 1: 22721680 ns
Execution Time - Trial 2: 22695600 ns
```

[illegible]

[illegible]

[illegible]

```
// Function to check and handle OpenCL errors
```

```
inline void
```

```
checkErr(cl_int err, const char * name)
```

```
{
    if (err != CL_SUCCESS) {
        std::cerr << "ERROR: " << name << " (" << err << ")" << std::endl;
        exit(EXIT_FAILURE);
    }
}
```

```
void CL_CALLBACK contextCallback(
```

```
const char * errInfo,
```

```
const void * private_info,
```

```
size_t cb,
```

```
void * user data)
```

```
{
    std::cout << "Error occurred during context use: " << errInfo << std::endl;
    // should really perform any cleanup and so on at this point
    // but for simplicity just exit.
    exit(1);
}
```

```
cl_ulong RunKernelForInputSignalWithSevenBySevenFilter()
```

```
{
    // Initialize Variables
    cl_uint index;
    cl_int errNum;
    cl_kernel kernel;
    cl_mem maskBuffer;
    cl_uint numDevices;
    cl_program program;
    size_t totalLength;
    cl_uint numPlatforms;
    cl_ulong stopTime = 0;
    cl_ulong startTime = 0;
    cl_command_queue queue;
    cl_ulong kernelTime = 0;
    cl_mem inputSignalBuffer;
    cl_context context = NULL;
    cl_mem outputSignalBuffer;
    cl_event startStopEvent = 0;
    cl_platform_id * platformIDs;
```

```

cl_device_id *deviceIDs = NULL;
cl_float inputSignal[inputSignalWidth][inputSignalHeight];
cl_float outputSignal[outputSignalWidth][outputSignalHeight];

// Initialize Input Signal to Random Values
for (int i = 0; i < inputSignalHeight; i++)
{
    for (int j = 0; j < inputSignalWidth; j++)
    {
        inputSignal[i][j] = rand() % 10;
    }
}

// First, select an OpenCL platform to run on.
errNum = clGetPlatformIDs(0, NULL, &numPlatforms);
checkErr((errNum != CL_SUCCESS) ? errNum :
    (numPlatforms <= 0 ? -1 : CL_SUCCESS), "clGetPlatformIDs");
platformIDs = (cl_platform_id *)alloca(sizeof(cl_platform_id) * numPlatforms);
errNum = clGetPlatformIDs(numPlatforms, platformIDs, NULL);
checkErr((errNum != CL_SUCCESS) ? errNum :
    (numPlatforms <= 0 ? -1 : CL_SUCCESS), "clGetPlatformIDs");

// Iterate through the list of platforms until we find one that supports
// a CPU device, otherwise fail with an error.
for (index = 0; index < numPlatforms; index++)
{
    errNum = clGetDeviceIDs(platformIDs[index],
        CL_DEVICE_TYPE_GPU, 0, NULL, &numDevices);

    if (errNum != CL_SUCCESS && errNum != CL_DEVICE_NOT_FOUND)
    {
        checkErr(errNum, "clGetDeviceIDs");
    }
    else if (numDevices > 0)
    {
        deviceIDs = (cl_device_id *)alloca(sizeof(cl_device_id) * numDevices);
        errNum = clGetDeviceIDs(platformIDs[index],
            CL_DEVICE_TYPE_GPU, numDevices, &deviceIDs[0], NULL);
        checkErr(errNum, "clGetDeviceIDs");

        break;
    }
}

// Next, create an OpenCL context on the selected platform.
cl_context_properties contextProperties[] =
{
    CL_CONTEXT_PLATFORM,
    (cl_context_properties)platformIDs[index],
    0
};

context = clCreateContext(contextProperties, numDevices,
    deviceIDs, &contextCallback, NULL, &errNum);
checkErr(errNum, "clCreateContext");

std::ifstream srcFile("Convolution.cl");
checkErr(srcFile.is_open() ? CL_SUCCESS : -1, "reading Convolution.cl");

std::string srcProg(std::istreambuf_iterator<char>(srcFile),
    (std::istreambuf_iterator<char>()));

const char *src = srcProg.c_str();
totalLength = srcProg.length();

```

```

// Create program from source
program = clCreateProgramWithSource(context, 1, &src, &totalLength, &errNum);
checkErr(errNum, "clCreateProgramWithSource");

// Build program
errNum = clBuildProgram(program, numDevices, deviceIDs, NULL, NULL, NULL);
if (errNum != CL_SUCCESS)
{
    // Determine the reason for the error
    char buildLog[16384];
    clGetProgramBuildInfo(program, deviceIDs[0], CL_PROGRAM_BUILD_LOG,
        sizeof(buildLog), buildLog, NULL);

    std::cerr << "Error in kernel: " << std::endl;
    std::cerr << buildLog;

    checkErr(errNum, "clBuildProgram");
}

// Create kernel object
kernel = clCreateKernel(program, "convolve", &errNum);
checkErr(errNum, "clCreateKernel");

// Now allocate buffers
inputSignalBuffer = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
    sizeof(cl_float) * inputSignalHeight * inputSignalWidth,
    static_cast<void *>(inputSignal),
    &errNum);
checkErr(errNum, "clCreateBuffer(inputSignal)");

maskBuffer = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
    sizeof(cl_float) * sevenBySevenMaskHeight * sevenBySevenMaskWidth, static_cast<void *>
    (sevenBySevenMask), &errNum);
checkErr(errNum, "clCreateBuffer(mask)");

outputSignalBuffer = clCreateBuffer(context, CL_MEM_WRITE_ONLY,
    sizeof(cl_float) * outputSignalHeight * outputSignalWidth, NULL, &errNum);
checkErr(errNum, "clCreateBuffer(outputSignal)");

// Pick the first device and create command queue.
queue = clCreateCommandQueue(context, deviceIDs[0],
    CL_QUEUE_PROFILING_ENABLE, &errNum);
checkErr(errNum, "clCreateCommandQueue");

errNum = clSetKernelArg(kernel, 0, sizeof(cl_mem), &inputSignalBuffer);
errNum |= clSetKernelArg(kernel, 1, sizeof(cl_mem), &maskBuffer);
errNum |= clSetKernelArg(kernel, 2, sizeof(cl_mem), &outputSignalBuffer);
errNum |= clSetKernelArg(kernel, 3, sizeof(cl_uint), &inputSignalWidth);
errNum |= clSetKernelArg(kernel, 4, sizeof(cl_uint), &sevenBySevenMaskWidth);
checkErr(errNum, "clSetKernelArg");

const size_t globalWorkSize[1] = { outputSignalWidth * outputSignalHeight };
const size_t localWorkSize[1] = { 1 };

// Queue the kernel up for execution across the array
errNum = clEnqueueNDRangeKernel(queue, kernel, 1,
    NULL, globalWorkSize, localWorkSize, 0, NULL, &startStopEvent);
checkErr(errNum, "clEnqueueNDRangeKernel");

// Wait for Events to End to Get Start and Stop Time
errNum = clWaitForEvents(1, &startStopEvent);
checkErr(errNum, "clWaitForEvents");
errNum = clGetEventProfilingInfo(startStopEvent,
    CL_PROFILING_COMMAND_START, sizeof(cl_ulong), &startTime, NULL);
checkErr(errNum, "clGetEventProfilingInfo");

```

```

errNum = clGetEventProfilingInfo(startStopEvent,
    CL_PROFILING_COMMAND_END, sizeof(cl_ulong), &stopTime, NULL);
checkErr(errNum, "clGetEventProfilingInfo");

// Compute Kernel Time
kernelTime = (stopTime - startTime);

// Read output buffer
errNum = clEnqueueReadBuffer(queue, outputSignalBuffer, CL_TRUE,
    0, sizeof(cl_uint) * outputSignalWidth * outputSignalHeight,
    outputSignal, 0, NULL, NULL);
checkErr(errNum, "clEnqueueReadBuffer");

// Write Input Signal
std::cout << "Input Signal:" << std::endl;

// Write each element of input signal
for (int y = 0; y < inputSignalHeight; y++)
{
    // Add Spacing
    std::cout << " ";

    // Step through and output each element in signal
    for (int x = 0; x < inputSignalWidth; x++)
    {
        std::cout << inputSignal[x][y] << " ";
    }

    // Add new line for next row
    std::cout << std::endl;
}

// Add New Line for Spacing
std::cout << std::endl;

// Write Output Signal
std::cout << "Output Signal:" << std::endl;

// Output the result buffer
for (int y = 0; y < outputSignalHeight; y++)
{
    // Add Spacing
    std::cout << " ";

    // Step through and output each element in signal
    for (int x = 0; x < outputSignalWidth; x++)
    {
        std::cout << outputSignal[x][y] << " ";
    }

    // Add new line for next row
    std::cout << std::endl;
}

// Return total kernel time
return kernelTime;
}

cl_ulong RunKernelForInputSignalWithFortyNineByFortyNineFilter()
{
    // Initialize Variables
    cl_uint index;
    cl_int errNum;
    cl_kernel kernel;
    cl_mem maskBuffer;

```

```

cl_uint numDevices;
cl_program program;
size_t totalLength;
cl_uint numPlatforms;
cl_ulong stopTime = 0;
cl_ulong startTime = 0;
cl_command_queue queue;
cl_ulong kernelTime = 0;
cl_mem inputSignalBuffer;
cl_context context = NULL;
cl_mem outputSignalBuffer;
cl_event startStopEvent = 0;
cl_platform_id * platformIDs;
cl_device_id *deviceIDs = NULL;
cl_float inputSignal[inputSignalWidth][inputSignalHeight];
cl_float outputSignal[outputSignalWidth][outputSignalHeight];

// Initialize Input Signal to Random Values
for (int i = 0; i < inputSignalHeight; i++)
{
    for (int j = 0; j < inputSignalWidth; j++)
    {
        inputSignal[i][j] = rand() % 10;
    }
}

// First, select an OpenCL platform to run on.
errNum = clGetPlatformIDs(0, NULL, &numPlatforms);
checkErr((errNum != CL_SUCCESS) ? errNum :
    (numPlatforms <= 0 ? -1 : CL_SUCCESS), "clGetPlatformIDs");
platformIDs = (cl_platform_id *)alloca(sizeof(cl_platform_id) * numPlatforms);
errNum = clGetPlatformIDs(numPlatforms, platformIDs, NULL);
checkErr((errNum != CL_SUCCESS) ? errNum :
    (numPlatforms <= 0 ? -1 : CL_SUCCESS), "clGetPlatformIDs");

// Iterate through the list of platforms until we find one that supports
// a CPU device, otherwise fail with an error.
for (index = 0; index < numPlatforms; index++)
{
    errNum = clGetDeviceIDs(platformIDs[index],
        CL_DEVICE_TYPE_GPU, 0, NULL, &numDevices);

    if (errNum != CL_SUCCESS && errNum != CL_DEVICE_NOT_FOUND)
    {
        checkErr(errNum, "clGetDeviceIDs");
    }
    else if (numDevices > 0)
    {
        deviceIDs = (cl_device_id *)alloca(sizeof(cl_device_id) * numDevices);
        errNum = clGetDeviceIDs(platformIDs[index],
            CL_DEVICE_TYPE_GPU, numDevices, &deviceIDs[0], NULL);
        checkErr(errNum, "clGetDeviceIDs");

        break;
    }
}

// Next, create an OpenCL context on the selected platform.
cl_context_properties contextProperties[] =
{
    CL_CONTEXT_PLATFORM,
    (cl_context_properties)platformIDs[index],
    0
};

```



```

context = clCreateContext(contextProperties, numDevices,
    deviceIDs, &contextCallback, NULL, &errNum);
checkErr(errNum, "clCreateContext");

std::ifstream srcFile("Convolution.cl");
checkErr(srcFile.is_open() ? CL_SUCCESS : -1, "reading Convolution.cl");

std::string srcProg(std::istreambuf_iterator<char>(srcFile),
    (std::istreambuf_iterator<char>()));

const char *src = srcProg.c_str();
totalLength = srcProg.length();

// Create program from source
program = clCreateProgramWithSource(context, 1, &src, &totalLength, &errNum);
checkErr(errNum, "clCreateProgramWithSource");

// Build program
errNum = clBuildProgram(program, numDevices, deviceIDs, NULL, NULL, NULL);
if (errNum != CL_SUCCESS)
{
    // Determine the reason for the error
    char buildLog[16384];
    clGetProgramBuildInfo(program, deviceIDs[0], CL_PROGRAM_BUILD_LOG,
        sizeof(buildLog), buildLog, NULL);

    std::cerr << "Error in kernel: " << std::endl;
    std::cerr << buildLog;

    checkErr(errNum, "clBuildProgram");
}

// Create kernel object
kernel = clCreateKernel(program, "convolve", &errNum);
checkErr(errNum, "clCreateKernel");

// Now allocate buffers
inputSignalBuffer = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
    sizeof(cl_float) * inputSignalHeight * inputSignalWidth,
    static_cast<void *>(inputSignal),
    &errNum);
checkErr(errNum, "clCreateBuffer(inputSignal)");

maskBuffer = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
    sizeof(cl_float) * fortyNineByFortyNineMaskHeight * fortyNineByFortyNineMaskWidth,
    static_cast<void *>(fortyNineByFortyNineMask), &errNum);
checkErr(errNum, "clCreateBuffer(mask)");

outputSignalBuffer = clCreateBuffer(context, CL_MEM_WRITE_ONLY,
    sizeof(cl_float) * outputSignalHeight * outputSignalWidth, NULL, &errNum);
checkErr(errNum, "clCreateBuffer(outputSignal)");

// Pick the first device and create command queue.
queue = clCreateCommandQueue(context, deviceIDs[0],
    CL_QUEUE_PROFILING_ENABLE, &errNum);
checkErr(errNum, "clCreateCommandQueue");

errNum = clSetKernelArg(kernel, 0, sizeof(cl_mem), &inputSignalBuffer);
errNum |= clSetKernelArg(kernel, 1, sizeof(cl_mem), &maskBuffer);
errNum |= clSetKernelArg(kernel, 2, sizeof(cl_mem), &outputSignalBuffer);
errNum |= clSetKernelArg(kernel, 3, sizeof(cl_uint), &inputSignalWidth);
errNum |= clSetKernelArg(kernel, 4, sizeof(cl_uint), &fortyNineByFortyNineMaskWidth);
checkErr(errNum, "clSetKernelArg");

const size_t globalWorkSize[1] = { outputSignalWidth * outputSignalHeight };

```

```

const size_t localWorkSize[1] = { 1 };

// Queue the kernel up for execution across the array
errNum = clEnqueueNDRangeKernel(queue, kernel, 1,
    NULL, globalWorkSize, localWorkSize, 0, NULL, &startStopEvent);
checkErr(errNum, "clEnqueueNDRangeKernel");

// Wait for Events to End to Get Start and Stop Time
errNum = clWaitForEvents(1, &startStopEvent);
checkErr(errNum, "clWaitForEvents");
errNum = clGetEventProfilingInfo(startStopEvent,
    CL_PROFILING_COMMAND_START, sizeof(cl_ulong), &startTime, NULL);
checkErr(errNum, "clGetEventProfilingInfo");
errNum = clGetEventProfilingInfo(startStopEvent,
    CL_PROFILING_COMMAND_END, sizeof(cl_ulong), &stopTime, NULL);
checkErr(errNum, "clGetEventProfilingInfo");

// Compute Kernel Time
kernelTime = (stopTime - startTime);

// Read output buffer
errNum = clEnqueueReadBuffer(queue, outputSignalBuffer, CL_TRUE,
    0, sizeof(cl_uint) * outputSignalWidth * outputSignalHeight,
    outputSignal, 0, NULL, NULL);
checkErr(errNum, "clEnqueueReadBuffer");

// Write Input Signal
std::cout << "Input Signal:" << std::endl;

// Write each element of input signal
for (int y = 0; y < inputSignalHeight; y++)
{
    // Add Spacing
    std::cout << " ";

    // Step through and output each element in signal
    for (int x = 0; x < inputSignalWidth; x++)
    {
        std::cout << inputSignal[x][y] << " ";
    }

    // Add new line for next row
    std::cout << std::endl;
}

// Add New Line for Spacing
std::cout << std::endl;

// Write Output Signal
std::cout << "Output Signal:" << std::endl;

// Output the result buffer
for (int y = 0; y < outputSignalHeight; y++)
{
    // Add Spacing
    std::cout << " ";

    // Step through and output each element in signal
    for (int x = 0; x < outputSignalWidth; x++)
    {
        std::cout << outputSignal[x][y] << " ";
    }

    // Add new line for next row
    std::cout << std::endl;
}

```

```

}

// Return total kernel time
return kernelTime;
}

// main() for Convolution example
int main(int argc, char** argv)
{
    // Initialize Variables
    const int numberOfTrials = 3;
    cl_ulong executionTimesForEachSevenBySevenTrial[numberOfTrials];
    cl_ulong executionTimesForEachFortyNineByFortyNineTrial[numberOfTrials];

    // Execute Trial for 7x7 Filter
    for (int i = 0; i < numberOfTrials; i++)
    {
        // Write header for starting trial
        std::cout << "7x7 Filter - Trial " << i << ":" << std::endl;

        // Run Trial and Record total time to execute
        executionTimesForEachSevenBySevenTrial[i] = RunKernelForInputSignalWithSevenBySevenFilter();

        // Add Line for Spacing
        std::cout << std::endl;
    }

    // Output Execution Time
    std::cout << std::endl;
    std::cout << "7x7 Filter - Execution Times:" << std::endl;

    // Write out execution time for each
    for (int i = 0; i < numberOfTrials; i++)
    {
        std::cout << " Execution Time - Trial " << i << ": "
            << executionTimesForEachFortyNineByFortyNineTrial[i] << " ns" << std::endl;
    }

    // Output Execution Time
    std::cout << std::endl;

    // Execute Trial for 49x49 Filter
    for (int i = 0; i < numberOfTrials; i++)
    {
        // Write header for starting trial
        std::cout << "49x49 Filter - Trial " << i << ":" << std::endl;

        // Run Trial and Record total time to execute
        executionTimesForEachFortyNineByFortyNineTrial[i] =
            RunKernelForInputSignalWithFortyNineByFortyNineFilter();

        // Add Line for Spacing
        std::cout << std::endl;
    }

    // Output Execution Time
    std::cout << std::endl;
    std::cout << "49x49 Filter - Execution Times:" << std::endl;

    // Write out execution time for each
    for (int i = 0; i < numberOfTrials; i++)
    {
        std::cout << " Execution Time - Trial " << i << ": "
            << executionTimesForEachFortyNineByFortyNineTrial[i] << " ns" << std::endl;
    }
}

```

```
// Write to Console Success
std::cout << std::endl;
std::cout << "Executed program successfully." << std::endl;
std::cout << std::endl;

// Return Success
return 0;
}
```

```
//  
// Book:      OpenCL(R) Programming Guide  
// Authors:   Aaftab Munshi, Benedict Gaster, Timothy Mattson, James Fung, Dan Ginsburg  
// ISBN-10:   0-321-74964-2  
// ISBN-13:   978-0-321-74964-2  
// Publisher: Addison-Wesley Professional  
// URLs:      http://safari.informit.com/9780132488006/  
//            http://www.openclprogrammingguide.com  
//  
  
// Convolution.cl  
//  
// This is a simple kernel performing convolution.  
  
__kernel void convolve(const __global float * const input,  
    __constant float * const mask,  
    __global float * const output,  
    const int inputWidth,  
    const int maskWidth)  
{  
    // Initialize Variables  
    float sum = 0;  
    const int x = get_global_id(0);  
    const int y = get_global_id(1);  
  
    for (int r = 0; r < maskWidth; r++)  
    {  
        const int idxIntmp = (y + r) * inputWidth + x;  
  
        for (int c = 0; c < maskWidth; c++)  
        {  
            sum += mask[(r * maskWidth) + c] * input[idxIntmp + c];  
        }  
    }  
  
    output[y * get_global_size(0) + x] = sum;  
}
```