Project Topic

Hardware Acceleration of Edge Detection for Images using Canny Edge Detection Algorithm

Discussion on Implementation:

Canny Edge Detection Algorithm Steps

1) Convolution with the Gaussian Filter

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

2) Applying Sobel Filter to calculate the Vertical and Horizontal Edges
   (a) Applying the convolution masks for x and y directions

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

3) Finding the gradient strength and its direction

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

4) Hysteresis: The final step. Canny does use two thresholds (upper and lower):
   (a) If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge
   (b) If a pixel gradient value is below the lower threshold, then it is rejected.
   (c) If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

Design Choice

1) Out of the 4 steps in algorithm, first 2 steps require Convolution.
2) These 2 steps are performed using the coprocessor.
3) The coprocessor returns the two matrices Gx and Gy, where the processor will compute the part (b) of step and onwards.
4) The calculation of convolution is performed on hardware/co-processor to reduce the execution time by exploiting hardware-level parallelism and algorithm-level optimizations.
5) In this design, the Horizontal and Vertical Sobel filter are applied on same set of inputs and hence that can be performed in parallel, which helped in reducing the time complexity.

Coding Choice

State Machine has been implemented using the 3 always block design.

Execution Time

1) Maximum achievable clock frequency is 60.04 MHz
2) The execution time for convolution 32X32 image is 106446 cycles (for step 1 and 2). In these cycles, the image was read from the memory byte by byte. The first step is convolution with a 5X5 filter with 32X32 matrix. The second step is convolution of 28X28 matrix with a 3X3 filter.
3) Also, to transfer the images inside the sub-modules, each data byte was sent in a clock cycle. Hence, the clock cycles required are very high in this case. If DMA is used for burst transfers and burst transfers are also used for sending the data within the sub modules as well, the clock cycles required will be very less. Each convolution operation was performed in 1 clock cycle. So, if burst transfer is used, output will be obtained very fast.

Simulation Results

Below results are obtained from the Simulation of the SystemVerilog code for the Steps 1 and 2 of the algorithm. The results matched the values obtained from the MATLAB simulation of the same steps.

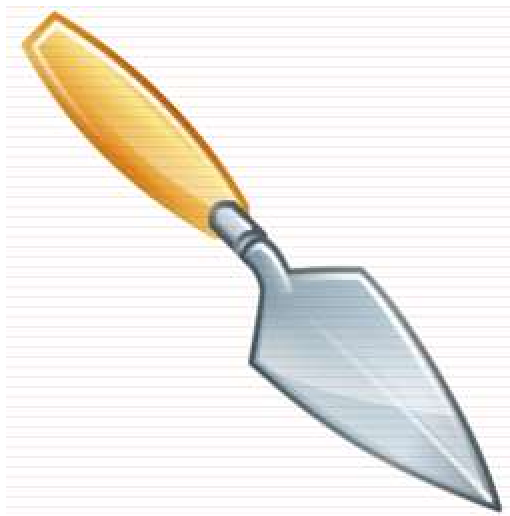
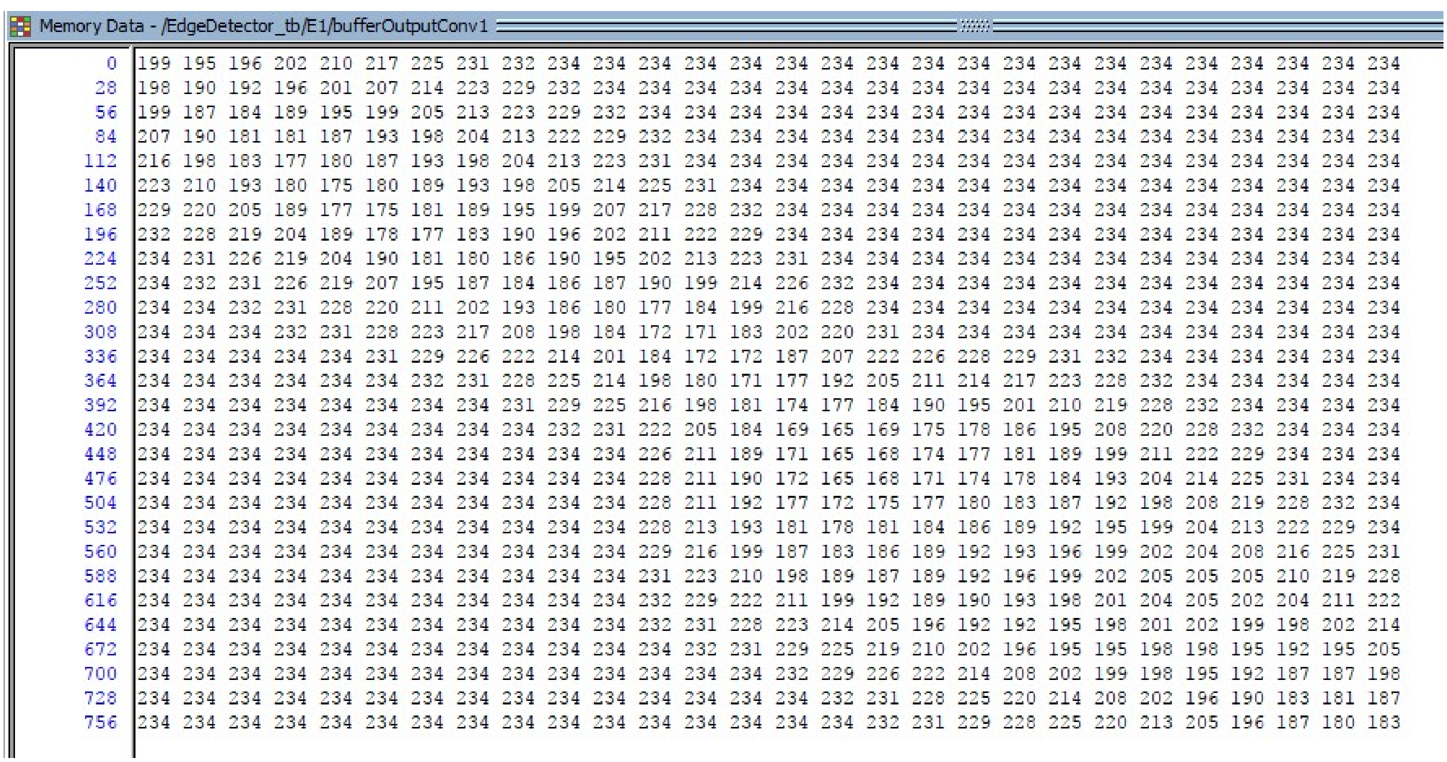


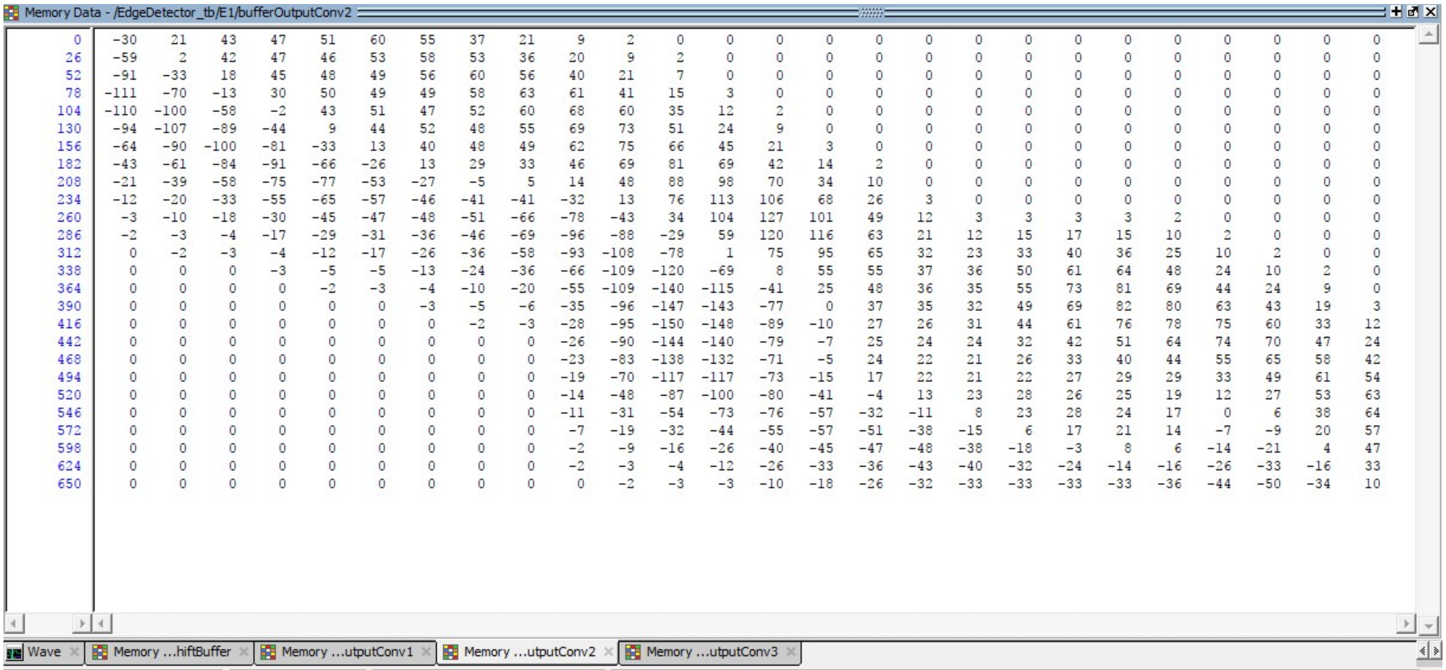
Figure 1: Input Image



Memory Data - /EdgeDetector_tb/E1/bufferOutputConv1

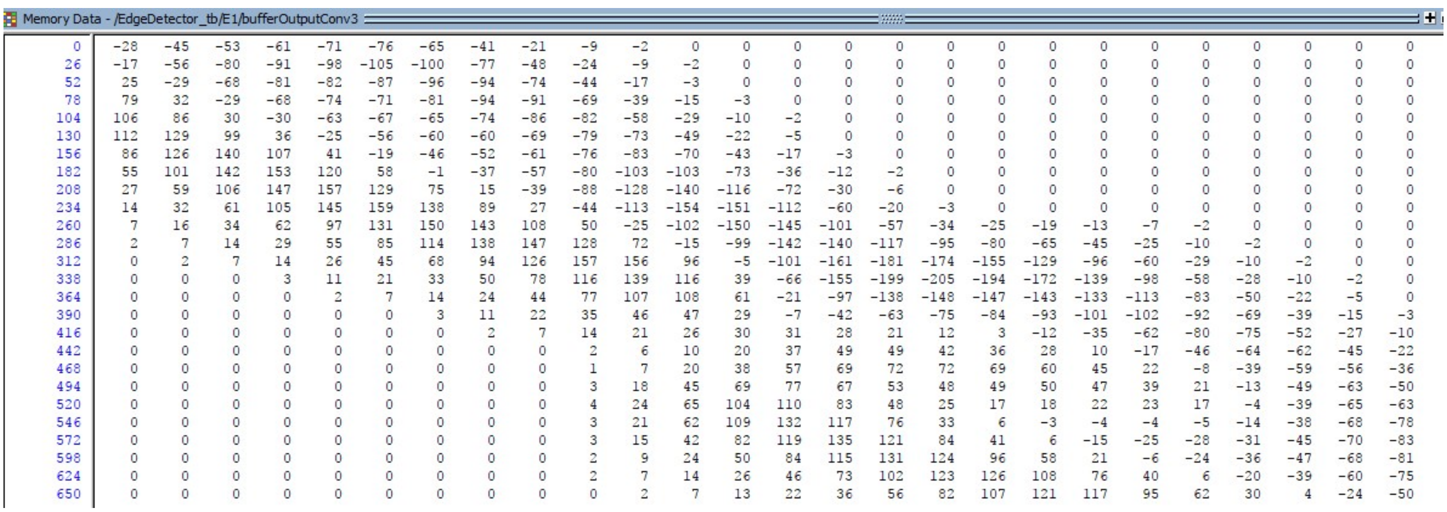
0	199	195	196	202	210	217	225	231	232	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234
28	198	190	192	196	201	207	214	223	229	232	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234
56	199	187	184	189	195	199	205	213	223	229	232	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234
84	207	190	181	181	187	193	198	204	213	222	229	232	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234
112	216	198	183	177	180	187	193	198	204	213	223	231	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234
140	223	210	193	180	175	180	189	193	198	205	214	225	231	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234
168	229	220	205	189	177	175	181	189	195	199	207	217	228	232	234	234	234	234	234	234	234	234	234	234	234	234	234	234
196	232	228	219	204	189	178	177	183	190	196	202	211	222	229	234	234	234	234	234	234	234	234	234	234	234	234	234	234
224	234	231	226	219	204	190	181	180	186	190	195	202	213	223	231	234	234	234	234	234	234	234	234	234	234	234	234	234
252	234	232	231	226	219	207	195	187	184	186	187	190	199	214	226	232	234	234	234	234	234	234	234	234	234	234	234	234
280	234	234	232	231	228	220	211	202	193	186	180	177	184	199	216	228	234	234	234	234	234	234	234	234	234	234	234	234
308	234	234	234	232	231	228	223	217	208	198	184	172	171	183	202	220	231	234	234	234	234	234	234	234	234	234	234	234
336	234	234	234	234	234	231	229	226	222	214	201	184	172	172	187	207	222	226	228	229	231	232	234	234	234	234	234	234
364	234	234	234	234	234	234	232	231	228	225	214	198	180	171	177	192	205	211	214	217	223	228	232	234	234	234	234	234
392	234	234	234	234	234	234	234	234	231	229	225	216	198	181	174	177	184	190	195	201	210	219	228	232	234	234	234	234
420	234	234	234	234	234	234	234	234	234	232	231	222	205	184	169	165	169	175	178	186	195	208	220	228	232	234	234	234
448	234	234	234	234	234	234	234	234	234	234	234	226	211	189	171	165	168	174	177	181	189	199	211	222	229	234	234	234
476	234	234	234	234	234	234	234	234	234	234	234	228	211	190	172	165	168	171	174	178	184	193	204	214	225	231	234	234
504	234	234	234	234	234	234	234	234	234	234	234	228	211	192	177	172	175	177	180	183	187	192	198	208	219	228	232	234
532	234	234	234	234	234	234	234	234	234	234	234	228	213	193	181	178	181	184	186	189	192	195	199	204	213	222	229	234
560	234	234	234	234	234	234	234	234	234	234	234	229	216	199	187	183	186	189	192	193	196	199	202	204	208	216	225	231
588	234	234	234	234	234	234	234	234	234	234	234	231	223	210	198	189	187	189	192	196	199	202	205	205	205	210	219	228
616	234	234	234	234	234	234	234	234	234	234	234	232	229	222	211	199	192	189	190	193	198	201	204	205	202	204	211	222
644	234	234	234	234	234	234	234	234	234	234	234	232	231	228	223	214	205	196	192	192	195	198	201	202	199	198	202	214
672	234	234	234	234	234	234	234	234	234	234	234	234	232	231	229	225	219	210	202	196	195	195	198	198	195	192	195	205
700	234	234	234	234	234	234	234	234	234	234	234	234	234	234	232	229	226	222	214	208	202	199	198	195	192	187	187	198
728	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	232	231	228	225	220	214	208	202	196	190	183	181	187
756	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	234	232	231	229	228	225	220	213	205	196	187	180	183

Figure 2: Output from Step 1 (performed in SystemVerilog)



0	-30	21	43	47	51	60	55	37	21	9	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	-59	2	42	47	46	53	58	53	36	20	9	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52	-91	-33	18	45	48	49	56	60	56	40	21	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
78	-111	-70	-13	30	50	49	49	58	63	61	41	15	3	0	0	0	0	0	0	0	0	0	0	0	0	0
104	-110	-100	-58	-2	43	51	47	52	60	68	60	35	12	2	0	0	0	0	0	0	0	0	0	0	0	0
130	-94	-107	-89	-44	9	44	52	48	55	69	73	51	24	9	0	0	0	0	0	0	0	0	0	0	0	0
156	-64	-90	-100	-81	-33	13	40	48	49	62	75	66	45	21	3	0	0	0	0	0	0	0	0	0	0	0
182	-43	-61	-84	-91	-66	-26	13	29	33	46	69	81	69	42	14	2	0	0	0	0	0	0	0	0	0	0
208	-21	-39	-58	-75	-77	-53	-27	-5	5	14	48	88	98	70	34	10	0	0	0	0	0	0	0	0	0	0
234	-12	-20	-33	-55	-65	-57	-46	-41	-41	-32	13	76	113	106	68	26	3	0	0	0	0	0	0	0	0	0
260	-3	-10	-18	-30	-45	-47	-48	-51	-66	-78	-43	34	104	127	101	49	12	3	3	3	3	2	0	0	0	0
286	-2	-3	-4	-17	-29	-31	-36	-46	-69	-96	-88	-29	59	120	116	63	21	12	15	17	15	10	2	0	0	0
312	0	-2	-3	-4	-12	-17	-26	-36	-58	-93	-108	-78	1	75	95	65	32	23	33	40	36	25	10	2	0	0
338	0	0	0	-3	-5	-5	-13	-24	-36	-66	-109	-120	-69	8	55	55	37	36	50	61	64	48	24	10	2	0
364	0	0	0	0	-2	-3	-4	-10	-20	-55	-109	-140	-115	-41	25	48	36	35	55	73	81	69	44	24	9	0
390	0	0	0	0	0	0	-3	-5	-6	-35	-96	-147	-143	-77	0	37	35	32	49	69	82	80	63	43	19	3
416	0	0	0	0	0	0	0	-2	-3	-28	-95	-150	-148	-89	-10	27	26	31	44	61	76	78	75	60	33	12
442	0	0	0	0	0	0	0	0	0	-26	-90	-144	-140	-79	-7	25	24	24	32	42	51	64	74	70	47	24
468	0	0	0	0	0	0	0	0	0	-23	-83	-138	-132	-71	-5	24	22	21	26	33	40	44	55	65	58	42
494	0	0	0	0	0	0	0	0	0	-19	-70	-117	-117	-73	-15	17	22	21	22	27	29	29	33	49	61	54
520	0	0	0	0	0	0	0	0	0	-14	-48	-87	-100	-80	-41	-4	13	23	28	26	25	19	12	27	53	63
546	0	0	0	0	0	0	0	0	0	-11	-31	-54	-73	-76	-57	-32	-11	8	23	28	24	17	0	6	38	64
572	0	0	0	0	0	0	0	0	0	-7	-19	-32	-44	-55	-57	-51	-38	-15	6	17	21	14	-7	-9	20	57
598	0	0	0	0	0	0	0	0	0	-2	-9	-16	-26	-40	-45	-47	-48	-38	-18	-3	8	6	-14	-21	4	47
624	0	0	0	0	0	0	0	0	0	-2	-3	-4	-12	-26	-33	-36	-43	-40	-32	-24	-14	-16	-26	-33	-16	33
650	0	0	0	0	0	0	0	0	0	0	-2	-3	-3	-10	-18	-26	-32	-33	-33	-33	-33	-36	-44	-50	-34	10

Figure 3: Output from step2 (convolution with Horizontal Edge Sobel Filter)



0	-28	-45	-53	-61	-71	-76	-65	-41	-21	-9	-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	-17	-56	-80	-91	-98	-105	-100	-77	-48	-24	-9	-2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52	25	-29	-68	-81	-82	-87	-96	-94	-74	-44	-17	-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
78	79	32	-29	-68	-74	-71	-81	-94	-91	-69	-39	-15	-3	0	0	0	0	0	0	0	0	0	0	0	0	0
104	106	86	30	-30	-63	-67	-65	-74	-86	-82	-58	-29	-10	-2	0	0	0	0	0	0	0	0	0	0	0	0
130	112	129	99	36	-25	-56	-60	-60	-69	-79	-73	-49	-22	-5	0	0	0	0	0	0	0	0	0	0	0	0
156	86	126	140	107	41	-19	-46	-52	-61	-76	-83	-70	-43	-17	-3	0	0	0	0	0	0	0	0	0	0	0
182	55	101	142	153	120	58	-1	-37	-57	-80	-103	-103	-73	-36	-12	-2	0	0	0	0	0	0	0	0	0	0
208	27	59	106	147	157	129	75	15	-39	-88	-128	-140	-116	-72	-30	-6	0	0	0	0	0	0	0	0	0	0
234	14	32	61	105	145	159	138	89	27	-44	-113	-154	-151	-112	-60	-20	-3	0	0	0	0	0	0	0	0	0
260	7	16	34	62	97	131	150	143	108	50	-25	-102	-150	-145	-101	-57	-34	-25	-19	-13	-7	-2	0	0	0	0
286	2	7	14	29	55	85	114	138	147	128	72	-15	-99	-142	-140	-117	-95	-80	-65	-45	-25	-10	-2	0	0	0
312	0	2	7	14	26	45	68	94	126	157	156	96	-5	-101	-161	-181	-174	-155	-129	-96	-60	-29	-10	-2	0	0
338	0	0	0	3	11	21	33	50	78	116	139	116	39	-66	-155	-199	-205	-194	-172	-139	-98	-58	-28	-10	-2	0
364	0	0	0	0	2	7	14	24	44	77	107	108	61	-21	-97	-138	-148	-147	-143	-133	-113	-83	-50	-22	-5	0
390	0	0	0	0	0	0	3	11	22	35	46	47	29	-7	-42	-63	-75	-84	-93	-101	-102	-92	-69	-39	-15	-3
416	0	0	0	0	0	0	0	2	7	14	21	26	30	31	28	21	12	3	-12	-35	-62	-80	-75	-52	-27	-10
442	0	0	0	0	0	0	0	0	0	2	6	10	20	37	49	49	42	36	28	10	-17	-46	-64	-62	-45	-22
468	0	0	0	0	0	0	0	0	0	1	7	20	38	57	69	72	72	69	60	45	22	-8	-39	-59	-56	-36
494	0	0	0	0	0	0	0	0	0	3	18	45	69	77	67	53	48	49	50	47	39	21	-13	-49	-63	-50
520	0	0	0	0	0	0	0	0	0	4	24	65	104	110	83	48	25	17	18	22	23	17	-4	-39	-65	-63
546	0	0	0	0	0	0	0	0	0	3	21	62	109	132	117	76	33	6	-3	-4	-4	-5	-14	-38	-68	-78
572	0	0	0	0	0	0	0	0	0	3	15	42	82	119	135	121	84	41	6	-15	-25	-28	-31	-45	-70	-83
598	0	0	0	0	0	0	0	0	0	2	9	24	50	84	115	131	124	96	58	21	-6	-24	-36	-47	-68	-81
624	0	0	0	0	0	0	0	0	0	2	7	14	26	46	73	102	123	126	108	76	40	6	-20	-39	-60	-75
650	0	0	0	0	0	0	0	0	0	0	2	7	13	22	36	56	82	107	121	117	95	62	30	4	-24	-50

Figure 4: Output from step2 (convolution with Vertical Edge Sobel Filter)

Comparison with the results obtained from MATLAB

-30	-52	-64	-73	-82
2	-33	-52	-57	-63
41	-2	-40	-54	-52
78	45	-6	-42	-49
93	89	48	-6	-41
80	106	97	52	-4
52	93	120	109	61
25	60	103	129	118
7	28	63	104	131

Figure 5: Output from MATLAB

Memory Data - /EdgeDetector_tb/E1/bufferOutputConv3

0	-28	-45	-53	-61	-71
26	-17	-56	-80	-91	-98
52	25	-29	-68	-81	-82
78	79	32	-29	-68	-74
104	106	86	30	-30	-63
130	112	129	99	36	-25
156	86	126	140	107	41
182	55	101	142	153	120
208	27	59	106	147	157

Figure 6: Output from RTL Implementation

Figure 5 and 6 are subset of output values taken from MATLAB and RTL implementation after the convolution with vertical Sobel edge detector filter.

The error in the values are propagated from the step 1 in which the division in case of Gaussian filter is done with 170.66 in RTL implementation instead of 159 in MATLAB. The division of 170.66 is implemented as:

$$x/512 + x/256 = 3x/512 = x/170.66.$$

In spite of the errors, the overall trend of the output is correct and the Figures 3 and 4 clearly indicate that the RTL implementation is able to detect the edges of the input image as precisely as the MATLAB implementation.