

# An FPGA Based Real-time Histogram Equalization Circuit for Image Enhancement

<sup>1</sup>Nitin Sachdeva, <sup>2</sup>Tarun Sachdeva

<sup>1</sup>YMCA University of Science & Technology, Faridabad, Haryana, India

<sup>2</sup>Infotel Business Sol. Ltd., Gurgaon, Haryana, India

E-mail: <sup>1</sup>n\_julka@rediffmail.com, <sup>2</sup>sachdeva\_tarun@rediffmail.com

## Abstract

This paper gives a design of real-time Histogram Equalization circuit for enhancement of images using Field Programmable Gate Arrays (FPGAs). This design makes use of counters in conjunction with a special decoder designed to compute the histogram statistics and equalization in parallel. The proposed system is fast, simple, and flexible with reasonable development cost. Timing simulation of the proposed system is verified using Altera software package (Quartus).

## Keywords

FPGA, Histogram Equalization.

## I. Introduction

In comparison with visible light images, thermal images suffer from low contrast and shortage of gray level representing the scene. Their images typically dominated by the background radiation at the average scene temperature leading to a narrow dynamic range. Therefore, a real-time thermal image enhancement is of a great interest for better image performance and quality. Histogram Equalization (HE) is an effective technique for image enhancement. It has been widely used in diverse fields that include radiology [1], thermal imaging [2], and speech recognition [3]. The essence of HE is to expand those pixels that have large gray quantity occurrences to adjacent gray level pixels and squeeze those pixels whose quantity gray occurrences are small, i.e., to enlarge the intensity difference among objects and background. For a digital image with gray levels in the range [0, 255] HE is computed as follows:

$$Y_k = \frac{1}{2^8} \sum_{j=0}^k n_j \quad (1)$$

where  $k = 0, 1, \dots, 255$ ,  $Y_k$  is the  $k_{th}$  pixel of the equalized image and  $n_k$  is the number of times the  $k_{th}$  pixel appears in the input image. The first step in realization of the HE is to compute Histogram Statistics (HS) of the captured image. A transformation look-up table that contains the modified new pixel gray values is then constructed. Each input pixel level is mapped to the corresponding output pixel level using this transformation table. The whole process must be performed during the frame display rate time, which necessitates the use of high computational speed for real-time processing. Research effort towards the application of FPGAs for HE computation has been reported. In [1], a combined Software/Hardware HE scheme based on FPGA is described. The HS and HE are realized in the traditional way using memory Read/Increment/Write sequences. The time required for the computation of HE is 50 ms, when the clock frequency is 30MHz and the image size is  $256 \times 256$  pixels. The approach of [1] has the disadvantages of being slow and sophisticated, which prevent its use in real-time processing. In [4], image enhancement is

performed by what is called gray rearranging that is based on HE. The enhancement method is implemented using both Digital Signal Processor (DSP) and FPGA hardware architectures in conjunction with software programmable environment. This hardware configuration is not capable of acquiring the image pixels in real-time; hence, only one pixel from three adjacent pixels is captured and processed. In [5, 6], a simple and straight forward method to realize the HE algorithm with reasonable cost and reduced development cycle is presented using FPGA. System analysis shows that the minimum time required for HS and HE computations for an image of size  $256 \times 256$  and clock frequency 50MHz is more than 13 ms. For images of higher dimensions (e.g.  $512 \times 512$ ), the timing restriction of this hardware architecture increases by a factor of 4 which prohibits its use in some critical applications. In thermal imaging, for example, the processing time should not exceed the detector integration time which is typically not more than 20 ms, assuming array size of  $512 \times 512$  and image display at 50 frames per second. In this paper, a novel scheme for the implementation of HE is proposed using FPGA. The technique relies on constructing the HS and HE in parallel using counters instead of memory arrays that are commonly used in such computations. The number of counters equals to the number of memory locations of the transformation array. The operation of the counters is controlled by a dedicated Switching Decoder (SD), which is specially designed to perform such a task. This parallel processing configuration has the advantage of reducing the processing time significantly, and will also make it possible to adapt the HE in real-time infrared thermal imagers to improve the quality of low contrast images.

## II. Proposed System

In our proposed scheme, each RAM location containing the number of the occurrences of the input pixel gray levels is replaced by the same number of bits counter. The significant advantage of using counters in this signal processing setting is that the number of clock cycles required to increment the location value is only one clock cycle compared to minimum three cycles in the RAM configuration presented in [5]. The development of the transformation look up table is performed in two steps. The first step is to compute the HS and HE, and the second one is to transfer the HE array from the counters to the look up table RAM. Hence, the total time ( $T_{total}$ ) required to construct the transformation look up table is composed of two parts, as indicated in the following equation:

$$T_{total} = \frac{m \times n}{f_{max}} + \frac{g}{f_{max}} \text{ seconds} \quad (2)$$

where  $m$  and  $n$  are the image vertical and horizontal sizes in pixels, respectively.  $f_{max}$  is the maximum possible frequency applied to the chip employed, and  $g$  is the number of the pixel gray level. Several conclusions can be drawn from Eq. (2).

- $T_{total}$  is drastically affected by the number of pixels ( $n \times m$ ) of the image.
- The number of the pixel gray levels ( $g$ ) partially contributes to  $T_{total}$ . Its percentage effect reduces when the image resolution increases.
- The maximum clock frequency of the FPGA chip used plays a major role in determining the processing speed,  $T_{total}$  inversely changes with  $f_{max}$ .

**III. The Circuit Description**

For demonstration purposes, the proposed HE circuit has been developed to compute and construct the HE transformation look up table for  $(256 \times 256)$  image with 256 pixel gray levels. The histogram circuit is mainly composed of four units: (16-bit) binary counters, (8 to 256) SD, multiplexers, and Timing and Control module. This is depicted in the schematic block diagram of Fig. 1. There are 256 counters as a replacement to the 256 locations look up table RAM. Each counter is responsible for two tasks. The first task is to calculate the number of the occurrences of the pixel gray levels that range from 0 to 255.

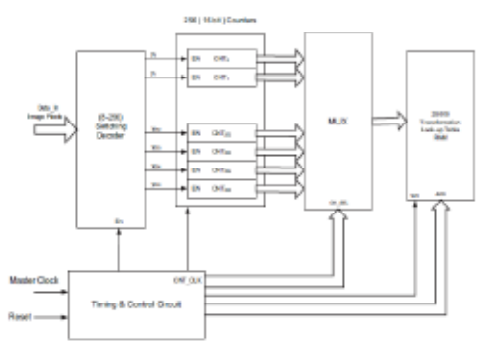


Fig. 1: Block diagram of the proposed histogram circuit.

The second task is to simultaneously accumulate the current gray level statistics with the statistics of all pixels whose values are less than the current pixel value. The results of this accumulation process are the final HE values. Since HE calculation is performed during the phase of computing HS, this adds an extra time saving. The implementation of such parallel steps is achieved by introducing the SD which assigns an array input vector for counters enable inputs which in turn activates the corresponding pixel gray level counter to be incremented. The SD is an (8 to 256) special purpose decoder which has an eight input lines and 256 outputs. Its function differs from the normal decoders that activate a unique output for each input combination. Specifically, SD has the additional property in that all the outputs associated with inputs having gray levels higher than the current input pixel value are forced to take on high state. Therefore, the main function of the SD is to enable the counter assigned to the current pixel and all the counters assigned to pixels whose values are greater than the current one so that the computation of HS and transformation function are performed at the same time. The SD will be explained in more details in the next section.

**IV. The Switching Decoder Circuit**

The concept of the conventional decoder circuit is to accept a set of inputs that represents a binary number and activates only a unique output that corresponds to that input number; all other outputs remain inactive. To further illustrate the concept of the SD function and for simplicity of presentation, let us

consider a (3 to 8) SD configuration as depicted in the truth table of Fig. 2.

No	Input			Output							
	a	b	c	$y_7$	$y_6$	$y_5$	$y_4$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1	0
2	0	1	0	1	1	1	1	1	1	0	0
3	0	1	1	1	1	1	1	1	0	0	0
4	1	0	0	1	1	1	1	0	0	0	0
5	1	0	1	1	1	1	0	0	0	0	0
6	1	1	0	1	1	0	0	0	0	0	0
7	1	1	1	1	0	0	0	0	0	0	0

Fig. 2: Truth table for (3 to 8) switching decoder.

The SD has three inputs a, b, and c (a is the most significant bit) and eight different outputs  $y_7$  to  $y_0$ . Unlike the conventional decoder, more than one output could be active (high state) at a time for the corresponding binary number representing the gray level of the image pixel at the input. The SD works as follows: When the input combination is 000, the  $y_0$  and all the outputs denoted by higher suffix are activated to high state. If the combination at the input is 001,  $y_1$  and all the next higher suffix outputs are high while the output denoted by the lower suffix ( $y_0$  in this case) remains at low state, when the combination at the input is 010,  $y_2$  and all the next higher suffix outputs are high while the outputs denoted by the lower suffix (in this case  $y_0$  and  $y_1$ ) remain low, and so forth. The easiest and straight forward way to implement the SD is to utilize the same size available for normal decoder. Therefore, the SD is simply composed of a normal decoder where every output is OR gated with the next output denoted by higher suffix, as shown in the schematic diagram of Fig. 3. The equivalent logic circuit is described by the following logic equations:

$$\begin{aligned}
 y_0 &= q_0 \\
 y_1 &= q_1 + y_0 \\
 y_2 &= q_2 + y_1 \\
 y_3 &= q_3 + y_2 \\
 y_4 &= q_4 + y_3 \\
 y_5 &= q_5 + y_4 \\
 y_6 &= q_6 + y_5 \\
 y_7 &= q_7 + y_6
 \end{aligned}$$

The eight different outputs produced in the above configuration are generated in a cascading manner, and this in turn will lead to a considerable accumulated propagation delay which is directly proportional to the number of the cascaded outputs in a linear manner. This means that if the gate delay for  $y_0$  equal to  $t_p$ , the delay for  $y_7$  will be  $8 \times t_p$  as a worst case propagation delay, and this will limit the maximum frequency applied for the system clock.

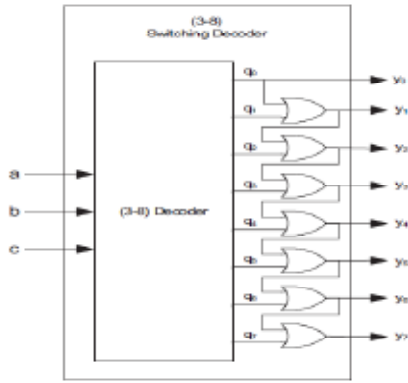


Fig. 3: Block diagram for (3 to 8) switching decoder.

Therefore, this configuration is not the suitable configuration for real-time processing applications. The above inherent problem becomes very significant when the number of outputs for the SD increases. However, this problem can be eliminated when we get rid of the cascaded outputs by generating all the outputs directly from their corresponding product terms using basic logical operators (the decoder is no longer needed), as described by the following logic equations:

$$\begin{aligned}
 y_0 &= a'.b'.c' \\
 y_1 &= a'.b'.c + a'.b'.c' = y_0 + a'.b'.c \\
 y_2 &= a'.b'.c' + a'.b'.c + a'.b.c' = y_1 + a'.b.c' \\
 y_3 &= a'.b'.c' + a'.b'.c + a'.b.c' + a'.b.c = y_2 + a'.b.c \\
 y_4 &= a'.b'.c' + a'.b'.c + a'.b.c' + a'.b.c + ab'.c' = y_3 + ab'.c' \\
 y_5 &= a'.b'.c' + a'.b'.c + a'.b.c' + a'.b.c + ab'.c' + ab'.c \\
 &= y_4 + ab'.c \\
 y_6 &= a'.b'.c' + a'.b'.c + a'.b.c' + a'.b.c + ab'.c' + ab'.c + abc' \\
 &= y_5 + abc' \\
 y_7 &= a'.b'.c' + a'.b'.c + a'.b.c' + a'.b.c + ab'.c' + ab'.c + abc' \\
 &+ abc = y_6 + abc
 \end{aligned}$$

As mentioned earlier, (3 to 8) SD is just a low scale configuration for demonstration purposes. In our case, the scale of the SD used in this design is much larger. As shown in the block diagram and the truth table of Figs. 4 and 5 respectively, the size of the SD used is an (8 to 256) and it has the same function and perform the same switching behavior, but in a larger scale. Hence, the design will be more sophisticated. Specifically, the SD accepts eight bits binary input and produces 28 (256) outputs denoted  $y_0$  to  $y_{255}$ . For instance, the output  $y_{255}$  as shown in the following logic equation is constructed by OR

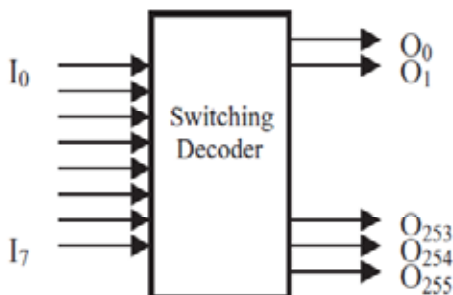


Fig. 4: Block diagram for (8 to 256) switching decoder

Input								Output					
$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$	$y_{255}$	$y_{254}$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	0	0	0	0	0	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	0	0	0	1	0	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0
1	1	1	1	1	0	1	0	1	1	0	0	0	0
1	1	1	1	1	0	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	1	1	0	0	0	0
1	1	1	1	1	1	0	1	1	1	0	0	0	0
1	1	1	1	1	1	1	0	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0

Fig. 5: Truth table for (8 to 256) switching decoder. gating 256 × 8 variables product terms:

$$\begin{aligned}
 Y_{255} &= I'_7 \cdot I'_6 \cdot I'_5 \cdot I'_4 \cdot I'_3 \cdot I'_2 \cdot I'_1 \cdot I'_0 \\
 &+ I'_7 \cdot I'_6 \cdot I'_5 \cdot I'_4 \cdot I'_3 \cdot I'_2 \cdot I'_1 \cdot I_0 \\
 &| + \dots + I_7 \cdot I_6 \cdot I_5 \cdot I_4 \cdot I_3 \cdot I_2 \cdot I_1 \cdot I'_0 \\
 &+ \dots + I_7 \cdot I_6 \cdot I_5 \cdot I_4 \cdot I_3 \cdot I_2 \cdot I_1 \cdot I_0
 \end{aligned}$$

This SD circuit is described and designed in a very efficient way using Hardware Description Language (HDL) provided with the software package of Quartus [4.2] as an available option for the design entry. Having described the function of the SD, the next step in the computation as indicated in the HE digitized formula is to divide the resulting accumulated values located in the 16-bit counters by 28. This division can be easily achieved in binary by shifting data to the right eight times, in other words, it is simply to consider only the eight Most Significant Bits (the Most Significant Byte of the counters contents). Once performed, the final step is to load the one byte data array of the computed HE to the transformation look-up table. The 256 words of the data are transferred to the look-up table via multiplexers. The transformation look-up table resides in 256 × 8 RAM. The computation process of HE is governed by the timing and control unit. The main tasks of this unit are to initialize the transfer and generate the proper sequences and the required timing signals and addresses. The HE starts by logging the digitized pixel gray levels (8 bit) of the image in real-time to the SD input, which in turn, enables the relevant counters to be incremented. This process, as shown in the flow chart of Fig. 6, is repeated  $n \times m$  times, till the last pixel in the image is reached.

**V. System Performance**

The design and synthesis of the HE functional units shown in Fig. 1 are built around Altera's FPGA Stratix II family chip). The Altera Quartus II version 4.2 is used to develop and simulate the circuit. For simulation purposes, a 64K ROM is constructed within the same chip and loaded with an image. The hardware set-up for simulation and verification purposes is composed of four different components as shown in the block diagram of Fig. 7. The compilation and time simulation reports show that the proposed system works properly. The simulated timing diagram for units addressing is depicted in Fig. 8 and the transformation look-up table RAM contents is shown in the print out of Fig. 9. The compilation report shows that the maximum frequency in the design is 250 MHz. This means, when we substitute in Eq. (2), the complete computation time including data transfer to the look up table RAM for (256 × 256) image HE, is given by

$$T_{total} = \frac{1}{f_{max}} [m \times n + g]$$

$$= \frac{1}{250 \times 10^6} [256 \times 256 + 256] = 0.263 \text{ ms}$$

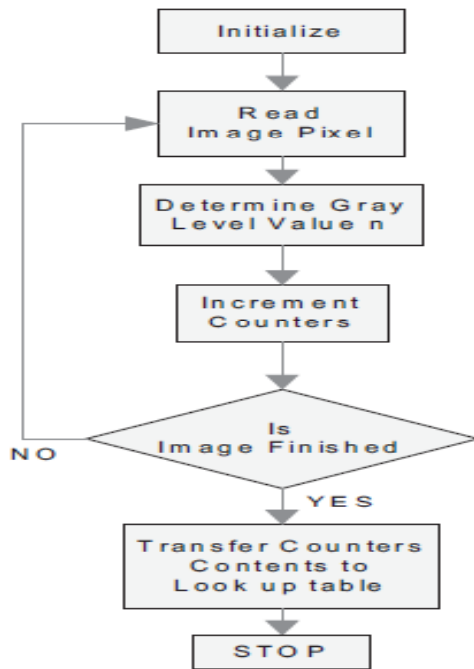


Fig. 6: Proposed system flow chart.

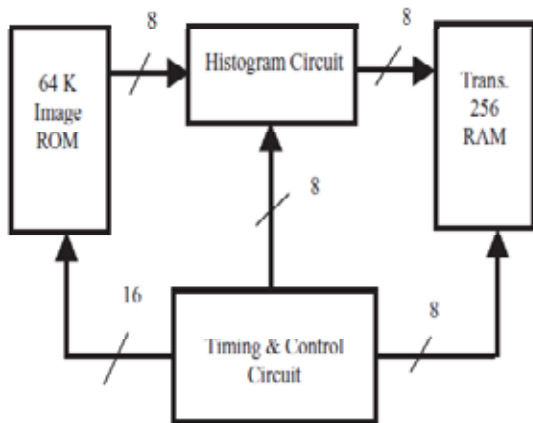


Fig. 7 The schematic block diagram for the simulation set-up.

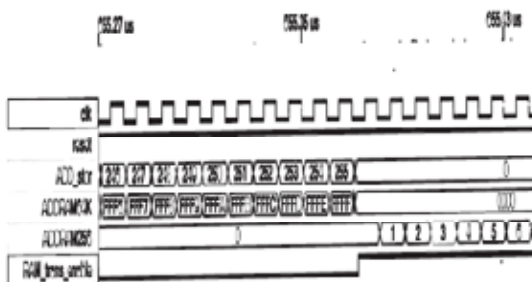


Fig. 8: The simulation timing diagram.

Date: June, 2010 RAM256\_image.tif Project: New\_algorithm

Addr	40	41	42	43	44	45	46	47	48	49	4a	4b	4c	4d	4e	4f
00	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
20	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
30	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
40	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
50	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
60	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
70	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
80	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
90	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
a0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
b0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
c0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
d0	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
e0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
f0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Fig. 9: The computed transformation RAM contents.

Table 1: The processing time for different image configurations

Image size(n×m) pixels	T <sub>total</sub> (m <sub>sec</sub> )	%T <sub>transfer</sub>
256×256	0.263	0.4
512×512	1.05	0.1
1024×1024	4.2	0.0244
2048×2048	16.78	0.006

The results shows that the time required to transfer the data at the end of the run which is considered as an additional effort is almost neglected compared to the HE computation. It becomes even insignificant in higher resolution image applications. In this set up, it consumes less than 0.4% of the total processing time. For higher resolution images of size (512 × 512) with 256 gray levels, which are commonly used for the commercial applications, the total processing time will be equal to

$$T_{total} = \frac{1}{f_{max}} [m \times n + g]$$

$$= \frac{1}{250 \times 10^6} [512 \times 512 + 256] = 1.05 \text{ ms}$$

Table 1 shows that the proposed system can handle in real-time images of size up to (2048 × 2048) pixel, assuming an integration time (frame rate) of 20 ms. and with a gray level of 256. Images with this size are considered to be of high resolution in the field of thermal imaging in both commercial and military applications

**VI. Conclusion**

The computational speed of image equalization has been improved using FPGAs. A new system based on counters in conjunction with a specially designed decoder has been developed for simultaneous computation of HS and HE. The proposed system is designed, synthesized and simulated using Stratix II family chip. This system has the advantages of being simple, flexible with reasonable development cost. The timing simulation results show that this system is faster than that of the previously published work [4] by at least a factor of 50, and the total time required to perform HE of an image of size 256 × 256 is 0.263 msec.

## References

- [1] Salcic, Z. and Sivaswamw, J. IMECO: A Reconfigurable FPGA-based Image Enhancement, Co-Processor Framework. Real-Time Imaging, 5, 1999, pp. 385–395.
- [2] Karim, M. A. Electro-optical displays. Marcel Dekker, Inc.
- [3] Torre, A., Peinado, A., Segura, J., Perez-Cordoba, J., Benitez, M. and Rubio, A. Histogram equalization of speech representation for robust speech recognition. IEEE Transaction on Speech and Audio Processing, 13(3), May 2005, pp. 355–366.
- [4] Lianfa, B., Xing, L., Qian, C. and Baomin, Z. The hardware design of a real-time infrared image enhancement system. IEEE Int. Conf. Neural Networks & Signal Processing. Dec. 2003, pp. 1009–1012.
- [5] Xiyang, L., Guqiang, N., Yanmei, C., Tian, P. and Yanli, Z. Real-time Image Histogram Equalization Using FPGA. Proc. Spie, Conf., 3561, 1998, pp. 293–299.
- [6] Dongsheng, G., Nansheng, Y., Defu, P. Min, H., Xiaoyan, S. and Reolan, Z. ADSP+FPGA based real-time histogram equalization system of infrared image. Proc. Spie, 4602, 2001, pp.160–165.



Mrs. Nitin Sachdeva received her M-Tech. degree in VLSI Design and CAD from THAPAR INSTITUTE OF ENGG. & TECHNOLOGY (Deemed University) Patiala, Punjab, India in 2005. She is now working as Assistant Professor in Electronics Deptt. at YMCA UNIVERSITY OF SCIENCE AND TECHNOLOGY, Faridabad (Haryana) India. Her area of interest is VLSI DESIGN. She has published 15 research papers in International and National Conferences, guided

several B-Tech projects, M-Tech projects & thesis. She is the coordinator of VLSI design group of post graduate program of YMCAUST, Faridabad. She is NATIONAL CHAMPION in Softball and awarded ROLL OF HONOUR by Chief Minister Punjab in year 1995-1996.



Tarun Sachdeva received his M-Tech. Degree in VLSI Design and CAD from THAPAR INSTITUTE OF ENGG. & TECHNOLOGY (Deemed University) Patiala, (Punjab) India in 2005. He is now working as Senior Engineer, in Infotel business solution Ltd. Gurgaon (Haryana) India. He has published 10 research papers in International and National Conferences. He is a member of IETE society.