Proceedings of the World Congress on Engineering 2014 Vol I,
WCE 2014, July 2 - 4, 2014, London, U.K.

Real Time Application Specific Image Processing Hardware

Abdul Raouf Khan, *Member, IAENG,* and Md. Al-Amin Bhuiyan

***Abstract*— Modern multimedia tools heavily depend on image manipulation. In many cases, manipulation of images is specific to certain directions or angles. Generally, the images are manipulated using software routines, which are time consuming, compared to manipulation using hardware techniques. This paper proposes hardware architecture, based on two dimensional cellular automata, for manipulating images in certain specific directions and angle. The proposed architecture can be easily implemented using VLSI technology.**

***Index Terms*— Image Processing, Cellular Automata, 2DCA, Real Time Processing.**

I. Introduction

IMAGE are being processed and manipulated at every step in modern multimedia tools. In most gaming devices and animation series, movement of images are confined or restricted to right, left, up & down directions only. The rotation of objects also remains limited to specific angles, like 90 degree or 180 degree. In addition, special effects like duplication or multiplication of images is quite common in many gaming devices and animation series.

The processing time of images and objects is also very important, and usually depend on the software routines used to manipulate the images. To enhance the processing speed of images various hardware solutions have been reported. In [1], [2] the hardware proposed is only for image rotation. In [3] a highly parallel machine CLIP4 was developed. In CLIP4 an array of identical cells corresponding to each pixel consists of a logical unit and four memories, where three of them are single bit buffers and one is a 32 bit RAM. Also there is a Boolean processor in each cell, which provides 16 Boolean functions. In [4] a similar kind of work was proposed and dealt with the implementation of 2D transform operators as a sequence of window shift operations.

In this paper we propose hardware architecture for manipulating binary images in specific directions and angles. The proposed architecture is based on the theory of Cellular Automata (CA) and is highly suitable for VLSI implementation. For reducing the complexity on the silicon floor, designers look for simple and regular blocks which can be cascaded and reused. Cellular automata have the advantage to fulfill these objectives. It has been demonstrated by us [5] that parallel processing architecture built around the CA machine is highly suitable for variety of applications. Further, various VLSI applications of Cellular Automata have been reported [5] - [10]. In addition, various theoretical developments of Cellular Automata have been reported in [11] - [14].

Abdul Raouf Khan is with the department of Computer Sciences, King Faisal University, P. O. Box 400, Alhassa 31982. Saudi Arabia. (corresponding author phone: +966 562506615; fax: +966 135899236; e-mail: raoufkhan@ kfu.edu.sa).

Md. Al-Amin Bhuiyan is with the department of Computer Engineering, King Faisal University, P. O. Box 400, Alhassa, 31982. Saudi Arabia. email: mbhuiyan@kfu.edu.sa.

II. CA Preliminaries

The CA structure investigated by Wolfram [15] can be viewed as discrete lattice of sites (cells) where each cell can assume either the value 0 or 1. The next state of a cell is assumed to depend on itself and on its two neighboring cells for a 3 neighborhood dependency. The cells evolve in discrete time steps according to some deterministic rule that depends only on local neighborhood. In effect, each cell as shown in Fig. 1, consists of a storage element (D- Flip Flop) and a combinational logic implementing the next state. Final Stage

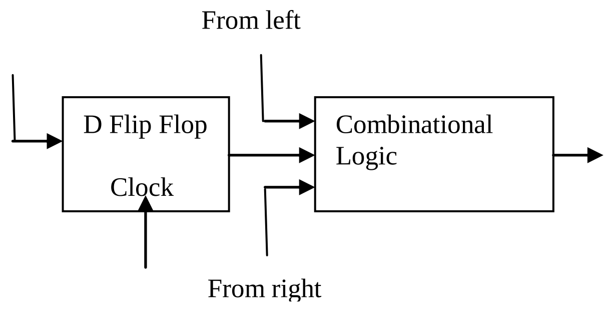

Figure 1 Basic CA cell

Mathematically, the next state q of the i^{th} cell of a CA is given by

$$q_i(t+1) = f[q_{i-1}(t), q_i(t), q_{i+1}(t)]$$

In 2D cellular automata the cells are arranged in two dimensional grid with connections among the neighborhood cells. The state of CA at any time instant can be represented by an $(m \times n)$ binary matrix. The neighborhood function specifying the next state of a particular cell of the CA is affected by the current state of itself and eight cells in its nearest neighborhood. Mathematically, the next state q of the $(i, j)^{th}$ cell of a 2D CA is given by

$$q_{i,j}(t+1) = f[q_{i,j}(t), q_{i,j-1}(t), q_{i,j+1}(t), q_{i+1,j-1}(t), q_{i+1,j}(t), q_{i+1,j+1}(t), q_{i-1,j-1}(t), q_{i-1,j}(t), q_{i-1,j+1}(t)]$$

Where, *f* is the Boolean function of nine variables.

. To express a transition rule of 2D CA, a specific rule convention proposed in [5] is noted below

64	128	256
32	1	2
16	8	4

The central box represents the current cell (that is the cell being considered) and all other boxes represent the eight nearest neighbors of that cell. The number within each box represents the rule number associated with that particular neighbor of the current cell – that is, if the next state of a cell is dependent only on its present state, it is referred to as rule 1. If the next state depends on the present state of itself and its right neighbor, it is referred to as rule 3 (=1+2). If the next state depends on the present state of itself and its right, bottom, left, and top neighbors, it is referred to as rule 171 (=1+2+8+32+128) and so on. The minimized expression for Rule 171 is given by

$$q_{i,j}(t+1) = [q_{i,j}(t) \oplus q_{i,j-1}(t) \oplus q_{i,j+1}(t) \oplus q_{i+1,j}(t) \oplus q_{i-1,j}(t)]$$

III. Mathematical Model

The 2D CA behavior can be analyzed with the help of an elegant mathematical model [5], where two fundamental matrices are used to obtain row and column dependencies of the cells. Let the two dimensional binary information matrix be denoted as X_t that represents the current state of a 2D CA configured with a specific rule. The next state of any cell will be obtained by XOR operation of the states of its relevant neighbors associated with the rule. The global transformation associated with different rules are made effective with following fundamental matrices referred to as $T_1 \& T_2$.

$$T_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \;\&\; T_2 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Next we define CA under various possible basic conditions. If same rule is applied to all the cells in a CA, then the CA is said to be Uniform or Regular CA. If different rules are applied to different cells in a CA, then the CA is said to be Hybrid CA. The CA is said to be a Periodic CA if the extreme cells are adjacent to each other. The CA is said to be a Null boundary CA if the extreme cells are connected to logic 0-state. If in a CA the neighborhood dependence is on XOR or XNOR only, then the CA is called additive CA, specifically, a linear CA employs XOR rules only. A CA whose transformation is invertible (i.e. all the states in the state transition diagram lie in some cycle) is called a Group CA, otherwise it is called a Non Group CA.

The following theorems [5], specifies the value of the next state of a 2D CA referred to as X_{t+1} given that its current state is X_t. The CA is assumed to be configured with primary rule only—that is it depends on only one of its nine neighbors.

Theorem: 1. *The next state transition of all the primary rules (1, 2, 4, 8, 16, 32, 64, 128, 256) with null boundary condition, can be represented as*

$$Rule1 \rightarrow [X_{t+1}] = [X_t]$$
$$Rule2 \rightarrow [X_{t+1}] = [X_t][T_2]$$
$$Rule4 \rightarrow [X_{t+1}] = [T_1][X_t][T_2]$$
$$Rule8 \rightarrow [X_{t+1}] = [T_1][X_t]$$
$$Rule16 \rightarrow [X_{t+1}] = [T_1][X_t][T_1]$$
$$Rule32 \rightarrow [X_{t+1}] = [X_t][T_1]$$
$$Rule64 \rightarrow [X_{t+1}] = [T_2][X_t][T_1]$$
$$Rule128 \rightarrow [X_{t+1}] = [T_2][X_t]$$
$$Rule256 \rightarrow [X_{t+1}] = [T_2][X_t][T_2]$$

Theorem:2. *The next state transition of all the primary rules with periodic boundary condition can be represented by*

$$Rule1 \rightarrow [X_{t+1}] = [X_t]$$
$$Rule2 \rightarrow [X_{t+1}] = [X_t][T_{2c}]$$
$$Rule4 \rightarrow [X_{t+1}] = [T_{1c}][X_t][T_{2c}]$$
$$Rule8 \rightarrow [X_{t+1}] = [T_{1c}][X_t]$$
$$Rule16 \rightarrow [X_{t+1}] = [T_{1c}][X_t][T_{1c}]$$
$$Rule32 \rightarrow [X_{t+1}] = [X_t][T_{1c}]$$
$$Rule64 \rightarrow [X_{t+1}] = [T_{2c}][X_t][T_{1c}]$$
$$Rule128 \rightarrow [X_{t+1}] = [T_{2c}][X_t]$$
$$Rule256 \rightarrow [X_{t+1}] = [T_{2c}][X_t][T_{2c}]$$

Where

$$T_{1c} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \;\&\; T_{2c} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

IV. Simulation and Results

We worked with various binary matrices of, considering binary “0” as white pixel and a binary “1” as a black pixel. We wrote programs in C language. Various rules were applied to different $(m \times n)$ matrices. In many cases, we obtained similar results and reported accordingly. Figure 2 summarizes certain outputs and were produced similar to our experimental results.

In fact, all primary rules except rule 1 can be used to translate the images, in all directions including corner directions, both with null and periodic boundary conditions (Fig. 2a, 2b). However, under null boundary condition, once the image has been translated up to the screen boundary, further translation means that images are lost and cannot be retrieved (Fig 2c). For similar translation under periodic boundary conditions, the images appear again on the other side of the screen boundary (Fig 2d). Rotation of images about x- axis and y- axis, can be implemented with the help of secondary rules i.e. rule 136 (128 + 8) and rule 34 (32+2) under null boundary conditions (Fig. 2e).

To achieve the required translation or rotation the CA is run for several clock periods and depends on the cycle length of a particular rule applied as well as the number of rows and columns of the CA $(m \times n)$.

To create special effects, like duplication of image (Fig. 2f), rule 3, rule 9, rule 33 and rule 129 can be used. It is possible to produce multiple images of an object with other rules. To zoom out and zoom in rule 170 (2+8+32+128) is applied (Fig. 2i, 2j).

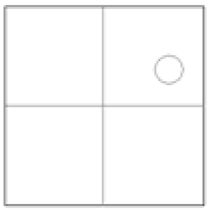
Figure 2a. RightShift

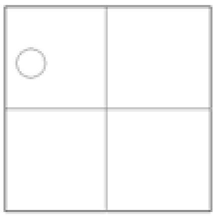
Figure 2b. Left Shift

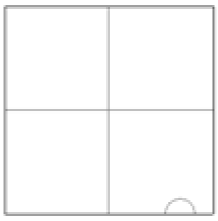
Figure 2c. Null Shift

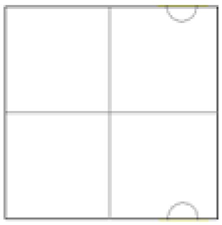
Figure 2d. Periodic Shift

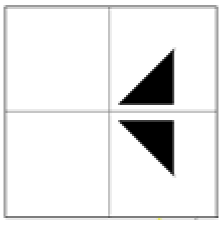
Figure 2e. X-rotation

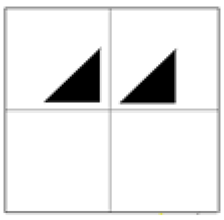
Figure 2f. Duplication

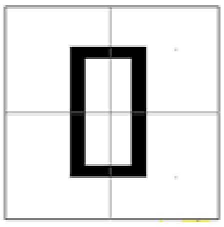
Figure 2g. Zoom in

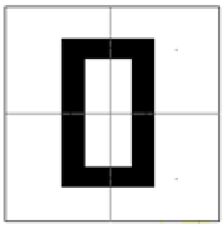
Figure 2h. Zoom out

V. Proposed Architecture

The architecture of the proposed hardware consists of an $(m \times n)$ matrix of cells. Each cell (as shown in Fig 3) consists of a memory element (D-flip Flop) and a Multiplexer. Eight inputs of the MUX (Fig. 4) are labeled as $L, R, T, B, LT, RT, LB, RB$. These stand for the eight outputs of the memory elements of the neighboring cells i.e. left, right, top, bottom, left top, right top, left bottom, right bottom respectively. The other two inputs R_x and R_y of the MUX are obtained from $T \oplus B$ and $L \oplus R$ respectively.

Inputs for duplication take the values $D_{xp} = Q \oplus R, D_{xn} = Q + L, D_{yp} = Q + T, D_{yn} = Q + B$ respectively.

For Zoom Out and Zoom in $Z_o = T \oplus B \oplus R \oplus L$ and $Z_i = Q \oplus T \oplus B \oplus R \oplus L$ inputs are applied to the MUX. The select lines of the MUX and the clock are same for all the cells.

For applying any of the transformations, the select lines of each MUX selects the particular input and the CA is evolved for some no. of cycles depending on the operation. For example if any binary image located at $x_1 \times y_1$ is to be translated to new location $x_2 \times y_2$ towards left in the x direction, the select lines select L input of the MUX and the CA is evolved for $|x_1 \times x_2|$ times. Similarly, other transformations are carried.

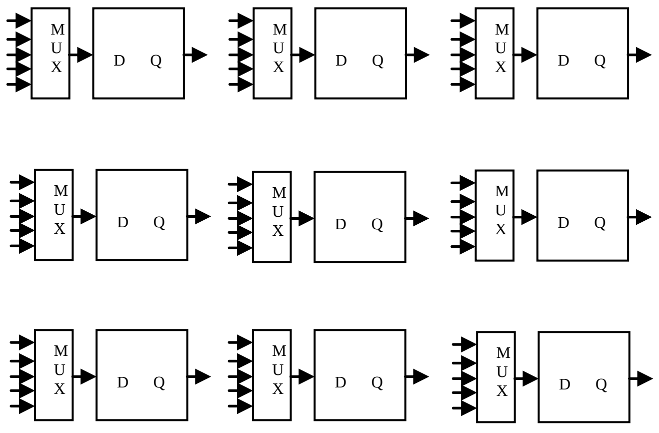

Figure 3 Details of each cell

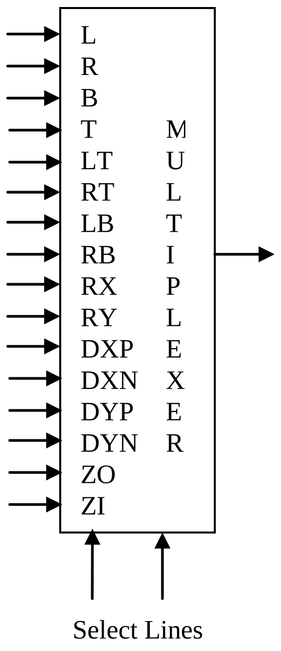

Figure 4 Details of the Multiplexer

VI. CONCLUSION

In the paper, we proposed the architecture of an application specific image processing hardware, based on 2D cellular automata. The local dependency of the cells (pixels) helps for faster real time operations. The proposed architecture can be extended to build hardware based games and animation.

REFERENCES

[1] I. Ghosh and B. Mujamdar, "Design of an application specific VLSI chip for image rotation", Proc. 7th Int. Conf. on VLSI Design, 1994, pp 275-278.

[2] N. Tsuchida, Y. Yamada Y and M. Weda, " Hardware for image rotation by twice skew transformation", IEEE Trans. ASSP, Vol. 34, No4, pp 527-532. 1987.

[3] M.J.B Duff "Review of the CLIP4 image processing system" University College London, England.

[4] G. P. Biswas, et al. "Cellular Architecture for Affine Transformation on Raster Images", "IEE Proc. Comput. Digit. Tech. Vol. 143, No2. 1996.

[5] A. R. Khan, P.P.Choudhury et.al. "VLSI architecture of a cellular automata machine", Int. Journal Computers and Mathematics with applications, Vol. 33 No. 5 pp79-94. 1997

[6] P.P. Choudhuri, D. R. Choudhuri, S. Nandi and S.Chatterjee "Additive Cellular Automata, Theory and Applications, Vol. 1, IEEE Computer Society Press, Calfornia. 1997

[7] D. R. Chowdhury, I. S. Gupta and P. P. Choudhuri, "A Class of two dimensional cellular automata and applications in random pattern testing", Journal of Electronic Testing: Theory and Applications, Vol. 5 pp 65-80, 1994.

[8] D. R. Chowdhury et. al., "A low cost high capacity associative memory design using cellular automata", IEEE Trans on Computers, Vol.44 No.10. 1995

[9] A. R. Khan "Architecture of Image Encryption Hardware Using Two Dimensional Cellular automata" Proc. Int. Conf. on IT Convergence and Security, Lecture notes in Electrical Engineering, 120. Springer. 2011.

[10] P. Sarkar. "A brief history of Cellular Automata" ACM Computing Surveys, Vol. 32 No. 1, pp 80-107. 2000.

[11] A. R. Khan A. "Classification of 2D Cellular Automata Uniform Group Rules", European Journal of Scientific Research, Vol. 64 No. 1 pp 51-57. 2011

[12] S. Munshi et.al., "An alalytical framework for characterizing restricted two dimensional cellular automata evolution", Journal of Cellular Automata, Vol. 3 No2, pp 313-335. 2008

[13] K. Nayak, et al. "Colour Graph: An efficient model for two dimensional cellular automata" Orissa Mathematical Society Conference, India. 2008

[14] Amal K. Gosh., P. P. Choudhury and A. Basuray, "Chotis fractals with multivalued logic in cellular automata", in Innovation and advanced techniques in computer and Information Sciences and engineering, pp 77-82, 2007. University of Bridgeport, Springer.

[15] S. Wolfram "Statistical Mechanics of Cellular Automata" Rev. Mod. Physics Vol. 55 No. 3 pp 601-644. 1983.