Fixed-point Simulink Designs for Automatic HDL Generation of Binary Dilation & Erosion

Gurpreet Kaur, Nancy Gupta, and Mandeep Singh

***Abstract*—**Embedded Imaging is a technique used to develop image processing based standalone hardware systems and involves the on-board implementations of image based operations on platforms like FPGAs. These implementations require low-level description of desired operation using VHDL/Verilog. A digital image is a 2-dimensional array of discrete pixel intensities. It involves lot of complexity for describing a pixel level image based operation in HDL. Tools like HDL-coder in MATLAB may reduce this complexity by automatically generating a synthesizable HDL code from Simulink Models that are designed using fixed-point conditions. This paper proposes the design of fixed-point coding scheme of Morphological Image Processing operations: *Binary dilation & erosion* for automatic generation of HDL codes. It is concluded that the output images generated through proposed fixed point Simulink models are not only qualitatively better but the generated VHDL codes also resulted in successful synthesization on Spartan6 FPGA.

***Keywords*—**Embedded Imaging, Fixed-point arithmetic, FPGA, HDL Coder, Morphological Image Procesing, Simulink

I. Introduction

DIGITAL Image processing is a branch of soft-computing that deals with subjecting a 2-d input image to a series of mathematical operations in order to obtain a desired result. A typical experimental setup used in most of the image processing based applications consists of a PC-based system as shown in figure 1.

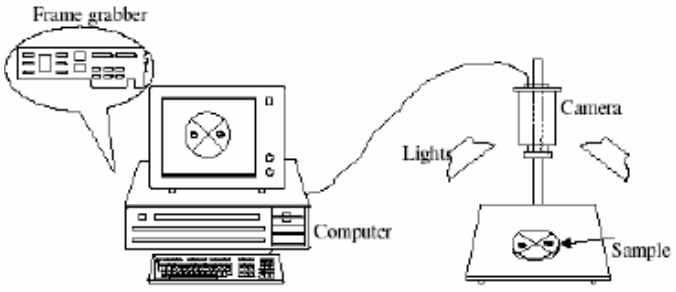

Fig. 1 PC-based experimental setup for image processing

Gurpreet Kaur, Assistant Professor, Department of ECE, CT Institute of Engineering, Management and Technology, Jalandhar, Punjab, India, PIN-144020, (E-mail: gpreet.09@gmail.com)

Nancy Gupta, Assistant Professor, Department of ECE, CT Institute of Engineering, Management and Technology, Jalandhar, Punjab, India, PIN-144020, (E-mail: nancy.gupta24@gmail.com)

Mandeep Singh, UG scholar, CT Institute of Engineering, Management and Technology, Jalandhar, Punjab, India, PIN-144020, (E-mail: er.mds94@gmail.com)

This system uses following major components [1] while performing image analysis:

i. An illumination device, which illuminates the sample under test
ii. A solid-state CCD array camera, to acquire an image
iii. A frame-grabber, to perform the A/D (analog-to-digital) conversion of scan lines into picture elements or pixels digitized in a N row by M column image
iv. A personal computer or microprocessor system, to provide disk storage of images and computational capability with vendor-supplied software and specific application programs
v. A high-resolution colour monitor, which aids in visualizing images and the effects of various image analysis routines.

A PC-based computational system may help in reducing computational time, but a hardware based standalone system may decreased it further to about 100 times shorter than manual computations [2]. A standalone system is designed using ***Embedded Imaging*** [3] techniques which involves realization of image based mathematical operations onto a dedicated hardware like FPGAs/DSP processors/ASICs etc.; such that the use of software is eliminated completely.

Embedded Imaging has found useful applications in developing many real-time applications. *Nelson E. A* [4] proposed implementation of morphological image processing operations on FPGA hardware. *Jiang H* [5] discussed various design issues in VLSI Implementation of image processing hardware accelerators methodology and implementation. *Pearson T* [6] proposed a fully functional hardware-based image processing system for high-speed inspection of grains. *Verma B & Sardana H K* [7] discussed the importance of standalone hardware systems & proposed a real-time Image segmentation using watershed algorithm on FPGA. *Kaur G & Verma B* [8] proposed an image processing based algorithm for automatic rice grading system using morphological operations viz. dilation & erosion, which may further be realized on hardware.

One of the major design issues in VLSI implementation of image processing is writing a pixel level HDL description for a complex 2-dimensional input (digital image). This complexity may be overcome with the help of MATLAB based hardware synthesis [9]. HDL-Coder in MATLAB (Simulink) provides an efficient mechanism that may be used to generate HDL codes from fixed point description of various mathematical operations [10]. However, HDL coder doesn't support automatic HDL conversion for every image based operation until the given operation is described using a 'fixed-

point coding' technique in a *user-defined function block* of Simulink.

This paper proposes a fixed-point coding scheme to design synthesizable user-defined blocks in Simulink for automatic HDL generation of morphological image processing operations: Binary Dilation & Binary Erosion. Section II of this paper discusses the design of various Simulink based systems & sub-systems used for performing different morphological operations. Section III proposes the scheme of designing fixed-point coding for these operations. The results obtained through these designs are compared with desired results in section IV. This section also discusses the steps for generating HDL codes along with the results of FPGA synthesis on Spartan6.

II. Simulink Model Design

Designing a Simulink model requires the complete knowledge of system components & its design flow. The components of any system can either be designed using only inbuilt blocks of Simulink or by integrating some user-defined functions into the design.

A. Simulink Model for Binary Dilation

Morphological Dilation operation in image processing can be of two types: *Binary & Grayscale*. The dilation operation deals with *growing* the size of object in image. Mathematically, it is defined as equation (1), and uses set theory for its implementation.

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset \quad (1)$$

Simulink model shown in figure 2(a) is designed for binary dilation operation using a square structuring element of size 3X3. In this model, an input image of size 256X256 is applied to two sections:

1. **First section** performs the direct dilation operation on image using an inbuilt dilate block. The output achieved using these two sections are finally compared with each other.
2. **Second section** *converts* the data type of input image, *serializes* the input pixels; performs the dilation operation using a *fixed-point* (hardware resembled) code; and *deserializes* the output pixels to reconstruct a 2-D image. This section consist of three sub-systems: Serialize, Dilation_hardware & Deserialize whose working can be described as follows:

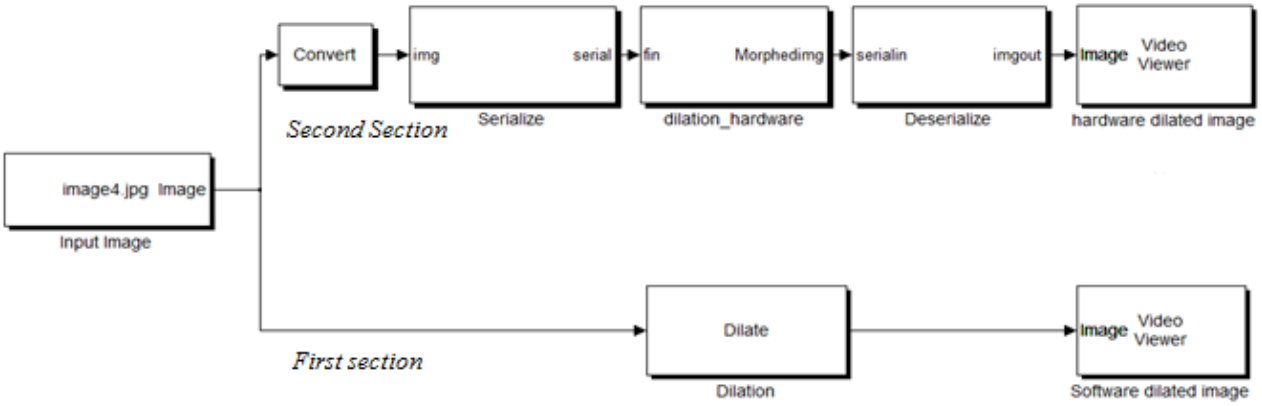

Fig 2(a) Simulink model for fixed-point coding based binary dilation

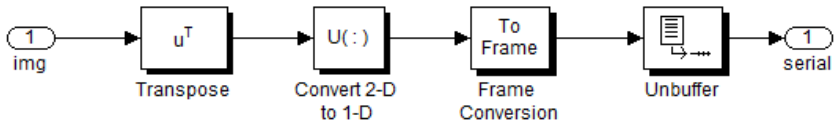

Fig 2(b) Serialize sub-system for Dilation Simulink model

Fig 2(c) Dilation_hardware sub-system for Dilation Simulink model

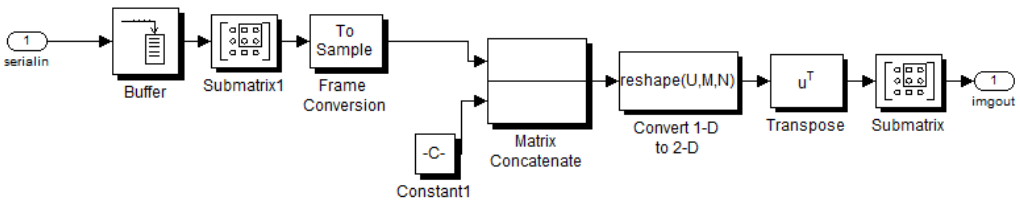

Fig 2(d) Deserialize sub-system for Dilation Simulink model

a) *Serialize:* This subsystem is used for serialization of input image as shown in figure 2(b). During serialization, the *Transpose* block transposes the M-by-N input matrix to size N-by-M. The Transpose block supports real & complex floating-point and fixed-point data types. Then, the *Convert 2-D to 1-D* block reshapes an M-by-N matrix input to a 1-D vector with length M*N. The input is reshaped column-wise. The *Frame Conversion* block passes the input to the output and sets the output sampling mode equal to the 'sampling mode of output signal parameter', which can be either Frame-based or Sample-based. Finally, *Unbuffer* block unbuffers an M-by-N input into a 1-by-N output i.e. inputs are unbuffered row-wise so that each matrix row becomes an independent time-sample in the output. The rate at which the block receives inputs is generally less than the rate at which the block produces outputs.

b) *Dilation_Hardware:* After serialization, another subsystem named *dilation_hardware* is used as shown in figure 2(c). This subsystem makes use of a *MATLAB Function* block named dilation_hw_filter with which a customized MATLAB script can be designed using fixed-point coding technique. The *MATLAB function* thus designed is used during simulation and generates code for a Simulink Coder target. This block can generate efficient and optimized HDL code. To generate HDL code, the *MATLAB Function* block relies on an analysis that determines the size, class, and complexity of each variable.

c) *Deserialize:* After performing the operation defined in hardware filter of dilation, the output is deserialized again using a subsystem shown in figure 2(d). It *unbuffers* the input and the *Submatrix* block extracts a 'contiguous submatrix' from the M-by-N input matrix. The block treats length-M unoriented vector input as an M-by-1 matrix. *Matrix concatenate* block Concatenate input signals of the same data type to create a contiguous output signal. Finally, the output pixels are *converted to 2-D* and *transposed* to create the sub-matrices of input image. The image is finally reconstructed through these sub-matrices.

B. Simulink Model for Binary Dilation

The erosion operation does the exact opposite of dilation operation. It deals with ***shrinking*** of the size of image object. Mathematically, it is expresses as equation 2.

$$A \ominus B = \{z \mid (B)_z \cap A^c = \emptyset \qquad (2)$$

The Simulink model shown in figure 3 is designed for binary erosion operation using a square structuring element of size 3X3. The output of binary erosion operation will be 1 or a Boolean true only if the input image (a sliding matrix same as the size of structuring element) as well as structuring element has non-zero value at all the locations. This model is designed using the same approach as is the dilation.

The simulink model uses same serialize and deserialize subsystems as shown in figure 2(b) and 2(d). The subsystem named *erosion_hardware* is again designed using *MATLAB function* block with the help of fixed-point MATLAB script whose connection scheme is again similar to the one shown in figure 2(c). The implementation differs only in setting threshold values in fixed-point script for erosion & dilation which is explained later in section III.

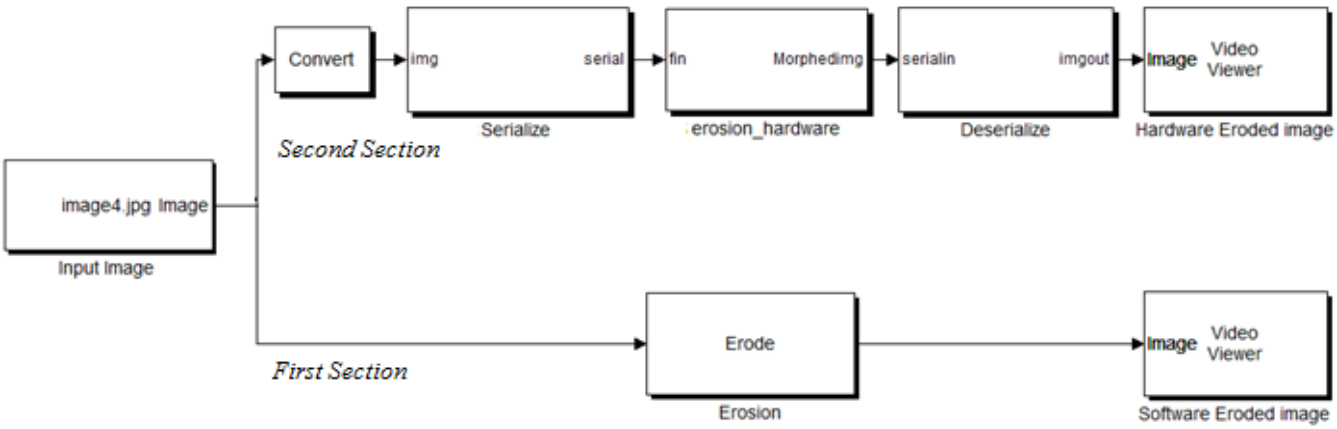

Fig 3 Simulink model for fixed-point coding based binary erosion

III. Designing Fixed-Point Code

A Fixed-point MATLAB script is designed with the help of functions that support fixed-point arithmetic. The biggest advantage of using this script is its compatibility for automatic generation of synthesizable HDL codes (VHDL/Verilog).

A. Fixed Point Script for Binary Dilation & Erosion

The *MATLAB function* block of Simulink computes binary dilation or binary erosion on input serialized image by creating 3X3 sub-images of input image whose array elements are given as defined in equation (3),

$$I = \begin{matrix} b11 & b12 & u \\ b21 & b22 & b23 \\ b31 & b32 & b33 \end{matrix} \quad (3)$$

Here, u is the first pixel of serialized input image. These array elements are then declared as arguments in a function such that the sum of array elements (fout) of every 3X3 sub-image is computed using fixed-point arithmetic as shown below:

```
function fout = f(u, b23, b33, b12, b22,
b32, b11, b21, b31)
a1 = fi(u+b23+b33, 1,11,0,fimath(u));
pa1 = row_shifter1(a1);
a2 = fi(b11+b21+b31, 1,11,0,fimath(u));
pa2 = row_shifter2(a2);
a3 = fi(b32+b22+b12, 1,11,0, fimath(u));
pa3 = row_shifter3(a3);
fout = fi(pa1+pa2+pa3,1,12,0,fimath(u));
end
```

The row-shifters are used to shift the operation from one row to another. In low level HDL description, these row shifters are treated as delay units. Similarly, column shifters are also designed to shift from one column to another in order to create a sliding 3X3 matrix as shown below:

```
b12_new = column_shifter1(u);
b11_new = column_shifter2(b12);
b23_new = row_shifter1(u);
b33_new = row_shifter2(b23);
b22_new = row_shifter3(b12);
b32_new = row_shifter4(b22);
b21_new = row_shifter5(b11);
b31_new = row_shifter6(b21);
```

Finally, the sum obtained by adding array elements of 3X3 sub-image (fout) is compared with a threshold value to decide the output. The output of this system can be either a Boolean true '1' or false '0', depending upon the required operation.

B. Deciding threshold value

A square structuring element (SE) of size 3, as shown in figure 4, is used for computing binary dilation or binary erosion.

1	1	1
1	1	1
1	1	1

Fig. 4 Square SE of size 3

Assume the two input sub-image I_1 & I_2 having array elements as shown in figure 5(a) & 5(b).

0	1	1
1	1	1
0	0	1

(a)

1	1	1
1	1	1
1	1	1

(b)

Figure 5(a): Sub-image I_1, (b): Sub-image I_2

Now, as per the definitions of operations , the output of binary dilation for both sub-images (I_1 & I_2) as well as the output of binary erosion of sub-image I_2 with given SE will be Boolean true '1'. Whereas, the output of binary erosion of I_1 with given SE will be a Boolean false '0'. This is because, the output of binary erosion is one only when all the array elements of input sub-image have same values as that of SE matrix at same locations. In other words, the image is eroded only when sum of array elements in binary sub-image is equal to 9, and the image can be dilated if the sum of array elements in binary sub-image is either equal to or greater than 1. These conditions are declared as threshold for obtaining outputs of binary dilation, equation (4), and binary erosion, equation (5), in fixed-point coding script.

$$\text{Output of binary dilation} = \begin{cases} 1, & fout \geq 1 \\ 0, & otherwise \end{cases} \quad (4)$$

$$\text{Output of binary erosion} = \begin{cases} 1, & fout = 9 \\ 0, & otherwise \end{cases} \quad (5)$$

IV. Results & Discussion

The original test image (256X256) used as an input for Simulink models of binary dilation & erosion is shown in figure 6(a). The output image obtained through hardware efficient fixed-point script of binary dilation after simulating Simulink model (shown in figure 2(a)) is represented as "Hardware dilated image" and is shown in figure 6(b), whereas, the output obtained through direct usage of inbuilt dilate block of Simulink is represented as "Software dilated image" and is shown in figure 6(c).

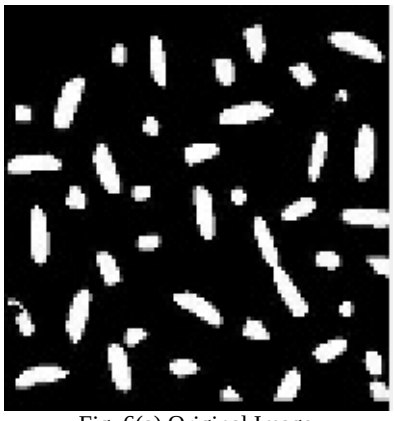

Fig. 6(a) Original Image

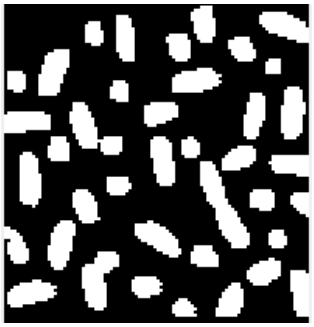
Fig. 6(b) Hardware dilated image

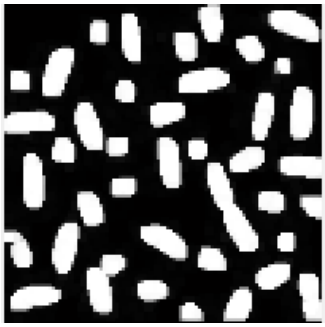
Fig. 6(c) Software dilated image

Similarly, after simulating the Simulink model shown in figure 3, the outputs obtained through hardware efficient fixed-point script of binary erosion is represented as "Hardware eroded image" and is shown in figure 6(d), whereas, the output obtained through direct usage of inbuilt erode block of Simulink is represented as "Software dilated image" and is shown in figure 6(e).


Fig. 6(d) Hardware eroded image

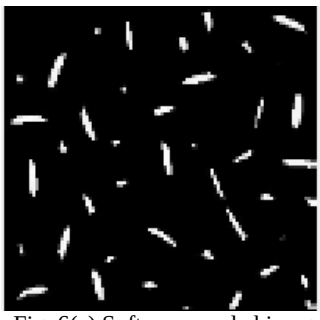
Fig. 6(e) Software eroded image

It is observed that the response of fixed-point description of dilation & erosion operations is close to desired response. Also, the output images obtained are qualitatively better. This implies that by generating an automatic HDL code for fixed-point description block of Simulink, the resultant codes would work effectively as desired morphological operations. Figure 7 below describe the steps for generating HDL in Simulink using HDL coder.

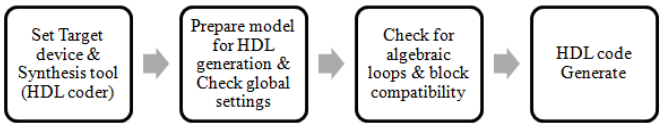

Fig. 7 Steps for generating HDL in Simulink HDL Coder

The HDL Workflow Advisor automatically converts MATLAB code from floating-point to fixed-point and generates synthesizable VHDL and Verilog code. The steps used in HDL coder to generate the HDL codes for morphological image processing operations: dilation & erosion are shown in figure 8. During this simulation, the term *morphology_hardware* is used to give a common representation to the blocks of *dilation_hardware* and *erosion_hardware* in their respective simulation models.

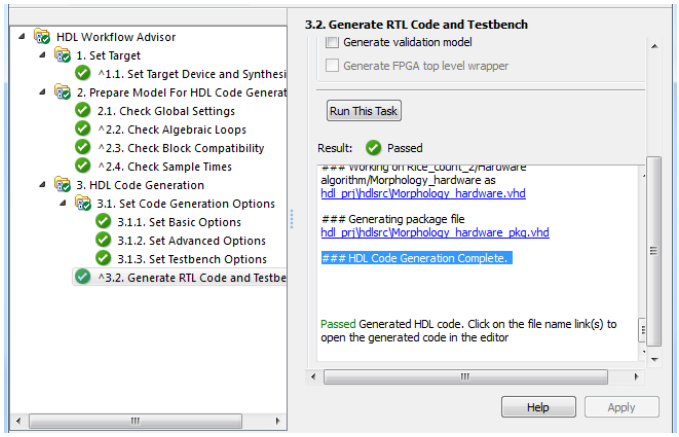

Fig. 8 Generation of HDL in Simulink HDL Coder for Morphology_hardware

As both of the models are using same architecture for processing an input image, therefore, there will be no difference in steps of generating HDLs for both models. Once the VHDL codes are generated, the FPGA synthesization is done using Xilinx ISE. Figure 9 represents the implementation of generated VHDL codes on Spartan6 FPGA.

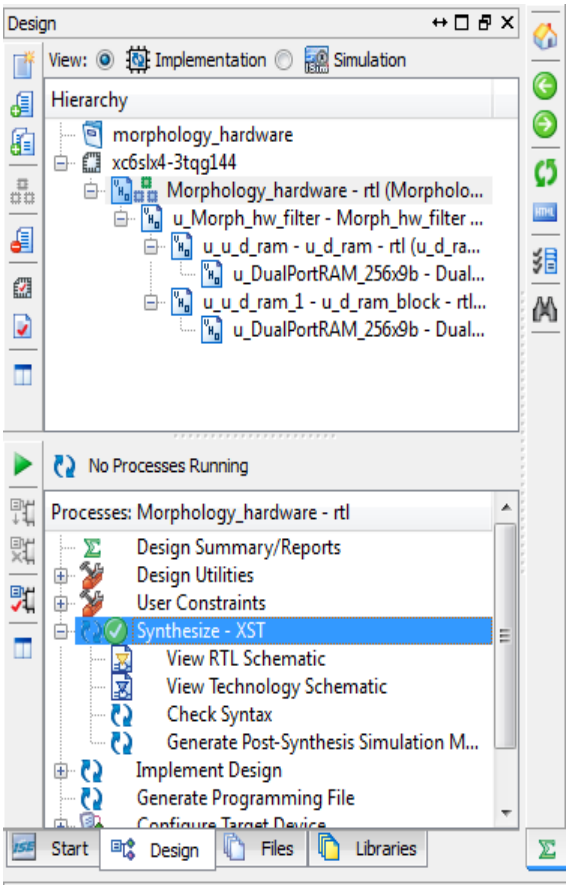

Fig. 9 Synthesis of VHDL codes in Xilinx ISE

It is also observed that the VHDL codes synthesized with no errors as shown in synthesis status report in figure 10.

Morphology_hardware Project Status			
Project File:	morphology_hardware.xise	**Parser Errors:**	No Errors
Module Name:	Morphology_hardware	**Implementation State:**	Synthesized
Target Device:	xc6slx4-3tqg144	**• Errors:**	No Errors
Product Version:	ISE 12.3	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Fig. 10 Synthesis Status of generated VHDL codes on Spartan6 FPGA

V. Conclusion

This paper has presented a fixed-point scheme for designing a hardware resembled code of morphological imaging operations: binary dilation and binary erosion. Following this methodology, Simulink models have been developed yielding the results qualitatively better in comparison to inbuilt functions of morphological operations. It is observed that the suggested fixed-point Simulink designs are also compatible for automatic HDL generation through HDL-coder which yields successful synthesization on Spartan6 FPGA. Thus, it is concluded that the proposed fixed-point Simulink designs may prove useful in reducing lot of efforts to design manual HDL codes for the implementation of imaging operations on powerful reconfigurable devices like FPGA.

Acknowledgment

The authors would like to thank the management of CT Institute of Engineering Management & Technology, Jalandhar for their constant motivation & support to carry out this research and to provide us opportunities to work in a collaborative environment with industry professionals.

References

[1] Narendra V G, Hareesh K S (2010), "Quality Inspection and Grading of Agricultural and Food Products by Computer Vision- A Review", International journal of computer applications (0975 – 8887), Manipal Institute of technology, Karnatka, Volume 2 – No.1

[2] Latala Z, Wojnar L (2001), "Computer aided versus manual grain size assessment in a single phase material", Materials Characterization 46, Elsevier, Cracow University of Technology, Poland, Pages 227 – 233

[3] Bailey D G (2011), "Design for embedded image processing on FPGAs" first edition, IEEE, John Wiley & Sons (Asia),

[4] Nelson, E A (2000), "Implementation of image processing algorithms on FPGA hardware", M.S. thesis, Department of Electrical engineering, Graduate School of Vanderbilt University, Nashville, TN

[5] Jiang H (2007), "Design Issues in VLSI Implementation of Image Processing Hardware Accelerators Methodology and implementation", Department of Electroscience, Lund University, Sweden, No. 66, ISSN 1402-8662

[6] Pearson T (2009), "Hardware-based image processing for high-speed inspection of grains", Computer and electronics in agriculture, Elsevier, 1515 College Ave, Manhattan, Pages 12-16

[7] Verma B, Sardana H K (2011), "Real Time Image Segmentation using watershed algorithm on FPGA", International Journal of Engineering Science and Technology (IJEST), Department of ECE, Lovely Professional University, Punjab, India, Vol. 3 No. 6, Pages 5282 - 5287

[8] Kaur G. & Verma B. (2013), "Measurement standards based grading of rice kernels by separating touching kernels for embedded imaging applications", International Journal of Electronics, Communication& Instrumentation Engineering Research and Development (IJECIERD) Vol, 3, pages 127-134

[9] Haldar M et al. (2001), "FPGA hardware synthesis from MATLAB", Fourteenth International Conference on VLSI Design, Bangalore, India (3–7 January, 2001), Pages 299–304

[10] MathWorks (2012), "Simulink HDL Coder User's Guide", The MathWorks Inc.