“Automatic Generation of Custom Parallel Processors for Morphological Image Processing”

Authors:
Emerson C Pedrino and Marcio M Fernandes
Federal University of São Carlos, Brazil
(Computer Science Department)
Contents

- Introduction
- Morphological Image Processing
- Reconfigurable Parallel Processor
- Automatic Generation
- Experimental Results
- Conclusions
- Future Work
- Acknowledgments
Motivation

- Morphological Image Processing: a nonlinear branch in Image Processing and very important in industrial applications.
- Requirements for high performance.
- Presents a high degree of achievable parallelism, making FPGA based platforms suitable candidates for this task.
Motivation

- However, ad-hoc FPGA implementations of specialized morphological functions to be applied to images of varying dimensions suffers from long development times and lack of flexibility.
Work Overview

- Strategy for generating automatically customized parallel processors for morphological image processing in one clock cycle.
Work Overview

- Structures based on a few textual parameters of basic morphological operations and image dimensions.

```
// DILATION: BASE TEMPLATE

if(SE[9] & orig[i-1-N]) || (SE[8] & orig[i-N]) ||
  (SE[1] & orig[i+1+N])
  result_temp[i] <= 1;
else
  result_temp[i] <= 0;
```
Morphological Image Processing

- Powerful tool for binary, grayscale, and color image processing.
- Used for: edge detection, noise suppression, skeletonization, segmentation, pattern recognition and enhancement.
- Developed by Matheron and Serra in the 1960’s.
Morphological Image Processing

- The basic dilation and erosion operations can be described by (1) and (2), respectively:

\[ A \oplus B = \bigcup \{ B + a \mid a \in A \} \quad (1) \]
\[ A \ominus (-B) = \bigcap \{ A + b \mid b \in B \} \quad (2) \]

- Powerful tools can be built using these operators in addition to logical operators (AND, OR and NOT).
Reconfigurable Parallel Processors
Parallel Architecture
ISRU Unit

```
ISRU (HDL Template)

Pk+2        Op(v_pixels[p1,p2,p3,pk+1,pk+2,pk+3,p2k+1, p2k+2, p2k+3])
            
Pk+3        Op(v_pixels[p2,p3,p4,pk+2,pk+3,pk+4,p2k+2, p2k+3, p2k+4])
```
Automatic Generation

- Pixel Mapping made by the developed tool ($Pi, SEj$):

  a) Structuring Element (SE)
  b) Input image
  c) SE over pixel P06 and neighbours
  d) SE applied to pixel P06 and neighbours using the vectorized image
Automatic Generation

- In order to configure the parallel architecture automatically, the SE mapping over the vectorized image is based on:
  - $L \times N$: the 2D image dimensions (line x columns);
  - MOp: the morphological operation;
Automatic Generation

- A prototype programming tool, using Matlab, is able to generate synthesizable Verilog to configure the parallel architecture based on $N$ and $MOp$.

- The supported morphological operations are:
  - Dilation, Erosion, Logical OR, AND, NOT and XOR.
Automatic Generation: Example of an output from the prototype tool

- Verilog code fragment aimed to execute the erosion operation over the pixel 6:

```verilog
// ISRU CONFIGURATION: 4x4 EROSION

// Code for pixels 1 to 5:
...

// Code for pixel 6:
if(SE1) if(~orig[6-1-4]) t1=0; else t1=1;
if(SE2) if(~orig[6-4]) t2=0; else t2=1;
if(SE3) if(~orig[6+1-4]) t3=0; else t3=1;
if(SE4) if(~orig[6-1]) t4=0; else t4=1;
if(SE5) if(~orig[6]) t5=0; else t5=1;
if(SE6) if(~orig[6+1]) t6=0; else t6=1;
if(SE7) if(~orig[6-1+4]) t7=0; else t7=1;
if(SE8) if(~orig[6+4]) t8=0; else t8=1;
if(SE9) if(~orig[6+1+4]) t9=0; else t9=1;
if(t1&&t2&&t3&&t4&&t5&&t6&&t7&&t8&&t9)
  result_temp[6]=1;
else
  result_temp[6]=0;

// Code for pixels 7 to 16:
...
```
// ISRU CONFIGURATION: 4x4 EROSION

// Code for pixels 1 to 5:
....

// Code for pixel 6:
if(SE1) if("orig[6-1-4]") t1=0; else t1=1;
if(SE2) if("orig[6-4]") t2=0; else t2=1;
if(SE3) if("orig[6+1-4]") t3=0; else t3=1;
if(SE4) if("orig[6-1]") t4=0; else t4=1;
if(SE5) if("orig[6]") t5=0; else t5=1;
if(SE6) if("orig[6+1]") t6=0; else t6=1;
if(SE7) if("orig[6+1+4]") t7=0; else t7=1;
if(SE8) if("orig[6+4]") t8=0; else t8=1;
if(SE9) if("orig[6+1+4]") t9=0; else t9=1;
if(t1&&t2&&t3&&t4&&t5&&t6&&t7&&t8&&t9)
    result_temp[6]=1;
else
    result_temp[6]=0;

// Code for pixels 7 to 16:
....
Automatic Generation: Example of an output from the prototype tool

- Considering this simple example, for N=4, we have: 16x13 lines of code generated.

- This example illustrates how tedious and prone to error could be the manual generation of this code, specially when using real images, with much larger values of N.
Automatic Generation: Reconfiguration Process

- The reconfiguration of the parallel architecture is made during the FPGA programming process.
- To run an application, a simple machine language was defined for this architecture.
Automatic Generation: Machine Language for the Parallel Architecture

a) Instruction format:

```
| Op | Op | Op | Op | L1 | L1 | L1 | L2 | L2 | L2 | L3 | L3 | L3 |
```

Opcode    SE line 1  SE line 2  SE line 3

b) Example instruction:

```
0 0 0 1 0 0 1 0 0 1 1 1 0 0 1 0
```

1h 2h 7h 2h

SE line 1 0 1 0
SE line 2 1 1 1
SE line 3 0 1 0

OpCode = 1h: Dilatation
SE = 2h, 7h, 2h: Cross structuring element

c) Meaning of example instruction
Experimental Results

- Target device for the experiments: Cyclone II EP2C35F672C6.
- Clock = 50MHz.
- Instruction rate: 1 clock cycle.
- Sequence of Morphological Operations applied to BW input images with the following sizes: 8x8, 16x16, 32x32, 64x64 (typically used during training by intelligent systems for fitness evaluation).
Experimental Results

- Several programs were employed in the experiments, consisting mainly of dilatation and erosion operations in sequence, with varying ordering and repetitions.

- The results of all the programs that run on the proposed architecture have been validated against equivalent software implementations using Matlab and its image processing libraries. In all cases, the final result was identical for both.
Experimental Results: Hardware Resources x LEs Complexity

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Logic Elements (LEs)</th>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>8x8</td>
<td>1063</td>
<td>224</td>
</tr>
<tr>
<td>16x16</td>
<td>4722</td>
<td>608</td>
</tr>
<tr>
<td>32x32</td>
<td>20700</td>
<td>2144</td>
</tr>
<tr>
<td>64x64</td>
<td>85873</td>
<td>8384</td>
</tr>
</tbody>
</table>

Quadratic Fitting
Experimental Results: Comparisons

<table>
<thead>
<tr>
<th>Filtering Application</th>
<th>Image Resolution</th>
<th>Processing Timing</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Work</td>
<td>64x64 pixels</td>
<td>3 µs</td>
<td>FPGA-Cyclone II-50 MHz</td>
</tr>
<tr>
<td>Matlab</td>
<td>64x64 pixels</td>
<td>4.3s</td>
<td>AMD-64-1.8 GHz</td>
</tr>
</tbody>
</table>
Conclusions

- This paper has presented a highly parallel reconfigurable architecture, able to execute a morphological operation in all of the image pixels in a single cycle.
- This system has been proposed as a high performance reconfigurable hardware unit to be used along with processing of low level images usually required by a fitness evaluation unit, from an intelligent system procedure.
Conclusions

- Although the work described in this paper is only concerned with black and white images, the architecture can be enhanced to handle grayscale images as well.

- A programming tool was also presented, aimed to automate the process of reconfiguring the architecture to handle images according to their dimensions.
Conclusions

- Based on the image dimensions and instruction opcodes, the tool generates synthesizable Verilog that can be used for static machine reconfiguration.
- Experimental results have shown that both, the programming tool and architecture operate as intended in terms of performance.
Future Work

*Verilog*
Acknowledgments

The authors would like to thank the Brazilian agencies FAPESP (grant no. 12/13899-9), and CNPq (grant no. 478084/2012-9), for their financial support.