Parallel Reconfigurable Hardware Architectures for Video Processing Applications

PhD Dissertation Presented by

Karim Ali

Supervised by

Prof. Jean-Luc Dekeyser
Dr. HdR. Rabie Ben Atitallah
Autonomous Electric Vehicles

1. Velodyne at 2.3 m
   Scan Angle = 360°

2. Stereo camera at 2.3 m

3. Stereo camera at 1.2 m

4. SICK at 0.74 m
   Scan Angle = 180°

5. SICK at 0.3 m
   Scan Angle = 180°
Why FPGAs for smart camera applications?

- Exploiting parallelism in video processing
- Reprogrammable platforms
- Low-power consumption
- Several functionalities can be integrated on a single Chip
- Flexible devices
- Dynamic reconfigurable architectures
- Exploiting parallelism in video processing
Design Challenges: Video Applications

01 Intensive signal application
For example, an image sensor of 60 colorful HD frame/s delivers 2.7 Gbps.

02 Real-time constraints
Video applications are processed under real-time constraints.

03 Multiple image sensors
Multiple image sensors are installed for obstacle detection, tracking and classification.

04 Image resolution and frame rate
There is a continuous demand for better resolution and frame rate.
Frame rate vs stopping distance

Assumptions:
1) ~7 frame latency
2) Braking distance: URL

Car Speed: 120 Km/Hr:
Car Speed: 100 Km/Hr:
Car Speed: 80 Km/Hr:
Car Speed: 60 Km/Hr:
Car Speed: 30 Km/Hr:

Frame-rate

Stopping Distance In Meters

©EETimes: https://www.eetimes.com/author.asp?doc_id=1329109
### Design Challenges: Parallel Architectures

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td><strong>Flexible parallel reconfigurable architecture</strong></td>
<td>Intensive signal applications where a huge amount of data is transferred from/to the computing nodes</td>
</tr>
<tr>
<td>02</td>
<td><strong>Exploiting inherent parallelism</strong></td>
<td>How can we choose the parallelism level to exploit the inherent parallelism in the application?</td>
</tr>
<tr>
<td>03</td>
<td><strong>Managing input/output pixel distribution</strong></td>
<td>How can we manage input/output pixel distribution to guarantee high processing rate?</td>
</tr>
<tr>
<td>04</td>
<td><strong>Low power consumption</strong></td>
<td>How can we scale the operating frequency to reduce the system power consumption without affecting the processing rate?</td>
</tr>
</tbody>
</table>
Design Challenges: Design Automation

01. Large FPGAs
Using conventional ways to design large FPGAs is a time-consuming process.

02. Design efforts
Design efforts are moved to higher abstraction levels to increase design productivity and to shorten time-to-market constraints.

03. Design space exploration
There is no unique design solution but a space of different design points. Automating the exploration process is necessary to rapidly find the solution which fits with design constraints.
Contributions

Flexible Parallel Reconfigurable Architecture

- Generic model for pixel distribution/collection
- HW parallelism for power consumption reduction

Efficient HW for Multi-window SAD stereo matching

- Developing ViPar tool for design space exploration
- Automatic code generation for parallel architectures

Design Automation Tool
Reconfigurable Architectures for Video Applications

01. Heterogenous Reconfigurable Platforms (Zynq

02. Soft Vector Processors (VESPA

03. Soft GPGPU Processors (FGPU

04. Soft VLIW Processors

05. Multi-FPGA Architectures

06. Dedicated Reconfigurable Architectures

1- Xilinx©
2- Yiannacouras[2012]
3- Al Kadi[2016]
4- Brost[2014]
5- Viswanathan[2015]
6- Tziortzios[2013]
FPGA Design Methodologies

HLS Design Space Exploration

DSE inside HLS tool
- Sengupta [2011]
- Prost-Boucle [2014]

DSE with HLS tool
- Schafer [2016]
- Zhong [2016]
- Lo [2016]
- Schafer [2017]

DSE Algorithm
- Genetic, Gao [2013]
- Machine learning, Liu [2013]
- Sim. annealing, Mahapatra [2014]
- Ant colony, Schafer [2016]
Outline

01. Generic Pixel Distribution/Collection Model
02. Hardware Parallelism for Reducing Power Consumption
03. Efficient Hardware Implementation for Multi-window SAD Algorithm
04. ViPar: A Tool for Design Space Exploration
Outline

1. Generic Pixel Distribution/Collection Model
2. Hardware Parallelism for Reducing Power Consumption
3. Efficient Hardware Implementation for Multi-window SAD Algorithm
4. ViPar: A Tool for Design Space Exploration
Video processing can be classified into single-pixel or macro-block.

In macro-block processing, the processing window slides horizontally and vertically.

We defined a generic model for flexible pixel distribution in both directions.

The hardware architecture was generated automatically to reduce the design efforts.

Experimental Validation: video downscaler (16:1) and convolution filter.
Model Parameters

- `frame_wid`
- `proc_num_cols`
- `H`
- `V`
- `hor_slide`
- `ver_slide`
- `frame_len`
- `proc_num_lines`
- `N`
Pixel Distributor Architecture

- **clk**
- **rst**

**Line Buffers**
- line buffer 1
- line buffer 2
- line buffer 3
- ... line buffer V

**Circular Vertical Shifter**

**Horizontal Shift Register**
- pixel< 1 >
- ...
- pixel< H >
- ...
- pixel< 2*H >
- ...
- pixel< 3*H >
- ...
- pixel< V*H >

**Controller**
- sof
- valid

**Video Data**
- wr_clk
- rd_clk
- wr_addr
- rd_addr
- wr_en_buff(i)
- vblank
- hblank
- act_video

**Pixel Distributor**
Pixel Collector Architecture

### Experimental Results: Code Generation

<table>
<thead>
<tr>
<th>Generated Files</th>
<th>Number of Code Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4x4</td>
</tr>
<tr>
<td>pixel_distributor.vhd</td>
<td>500</td>
</tr>
<tr>
<td>(Pixel distributor top level)</td>
<td></td>
</tr>
<tr>
<td>circular_ver_shifter.vhd</td>
<td>80</td>
</tr>
<tr>
<td>(Circular vertical shifter)</td>
<td></td>
</tr>
<tr>
<td>hor_shift_register.vhd</td>
<td>83</td>
</tr>
<tr>
<td>(Horizontal shift register)</td>
<td></td>
</tr>
<tr>
<td>buffer.vhd</td>
<td>60</td>
</tr>
<tr>
<td>(Line buffers)</td>
<td></td>
</tr>
<tr>
<td>Total generated code lines</td>
<td>723</td>
</tr>
</tbody>
</table>
## Experimental Results: Video downscaler (16:1)

<table>
<thead>
<tr>
<th>Component</th>
<th>Register</th>
<th>LUT</th>
<th>BRAM18</th>
<th>BRAM36</th>
<th>DSP48E1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Video processing system architecture</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video timing controller 0</td>
<td>2302</td>
<td>2344</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VITA image sensor</td>
<td>5941</td>
<td>6331</td>
<td>0</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>Image pipeline</td>
<td>10565</td>
<td>9733</td>
<td>19</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>RGB-to-YCbCr422</td>
<td>254</td>
<td>202</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td><strong>Video downscaler (1:16)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pixel distributor (R,G,B)</td>
<td>669</td>
<td>639</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Video scaling (R,G,B)</td>
<td>1140</td>
<td>756</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DeMux / Mux (R,G,B)</td>
<td>93</td>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pixel collector (R,G,B)</td>
<td>154</td>
<td>307</td>
<td>0</td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>Pixel distribution area</td>
<td>2056</td>
<td>1753</td>
<td>12</td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total application area</strong></td>
<td>21118</td>
<td>20363</td>
<td>31</td>
<td>119</td>
<td>17</td>
</tr>
<tr>
<td><strong>Resource Utilization (%)</strong></td>
<td>4,83</td>
<td>9,32</td>
<td>2,8</td>
<td>22</td>
<td>1,89</td>
</tr>
</tbody>
</table>
Outline

01

Generic Pixel Distribution/Collection Model

02

Hardware Parallelism for Reducing Power Consumption

03

Efficient Hardware Implementation for Multi-window SAD Algorithm

04

ViPar: A Tool for Design Space Exploration
HW Parallelism for Power Reduction

01. Frequency scaling is used as a technique for reducing power consumption.

02. We calculated the parallelism level required to keep the same processing rate.

03. We formulated the equations for calculating the maximum FIFO depth.

04. By varying the parallelism level and operating frequency, we obtained set of different design points in hardware utilization and power consumption.
HW Parallelism for Power Reduction

60 HD frame/s (FREQ1 = 148.5 MHz)
HW Parallelism for Power Reduction

- Processing condition to bound the maximum depth of FIFO:
  \[ \text{Processing time} \left| \text{macro-blocks} \right| \leq \text{idle time} \]
Calculating Level of Parallelism

\[
\text{Level of Parallelism} = \frac{\text{comp\_delay} \times N\_mblocks \times \text{rd\_clk}}{\text{distributing\_cycle}}
\]

\[
= \frac{\text{comp\_delay} \times N\_mblocks \times \text{rd\_clk}}{V \times \text{line\_period} \times \text{wr\_clk}}
\]

\[
= \frac{\text{comp\_delay} \times N\_mblocks \times \frac{1}{\text{FREQ2}}}{V \times \text{line\_period} \times \frac{1}{\text{FREQ1}}}
\]

\[
= \frac{\text{comp\_delay} \times N\_mblocks \times \text{FREQ1}}{V \times \text{line\_period} \times \text{FREQ2}}
\]

Where,
- \(\text{distributing\_cycle}\) is the time required to stream \(V\) lines of pixels,
- \(\text{comp\_delay}\) is the number of clock cycles required by PE to process one macro-block,
- \(N\_mblocks\) is the number of macro-blocks per distributing cycle,
- \(V\) is the vertical dimension of the macro-block,
- \(\text{line\_period}\) is the time required to stream one line of pixels in the horizontal direction,
- \(\text{wr\_clk}\) is the clock period for FIFO writing clock frequency (FREQ1)
- \(\text{rd\_clk}\) is the clock period for FIFO reading clock frequency (FREQ2)
• **Case 1:** When all processing elements are not yet activated by the end of distributing time \( (N_{PE} \times rd_{clk} > distributing_{time}) \)

\[
\text{FIFO depth} = N_{mblocks} - N_{act_{PE}} + 2 \\
= N_{mblocks} - \frac{N_{pixels\_line} \times FREQ2}{FREQ1} + 2
\]
Calculating FIFO Depth

**Case 2:** When all processing elements are activated at least once during the distributing time \((N_{PE} \cdot rd\_clk \leq distributing\_time)\)

\[
\text{FIFO depth} = N_{mblocks} - \frac{distributing\_time \cdot N_{PE}}{rd\_clk \cdot comp\_delay} + 2
\]

\[
= N_{mblocks} - \frac{N_{pixels\_line} \cdot \text{FREQ2} \cdot N_{PE}}{\text{FREQ1} \cdot comp\_delay} + 2
\]
### Experimental Results: Design Points

<table>
<thead>
<tr>
<th>Design Point</th>
<th>Level of Parallelism</th>
<th>CLK1 (MHz)</th>
<th>CLK2 (MHz)</th>
<th>FIFO depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>3</td>
<td>148,5</td>
<td>148,5</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>3</td>
<td>148,5</td>
<td>74,25</td>
<td>242</td>
</tr>
<tr>
<td>D3</td>
<td>3</td>
<td>148,5</td>
<td>37,125</td>
<td>362</td>
</tr>
<tr>
<td>D4</td>
<td>6</td>
<td>148,5</td>
<td>74,25</td>
<td>2</td>
</tr>
<tr>
<td>D5</td>
<td>6</td>
<td>148,5</td>
<td>37,125</td>
<td>242</td>
</tr>
<tr>
<td>D6</td>
<td>6</td>
<td>148,5</td>
<td>18,5625</td>
<td>362</td>
</tr>
<tr>
<td>D7</td>
<td>12</td>
<td>148,5</td>
<td>37,125</td>
<td>2</td>
</tr>
<tr>
<td>D8</td>
<td>12</td>
<td>148,5</td>
<td>18,5625</td>
<td>242</td>
</tr>
<tr>
<td>D9</td>
<td>12</td>
<td>148,5</td>
<td>9,28125</td>
<td>362</td>
</tr>
</tbody>
</table>

**AES Encryption**
Power Analysis: AES Encryption

![Power Consumption Graph](image)

- Estimated power
- Measured power
- Slice register cost

Power Consumption:
- 1038 mW
- 982 mW
Power Analysis: Video Downscaler (16:1)

1289 mW

1037 mW
Estimated vs Measured Power

Video downscaler

- Estin Meas: 53%
- Clocks: 7%
- Static: 11%
- BRAM: 15%
- Signals & Logic: 14%
- Other: 14%

AES encryption

- Estin Meas: 52%
- Clocks: 11%
- Static: 11%
- BRAM: 12%
- Signals & Logic: 14%
- Other: 14%
Outline

01. Generic Pixel Distribution/Collection Model
02. Hardware Parallelism for Reducing Power Consumption
03. Efficient Hardware Implementation for Multi-window SAD Algorithm
04. ViPar: A Tool for Design Space Exploration
### Stereo Matching Algorithm

![Stereo Matching Diagram](image)

The formula for depth calculation is:

\[
\text{depth} = \frac{\text{baseline} \ast \text{focal length}}{\text{disparity}} = \frac{b \ast f}{(X_R - X_L)}
\]
5-window SAD Stereo Algorithm

Score at Pixel \( P \) = \( \text{Score}_E + \min \text{ two score values } \{ \text{Score}_A, \text{Score}_B, \text{Score}_C, \text{Score}_D \} \)
High-level Synthesis Optimizations

Optimizations for hardware implementation

- Dividing an image into strips
- Arbitrary precision data types
- I/O communication protocol
- Grouping pixels at I/O ports

Optimizations for exploiting the inherent parallelism

- Task-level parallelism
- Pipeline-level parallelism
- Data-level parallelism
### High-level Synthesis Optimizations

<table>
<thead>
<tr>
<th>Design</th>
<th>Optimization</th>
<th>Slice</th>
<th>FF</th>
<th>LUT</th>
<th>BRAM 18K</th>
<th>Exec. Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW version</td>
<td>380 ms on core i7 @ 2.7 GHz and 16 GB of RAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#1</td>
<td>First synthesis</td>
<td>X</td>
<td>2637</td>
<td>5918</td>
<td>7392</td>
<td>X</td>
</tr>
<tr>
<td>#2</td>
<td>Horizontal-Vertical aggregation</td>
<td>898</td>
<td>1743</td>
<td>2735</td>
<td>155</td>
<td>30080</td>
</tr>
<tr>
<td>#3</td>
<td>Vertical-Horizontal aggregation</td>
<td>859</td>
<td>1758</td>
<td>2659</td>
<td>113</td>
<td>22410</td>
</tr>
<tr>
<td>#4</td>
<td>All pixels aggregation</td>
<td>1400</td>
<td>2552</td>
<td>3738</td>
<td>75</td>
<td>8163</td>
</tr>
<tr>
<td>#5</td>
<td>Arbitrary precision data type</td>
<td>983</td>
<td>1525</td>
<td>2567</td>
<td>47</td>
<td>5786</td>
</tr>
<tr>
<td>#6</td>
<td>Adding TLAST</td>
<td>996</td>
<td>1575</td>
<td>2619</td>
<td>49</td>
<td>6307</td>
</tr>
<tr>
<td>#7</td>
<td>Group pixels</td>
<td>1135</td>
<td>1820</td>
<td>3080</td>
<td>49</td>
<td>5865</td>
</tr>
<tr>
<td>#8</td>
<td>Task-level Parallelism (pipe1)</td>
<td>1110</td>
<td>2002</td>
<td>3339</td>
<td>67</td>
<td>2658</td>
</tr>
<tr>
<td>#9</td>
<td>Calculating 4 disparity lines (pipe4)</td>
<td>2790</td>
<td>4578</td>
<td>7796</td>
<td>102</td>
<td>815</td>
</tr>
<tr>
<td>#10</td>
<td>Calculating 8 disparity lines (pipe8)</td>
<td>5012</td>
<td>8502</td>
<td>14027</td>
<td>204</td>
<td>432</td>
</tr>
<tr>
<td>#11</td>
<td>Calculating 12 disparity lines (pipe12)</td>
<td>6594</td>
<td>12563</td>
<td>18476</td>
<td>252</td>
<td>339</td>
</tr>
<tr>
<td>#12</td>
<td>Loop pipelining</td>
<td>1161</td>
<td>2004</td>
<td>3546</td>
<td>67</td>
<td>1174</td>
</tr>
<tr>
<td>#13</td>
<td>Removing false dependency</td>
<td>1115</td>
<td>2030</td>
<td>3433</td>
<td>67</td>
<td>1002</td>
</tr>
<tr>
<td>#14</td>
<td>Data-level parallelism</td>
<td>2771</td>
<td>6365</td>
<td>8155</td>
<td>59</td>
<td>313</td>
</tr>
</tbody>
</table>
### Pipeline-level Parallelism (Code Restructuring)

<table>
<thead>
<tr>
<th>Design</th>
<th>Optimization</th>
<th>Slice</th>
<th>FF</th>
<th>LUT</th>
<th>BRAM 18K</th>
<th>Exec. (ms)</th>
<th>Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>#8</td>
<td>Calculating 1 disparity lines (pipe1)</td>
<td>1110</td>
<td>2002</td>
<td>3339</td>
<td>67</td>
<td>2658</td>
<td>70 %</td>
</tr>
<tr>
<td>#9</td>
<td>Calculating 4 disparity lines (pipe4)</td>
<td>2790</td>
<td>4578</td>
<td>7796</td>
<td>102</td>
<td>815</td>
<td>47 %</td>
</tr>
<tr>
<td>#10</td>
<td>Calculating 8 disparity lines (pipe8)</td>
<td>5012</td>
<td>8502</td>
<td>14027</td>
<td>204</td>
<td>432</td>
<td>22 %</td>
</tr>
<tr>
<td>#11</td>
<td>Calculating 12 disparity lines (pipe12)</td>
<td>6594</td>
<td>12563</td>
<td>18476</td>
<td>252</td>
<td>339</td>
<td></td>
</tr>
</tbody>
</table>
Experimental Setup

Diagram showing the architecture of a parallel reconfigurable hardware system for video processing applications. The system includes components such as AXI interconnects, AXI DMA modules, an HLS_Laplace filter, HLS_SAD, and an HDMI interface. The diagram illustrates the connections and signals between these components, highlighting the processing system, programmable logic, and relevant interfaces such as Ethernet and S_AXI_LITE.
Outline

01
Generic Pixel Distribution/Collection Model

02
Hardware Parallelism for Reducing Power Consumption

03
Efficient Hardware Implementation for Multi-window SAD Algorithm

04
ViPar: A Tool for Design Space Exploration
ViPar: A Tool for Design Space Exploration

ViPar tool automates the design space exploration through the following steps:

2. Introducing an empirical power model to estimate the power consumption.
3. Generating the high-level codes for the parallel video architectures.

Different design solutions with different system constraints (area, power, frequency, performance, etc).
ViPar Design Flow

**Initial Design**
- Initial Design Metrics

**Design Metrics Estimation**
- Area Estimation
- Power Estimation
- Performance Estimation

**Estimations**
- Design constraints
- Metrics Estimation for Different Design Alternatives

**Design Space Exploration**

**Candidate Designs**
- System Specification File
- Processing Element C++ File

**Automatic High-level Code Generation**

**HLS Files**

**ViPar Tool**
Hardware Resource Estimation

Default strategies at 100 MHz  
Default strategies at 200 MHz  
Performance_Explore at 100 MHz
LUT, FF, BRAM and Slice Estimation

For estimating LUT, FF, BRAM and Slice utilization:

\[
\text{Estimated Utilization}_{\text{parallelism level } = N} = \text{Base cost} + N \times \text{Utilization}_{\text{parallelism level } = 1}
\]

Where,

* **Base cost** is the hardware cost for the basic blocks in the design,

* **N** is the level of parallelism.
Power Estimation Model

50 MHz ------- 100 MHz ------- 150 MHz ------- 200 MHz -------
Statistical Results for Regression Models

<table>
<thead>
<tr>
<th></th>
<th>Linear model</th>
<th>Pure-Quadratic model</th>
<th>Full-Quadratic model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression SS</td>
<td>6384.396</td>
<td>6957.231</td>
<td>6958.076</td>
</tr>
<tr>
<td>Residual SS</td>
<td>614.766</td>
<td>41.93071</td>
<td>41.0856</td>
</tr>
<tr>
<td>Total SS</td>
<td>6999.162</td>
<td>6999.162</td>
<td>6999.162</td>
</tr>
<tr>
<td>Significance F</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Multiple R</td>
<td>0.955074</td>
<td>0.997</td>
<td>0.997061</td>
</tr>
<tr>
<td>R-square (R^2)</td>
<td>0.912166</td>
<td>0.994009</td>
<td>0.99413</td>
</tr>
<tr>
<td>Adjusted R^2</td>
<td>0.912163</td>
<td>0.994009</td>
<td>0.994129</td>
</tr>
<tr>
<td>Observations</td>
<td>100152</td>
<td>100152</td>
<td>100152</td>
</tr>
</tbody>
</table>

Where,

**Significance-F** is the probability that the regression equation does not explain the variation in the dependent variable (power).

**Multiple-R** explains how strong the relationship between the dependent and independent variables is,

**R^2** tells how many points fall on the regression line,
Evaluating the Unknown Coefficient $\beta$

- **Full-quadratic:**

$$
\text{Power} = \beta_0 + \beta_1 \times \text{Slice} + \beta_2 \times \text{BRAM} + \beta_3 \times \text{Frequency} + \beta_4 \times \text{Slice} \times \text{BRAM} + \beta_5 \times \text{Slice} \times \text{Frequency} + \beta_6 \times \text{BRAM} \times \text{Frequency} + \beta_7 \times \text{Slice}^2 + \beta_8 \times \text{BRAM}^2 + \beta_9 \times \text{Frequency}^2
$$

- **Null hypothesis** is named "null" because we try to nullify it, but it does not mean that the statement is null by itself (for example, the null hypothesis for the regression model is $\beta=0$).

- **P-value** is used in the hypothesis test for either supports or rejects the null hypothesis.
Evaluating the Unknown Coefficient $\beta$

- We chose confidence level = 99%, so the null hypothesis is rejected if P-value ≤ 0.01.

<table>
<thead>
<tr>
<th>Intercept</th>
<th>Slice * Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coefficient $\beta_0$ = 0.222776</td>
</tr>
<tr>
<td></td>
<td>P-value = 0</td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_5$ = 7.03e-08</td>
</tr>
<tr>
<td></td>
<td>P-value = 0</td>
</tr>
<tr>
<td>Slice</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_1$ = 1.29e-06</td>
</tr>
<tr>
<td></td>
<td>P-value = 3e-161</td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_6$ = 5.51e-07</td>
</tr>
<tr>
<td></td>
<td>P-value = 0</td>
</tr>
<tr>
<td>BRAM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_2$ = 9.39e-05</td>
</tr>
<tr>
<td></td>
<td>P-value = 8.81e-31</td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_7$ = -4.6e-11</td>
</tr>
<tr>
<td></td>
<td>P-value = 0</td>
</tr>
<tr>
<td>Frequency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_3$ = 0.000116</td>
</tr>
<tr>
<td></td>
<td>P-value = 2.28e-60</td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_8$ = -9.5e-08</td>
</tr>
<tr>
<td></td>
<td>P-value = 1.3e-147</td>
</tr>
<tr>
<td>Slice * BRAM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_4$ = 2.38e-09</td>
</tr>
<tr>
<td></td>
<td>P-value = 1.19e-54</td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_9$ = 4.91e-07</td>
</tr>
<tr>
<td></td>
<td>P-value = 9.91e-81</td>
</tr>
<tr>
<td>BRAM $^2$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_8$ = -9.5e-08</td>
</tr>
<tr>
<td></td>
<td>P-value = 1.3e-147</td>
</tr>
<tr>
<td>Frequency $^2$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coefficient $\beta_9$ = 4.91e-07</td>
</tr>
<tr>
<td></td>
<td>P-value = 9.91e-81</td>
</tr>
</tbody>
</table>
Automatic High-level Code Generation

Parallel Reconfigurable Hardware Architectures for Video Processing Applications, PhD dissertation by Karim M. A. Ali
System Specification File

```plaintext
### System_Properties ###
input_image.width = 640
input_image.height = 480
output_image.width = 320
output_image.height = 240
Parallelism_Level = 32
### ENDOF_System_Properties ###

### Processing_Element ###
Name = VideoDownScaler
Num_of_inputs = 1
Input_0.name = image[IMG_WIDTH][WIN_HEIGHT]
Input_0.type = unsigned char
Input_0.src = data_img[STRIP_SIZE_PARA32_8]
Input_0.store_scanlines_from = 0
Input_0.store_scanlines_to = 1
Input_0.shift_step = 2
Num_of_outputs = 1
Output_0.name = image_result[IMG_WIDTH_2]
Output_0.type = unsigned char
Output_0.sink = img_result[IMG_WIDTH_2_PARA32_8]
Output_0.store_scanlines_from = 0
Output_0.store_scanlines_to = 0
Output_0.shift_step = 1
### ENDOF_Processing_Element ###
```

- Number of parallel processing elements
- For video downscaler (4:1), two scanlines are distributed to each processing element
Automatic High-level Code Generation

**Candidate Designs Input Files**
- System Specification File
- Processing Element C++ File

**Properties Extraction**
- Extracting system properties (parallelism level, input/output image size)
- Extracting TopLevelFunc port properties for input ports ($i = 0 \rightarrow m$) and output ports ($O = 0 \rightarrow n$)
- Extracting Processing Element port properties for input ports ($X = U \rightarrow i$) and output ports ($Y = 0 \rightarrow j$)

**Generating Input/Output Ports Properties for Processing Elements**
- Generating input/output ports properties for Processing Elements $PE = 1 \rightarrow N$
- Generating arrays for both pixel distributor and pixel collector

**PE Properties Generation**
- Storing scanlines for each input $i_0, i_1, ..., i_n$ in the arrays according to the distribution pattern
- Generating N parallel instances of the processing element $PE \rightarrow 0 \rightarrow N$
- Streaming the outputs $O_0, O_1, ..., O_n$ by reading pixel collector arrays

**Building Parallel Architecture**

**HLS Output Files**
- Design C++ Files

2/8/2018 Parallel Reconfigurable Hardware Architectures for Video Processing Applications, PhD dissertation by Karim M. A. Ali 48
Experimental Results: Design Points

Pipeline parallelism (pipe4, pipe8, pipe12)

Operating frequency (100, 150, 200 MHz)

Number of parallel processing elements (1, 2, 3, 4, ......)

<table>
<thead>
<tr>
<th>#</th>
<th>Freq. in MHz</th>
<th>level of Parallelism</th>
<th>Slice (54650)</th>
<th>FF (347200)</th>
<th>LUT (218600)</th>
<th>BRAM (1090)</th>
<th>Power in mW</th>
<th>Frame Exec. Time in ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>1</td>
<td>10534</td>
<td>28903</td>
<td>31163</td>
<td>131</td>
<td>342.91</td>
<td>84.875</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>44571</td>
<td>150451</td>
<td>178283</td>
<td>803</td>
<td>692.85</td>
<td>15.7</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>50701</td>
<td>170709</td>
<td>262833</td>
<td>915</td>
<td>710.87</td>
<td>14.32</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>1</td>
<td>10111</td>
<td>27410</td>
<td>31140</td>
<td>131</td>
<td>390.71</td>
<td>56.733</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>17754</td>
<td>46175</td>
<td>55636</td>
<td>243</td>
<td>494.28</td>
<td>29.975</td>
</tr>
<tr>
<td>20</td>
<td>200</td>
<td>1</td>
<td>33616</td>
<td>114895</td>
<td>129040</td>
<td>579</td>
<td>929.78</td>
<td>10.387</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1</td>
<td>13608</td>
<td>47902</td>
<td>48170</td>
<td>203</td>
<td>391.73</td>
<td>44.85</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td>28745</td>
<td>87339</td>
<td>89700</td>
<td>387</td>
<td>510.4</td>
<td>24.838</td>
</tr>
<tr>
<td>29</td>
<td>150</td>
<td>1</td>
<td>16025</td>
<td>44339</td>
<td>48591</td>
<td>203</td>
<td>729.75</td>
<td>12.167</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>200</td>
<td>1</td>
<td>16278</td>
<td>48644</td>
<td>53711</td>
<td>203</td>
<td>548.49</td>
<td>22.574</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>100</td>
<td>1</td>
<td>26073</td>
<td>67071</td>
<td>66804</td>
<td>275</td>
<td>475.8</td>
<td>30.965</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td>49292</td>
<td>125497</td>
<td>120967</td>
<td>531</td>
<td>652.76</td>
<td>17.854</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td>46783</td>
<td>183923</td>
<td>187130</td>
<td>787</td>
<td>674.48</td>
<td>12.774</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td>24133</td>
<td>61240</td>
<td>69833</td>
<td>275</td>
<td>567.12</td>
<td>20.603</td>
</tr>
<tr>
<td>41</td>
<td>150</td>
<td>1</td>
<td>45392</td>
<td>113835</td>
<td>133017</td>
<td>531</td>
<td>818.27</td>
<td>11.927</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td>49051</td>
<td>166430</td>
<td>190201</td>
<td>787</td>
<td>893.56</td>
<td>8.531</td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
<td>22127</td>
<td>67339</td>
<td>68978</td>
<td>275</td>
<td>646</td>
<td>15.581</td>
</tr>
<tr>
<td>44</td>
<td>200</td>
<td>1</td>
<td>41461</td>
<td>128033</td>
<td>131286</td>
<td>551</td>
<td>957.05</td>
<td>8.976</td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td>48990</td>
<td>184727</td>
<td>193504</td>
<td>787</td>
<td>1063.59</td>
<td>6.418</td>
</tr>
</tbody>
</table>
Hardware Utilization Estimation Error

![Graph showing hardware utilization estimation error for different designs](image)

- **Design Number**
- **% Estimation Error**
- **Slice**
- **LUT**
- **FF**
Power Consumption Estimation Error

![Graph showing the power consumption estimation error for different design numbers. The x-axis represents the design number, and the y-axis represents the percentage estimation error. The graph shows a range of error values from approximately -60% to -40%.](image-url)
Estimated Power vs Measured Power

![Graph showing estimated and measured power vs design number]

- Estimated power
- Measured power
Design Space Exploration

System constraints:
- \( \text{LUT} \leq 150000 \)
- \( \text{FF} \leq 120000 \)
- \( \text{BRAM} \leq 700 \)
- Frequency \( \leq 150 \text{ MHz} \)
- Execution time \( \leq 15 \text{ ms} \)
Design Space Exploration

Design constraints
- design #7
- design #31
- design #42
- design #43
Conclusion

01 We used FPGAs to build parallel architectures for video processing applications.

02 There is a continuous demand for increasing the frame rate and image resolution.

03 Different challenges were tackled to implement parallel reconfigurable architectures.

04 We presented a generic model for pixel distribution/collection (Ali_ReConFig_2014).
Conclusion

05. Hardware parallelism was used for reducing power consumption (Ali_ReCoSoC_2015).

06. We used HLS tool to build an efficient hardware implementation for 5-window SAD stereo matching algorithm (Ali_ARC_2017).

07. We developed ViPar tool for design space exploration
   - By estimating design metrics like area, power and performance
   - Automatic code generation for parallel architecture.
Perspectives

01 Multi-application design space exploration
- In autonomous vehicle domain, it is normal to have multiple applications running concurrently for object tracking and classification.
- Application profiling to figure out the bottleneck computation tasks.
- Searching algorithms (exact or heuristic) for DSE.
- Exploring how the tasks are mapped either to CPU or FPGA

02 Self-adaptivity
- The ability of the system to adapt itself due to external changes.
- The quality of results can be affected by the environmental conditions.
- FPGAs are dynamic and partial reconfigurable architectures.
- Self-adaptivity can be used to avoid system failures scenarios.
- Two different self-adaptive scenarios: predefined or at run-time
Perspectives

System scalability
- Several functions are integrated for detecting traffic light signs, lanes, etc.
- Multi-FPGA boards are common in the market.
- Exploring which tasks are mapped to which FPGA then to explore the different alternatives for implementation on the selected FPGA.
- Can the same application be mapped to two different FPGAs

Deep learning and reconfigurable technology
- Autonomous vehicles have intelligent behavior like steering to avoid obstacles.
- Multi-sensor data fusion is required for correct decision making.
- Using deep learning in video applications for obstacle detection, tracking and classification for autonomous vehicles.
- Examples: DRIVE PX from Nvidia and GO COMPUTER from Intel.


THANK YOU

KARIM ALI