Self-Adaptive FPGA-Based Image Processing Filters Using Approximate Arithmetics

Jutta Pirkl, Andreas Becher, Jorge Echavarria, Jürgen Teich, and Stefan Wildermann
Hardware/Software Co-Design, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
SCOPES'17, St. Goar, Germany, June 12, 2017
Approximate Computing – A New Design Paradigm

Underlying Idea

Trading **accuracy** of computations against disproportionate improvements with respect to **power consumption** and/or **performance** and/or **circuit area**.

Motivation

Problem:

• Error-tolerance depends on both input data and application context
• **Quality-configurability** is the key principle of prospective AC platforms

**Approach**: Dynamic autonomous swapping of filters with different degrees of approximation utilizing reconfigurable hardware

---

Outline

Concepts of Self-Adaptive Image Processing
  Approximate 2D-Convolution Filters
  Quality Evaluation
  Reconfiguration Management

Experimental Results
  Quality-Configurable Control Mechanism
  Partial Reconfiguration Overhead

Summary
Concepts of Self-Adaptive Image Processing
Approximate 2D-Convolution Filters

- **Basic filter building block**: 2D-convolution filter wrapper with a kernel size of $3 \times 3$

  \[ Y[m, n] = \sum_i \sum_j H[i, j] \cdot X[m - i, n - j] \]

- Parallel Multiply-Accumulate (MAC) operation in a pipelined adder tree structure
- Replacement of all adders by the same approximate version
Approximate Adder Structures on FPGAs

Most Significant Part

Least Significant Part

Case 1: Carry suppression

MSP LSP

0 1 1 1 1 1 1 1
+ 0 0 1 0 1 1 1 1
1 1 1 1 1 1 1 1

0 1 1 1 1 1 1 1
+ 0 0 1 0 1 1 1 1
1 0 0 0 0 0 0 0

Approximate Adder Structures on FPGAs

**Case 1: Carry suppression**

MSP  LSP

\[\begin{array}{cccccc}
0 & 1 & 1 & 1 & 1 & 1 \\
+ & 0 & 0 & 1 & 0 & 1 \\
\hline
1 & 0 & 0 & 1 & 1 & 1 \\
\end{array} \quad \begin{array}{c}
31 \\
11 \\
11 \\
39 \text{ or } 42 \\
\end{array} \]

⇒ error reduction mechanism

---

Approximate Adder Structures on FPGAs

Most Significant Part

Least Significant Part

Case 1: Carry suppression

\[
\begin{array}{c}
MSP \\
\downarrow \\
0 \ 1 \ 1 \ 1 \ 1 \ 1 \\
+ \ 0 \ 0 \ 1 \: 0 \ 1 \ 1 \\
\hline
1 \ 0 \ 0 \ 1 \ 1 \ 1 \\
\end{array}
\]

⇒ error reduction mechanism

Case 2: Carry prediction

\[
\begin{array}{c}
MSP \\
\downarrow \\
0 \ 1 \ 1 \ 1 \ 1 \ 1 \\
+ \ 0 \ 0 \ 1 \: 1 \ 1 \ 1 \\
\hline
1 \ 0 \ 1 \ 1 \ 1 \ 0 \\
\end{array}
\]

⇒ no approximation error

Case Study – Approximate Gaussian Lowpass Filter

Artifacts:

- Brightness decrease → underestimating adder
- “Cartoon effect”
Impact of the Carry Chain Splitting Point on the Output Quality

Dependency of the average Peak Signal-to-Noise Ratio (PSNR) on $m$ among the Kodak Lossless True Color Image Suite$^3$

\[ \text{Splitting Position of the Carry Chain (m)} \]

\[ \text{Average PSNR [dB]} \]

Quality Evaluation

- **Problem**: Requirement of a *no-reference* metric to assess the quality at runtime

- **Approach**: Feature extraction from the histograms of in- and output images

- More and more pixels are mapped onto exactly the same brightness values → “cartoon”-effect
Quality Evaluation

- Distinctive peaks created by the \( all1 \)-signal → erroneous sums are mapped onto \( 2^{n-m} \) values

- Example: \( m = 9 \), output bit width after normalization \( n = 8 \)
  
  before normalization: \( x_{20} \cdots x_9 \mid 111111111 \)

  after normalization by 16: \( x_{11}x_{10}x_9 \mid 11111 \)

  → smallest collection bin at \( 000111111 \)\(_b \) \( (31\_d) \)

  → further peaks at a distance of \( 2^5 = 32 \)
Quality Evaluation

• Distinctive peaks created by the *all*1-signal
  → erroneous sums are mapped onto $2^{n-m}$ values

• **Example**: $m = 9$, output bit width after normalization $n = 8$
  before normalization: $x_{20} \cdots x_9 | 111111111$
  after normalization by 16: $x_{11}x_{10}x_9 | 11111$
  → smallest collection bin at $000111111_b (31_d)$
  → further peaks at a distance of $2^5 = 32$
Quality Evaluation

- Amount of counters to sample the histograms constitutes a trade-off between overhead and fidelity

- Counting of the pixels with gray levels 31, 63, 127 and 191 in both in- and output image

**Definition QM**

Ratio of the maximum peak height of the four bins and the corresponding amount in the input image

- Large metric value indicates bad quality

Progression of $QM$ with increasing $m$
Case Study – Approximate Gaussian Lowpass Filter

Artifacts:
- Brightness decrease → underestimating adder
- “Cartoon effect”
Reconfiguration Management

- **Objective:**
  Minimization of the critical path while maintaining a given quality boundary → successive approximation of $m$ to the fastest configuration $m_{opt}$

- Based on a bang-bang controller with integrated hysteresis

### Decision logic for setting the degree of approximation

- **Case 1:** Quality is still acceptable
  → in- or decrement the splitting position $m$ in the direction of $m_{opt}$

- **Case 2:** Quality boundary is exceeded
  → in- or decrement $m$ in the opposite direction of $m_{opt}$

- **Case 3:** Quality metric is within dead zone
  → keep configuration
Experimental Results
Results – Input-Based Adaptivity

System behavior at runtime for the approximate Gaussian filter
Results – Input-Based Adaptivity

System behavior at runtime for the approximate Gaussian filter

![Graph showing system behavior over frames with adaptive and static parameters.](image-url)
Results – Input-Based Adaptivity

System behavior at runtime for the approximate Gaussian filter
Results – Input-Based Adaptivity

System behavior at runtime for the approximate Gaussian filter

![Graph showing system behavior](image-url)

- Dynamic behavior
- Static behavior with m = 5
- Static behavior with m = 6

Frames vs. Quality Metric (QM) over time.
Results – Input-Based Adaptivity

System behavior at runtime for the approximate Gaussian filter

![Graph showing system behavior with dynamic and static modes]
Results – Requirement-Based Adaptivity

System evaluation results\(^4,5\)

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{Quality Requirement} & \text{psnr}_{\text{avg}} & \text{PSNR}_{\text{avg}} [\text{dB}] \\
\hline
\text{DoA} & \text{aspen} & \text{redkayak} & \text{snowmnt} & \text{touchdownpass} & \text{pedestrian area} & \text{demo video} \\
\hline
\end{array}
\]

\(^4\)Test videos: Derf’s collection + self-shot demo video, resolution of 640 \(\times\) 480, grayscale 8 bits/pixel

\(^5\)Evaluation Parameters: Adaptation rate of \(\frac{2}{\text{sec}}\) at a frame rate of 30 fps
Analysis of the Partial Reconfiguration Overhead

Reconfiguration time for the partial bitstreams\(^6\)

<table>
<thead>
<tr>
<th>XC7Z020</th>
<th>Partial Bitstream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitstream Size [KB](^7)</td>
<td>637.44</td>
</tr>
<tr>
<td>Reconfiguration Time [ms]</td>
<td>1.62</td>
</tr>
<tr>
<td>Download Rate [MB/s]</td>
<td>374.48</td>
</tr>
</tbody>
</table>

- Approximately linear correlation between configuration time and bitstream size
- Remaining time slot for the filtering process at 30 fps:
  - \(33.33\, \text{ms} - 1.62\, \text{ms} = 31.71\, \text{ms}\)
  - Partial reconfiguration requires \(4.86\%\) of the time frame

\(^6\) This table contains only the largest bitstream among the approximate variants for the Gaussian filter which determines the slowest transfer

\(^7\) Full binary bitstream size for the xc7z020 device: 4,045,564 Bytes
Summary
Summary

• **Challenge**: Input-dependent approximation error behavior requires self-adaptive methods

• Proposition of a *no-reference metric* for online output quality monitoring based on histogram information

• Our concept offers
  • better exploitation of a given error tolerance than static approximation
  • a user *control knob* to select the desired output quality at runtime
Summary

- **Challenge**: Input-dependent approximation error behavior requires self-adaptive methods

- Proposition of a **no-reference metric** for online output quality monitoring based on histogram information

- Our concept offers
  - better exploitation of a given error tolerance than static approximation
  - a user **control knob** to select the desired output quality at runtime
Summary

- **Challenge**: Input-dependent approximation error behavior requires self-adaptive methods

- Proposition of a no-reference metric for online output quality monitoring based on histogram information

- Our concept offers
  - better exploitation of a given error tolerance than static approximation
  - a user **control knob** to select the desired output quality at runtime
Summary

- **Challenge**: Input-dependent approximation error behavior requires self-adaptive methods

- Proposition of a **no-reference metric** for online output quality monitoring based on histogram information

- Our concept offers
  - better exploitation of a given error tolerance than static approximation
  - a user **control knob** to select the desired output quality at runtime

Thank you for listening!
Any questions?
System Level Overview

**Software**

- **Reconfiguration Manager**: Quality-control loop
- **Linux device drivers as hardware interfaces**

**Hardware**

- **Approximate Filter Operators**: Partial bitstreams for various degrees of approximation
- **Quality Evaluation**: Online quality monitoring
Correlation Between the Proposed Quality Metric and PSNR

Inverse relation: Increasing tendency of the metric with decreasing PSNR