FPGA-based Real-Time Super-Resolution System for Ultra High Definition Videos

Zhuolun He, Hanxian Huang, Ming Jiang, Yuanchao Bai, and Guojie Luo
Peking University

FCCM 2018
Ultra High Definition (UHD) Technology

- UHD Television
- UHD Projector
- UHD Phone
- UHD Camera

Content?
- Limited Creators
- High network bandwidth cost
- Huge storage cost
High-Resolution $\leftrightarrow$ Low-Resolution

Desired HR Image $X$

Blur

Down-Sampling

Noise $n$

Observed LR Image $Y$
Spectrum of Super Resolution Methods

Interpolation
- Fast
- Easy to implement
- Blurry results

Model-based
- Interpretable
- High complexity
- Assumed known blur kernel/noise

Example-based
- State-of-the-art quality
- High complexity
- Training data needed

Complicated
Simple
Model-based Method is also Compute-Intensive

Desired HR Image $X$

Blur

Down-Sampling

Noise $n$

Observed LR Image $Y$

Model-based methods may not be needed
• The computation also has a layered structure
• We can use a neural network to approximate
Fact:
Blocks contain DIFFERENT amount of information (NOT equally important)

Insight:
Use DIFFERENT upscaling methods for different blocks
A Hybrid Algorithm

INPUT: LR Image $Y$

1. Crop $Y$ into sub-images $\{y\}$

2.1. $x \leftarrow \text{Upscale}(y)$ IF $M(x) > T$

2.2. ELSE $x \leftarrow \text{CheapUpscale}(y)$

3. Mosaic $X$ with $\{x\}$

OUTPUT: HR Image $X$

M: Total Variation (TV)

Upscale: FSRCNN-s

CheapUpscale: Interpolation
Overall System

Dispatcher

Pipelined Neural Network

Conv(1, 5, 32) → Conv(32, 1, 5) → Conv(5, 3, 5) → Conv(5, 1, 32) → Deconv(32, 9, 1)

Feature Extraction → Shrinking → Mapping → Expanding → Deconvolution

Interpolator

Accelerator

Low-Res Image → High-Res Image
Stencil Access of TV Computation

\[(\forall x)_{\text{offset}} = \text{abs}(x[\text{right}] - x[\text{offset}]) + \text{abs}(x[\text{down}] - x[\text{offset}])\]
Micro-architecture for Stencil Computation

Buffering System for array x

- Buffer1(N-1)
  - x[i][j]...x[i-1][j+2]
  - x[i][j] (x[down])

- Buffer2(1)
  - x[i-1][j+1]
  - x[i-1][j+1] (x[right])
  - x[i-1][j] (x[offset])

Computation Kernel

(\nabla x)_{i,j}
Convolutional Neural Network

Feature Extraction

Shrinking

Mapping

Expanding

Deconvolution

Conv(1, 5, 32)

Conv(32, 1, 5)

Conv(5, 3, 5)

Conv(5, 1, 32)

Deconv(32, 9, 1)

Pipelined Neural Network
Convolution

\[ N_i \] \[ N_i+1 \]
\[ n_i \] \[ c_i \] \[ f_i \]

Input Compute Output

sliding window(s)

1

Conv(c_i, f_i, n_i)
Deconvolution

\[ \text{Deconv} (c_i, f_i, n_i) \]

Input Compute Output

\( N_i \)

\( N_{i+1} \)

\( c_i \)

\( f_i \)

\( n_i \)

sliding window(s)
## Pipeline Balancing

<table>
<thead>
<tr>
<th>Layer</th>
<th>$c_i$</th>
<th>$f_i$</th>
<th>$n_i$</th>
<th>$N_i$</th>
<th>#Mult.</th>
<th>Ideal #DSP</th>
<th>Ideal II</th>
<th>Alloc. #DSP</th>
<th>Alloc. II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extraction</td>
<td>1</td>
<td>5</td>
<td>32</td>
<td>36</td>
<td>819200</td>
<td>201</td>
<td>4076</td>
<td>200</td>
<td>4096</td>
</tr>
<tr>
<td>Shrinking</td>
<td>32</td>
<td>1</td>
<td>5</td>
<td>32</td>
<td>163840</td>
<td>40</td>
<td>4096</td>
<td>32</td>
<td>4096</td>
</tr>
<tr>
<td>Mapping</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>32</td>
<td>202500</td>
<td>50</td>
<td>4050</td>
<td>45</td>
<td>4500</td>
</tr>
<tr>
<td>Expanding</td>
<td>5</td>
<td>1</td>
<td>32</td>
<td>30</td>
<td>144000</td>
<td>35</td>
<td>4115</td>
<td>32</td>
<td>4500</td>
</tr>
<tr>
<td>Deconvolution</td>
<td>32</td>
<td>9</td>
<td>1</td>
<td>30</td>
<td>2332800</td>
<td>573</td>
<td>4072</td>
<td>519</td>
<td>4500</td>
</tr>
<tr>
<td>Overall</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3662340</td>
<td>899</td>
<td>828</td>
<td>4500</td>
</tr>
<tr>
<td>Available (ZC706)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>900</td>
<td>-</td>
</tr>
</tbody>
</table>
Sub-image Size

• Padding
  • $N_i \equiv k + \sum_i^{\#Conv} (f_i - 1)$

• If sub-image size too small
  • Large border-to-block ratio
  • Limited by memory bandwidth

• If sub-image size too large
  • Large feature maps
  • Limited by on-chip BRAM capacity
Sub-image Size vs. Performance vs. #mult.

![Graph showing PSNR and SSIM vs. Block Size and Multiplications vs. Block Size](image-url)
## Overall Comparisons

- Compared six configurations

<table>
<thead>
<tr>
<th>No.</th>
<th>Preprocessing</th>
<th>Upscaling</th>
<th>#Mult.</th>
<th>PSNR(dB)</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
<td>Interpolation</td>
<td>$6.6 \times 10^7$</td>
<td>35.51</td>
<td>0.9138</td>
</tr>
<tr>
<td>2</td>
<td>None</td>
<td>Neural Network</td>
<td>$8.2 \times 10^9$</td>
<td>38.55</td>
<td>0.9421</td>
</tr>
<tr>
<td>3</td>
<td>Blocking</td>
<td>Interpolation</td>
<td>$6.6 \times 10^7$</td>
<td>35.51</td>
<td>0.9138</td>
</tr>
<tr>
<td>4</td>
<td>Blocking</td>
<td>Neural Network</td>
<td>$8.4 \times 10^9$</td>
<td>38.55</td>
<td>0.9420</td>
</tr>
<tr>
<td>5</td>
<td>Blocking</td>
<td>Mixed-Random</td>
<td>$2.2 \times 10^9$</td>
<td>36.10</td>
<td>0.9211</td>
</tr>
<tr>
<td>6</td>
<td>Blocking</td>
<td>Mixed-TV</td>
<td>$2.2 \times 10^9$</td>
<td>37.36</td>
<td>0.9287</td>
</tr>
</tbody>
</table>

- +3.04dB  No Performance Loss
- +1.26dB
- -1.19dB -75%
- >100x

-75%  -1.19dB
Example Outputs

- Configuration 1: None/Interpolation
- Configuration 2: None/Neural Network
- Configuration 3: Blocking/Interpolation
- Configuration 4: Blocking/Neural Network
- Configuration 5: Blocking/Mixed-Random
- Configuration 6: Blocking/Mixed-TV
Summary Flow

• Crop each frame into blocks
  • Suitable for low-level (pixel-level) tasks
  • GOOD: on-chip buffer friendly
  • BAD: Computation overheads

• Dispatch blocks according to TV value
  • Micro-architecture for buffering system

• Fully-pipelined CNN for upscaling
  • Sliding window for convolution/deconvolution
  • Pipeline balancing

• Performance
  • Full-HD (1920x1080) -> Ultra-HD (3940x2160): 31.7fps
Thank you!
TV Threshold vs. Performance vs. #mult.

![Graph showing PSNR, SSIM, and Multiplications vs. TV Threshold]
## Resource Utilizations

<table>
<thead>
<tr>
<th>Component</th>
<th>BRAM</th>
<th>DSP</th>
<th>FF</th>
<th>LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatcher</td>
<td>1</td>
<td>2</td>
<td>618</td>
<td>1138</td>
</tr>
<tr>
<td>Neural Network</td>
<td>178</td>
<td>844</td>
<td>63149</td>
<td>98439</td>
</tr>
<tr>
<td>Interpolator</td>
<td>0</td>
<td>10</td>
<td>1414</td>
<td>3076</td>
</tr>
<tr>
<td>Total</td>
<td>327</td>
<td>858</td>
<td>66261</td>
<td>103714</td>
</tr>
<tr>
<td>Available</td>
<td>1090</td>
<td>900</td>
<td>437200</td>
<td>218600</td>
</tr>
<tr>
<td>Utilization (%)</td>
<td>30</td>
<td>95</td>
<td>15</td>
<td>47</td>
</tr>
</tbody>
</table>