VLSI Design of QCIF to VGA

Asmar Azar Khan  
M.Sc. Project  
Computer Engineering  
Lahore University of Management Sciences

Objective

- Study image scaling and interpolation techniques
- To propose a hardware based approach for QCIF to VGA resolution conversion
Introduction

- Interoperability of multimedia devices
- Each device has different encoder and hence decoding schemes
  - Broadband TV and Video on demand
  - PDA and Mobile
  - Online Gaming
  - Internet Telephony
- Role of Transcoders and Image Scaling
Transcoding

- A steer demand of multimedia on network have given rise to challenges
  - Heterogeneous Encoders and Decoders
  - Bit Rate (video)
  - Delay (voice and video)
  - Quality (multimedia)
- A technique where we change the encoded bit stream on the fly according to receiver compatibility
  - Encoded scheme
  - Error Correction Techniques
  - Spatial and Temporal Resolution

Image Scaling

- Spatial Resolution of Image and Video
  - HDTV (1620 x 1200) etc...
  - PC Monitors (1280 x 800) XVGA
  - PDA (640 x 480) VGA
  - Mobile (176 x 144) QCIF
Literature Review

- Software based approaches
  - Interpolation Techniques
    - Nearest Neighbor
    - Linear
    - Cubic Spline
    - Bicubic Spline
  - Hardware based Nearest Neighbor
  - Simulations results for Advanced Techniques
  - Recently DCT domain Interpolation has been presented

Interpolation

- Whenever an image is desired to be re-sampled
  - It is first interpolated to continuous image
  - Then the image is sampled
Interpolation Methods

- Nearest Neighbor
- Linear
- Bicubic
- Quadratic
- Cubic B Spline

Nearest Neighbor

- Nearest Pixel Value
- Less Complex
- Low Quality
- Edge handling

\[ h(x) = \begin{cases} 
1, & 0 < |x| < 0.5 \\
0, & \text{else} 
\end{cases} \]
Linear Interpolation:

\[ h(x) = \begin{cases} 
1 - |x|, & 0 < |x| < 1 \\
0, & \text{else} 
\end{cases} \]

Quadratic Interpolation

\[ h(x) = \begin{cases} 
A_1|x|^2 + B_1|x| + C_1, & 0 < |x| < 0.5 \\
A_2|x|^2 + B_2|x| + C_2, & 0.5 < |x| < 1.5 \\
0, & \text{else} 
\end{cases} \]
Cubic B-Spline Interpolation

\[ h(x) = \begin{cases} 
\frac{1}{2}x^3 & \text{if } 0 < |x| < 1 \\
-\frac{1}{6}x^3 + x^2 + 2/3 & \text{if } 1 < |x| < 2 \\
0 & \text{else}
\end{cases} \]


Bicubic Interpolation

\[ P(x,y) = (t_1A + 2E + 3I + 4N)*45 + (t_1B + 2F + 3J + 4M)*16 + (t_1C + 2G + 3K + 4O)*47 + (t_1D + 2H + 3L + 4P)*8 \]

Courtesy: Marco Aurelio Nuño-Maganda, National Institute for Astrophysics, Optics and Electronics (INAOE)
Adopted Techniques

QCIF to VGA

Nearest Neighbor

- Interpolate with the nearest pixel value
- Less complexity
- Boundary problem
- Side lobe energy

Courtesy: Thomas M. Lehmann, Survey: Interpolation Methods in Medical Image Processing
QCIF to VGA

144  176  640
QCIF  480  VGA

Non Integer Factor of Scaling

144  176  640
144  480

gcd(176,640) = 16  ⇔  40/11
gcd(144,480) = 48  ⇔  10/3
Nearest Neighbor; Re sampling

- Low pass filter is applied to avoid aliasing
- Up sampling is done first in horizontal direction means column wise and then vertical direction i.e. row wise.
- 40/11 is non integer factor
  - Up sample by 40
  - Down sample by 11
- Similarly 10/3 factor
Preprocessing; Low Pass Filter

- A 7-tap filter used for CCIR-601 to SIF conversion
  \[ [-29 0 88 138 88 0 -29] \times \frac{1}{256} \]

LP Filter Contd.

- Filter is linear phase
- Easy to implement
- Use of power of two easy for hardware
- Fixed point multiplication
- It removes the chances of aliasing to occur
Slice based approach

- A parallel architecture is possible
  - More Area
  - Less delay
- The advantage is speed and real time application (video) is possible

Mapping onto scaled image

There will be 768 blocks/slices of original and scaled image
Memory Requirement

- 11 x 3 ↔ 40 x 10
- 768 blocks in total

<table>
<thead>
<tr>
<th></th>
<th>Memory in</th>
<th>Memory out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row wise</td>
<td>768[176x3]</td>
<td>768[640x10]</td>
</tr>
</tbody>
</table>

Total Memory = 5.33 Mbytes

Post Processing

- A 7-tap filter used to convert SIF to CCIR-601
  
  \[-12 \ 0 \ 140 \ 256 \ 140 \ 0 \ -12 \] * 1/256

- Removes the blocking effects from the interpolated image by introducing blurring

- As nearest neighbor techniques introduces sudden changes due to boundary value problems
Proposed Architecture

Controller based approach
- Distributed memory architecture
- State Machine based hardware
- Pre processing filtering
- Post processing filtering
- Memory Read and Write
Horizontal Scaling
11 to 40 Mapping

Vertical Scaling
Vertical Scaling
3 to 10 Mapping

\[
\begin{align*}
abc & \\
\downarrow & \\
\text{aaaa} & \text{bbbb} & \text{cccc}
\end{align*}
\]

System Block Diagram
Memory Requirement

<table>
<thead>
<tr>
<th>Memory in</th>
<th>Memory out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row wise</td>
<td>144[176]</td>
</tr>
<tr>
<td>Col wise</td>
<td>640[144]</td>
</tr>
</tbody>
</table>

**Total Memory = 522.24Kbytes**

Memory Requirement

\[
176 \times 640 = 112640 \text{ bytes}
\]

\[
640 \times 640 = 409600 \text{ bytes}
\]

**Total Memory Requirement**

\[
= 522 \text{ Kbytes}
\]

Total Memory requirement can be reduced by using block RAM
Hardware Requirement

- Gate Count for 1 Unit = 3795
- This 1 unit will convert 176 into 640
- There will be 144 rows for horizontal scaling
- And 640 cols of 144 length

\[ 144 + 640 = 784 \text{ units} \]
\[ 3795 \times 784 = 2975280 \text{ gates} \]

Delay Requirement

- At 100MHz
- \[ 176 \leftrightarrow 640 \ (900 \mu \text{sec}) \] 90000cc
- \[ 784 \times 900 = 705600 \mu \text{sec} = 0.705 \text{sec} \]
- 30 frames = 0.705sec x 30 = 21.168 sec

<table>
<thead>
<tr>
<th></th>
<th>Memory</th>
<th>Processing Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Architecture</td>
<td>176 + 640</td>
<td>21.168 sec</td>
</tr>
<tr>
<td>Parallel Architecture</td>
<td>522KB</td>
<td>0.705 sec</td>
</tr>
</tbody>
</table>
Future Proposals

- Advanced Interpolation methods
  - Cubic Spline
  - Normal Spline
- Generic Conversion
  - Generic scaling ratio

Questions !
I thank you for your time..😊