GESTURE RECOGNITION USING FPGA

Final Year Project Evaluation
2010 – 2011

15/03/2011
Guide: Jagadeesh Kumar P

<table>
<thead>
<tr>
<th>Roll no</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECU07201</td>
<td>Aarathy Udayabhanu</td>
</tr>
<tr>
<td>ECU07214</td>
<td>Franklin Geo Francis</td>
</tr>
<tr>
<td>ECU07225</td>
<td>Namsheer K</td>
</tr>
<tr>
<td>ECU07228</td>
<td>Nithin Sarath</td>
</tr>
<tr>
<td>ECU07234</td>
<td>Rizwana A</td>
</tr>
</tbody>
</table>

Gesture Recognition using FPGA

15/03/2011
At a Glance….

• Introduction
• Basic Overview
• Video Decoder
• Video Receiver
• Edge Detection & Feature Extraction
• VGA Display
• Project Status
• Future Scope
• References

Gesture Recognition using FPGA

15/03/2011
Gestures Tell A Lot!!
AIM

• Implement a system to recognize a Gesture

• Control a Windows Interface based on this Gesture
Video Decoder

• Why??

- Camera outputs are normally analog
- A decoder is used to digitize the analog video input
- Digitizes the (NTSC/PAL/SECAM) 4:2:2 YCbCr format
Digilent Video Decoder Board

Features

- **Inputs**: Component, Composite and S Video I2C Compatible Serial Bus
- High Speed **Hirose** FX2 Connector
- **Outputs**: 8/16 bit YCbCr, HS, VS and Field Signals
- Programmable Controls: Hue, Brightness, Saturation, Contrast

• Carrier for the Analog Devices ADV7183B Video Decoder Chip

15/03/2011
SDV Multiformat SDTV Video Decoder

ADV7183B

- High quality, single chip, multiformat video decoder
- Input (Composite, S video, Component video) → Output Digital ITU-R BT.656
- Front End:
  - Three 10 Bit ADCs
  - 12 channel input Mux
  - Current and Voltage Clamps
- Two wire Serial MPU interface: I²C Compatible

Gesture Recognition using FPGA
Features

- ADV7183B is fabricated in a 3.3 V CMOS process
- Automatic NTSC/PAL/SECAM identification
- Digital output formats (8-bit or 16-bit)
- ITU-R BT.656 YCrCb 4:2:2 output + HS, VS, and FIELD
- Integrated analog muxing section that allows more than one source of video signal to be connected to the decoder
- Programmable video controls
- Peak white/hue/brightness/saturation/contrast
- Integrated on-chip video timing generator
- Power-down mode
I\(^2\)C Interface

- Two inputs: Serial Data (**SDA**) & Serial Clock(**SCK**)
- Carries information b/w ADV7183B & FPGA
- Each device has unique address & operates as either slave or master

- **Master** : Initiates data transfer
- **(FPGA)** : Generates Clock Signals
- **Slave** : Receives data sent by Master
- **(ADV 7183B)** : Generates ACK bit

Gesture Recognition using FPGA
• **START**  High to Low on SDA  
  High in SCL

• 8 bits shifted: Address (7 bit ) + R/W

• Peripheral that recognizes the transmitted address  \( \rightarrow \) pulls **Data line Low** (the 9\(^{th} \) clock pulse) \( \rightarrow \) **Acknowledge bit**.

• **STOP**  Low to High on SDA  
  High on SCL

Gesture Recognition using FPGA  

15/03/2011
I²C Communication in ADV7183B

- ADV 7183B has two slave address for Read and Write

<table>
<thead>
<tr>
<th>ALSB</th>
<th>R/W</th>
<th>Slave Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0x40</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0x41</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0x42</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0x43</td>
</tr>
</tbody>
</table>

- The data sent on the SDA line to the ADV7183 includes
  - 7 bit slave address + 1 bit (R/W)- to identify the slave
  - 8 bit register address - to select one of the 249 sub addresses, thus enable access to the internal registers
  - 8 bit data - the data to be stored in the selected register
Bus Communication

- **START** condition
- Address of ADV 7183B SDA Line
- **ACK** bit occurs
- Sub address of the register
- **ACK** bit occurs
- Data sent from ADV 7183B FPGA
- **ACK** bit occurs
- **STOP** condition
VHDL Implementation

VDEC CARD with ADV7183B

CONTROLLER implemented on FPGA

Gesture Recognition using FPGA
Video Receiver

Line Field Decoder

Memory Controller (RAM)

Memory (BRAM)

Obtain the Luminance (Y) from the Video Stream

Gesture Recognition using FPGA
ITU-R BT 656 4:2:2 YCrCb Video Format

Parallel Data Format for NTSC video frames

Gesture Recognition using FPGA 15/03/2011
The XY status word is an SAV / EAV sequence defined as:

- F = 0 for odd fields; F = 1 for even fields
- V = 1 during vertical blanking
- H = 0 at SAV, H = 1 at EAV
- P3-P0 = protection bits
  - P3 = V xor H
  - P2 = F xor H
  - P1 = F xor V
  - P0 = F xor V xor H
Timing Reference Definition (word XY above) where XY = 1, F, V, H, P3, P2, P1, P0

<table>
<thead>
<tr>
<th>Line #</th>
<th>F</th>
<th>V</th>
<th>H(EAV)</th>
<th>H(SAV)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Blanking, Lines 1-9, 9 Lines</td>
</tr>
<tr>
<td>4-19</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Blanking, Lines 10-19, 10 Lines (optional ancillary data except line 14)</td>
</tr>
<tr>
<td>20-263</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Field 1 (Odd) Active Video, 244 Lines</td>
</tr>
<tr>
<td>264-265</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Blanking, Lines 264-272, 9 Lines</td>
</tr>
<tr>
<td>266-282</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Blanking, Lines 273-282, 10 Lines (optional ancillary data except line 277)</td>
</tr>
<tr>
<td>283-525</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Field 2 (Even) Active Video, 243 Lines</td>
</tr>
</tbody>
</table>

Protection Bits Definition:

- \( P3 = V \text{ (XOR) } H \)
- \( P2 = F \text{ (XOR) } H \)
- \( P1 = F \text{ (XOR) } V \)
- \( P0 = F \text{ (XOR) } V \text{ (XOR) } H \)
Memory

- Virtex II P
- 18Kb blocks of memory
- Generated Using Core Generator
- Stores 640x244 Pixels
Component Name: bram

Memory Type:
- Single Port RAM
- Simple Dual Port RAM
- True Dual Port RAM
- Single Port ROM
- Dual Port ROM

Algorithm:
Defines the algorithm used to concatenate the block RAM primitives. See the datasheet for more information.
- Minimum Area
- Fixed Primitives
  - Primitive (Write Port A): 8x2
  - Actual Primitive(s) Used: 8x2
Memory Controller

• Controls Read & Write Operation

 Writing

  ▪ Checks for EAV and SAV Code
  ▪ Stores only 244 lines
  ▪ Next “WRITE” occurs only after entire memory is Read

 Reading

  ▪ After entire WRITE operation is over, data is Read
  ▪ o/p Passed over to the Edge Detection Module
  ▪ Has to read data in 3x3 Matrix form

Gesture Recognition using FPGA

15/03/2011
Gesture Recognition using FPGA

3x3 Matrix Format
Implementation

- In VHDL
EDGE DETECTION

• Fundamental step in image processing
• Images contain redundant data
• Most important data → Edges
• Represents regions with strong image contrasts
• Filters out useless information
• Gradient Method
  • Edge Detection
  • Laplacian

• Our project □ Gradient Method
  □ Robert Cross
  □ Prewitt
  □ Canny
  □ Sobel

Gesture Recognition using FPGA  15/03/2011
Sobel Operator

• Performs a 2-D spatial gradient measurement on an image
• Two 3x3 Convolution Masks
  - Gradient in x-direction $G_x$
  - Gradient in y-direction $G_y$

• ‘A’ ▪ Input Image

```
G_x = [+1  +2  +1]
     [ 0    0    0]
     [-1   -2   -1] * A
```

```
G_y = [+1  0  -1]
     [+2  0  -2]
     [+1  0  -1] * A
```

• Magnitude of the gradient is calculated using the equation

$$|G| = \sqrt{(G_x)^2 + (G_y)^2}$$

15/03/2011
Implementation

Gesture Recognition using FPGA
MATLAB Outputs

Input Image

15/03/2011
Gesture Recognition using FPGA

15/03/2011
Horizontal + Vertical Edges

Gesture Recognition using FPGA

Edge Detection Output

15/03/2011
VHDL Implementation

- Edge detection for a 3x3 input pixels was implemented in VHDL
- Simulated using ISE 10.1
- Input for the operation was read from BRAM
- “Non-Restoring Square root Algorithm” was written in VHDL package for the calculation of the absolute value
- Based on the threshold value, a ‘1’ or ‘0’ was assigned
Testing

Gesture Recognition using FPGA

15/03/2011
Simulation Output

Gesture Recognition using FPGA 15/03/2011
Simulation Output

Gesture Recognition using FPGA 15/03/2011
Feature Vectors

• Mathematical Description of Image

• Feature vectors:
  • Area
  • Perimeter
  • Length and Width of fingers
  • Moment of Area
  • Centroid

• Compare the obtained values of feature vector with the reference values
FEATURE VECTORS

- **Perimeter**: No. of white pixels in the boundary
- **Area**: No. of black pixels inside the white contour
- **Length**: Difference b/w top and bottom most white pixels
- **Width**: Difference b/w left and right white pixels

Gesture Recognition using FPGA

15/03/2011
In a nutshell ....

Image (Y values from BRAM)

\[ G_x \text{ & } G_y \text{ Computed} \]

Magnitude Computed

Thresholding Performed

Feature Vectors Calculated

Edge Obtained

Gestures recognized

15/03/2011
VGA Display

Gesture Recognition using FPGA
Gesture Recognition using FPGA
Timing Diagram

Gesture Recognition using FPGA 15/03/2011
Video Parameters

- HSYNC
- VSYNC
- FRONT PORCH
- BACK PORCH

Gesture Recognition using FPGA  
15/03/2011
Implementation

Gesture Recognition using FPGA 15/03/2011
Video Controller Settings

- Output Format: 640 x 480 @ 60 Hz
- Pixel Clock: 25 MHz

- H Active Pixels: 640  V Active Pixels: 480
- H Front Porch: 16   V Front Porch: 9
- H Back Porch: 48    V Back Porch: 29

- H Total: 800  V Total: 520
Implemented Windows

Gesture Recognition using FPGA

15/03/2011
Project Status

- Configuring ADV 7183b
  - Synthesizable VHDL Code generated
  - Debugging Process Going on
- Video Receiver
- Memory & its Controller
- Sobel Edge Detection & Feature Extraction
- VGA Display

Gesture Recognition using FPGA  15/03/2011
Future Applications

- Help for disabled people
- Medicine
- Communication
- Entertainment
References


Non Restoring Square Root Algorithm
Hirose Connector