Case Study
JPEG Encoder

Speaker: 沈文中
Code Supported by NCTU
Outline

- *How to design an IP with AMBA standard*
- JPEG Spec.
- Lab requirement
Memory definition

- Self-designed IP can be located in the *Bus error response* area
- Defined in *AHBDenoder.v*
AMBA IP design

- Software part
  - Write a function to control hardware
  - Delay number of clocks by NOOP (asm) instruction

- Hardware part
  - Add MYIP.v into top module
  - Change ahbdecoder.v
  - Change AHBMuxS2M.v
  - Change ahhbahbtop.v
Add `MYIP.v` into the top module

- Write your own module in AMBA interface
Change ahbdecoder.v

- Add **HSELMYIP** signal to select your own slave IP to response
- Address are defined in decoder
IP design-HW

- Change AHBMuxS2M.v
  - Use mux to select slave which can use HRDATA
IP design-HW

- Change ahabhbtol.v
  - Add your module in AMBA Bus
  - Connect the above net connection
Outline

- How to design an IP with AMBA standard
- JPEG Spec.
- Lab requirement
JPEG spec.

- **Target**: JPEG baseline sequential DCT-based encoder

- **Content**
  - RGB->YUV
  - DCT (discrete cosine transform)
  - DQT (quantization)
  - Zig-Zag scan
  - Entropy coding (run length coding + Huffman coding)
JPEG encoder overview

- Split into 8x8 pixels per block
- Use FDCT, Quantizer, and Entropy encoder to compress data
RGB-\(\rightarrow\)YUV

\[
\begin{align*}
\text{Y} &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\
\text{Cb} &= -0.1687 \times R - 0.3313 \times G + 0.5 \times B + 128 \\
\text{Cr} &= 0.5 \times R - 0.4187 \times G - 0.0813 \times B + 128
\end{align*}
\]

- Y-Luminance (亮度) Cb, Cr-Chrominance domain (色差)
YUV arrange

- Order of block is from left to right, and then from up to down.
- Each block obeys the order of Y->U->V
- YUV can be sub-sampled as 4:2:2 ratio
DCT

- Encoding procedure uses 8x8 pixels as a compressing unit
- FDCT is composed of 2 times of 1D-DCT
Quantization

- Less bits is needed to code the quantization coefficient
- static BYTE std_luminance_qt[64] = {
  16, 11, 10, 16, 24, 40, 51, 61,
  12, 12, 14, 19, 26, 58, 60, 55,
  14, 13, 16, 24, 40, 57, 69, 56,
  14, 17, 22, 29, 51, 87, 80, 62,
  18, 22, 37, 56, 68, 109, 103, 77,
  24, 35, 55, 64, 81, 104, 113, 92,
  49, 64, 78, 87, 103, 121, 120, 101,
  72, 92, 95, 98, 112, 100, 103, 99
};
- static BYTE std_chrominance_qt[64] = {
  17, 18, 24, 47, 99, 99, 99, 99,
  18, 21, 26, 66, 99, 99, 99, 99,
  24, 26, 56, 99, 99, 99, 99, 99,
};
Zig-Zag scan

Zig-zag order

Figure 5 – Preparation of quantized coefficients for entropy encoding
Entropy Coding

- **Entropy coding** transfer the quantized data to compressed data
- Entropy coding is composed of **Run Length Coding** and **Variable Length Coding (Huffman Coding)**

- 15,0,-2,-1,-1,-1,0,0,…
  - ->15,(1,-2)(0,-1)(0,-1)(0,-1),…
  - ->101111…

Quantized data
- After RLC
- After VLC
AC Run Length Coding

Zero is the value occurs at highest frequency in quantized data
Variable Length Coding
(DC Huffman Coding)

<table>
<thead>
<tr>
<th>SSSS</th>
<th>DIFF values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1,1</td>
</tr>
<tr>
<td>2</td>
<td>-3,-2,2,3</td>
</tr>
<tr>
<td>3</td>
<td>-7,-4,4,7</td>
</tr>
<tr>
<td>4</td>
<td>-15,-8,8,15</td>
</tr>
<tr>
<td>5</td>
<td>-13,-16,16,31</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

DC Diff magnitude category

<table>
<thead>
<tr>
<th>Category</th>
<th>Code length</th>
<th>Code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>010</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>101</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

DC Huffman Table

DC code word = Category code word + DIFF value code word

15 → 101 1111
### Variable Length Coding (AC Huffman Coding)

<table>
<thead>
<tr>
<th>SSSS</th>
<th>AC coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1,1</td>
</tr>
<tr>
<td>2</td>
<td>-3,-2,2,3</td>
</tr>
<tr>
<td>3</td>
<td>-7..-4,4..7</td>
</tr>
<tr>
<td>4</td>
<td>-15..-8,8,15</td>
</tr>
<tr>
<td>5</td>
<td>-13..-16,16..31</td>
</tr>
<tr>
<td>6</td>
<td>-127..-</td>
</tr>
<tr>
<td>....</td>
<td>64,64..127</td>
</tr>
</tbody>
</table>

#### AC coefficient magnitude category

#### AC Huffman Table

<table>
<thead>
<tr>
<th>Run/Size</th>
<th>Code length</th>
<th>Code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0 (EOB)</td>
<td>4</td>
<td>1010</td>
</tr>
<tr>
<td>0/1</td>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>....</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>1/2</td>
<td>5</td>
<td>11011</td>
</tr>
<tr>
<td>....</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>2/1</td>
<td>5</td>
<td>11100</td>
</tr>
<tr>
<td>....</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

AC code word = Run/Size code word + AC coefficient code word

\[
\begin{align*}
\langle 1,-2 \rangle & \quad \langle 0,-1 \rangle & \quad \langle 0,-1 \rangle & \quad \langle 0,-1 \rangle & \quad \langle 2,-1 \rangle & \quad \langle \text{EOB} \rangle \\
1101101 & \quad 000 & \quad 000 & \quad 000 & \quad 1110000 & \quad 1010
\end{align*}
\]
Outline

- How to design an IP with AMBA standard
- JPEG Spec.
- Lab requirement
Lab requirement

- Try to explain the given dct IP content
- Try to partition the hw/sw part for implement
- Implement the JPEG encoder
- Compare the timing constraint of your own design with the one of reference code
Reference

- `lm_xcv600e_revc.pdf`
- `DUI146C_LM600_UG.pdf`
- Code Example supplied by NCTU
- JPEG Still Image Specification