JPEG Lossless Codec 17

Like the JPEG baseline sequential compressor, the JPEG lossless compressor is designed to compress still continuous-tone images. The difference between the two is that the baseline sequential compressor is a lossy compressor: once an image has been compressed, you cannot recover the original image samples. The JPEG lossless compressor does allow you to recover these samples; however, the lossless compressor produces much lower compression ratios than the lossy compressor. The compression ratio for the lossless compressor is in the neighborhood of 2:1. One other difference between the two compressors is that while the baseline sequential compressor works only with `XIL_BYTE` images, the lossless compressor can work with both `XIL_BYTE` and `XIL_SHORT` images.

The remainder of this chapter is divided into three sections. The first section provides an overview of how the JPEG lossless compressor works. The second explains briefly how to create a JPEG lossless CIS. The third discusses a set of CIS attributes that apply specifically to the JPEG lossless codec (as opposed to the general CIS attributes covered in the section "General CIS Attributes" on page 257).

17

How the JPEG Lossless Codec Works

The JPEG lossless compressor is not based on the DCT like the baseline sequential encoder. Instead, it uses a predictive method, as shown in Figure 17-1.

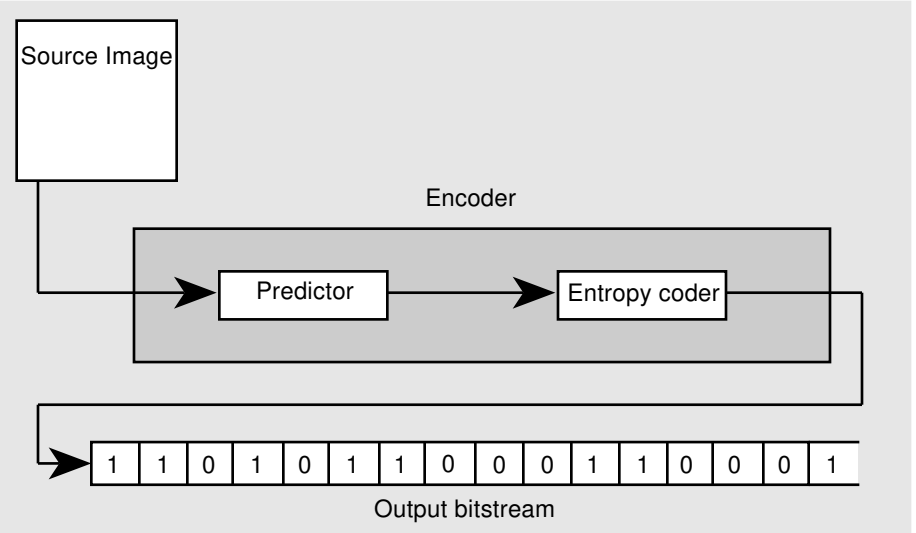

Figure 17-1 JPEG Lossless Compressor

Each sample in the source image is encoded as follows: The predictor makes a guess at the sample's value based on its knowledge of the values of neighboring samples and then subtracts the predicted value of the sample from its actual value. The difference calculated by the predictor is then passed on to the entropy coder, which does a lookup in a Huffman table and writes encoded data to the bitstream.

The prediction and entropy-coding steps in the encoding process are discussed in more detail in the following sections.

Note – The XIL library's JPEG lossless encoder actually lets you perform an operation called a point transformation on your source image before the encoding summarized above begins. The point transformation can be done on a band-by-band basis. What happens is this. If you supply a point-transform value other than 0 for a band of an image, each sample in the band is divided by 2 raised to the power *x*, where *x* is the point-transform value. This operation leads to a higher compression ratio because smaller values will need to be encoded; however, it is not actually part of the lossless-encoding process because it can result in the loss of some data.

Prediction

As mentioned above, the predictor predicts the value of an image sample based on its knowledge of the values of neighboring samples. The neighboring samples it can take into consideration are shown in Figure 17-2 and are labeled A, B, and C. The sample whose value is being predicted is labeled P.

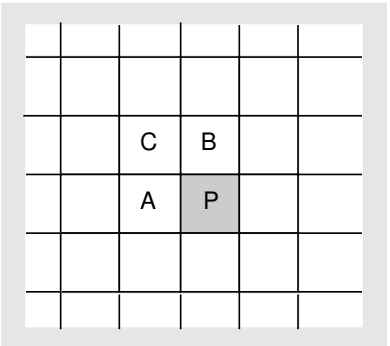

Figure 17-2 Predicting Values in the JPEG Lossless Compressor

Seven prediction methods are defined. These are shown below.

1. P = A
2. P = B
3. P = C
4. P = A + B - C
5. P = A + ((B - C) / 2)
6. P = B + ((A - C) / 2)
7. P = (A + B) / 2

By default, the compressor uses prediction method 1 for all bands in an image. However, you can select a different prediction method for a particular band using the CIS attribute `LOSSLESS_BAND_SELECTOR`. See the section "LOSSLESS_BAND_SELECTOR" on page 334.

After the predictor has predicted the value of a sample, it subtracts the predicted value of the sample from its actual value. It is this difference that is encoded by the entropy coder.

Entropy Coding

The entropy coder takes as input a difference, the difference between a predicted and actual sample value. On the output side, the entropy coder writes a sequence of bits to the JPEG bitstream.

To produce its output, the entropy coder uses the number of bits required to represent a difference as an index into a Huffman table. At the proper location in the table, the encoder reads a code word—a sequence of bits—and then writes this code word to the bitstream. Immediately after the code word, the encoder writes to the bitstream the difference itself. The encoder performs these steps for each sample in the image.

JPEG Lossless Compressor Attributes

The attributes discussed below affect the way that the JPEG lossless compressor works.

BAND_HUFFMAN_TABLE

This attribute enables you to specify that a particular Huffman table be used in encoding a particular band of your image. Before calling `xil_cis_set_attribute()` to set this attribute, you must declare and assign values to the members of a structure of type `XilJpegBandHTable`.

```
typedef struct {
    int band;
    int table;
    XilJpegHTableType type;
} XilJpegBandHTable;
```

The member `band` can have a value in the range 0 to 255 and represents the image band to be associated with a particular Huffman table. The member `table` can have a value between 0 and 3 and specifies one of four possible Huffman tables, and the member `type` must be set to `DC`.

The code fragment below associates table 0 (type `DC`) with band 1 of the image to be encoded.

```
XilCis cis;
XilJpegBandHTable table_for_band = {1, 0, DC};

xil_cis_set_attribute(cis, "BAND_HUFFMAN_TABLE",
    (void *)&table_for_band);
```

By default, table 0 is used in encoding band 0 of an image, and table 1 is used in encoding all other bands (1 to 255).

COMPRESSED_DATA_FORMAT

COMPRESSED_DATA_FORMAT is a set-only attribute that defines the format of the JPEG lossless compressor's output. It can be set to either INTERCHANGE or ABBREVIATED_IMAGE.

Setting the attribute to INTERCHANGE causes the compressor to produce output in JPEG interchange format. In this format, the Huffman tables required by the decompressor are included in each compressed frame.

The default value, ABBREVIATED_IMAGE, causes the compressor to produce output in JPEG abbreviated format. In this format, Huffman tables are not included in a compressed frame if the tables needed to decompress that frame have been defined in a previous frame in the sequence. If a table changes during the course of the sequence, the new table definition is included in the first compressed frame that uses the new table values.

The code fragment below shows COMPRESSED_DATA_FORMAT being set to INTERCHANGE.

```
XilCis cis;

xil_cis_set_attribute(cis, "COMPRESSED_DATA_FORMAT",
    (void *)INTERCHANGE);
```

ENCODE_INTERLEAVED

ENCODE_INTERLEAVED is a set-only attribute that determines whether the compressor produces an interleaved bitstream when compressing multiband images. If the attribute is set to TRUE, the default value, the compressor produces an interleaved bitstream, and if the attribute is set to FALSE, the compressor produces a noninterleaved bitstream.

This attribute is ignored if the image being compressed has more than four bands, or if different point transformations or different predictors are being used in encoding different bands. In these instances, the compressor never produces an interleaved bitstream.

In the example below, ENCODE_INTERLEAVED is being set to FALSE.

```
XilCis cis;

xil_cis_set_attribute(cis, "ENCODE_INTERLEAVED", (void *)FALSE);
```

HUFFMAN_TABLE

This attribute enables you to supply the compressor with a Huffman table to use in encoding difference values. By default, the compressor's four Huffman tables are loaded as follows: Tables 0 and 2 contain the values specified in Table K.3 of the ISO JPEG specification (extended to 17 entries). These values are useful for encoding pixel differences for the luminance band of 8-bit YC_bC_r images. Tables 1 and 3 contain the values specified in Table K.4 of the ISO JPEG specification (again extended to 17 entries). These values are useful for encoding pixel differences for the chrominance bands of 8-bit YC_bC_r images.

Before calling xil_cis_set_attribute() to set this attribute, you must store information about the Huffman table in a structure of type XilJpegHTable.

```
typedef struct {
    int table;
    XilJpegHTableType type;
    XilJpegHTableValue *value;
} XilJpegHTable;
```

The member table must be set to a value between 0 and 3, and the member type must be set to DC. The final member of the XilJpegHTable structure, value, is an array of 17 structures of type XilJpegHTableValue.

```
typedef struct {
    int bits;
    int pattern;
} XilJpegHTableValue;
```

Each structure of this type defines a pair of values to be written to the Huffman table. The member `bits` is the length of a Huffman code in bits, and `pattern` contains the code itself (the `bits` least significant bits of `pattern` are the code).

The example below shows table 0 (type `DC`) being loaded with values suitable for encoding difference values.

```
XilCis cis;
XilJpegHTable huffman_table;

XilJpegHTableValue huffman_values[17] = {
    {2, 0x0000}, {3, 0x0002}, {3, 0x0003}, {3, 0x0004},
    {3, 0x0005}, {3, 0x0006}, {4, 0x000e}, {5, 0x001e},
    {6, 0x003e}, {7, 0x007e}, {8, 0x00fe}, {9, 0x01fe},
    {10, 0x03fe},{11, 0x07fe}, {12, 0x0ffe}, {13, 0x1ffe},
    {14, 0x3ffe}
};

huffman_table.table = 0;
huffman_table.type = DC;
huffman_table.value = huffman_values;
xil_cis_set_attribute(cis, "HUFFMAN_TABLE",
    (void *)&huffman_table);
```

LOSSLESS_BAND_SELECTOR

As mentioned in the section "Prediction" on page 329, there are seven methods the predictor can use to predict the values of an image sample. This attribute enables you to specify which of these methods the compressor should use in encoding a particular band of an image.

Before calling `xil_cis_set_attribute()` to set this attribute, you must declare and assign values to the members of a structure of type `XilJpegLLBandSelector`.

```
typedef struct {
    int band;
    XilJpegLLBandSelectorType selector;
} XilJpegLLBandSelector;
```

The structure member `band` must be set to a value between 0 and 255 (where band 0 is the first band in an image). The member `selector` can have any of the values shown in the following enumeration.

```
typedef enum {
    ONE_D1=1, ONE_D2, ONE_D3, TWO_D1, TWO_D2, TWO_D3, TWO_D4
} XilJpegLLBandSelectorType;
```

The correspondence between these enumeration constants and the prediction methods described earlier is shown in Table 17-1.

Table 17-1 JPEG Lossless Prediction Methods

Constant	Prediction Method
`ONE_D1`	P = A
`ONE_D2`	P = B
`ONE_D3`	P = C
`TWO_D1`	P = A + B - C
`TWO_D2`	P = A + ((B - C) / 2)
`TWO_D3`	P = B + ((A - C) / 2)
`TWO_D4`	P = (A + B) / 2

The following code fragment shows the prediction formula P = A + ((B - C) / 2) being selected for band 2 of an image.

```
XilCis cis;
XilJpegLLBandSelector predictor_band = {2, TWO_D2};

xil_cis_set_attribute(cis, "LOSSLESS_BAND_SELECTOR",
    (void *)&predictor_band);
```

LOSSLESS_BAND_PT_TRANSFORM

This attribute enables you to request that a point transformation be performed on a particular band of an image before the image is encoded. If you request this operation, each sample in the band is divided by 2 raised to power of *x*, where *x* is the prediction-transformation value. The default prediction-transformation value for all bands is 0.

Before calling `xil_cis_set_attribute()` to set this attribute, you must declare and assign values to the members of a structure of type `XilJpegLLBandPtTransform`.

```
typedef struct {
    int band;
    int PtTransform;
} XilJpegLLBandPtTransform;
```

The structure member `band` must be set to a value between 0 and 255 (where band 0 is the first band in an image), and the `PtTransform` member must be set to an integer in the range 0 to 15 that represents the power of two by which you want to divide all the samples in a band.

The code below requests that the samples in band 2 of an image be divided by 16 (2^4) before the lossless encoding process takes place.

```
XilCis cis;
XilJpegLLBandPtTransform pt_transform = {2, 4};

xil_cis_set_attribute(cis, "LOSSLESS_BAND_PT_TRANSFORM",
    (void *)&pt_transform);
```