



XAPP621 (v1.1) January 31, 2005

Variable Length Coding

Author: Latha Pillai

Summary

This application note describes the implementation of Variable Length Coding (VLC) on Xilinx devices. Zig-zag coding and run length coding are done in an MPEG-2 encoder. The zig-zag coding arranges the DCT coefficients in the order of increasing frequency. These coefficients are then coded as a run-length pair where run is the number of occurrences of a value and the length is the amplitude.

Introduction

Variable Length Coding (VLC) is the final lossless stage of the MPEG video compression unit. VLC is done to further compress the quantized image. VLC consists of three steps: zig-zag scanning, Run Length Encoding (RLE), and Huffman coding. At the decoder, VLC is the first step in the decoding process. [Figure 1](#) shows VLC at the encoder.



x621_01_031803

Figure 1: VLC at the Encoder

Zig-zag Scan

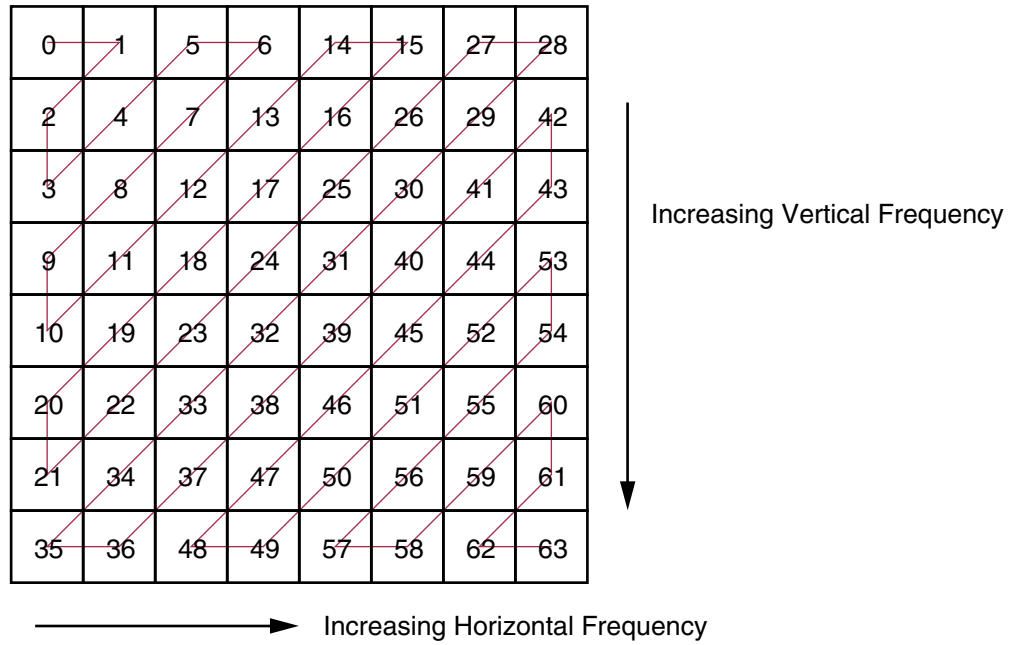
In zig-zag scanning the quantized coefficients are read out in a zig-zag order. By arranging the coefficients in this manner, RLE and Huffman coding can be done to further compress the data. The scan puts the high-frequency components together. These components are usually zeroes.

For MPEG-2 interlaced video, a top field and a bottom field consist of a frame frozen at different instances in time. If the frame has a moving object, the intra-coded blocks containing this moving object exhibit high vertical frequencies. These high frequencies are less likely to quantize to zero. In this case, implementing an alternate scan (alternate zig-zag scanning pattern) facilitates faster reception of the non-zero high frequencies. The alternate scan is an additional choice in the MPEG-2 standard. The MPEG-1 standard uses only the regular zig-zag scanning order. [Figure 2](#) shows the zig-zag scan and the alternate scan.

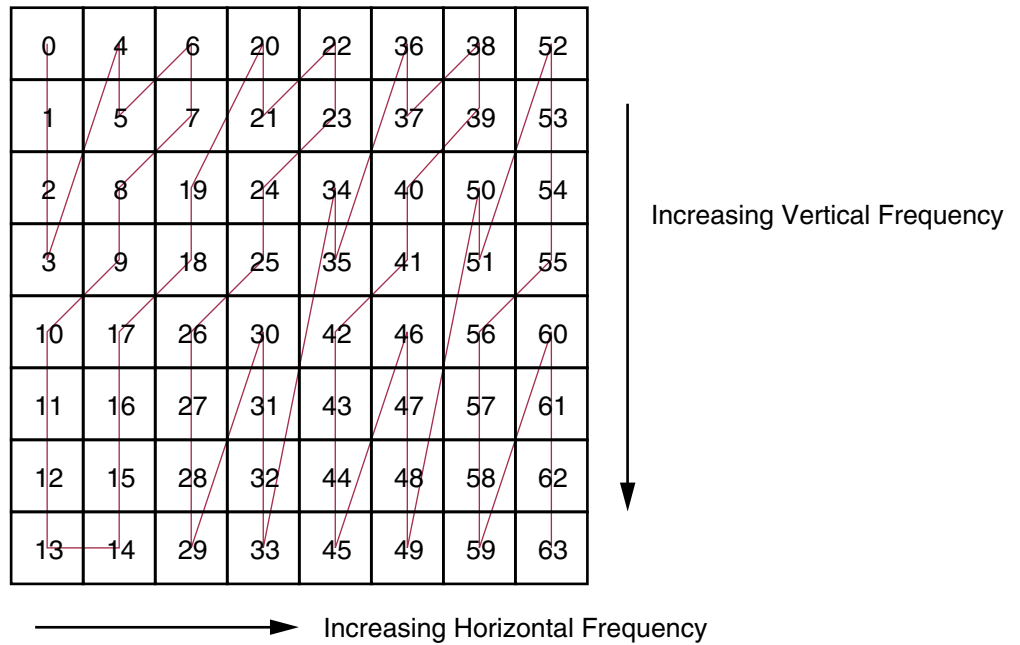
© 2003-2005 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

ZigZag Scan Orders Allowed In MPEG-1 and MPEG-2



Alternate Scan Orders Allowed In MPEG-2 Only



x621_02_032003

Figure 2: Zig-zag Scan Orders and Alternate Scan Orders (MPEG-2 Only)

Run Length Encoding (RLE)

Basics

The quantized coefficients are read out in a zig-zag order from DC component to the highest frequency component. RLE is used to code the string of data from the zig-zag scanner. Run length encoding codes the coefficients in the quantized block into a run length (or number of occurrences) and a level or amplitude. For example, transmit four coefficients of value "10" as: {10,10,10,10}. By using RLE, the level is 10 and the run of a value of 10 is four. By using RLE, {4,10} is transmitted, reducing the amount of data transmitted. Typically, RLE encodes a run of symbols into two bytes, a count and a symbol.

By defining an 8 x 8 block without RLE, 64 coefficients are used. To further compress the data, many of the quantized coefficients in the 8 x 8 block are zero. Coding can be terminated when there are no more non-zero coefficients in the zig-zag sequence. Using the "end-of-block" code terminates the coding.

The compression ratio obtained by RLE is:

$$\text{Compression Ratio} = \text{original size/compressed size} : 1$$

Two examples of compression using RLE are described:

1. Consider the sequence --- aa0005555555zzzzaaaa

This string of characters can be compressed to form --- 2(a), 3(0), 7(5), 4(z), 4(a)

The 20-byte original string would only require 10 bytes of data to represent the string. In this case, RLE yields a compression ratio of 2 : 1.

2. Consider the string of numbers as follows --- 2, 0, 1, 2, 0, 1, 2, 0, 1

This string can be compressed to form --- 3 (2,0,1). Giving a compression ratio of = 9/4 : 1, almost 2 : 1.

RLE will not always result in compression. The amount of compression achieved depends on the data being compressed.

Consider the string --- 1, 2, f, 4, a, a,
After RLE gives --- 1(1),1(2),1(f),1(4),2(a).

To express the original string of 6 bytes, 10 bytes are used. Instead of compression, the original string has expanded. If each character in a string of 128 characters is different from the previous, each character will require 2 bytes to denote the run and level. In this case, 2 x 128 bytes are required to denote the original 128 bytes of information, with no compression available.

RLE is most often used to compress black and white or 8-bit indexed color images where long runs are likely. RLE is not generally used for high color images. In photography, each pixel generally varies from the last. RLE cannot achieve high compression ratios compared to other compression methods, although it is easy to implement. Run length encoding is supported by most bitmap file formats (TIFF, BMP, and PCX). Long runs are rare in certain types of data (ASCII text files), and RLE is not a good compression choice. To encode a run in RLE, a minimum of two characters worth of information is required. A run of single character takes more space.

RLE Implementation

RLE in MPEG consists of denoting the "run of zeroes" followed by the value of the coefficient. Each AC coefficient is represented by the number of zeroes before the coefficient followed by the coefficient. RLE is very simple to implement. A flow chart of an RLE implementation is shown in [Figure 3](#).

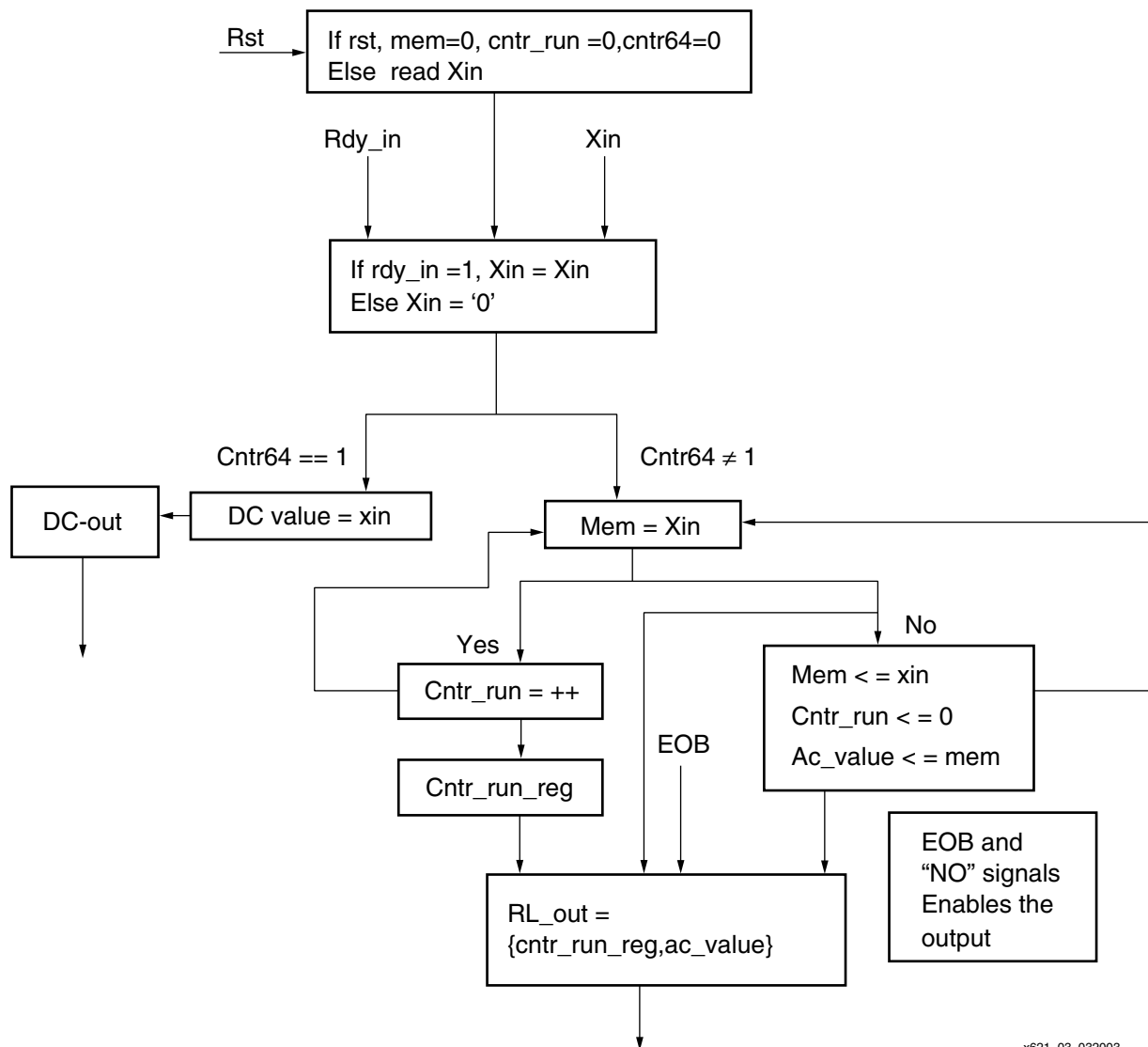


Figure 3: Block Diagram of a Variable Length Coder Implementation

Reference Design

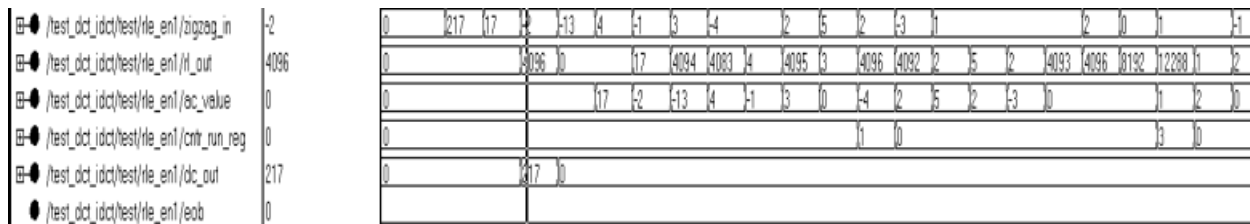
Resource Utilization

Table 1: Zig-zag Encoding

Device	Post-Route (Synthesis Constraint)	LUTs (Flip-Flops)
XCV300E -8 BG352	102.29 (100 MHz)	389 (23)
XC2V250 -6 FG456	131.87 (100 MHz)	360 (16)
XCV300 -6 PQ240	63.1 (100 MHz)	385 (60)
XC2S200 -6 FG256	89.16 (75 MHz)	391 (58)

The performance can be improved by adding more pipeline stages in the design. The place and route effort level was kept at "High". To get an update on performance and utilization, always re-run the designs using the latest Xilinx software.

Figure 4 shows the run-length coded output (r1_out) of the variable length coder (VLC). The input of the VLC (zigzag_in) is the output of the zig-zag encoder.



x621_04_032103

Figure 4: Run Length Coded Output of the VLC

The reference design, in both VHDL and Verilog is available on the Xilinx FTP site at:

<http://www.xilinx.com/bvdocs/appnotes/xapp621.zip>

Conclusion

Zig-zag encoding and run-length encoding done in MPEG-2 compression systems are explained in this application note. The design files show the efficient implementation of the algorithms in Xilinx devices. The code can be used to target any Xilinx device. The code can be optimized by instantiating the adder/subtractor and multiplier units when targeting Virtex devices. Adding more pipeline stages can increase the performance of both blocks. The zigzag_decode.v module can be used at the decoder. Both the encoder and decoder wait for 64 clock cycles before starting the zig-zag process. One set of 64 DCT values will be available for the zig-zag process. This ensures uninterrupted data flow at the output of the zig-zag encoder. The run-length coding module takes in the zig-zag output data and sends it out as a length of zeroes followed by the non-zero value.

References

1. Image and Video Compression Standards, Second Edition, by Vasudev Bhaskaran and Konstantinos Konstantinides, ISBN 0-7923-9952-8
2. MPEG Video Compression Standard, by Mitchell, Pennebaker, Fogg, and LeGall, ISBN 0-412-08771-5
3. MPEG2 Video IS document, ISO/IEC 13818-2: 1995(E)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/09/03	1.0	Initial Xilinx release.
01/31/05	1.1	Updated link to reference design.