VLSI IMPLEMENTATION OF A REVERSIBLE VARIABLE LENGTH ENCODER/DECODER

Mario Novell and Steve Molloy

UCLA Department of Electrical Engineering
Engineering IV, 53-138 N12
Los Angeles, CA 90095

ABSTRACT

Variable Length Codes (VLCs) are known for their efficient compression, but are susceptible to noisy environments due to synchronization losses that can occur from bit error propagation. Recent interest in Reversible Variable Length Codes (RVLCs) has come about due to the growing need for wireless exchange of compressed image and video signals over noisy channels and the ability for RVLCs to provide greater error robustness than their non-reversible counterparts (VLCs). With the current ITU H.263+ and ISO MPEG-4 standards already using RVLCs, low power implementations of the RVLC are essential in providing error robustness in real-time systems, while minimizing power consumption. This paper will present the first published VLSI architectures of a low power reversible variable length encoder and decoder. Results show power consumption of less than 1 mW for both encoder and decoder, with an additional 65% increase in area for the decoder over that of a conventional VLD design.

1. INTRODUCTION

With the growing emergence of image and video coding over wireless channels, new techniques and algorithms are providing better ways to achieve error robustness in real-time systems [1] [2]. The use of Reversible Variable Length Codes (RVLCs) has recently emerged as a substitute for standard VLCs, while providing greater error robustness. Already being used by the emerging ITU H.263+ and ISO MPEG-4 standards, RVLCs are fast becoming an integral component for providing greater error resilience in wireless environments.

VLCs are known for their efficient compression, but are very vulnerable in noisy environments due to synchronization losses that can occur from bit error propagation [3]. The idea behind RVLCs is that decoding can be performed by processing the received bitstream forwards and backwards, enabling localization of errors in contrast to conventional decoding. The inherent nature of RVLCs allows for them to be two-way decodable, such that any codeword can be equivalently represented by the result of forward and backward decoding. In addition to two-way codeword decoding, two-way run length decoding can provide even better results. This will be elaborated upon in the next section.

In this paper we present VLSI architectures for both an RVLC encoder (RVLE) and decoder (RVLD). The architectures are based on a wavelet-based image codec design, but are also applicable to standard image/video DCT-based codecs. The rest of the paper is organized as follows. In Section 2, error detection and concealment techniques will be discussed, which will aid in the control design for the decoder. In Section 3, the RVLE architecture is discussed. And in Section 4, the RVLD architecture is discussed. Both architectures use low-power Look-Up Table (LUT) partitioning for implementing the RVLC codeword tables. Finally, in Section 5, the results are summarized, with architecture trade-offs and layout results.

2. ERROR DETECTION/CONCEALMENT

Two-way decodability requires the ability for the decoder to process "reversible" codewords. The RVLCs used in the design here are reversible codes derived from exponential-Goulomb (EG) codes, where the number of codewords of a given length grows exponentially with length. Basically, codewords are grouped by having equivalent odd-indexed bits (with same codeword length), and the even-indexed bits distinguish each individual codeword within the same group (codeword length). This results in $2^{(l/2)-1}$ possible codewords of length l. Single bit errors on these codewords can be classified as either propagating bit errors or non-propagating bit errors, depending on whether bit errors occur on odd-indexed or even-indexed bits. Therefore, even-indexed bit errors will not result in a loss of synchronization at the decoder. More about the nature of these codewords, and analytical results about their error propagation can be found in [4].

Within the reversible coded bitstream, there are basically four different types of errors that can occur. These are: *i)* non-propagating bit errors, *ii)* propagating bit errors, *iii)* illegal codewords, and *iv)* synchronization marker (SM) errors. The first type of error *(i)* will cause the wrong codeword to be decoded, but no loss of synchronization at the decoder. As mentioned above, these are bit errors occurring on even-indexed positions of an RVLC. The second type of error *(ii)* will also cause the wrong codeword to be decoded, and in addition, causes loss of synchronization at the decoder. This in turn can cause insertion or deletion of subsequent decoded codewords in the bitstream. These are a result of bit errors occurring on odd-indexed positions of an RVLC. The third type of error *(iii)* is due to a bit error which results in an unrecognizable codeword to the decoder. Finally, *(iv)* is an error

occurring on the SM. SMs are fixed-length codes inserted by the encoder to divide the different coded subbands and establish boundaries for forward and backward decoding. The decoder finds and removes the SMs before forward and backward decoding each subband. Errors occurring on SMs cause a loss in synchronization at the decoder with respect to subbands.

The general decode procedure is as follows. When the bitstream is received by the decoder, the first subband (indicated by the beginning of the bitstream and the first SM), is buffered. After the first subband has been buffered, the next subband is buffered in a second buffer, while the decoder processes the first RVLC coded subband. The two buffers alternate in a ping-pong fashion until the end of the frame. In conjunction with the processing of the RVLC coded bitstream, Run-Length decoding is done on the run-level pair results. Since each of the coded subbands are being decoded bidirectionally, two-way Run Length Decoding (RLD) is done on the resulting run-level pairs, resulting in coefficient expansion occurring bidirectionally. This avoids the need to buffer the resulting run-level pairs before decoding them, resulting in a simple RVLD/RLD pipeline design. Figure 1 illustrates the general decoding procedure.

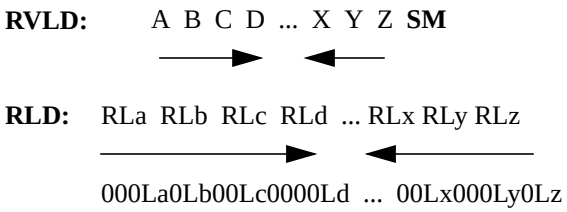

Figure 1: **Bidirectional RVLD and RLD.**

Now going back to the different types of errors that can occur in an RVLC-coded bitstream, several exceptions arise that need to be accounted for in the design of a reversible variable length decoder. These exceptions are listed below:

- ***Incorrect number of coefficients decoded in a particular subband.*** This results from a propagating error resulting in insertion and/or deletion of codewords by the decoder. The overall result is a number of run-length expanded coefficients that is less than or greater than the actual coefficient count for that particular subband.
- ***Illegal codeword decoded in the forward direction.*** This occurs when an illegal codeword is detected by the decoder while forward decoding the bitstream.
- ***Illegal codeword decoded in the backward direction.*** This occurs when an illegal codeword is detected by the decoder while backward decoding the bitstream.
- ***SM not found by the decoder.*** This occurs when there is a bit error on the SM, and the SM is unrecognizable to the decoder. There may also be instances where no SM is found due to an RVLC buffer overflow. In this case, the capacity of the buffer size is surpassed in a particular subband. Although highly improbable, overflow is statistically possible, and highly dependent on compression value and subband number.

When the above exceptions occur, appropriate actions have to be taken in order to conceal these errors. For the first exception listed above, if the number of expanded coefficients exceeds the subband size, the remaining coefficients and undecoded run-level pairs/codewords are discarded. Decoding resumes in the next subband. If the number of coefficients is less than the actual coefficient count for that particular subband, the remaining coefficient values are set to zero for the rest of the subband. Note that two-way RLD has the added advantage that unwanted runs of zeros don't propagate to the end of the subband (as in conventional forward RLD), only up to where forward and backward decoding ends.

The second and third exceptions listed above occur when illegal codewords are detected while forward or backward decoding. If the illegal codeword occurs during forward decoding, the remainder of the subband is zeroed out, including those coefficients which were backward decoded. If the illegal codeword occurs during backward decoding, backward decoding is discontinued and the rest of the subband is forward decoded up to the illegal codeword, where the remaining coefficients are zeroed out. Finally, an exception caused by a missing SM results in a loss of synchronization at the decoder. Since the decoder can no longer keep track of which subband is currently being decoded, the rest of the coefficients for that particular frame are zeroed out.

3. RVLE ARCHITECTURE

An architecture for implementing RVLC and low-power table partitioning [5] for a variable length encoder is illustrated in Figure 2. This architecture converts run-level pairs to reversible variable length codes and packs the resulting codes into an output bitstream. The architecture consists of the partitioned LUTs, a LUT select module, and a shifting datapath for the bit-packing. The LUT select module computes a run-level distance metric, and uses the result to select one of four LUTs. The encoder LUTs are partitioned according to bounded run-level distance metric values, in effect placing the more probable codewords in small, lower power LUTs and less probable codewords in larger, higher power LUTs [5]. So on average, the lower power LUTs are selected most of the time, reducing overall power consumption.

Registers with enable at the input to the LUTs serve to isolate the LUTs to effectively select one LUT per cycle. The outputs of the LUTs are then multiplexed, with the resulting RVLC codeword and its length used by the shifting datapath. The shifting datapath consists of a barrel shifter which takes the RVLC codeword and concatenates it into an output bitstream, an accumulator which provides the shifter with the number of bits to shift, and three output registers which effectively contain the next three "words" of the output bitstream. The maximum codeword length is 20 bits, with 16 bit outputs and 7 bit run and 5 bit level inputs. The RVLE architecture has a constant output rate, capable of encoding one run-level pair per cycle.

The RVLE was implemented in synthesizable VHDL, synthesized to a standard cell netlist, and mapped to layout in 0.5 *u*m CMOS. Power consumption measurements were extracted from layout at

a clock rate of 7.16 MHz at 3.3 V using a commercial power analysis tool. The power analysis results of the partitioned LUTs are shown in Table 1. The total power consumption of the RVLE was found to be 51 uW for a 0.028 activity factor in a typical wavelet codec system.

.

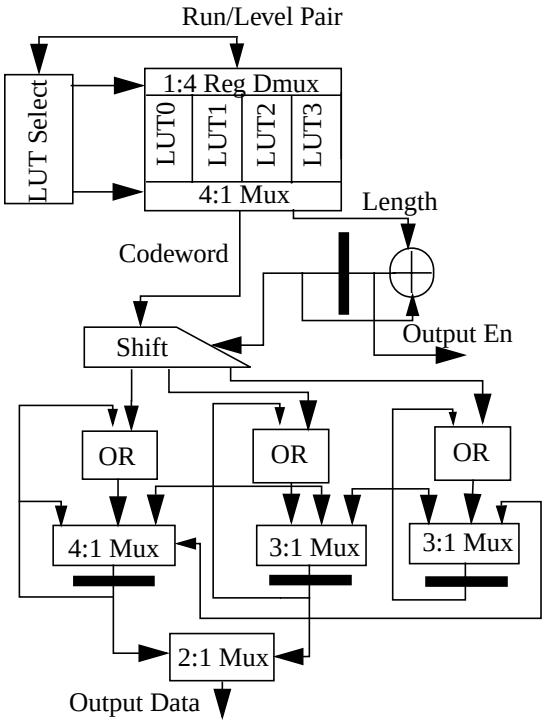

Figure 2: RVLE Architecture

4. RVLD ARCHITECTURE

The RVLD is functionally the dual of the RVLE, where the incoming received bitstream is unpacked into variable length codes, and converted into run-level pairs. An architecture for implementing RVLC and low-power table partitioning [5] for a variable length decoder is illustrated in Figure 3. The architecture consists of parallel forward and backward shifting datapaths, partitioned LUTs, a codeword length LUT, an LUT select module, a buffer controller, and two local RVLC buffers. For simplicity, only the forward shifting datapath is shown in Figure 3. The LUT selection module for the decoder counts the number of odd-indexed leading ones in the prefix of each unpacked codeword, and selects one of four LUTs based on its value along with a codeword length. This architecture also has a maximum decoding rate of one codeword per cycle.

The main difference between this architecture and a conventional VLD architecture [6] is in the additional buffering and bitstream processing that needs to occur before the resulting bitstream is sent to the shifting datapath(s). This allows for bidirectional decodability, which is needed for RVLC decoding. The buffer controller in Figure 3 is responsible for interacting with the local RVLC buffers, the incoming bitstream, the shifting datapaths, and the RLD (not shown here). The incoming bitstream is routed to the inactive RVLC buffer (the buffer not being read from) in segments, where a segment is a subband. After a subband is written to the inactive buffer, the controller waits until the current subband is fully decoded before beginning decoding on this new subband. The buffers operate in a ping-pong fashion until the complete RVLC-coded bitstream is processed. Simultaneously, as the incoming bitstream is being routed to one of the buffers, the controller also routes data from the other buffer to the forward or backward decode (not shown in Figure 3) datapaths alternatively every cycle. So each shifting datapath is active every other cycle.

For backward decodability, the data being read from the active buffer is bit-reversed so that the correct end of the data is shifted out of the barrel shifter. The result of the shifter datapath is once again bit-reversed for appropriate table look-up. Stalling mechanisms are also used by the controller to maintain a continuous throughput through the RVLD/RLD pipeline. The local RVLC buffer sizes were properly designed to buffer each RVLC-coded subband in the received bitstream for up to 0.8 bpp compression for 256x256 greyscale images. Each buffers are 1Kx16 bit SRAMs, with enable for low power consumption. Using the same synthesis flow as the RVLE, power analysis results of LUT partitioning are shown in Table 1, and the total power of the RVLD was found to be 619 uW, with the same activity factor as the encoder.

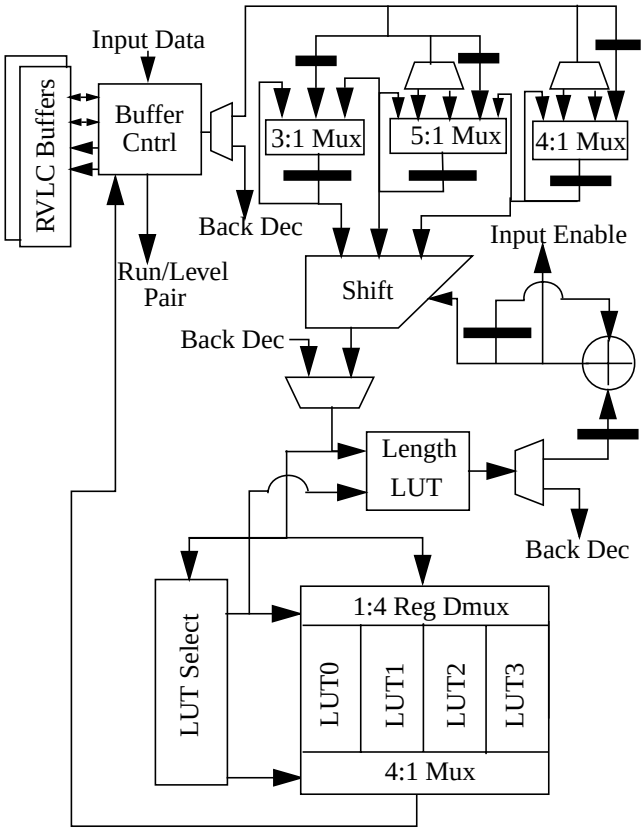

Figure 3: RVLD Architecture.

5. SUMMARY

This paper presents the first published VLSI architectures for a reversible variable length encoder and decoder. Although [7] contains an RVLC coder, it does not describe the architecture. Both architectures provide constant output rates, capable of encoding/decoding one run-level pair/codeword respectively per cycle. The decoder architecture is capable of decoding bidirectionally for improved error resilience. Both architectures contain low power table partitioning, and low power SRAMs for the decoder. The dominant source of power consumption for the RVLE is the shifter datapath, and for the RVLD, the RVLC buffers. Architecture trade-offs for both implementations are given in Table 2. A typical VLD design, with the use of LUT partitioning, is about 65 percent smaller then the RVLD design mentioned here, mainly due to the use of SRAMs for RVLC buffering. There's relatively no area difference between a VLE and the RVLE design mentioned here. Figures 4 and 5 show the layouts in 0.5 *u*m CMOS for both encoder and decoder respectively. Although the architectures here were designed for a wavelet-based codec design, the same architectures can be used for DCT-based video/image codecs. Synchronization markers can be used to segment the bitstream to a discrete number of MCUs or macroblocks for bidirectional decoding.

Table 1: RVLE/RVLD Table Partitioning Results

Table	Entries	Gate Count	Probability of Access	Weighted Power (mW)
0	19/ 15	59/ 37	0.714/ 0.725	0.035/ 0.025
1	64/ 48	168/ 95	0.188/ 0.215	0.061/ 0.021
2	239/192	503/ 308	0.090/ 0.051	0.086/ 0.054
3	412/479	811/ 622	0.008/ 0.009	0.019/ 0.023
Total	734	1441/ 1062	1.000	0.201/ 0.123

Table 2: Architecture Trade-offs

Datapath	Area	Gate Count	RAM	Power (mW)
RVLE	1.9 mm^2	2707	-	0.051
RVLD	7.0 mm^2	4630	32 Kb	0.619

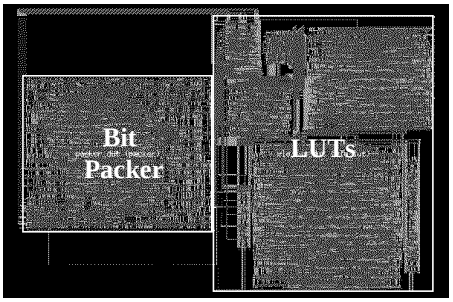

Figure 4: VLSI Layout for RVLE.

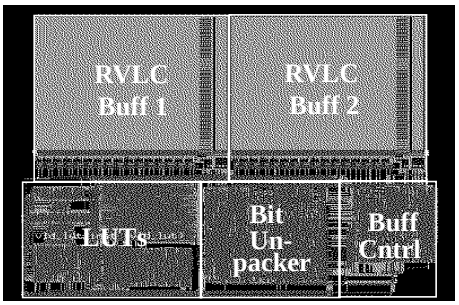

Figure 5: VLSI Layout for RVLD.

6. REFERENCES

[1] A. Chandrakasan, et al., "A Low Power Chipset for Portable Multimedia Applications," *IEEE International Solid-State Circuits Conference*, San Francisco, February 1994.

[2] T. Meng, et al., "Video Compression for Portable Communication Using Pyramid Vector Quantization of Subband Coefficients," *Proceedings 1993 IEEE Workshop on VLSI SP*, pp. 444-445.

[3] M.R. Titchener, "The synchronization of variable-length codes," *IEEE Trans. Inf. Theory*, vol.43, no.2, pp. 683-691, March 1997.

[4] J. Wen and J. Villasenor, "Reversible Variable LengthCodes for Robust Image and Video Transmission," *Proceedings of the Asilomar Conference on Signals, Systems & Computers*, Pacific Grove, CA, Nov. 1997.

[5] S. Molloy and R. Jain, "Low Power VLSI Architectures for Variable-Length Encoding and Decoding," *IEEE Midwest Symposium on Circuits and Systems*, Sacramento, CA, August 1997.

[6] M.T. Sun and S.M. Lei, "A Parallel Variable-Length-Code Decoder for Advanced Television Applications," *Proceedings of the Third International Workshop on HDTV*, Torino, Italy, August 1989.

[7] M. Takahasi, et al., "A 60 mW MPEG4 Video Codec Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme," *IEEE International Solid-State Circuits Conference*, February 5, 1998.

7. ACKNOWLEDGEMENTS

This research was supported by DARPA under contract #DABT63-95-C0100.