EFFICIENT VLSI IMPLEMENTATION OF A VLC DECODER FOR GOLOMB-RICE CODE USING ALTERNATING CODING

Shang Xue and Bengt Oelmann

Department of Information Technology and Media, Mid Sweden University
SE-851 70 Sundsvall, Sweden
xue.shang@mh.se

ABSTRACT

Variable length code (VLC) is used in a large variety of lossless compression applications. Golomb-Rice code (GR code) is one type of VLC that is often encountered in the coding of video and image data. In this work we develop an efficient decoder for GR codes. Unlike the conventional variable length decoders, this new type of decoder needs neither codeword tables nor barrel shifters, while the codeword tables and barrel shifters usually occupy the largest part of the area in the design and both are included in the critical timing path. This proposed decoder is built on the basis of a new coding method for GR codes, which is also proposed in this paper, under the name "Alternating Coding" (ALT). We compare the ALT decoder with the decoder called "VLC decoder using plane separation" (PLS) which is claimed to be one of the most effective VLC decoders. Our results show that the ALT decoder is up to 1.52 times faster, two times smaller, and consumes at most 28% power of the PLS decoder. Moreover, its unique structure also gives this GR decoder great flexibility in decoding different sets of GR codes with constant performances.

1. INTRODUCTION

Image and video coding standards (e.g. JPEG, H.26X, MPEG) all utilize entropy coding in the form of variable length codes (VLCs) for its efficient compression. Although VLCs are efficient in compression, the variable code length of VLCs also limits the decoding throughput. The decoding process needs to identify the codeword boundaries, each of which depends recursively on the previous codeword boundary. Parallelizing VLC decoders are usually done by implementing the decoder with look-up tables and a shifting scheme[3,4]. Codewords and codeword lengths are stored in look-up tables so that they can be matched out according to the input data. The shifting scheme shifts the input data according to the codeword lengths in order to perform decoding continuously. The codeword tables can be implemented with ROM or PLA and the shifting scheme is usually implemented with barrel shifters. These two parts in a VLC decoder occupy the largest portion of the area and as they are the two crucial parts in determining the codeword boundaries, they are both included in the critical timing path of the decoder. Look-up tables and barrel shifters are therefore the performance limiting components in a VLC decoder.

Specially constructed VLCs such as Golomb-Rice code (GR) are developed for different types of image and video data. GR code was first proposed in [1,2] and has recently been applied for coding of prediction errors in lossless image coding applications [5]. GR code belongs to the VLC family, so GR decoders are usually implemented using the general architecture for VLCs, i.e. using look-up tables and a shifting scheme. With the development in mobile video communications, the construction of smaller, faster, and less power-consuming video CODECS becomes increasingly important. In this paper we present a new type of GR decoder based on a coding method that we call "Alternating Coding" method (ALT). It takes advantage of the special properties of GR codes. It does not contain look-up tables, and it is also free of barrel shifters. Therefore it is faster, much more smaller and less power-consuming. In the paper, we compare the performances of the proposed GR decoder with a decoder developed by Jae Ho Jeon et al. [6], under the name of "Fast Variable-Length Decoder Using Plane Separation" (PLS), which was claimed to be one of the most effective VLC decoders. We compare the ALT decoder to the PLS decoder in delay, area and power consumption. Our results show that according to different sets of GR codes, the ALT decoder is up to 1.52 times faster, two times smaller, and consumes at most 28% power in comparison to the PLS decoder. In addition, the ALT decoder has a detachable structure which makes it easy to be reconfigured for different GR codes with constant performances.

The outline of this paper is as follows. First the coding method, "Alternating Coding", for GR codes is described. Then the ALT decoder is presented. After that we present a comparison of the performance of the ALT deocder to the PLS decoder. Finally we draw some conclusions.

2.ALTERNATING CODING

GR code is nearly optimal for coding of exponentially distributed non-negative integers, and describes an integer n in terms of a quotient and a remainder [1,2]. For simplicity, the divisor is often chosen to be a power of 2, 2^k, and is parameterized by k. Therefore a GR code consists of a prefix and a suffix. The prefix of a GR code is a unary expression of the quotient and the suffix of a GR code is a k-bit fixed length binary code representing the remainder. For example, for a GR code with $k = 2$, the number 9 would be represented as 11001. By considering prefixes and suffixes of the code separately, it can be seen that the prefixes are just a set of unary codes whose lengths grow linearly with the values of the quotients. As they are unary, it does not matter whether all ones or all zeros are used to represent them. When transmitting only the prefixes, all-one codes and all-zero codes can be used alternatingly in a sequence. Thus the codeword boundaries can be easily determined by detecting the changing of the value of a bit in the prefix series. While the suffixes are some fix-length codes and when only the suffixes are transmited, codeword boundaries can be determined by counting the bits. Therefore, if the prefixes and the suffixes are separately transmitted, the codeword boundary detection will be simplified because the need of a recursive procedure is eliminated. For instance, with $k = 2$, a GR series with four codewords 11000 1011 111001 1001 , will be turned into a 111 00 1111 00 prefix series and a 00 11 00 01 suffix series. The coding scheme of alternating coding is shown in Figure 1. The alternating coding can easily be achieved in the GR encoder by replacing the codeword table with an all-one code table or an all-zero code table and by inversing the prefix code every other clock cycle.

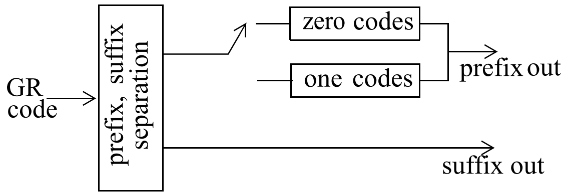

Fig. 1: Alternative Coding Method

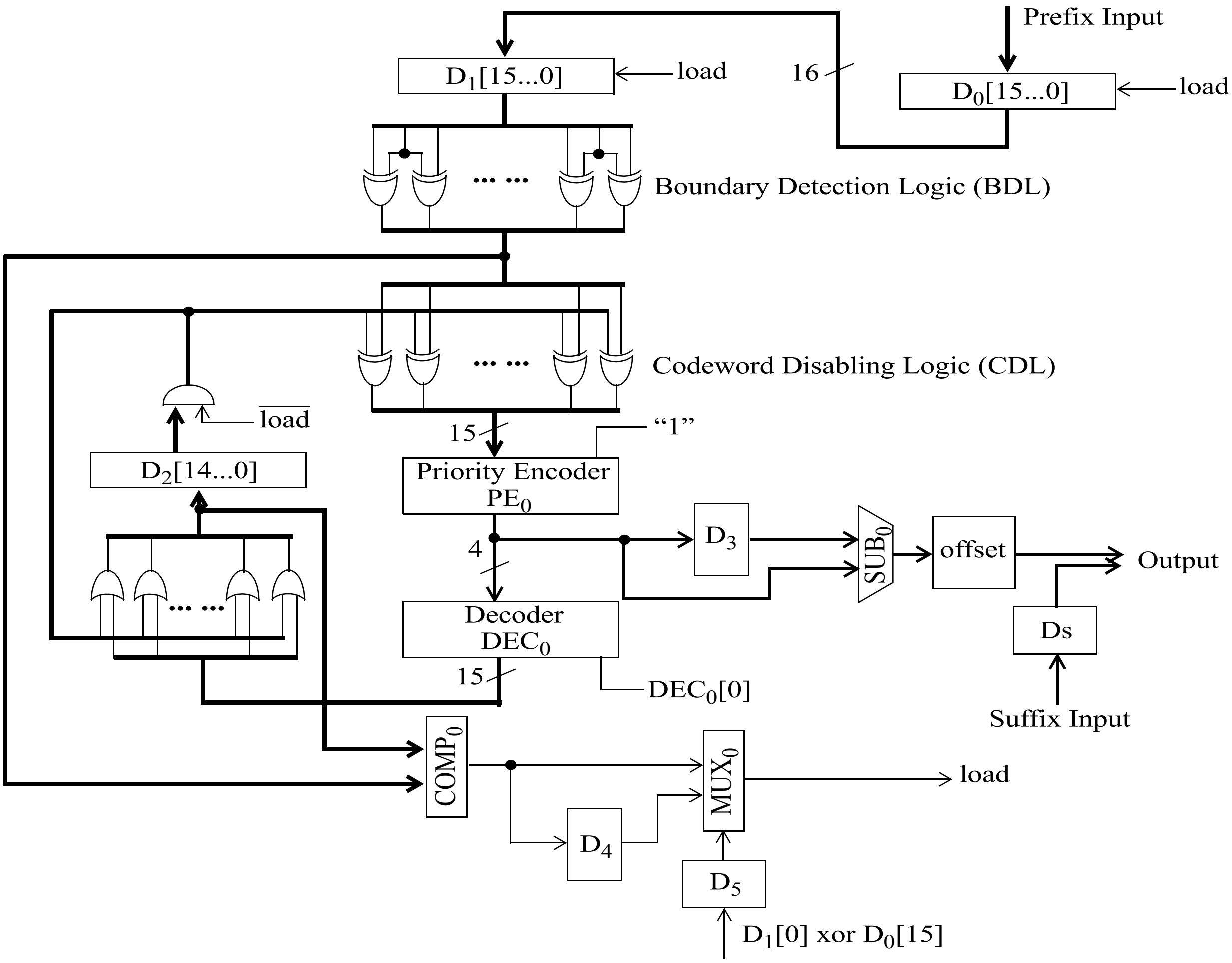

Fig. 2: ALT decoder

3.ALT DECODER

The ALT decoder proposed in this paper is based on the "Alternating Coding" method. The architecture is described in Figure 2. We assume the maximum prefix length to be 16 bits.

The ALT decoder has two inputs for the separated prefix and suffix series. One is the prefix input and the other is the suffix input. The decoder consists of one 16-to-4 priority encoder (PE_0), one 4-to-16 decoder (DEC_0), two 16-bit buffers (D_0 and D_1), one 15-bit register D_2, one 4-bit register D_3, one 15-bit comparator ($COMP_0$), one 4-bit subtractor (SUB_0), one 1-bit 2:1 multiplexer (MUX_0), one n-bit register D_s (n is the length of the suffix) and two 1-bit registers (D_4 and D_5). The prefix input of the decoder is put into the two buffers D_0 and D_1, the first two bytes in D_1 and the second two bytes in D_0. The first two-byte prefix series is then fed to the xor-gates in the "Boundary Detection Logic" (BDL) where two consecutive bits are xored with each other. As the prefixes are now denoted in alternating all-one and all-zero codes, only at each prefix boundary a "1" will be generated by the xor operations. Therefore, each "1" indicates a prefix boundary. The output after the BDL is then fed into the priority encoder PE_0 in order to generate the position of the first codeword boundary. Register D_3 is originally loaded with the number 16 (that is "0000" in a 4-bit binary code). The length of the first prefix is then calculated by SUB_0 and at the same time D_3 is updated with the position of the first codeword boundary. The 4-to-16 bit decoder DEC_0 generates the position of the first codeword boundary and disables the first "1" of the input of the priority encoder by using the or-gates and the "Codeword Disabling Logic" (CDL). In the next clock cycle, the second codeword boundary is encoded into PE_0. Again the second codeword boundary is put to D_3 and its position is decoded by DEC_0. The same operations are then repeated. As the prefix of a GR code is the unary expression of a quotient, the quotient itself can be easily generated by offsetting the integer which represents the prefix length. Therefore, by offsetting the output of SUB_0 the value of the quotient can be generated. The suffix of a GR code is already a binary expression, so the actual integer a GR code represents can be generated simply by concatenating the suffix and the decoded prefix. When decoding is performed till the end of D_1, the output of D_2 will then be accumulated to be the same as the output of BDL, and the output of $COMP_0$ is set high. The operation $D_1[0]$ xor $D_0[15]$ is used to find out if the prefix in D_1 still continues in D_0. If the prefix continues, the "load" signal is generated immediately and new data are loaded into the buffers. If the end of D_1 is the end of a prefix, then the load signal needs to be delayed to the next clock cycle. A multiplexer MUX_0 and a 1-bit register D_4 are used to complete this.

In this ALT decoder, neither look-up tables nor shifting scheme are needed, and it is capable of decoding one codeword per clock cycle.

4.COMPARISON OF PERFORMANCE

The ALT decoder is compared with the PLS decoder developed by Jae Ho Jeon et al.[6]. Their decoder can be described as in Figure 3.

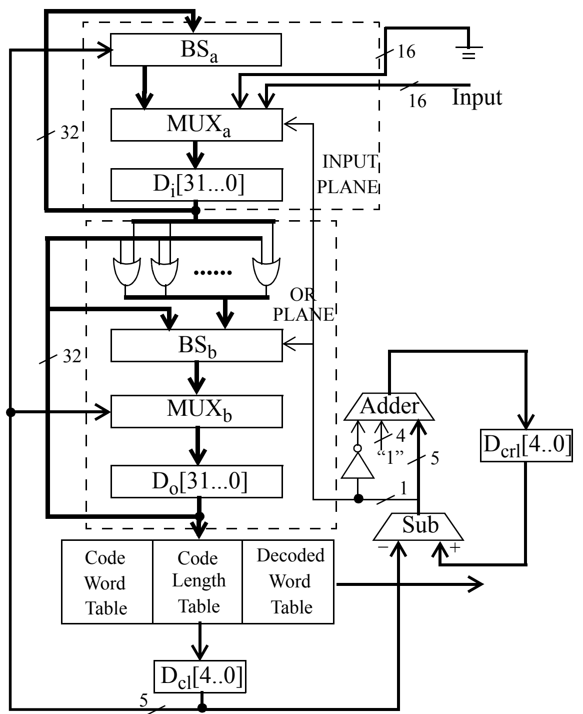

Fig. 3: PLS decoder

For a set of GR codes with maximum codeword length of 16 bits, the decoder consists of two separate planes. Each plane consists of a barrel shifter, a 32-bit 2:1 multiplexer, and a 32-bit output register. The codeword table in this case is loaded with a GR codeword table and so is the code length table. This decoder is capable of decoding one codeword per clock cycle and the design makes the coding process parallel by using an "or plane". However, feeding the codeword length from the look-up tables back to the barrel shifters still limits the decoding throughput. All the possible codewords, codeword lengths and decoded integers need to be implemented in the look-up tables, and two types of barrel shifters are included. These all limit the efficiency of the PLS decoder. According to our synthesis results, look-up tables and barrel shifters take as much as at least 67% of the total area of the PLS decoder.

We compare the delay, area and power consumption of the ALT decoder to those of the PLS decoder. Both of the decoder types have been implemented in synthesizable VHDL and their performance has been estimated according to the synthesis results. For each type, three decoders for GR codes have been implemented: without suffix, with 1-bit suffix and 2-bit suffix. The maximum prefix length is kept constant as 16 bits. The results are shown in Figure 4. Both types of decoders are implemented in VHDL and synthesized using *Design Compiler* from Synopsys. The delay has been obtained from static timing analysis and the figures for power consumption from Synopsys' *Power Compiler*. A standard cell library in a 0.5µm CMOS process has been used.

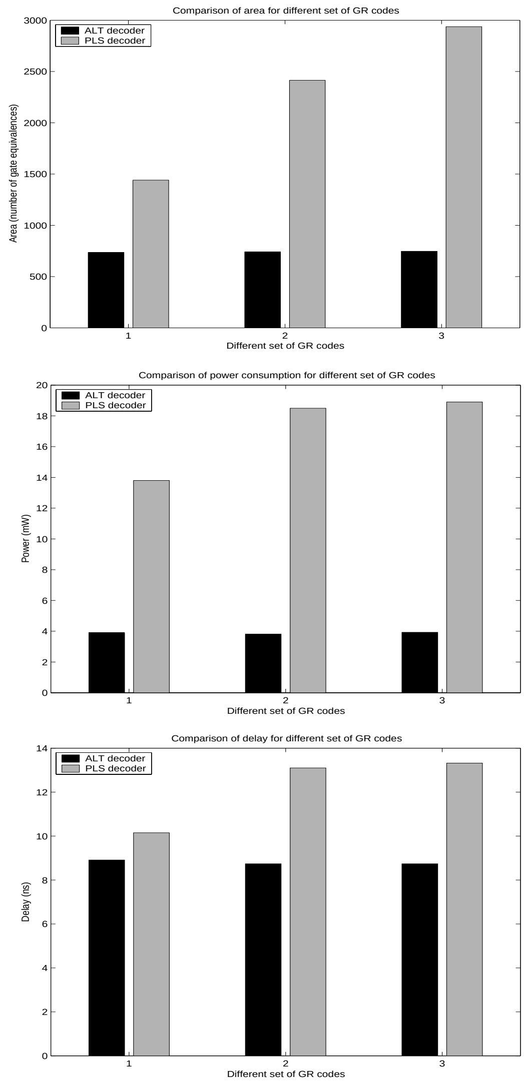

Fig. 4: Comparison of performances of PLS and ALT decoder

In Figure 4, the numbers 1, 2 and 3 on the x-axis represent three different sets of GR codes, 1 stands for GR codes without suffix, 2 for GR codes with 1-bit suffix, and 3 for GR codes with 2-bit suffix. From these graphs it is obvious that the ALT decoder performs much better than the PLS decoder in area, power and delay. The improvements are dramatic for area and power. For GR codes without suffix, the ALT decoder gets only 87% delay, 51% area and 28% power consumption of those of the PLS decoder. For GR codes with 2-bit suffix, the related performances are as good as 65% delay, 25% area and 20% power consumption of that of the PLS decoder. Moreover, the performances are constant for different set of GR codes, whereas the performance of the PLS decoder degrades quite rapidly as the suffix length grows. When the maximum codeword length increases from 16 bits to more than 16 bits yet less than 32 bits, the barrel shifters in the PLS decoder need 5 bits instead of 4 bits to count the number of bits needed to be shifted. Therefore, when 1-bit suffix is added to the prefix that has the maximum prefix length of 16 bits, there are abrupt increases in delay, power and area in the PLS decoder, and this makes the ALT decoder comparatively better.

5.CONCLUSIONS

We propose the ALT decoder for decoding GR codes. This decoder is based on a coding method that we call "Alternating Coding". It can be seen that the ALT decoder is up to 1.52 times faster, two times smaller, and 3.5 times less power-consuming than the PLS decoder, while the PLS decoder is declared to be one of the best decoders for variable length codes. In addition, its unique structure gives the ALT decoder great flexibility in decoding different sets of GR codes with constant performances, which is a great advantage in practice.

6. REFERENCES

[1] S. W. Golomb, "Run-length endodings," in *IEEE. Trans. Inf. Theory*, vol. IT-12, pp. 399-401, July 1966.

[2] R. F. Rice, "Some practical universal noiseless coding techniques," in *Tech. Rep.*, JPL-79-22, Jet Propulsion Laboratory, Pasadena, CA, March 1979.

[3] M. T. Lei, M. T. Sun, "An entropy coding system for digital HDTV applications," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 1, pp. 147-155, March 1991.

[4] H. D. Lin, D. G. Messerchmitt, "Designing high-throughput VLC decoder Part II-Parallel decoding methods", in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 197-206 June 1992.

[5] Jiangtao Wen, John D. Villasenor, "Reversible Variable Length Codes for Efficient and Robust Image and Video Coding", in *Data Compression Conference.*, pp. 471-480, 1998.

[6] Jae Ho Jeon et al, "A fast variable-length decoder using plane separation," in *IEEE. Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 806-812, Aug. 2000.