A high speed and efficient architecture of VLD for AVS HD video decoder

Yutong Liu, Zhenqiang Yang, Huizhu Jia, Don Xie
National Engineering Laboratory for Video Technology
Peking University
Beijing, China

***Abstract*—In this paper, we present a high speed and efficient architecture of Variable Length Decoder for AVS video standard targeted for all-hardware implementation. Besides the regular operations of decoding Fixed Length Code, unsigned or signed k-th Exp-Golomb Code and 2D Variable Length Code, the proposed design provides other functions such as de-stuffing or pre-fetching the Bitstream. It can perform decoding syntax elements of sequence, frame, slice, and macro block. The complete architecture has been described in Verilog HDL, simulated with Modelsim SE 6.3c simulator, implemented using FPGA of Xinlinx Vertex5 VLX330. Without any strict constraint, the design can achieve a working frequency at 190 MHz after synthesis with Synplify_pro9.4, and the critical path is less than 6.5ns after place & route. The throughput of the design is 1 codeword per clock. In all, the architecture fully meets the demands of AVS HD decoder. It can support real-time decoding for 1080P @ 30 frame/s or 1080i @ 60 field/s videos. Inevitably, the cost for such a high speed design is consuming more hardware resources. Report of Place & Route shows about 9.1K LUTs (4% of the total LUTs in FPGA chip) are consumed by our design. Although the VLD architecture was originally designed for AVS video standard, the idea of the design can be easily adapted to other video standards.**

I. INTRODUCTION

As a second-generation video standards, AVS [1] video coding standard also uses Variable Length Code (VLC) as the main entropy coding technique. The main idea of VLC is to assign shorter codewords to source symbols of high probability and longer codewords to less frequenty symbols, when the probability distribution of the these symbols is given. As a result, compared with the fixed-length code, the total bits used for coding the source symbols are reduced.

Different from MPEG-2 [2], which adopts a two dimensional VLC (2D-VLC) with two tables, AVS uses as many as 19 tables for entropy coding, named Context-based Adaptive 2D-VLC (CA-2D-VLC). With these VLD tables, we can transform run-level pairs to codenums, meanwhile, next table number can be decided according to current value of level. In order to achieve better coding performance, AVS adopts signed k-th Exp-Golomb Code for looking up VLC tables. So when decoding a codeword, we have to decode Exp-Golomb Code to get the corresponding codenum first, then we use the codenum to look up a certain VLD table for run-level pairs, and finally, we get table number and Golomb Coding rank for the next codeword. As can be seen, the process for Context-based Adaptive Variable Length Decoding (CA-2D-VLD) is quite complicated.

Researchers have provided many architectures of VLD for video decoder [3]-[6]. Some of them can be categorized to a tree-based architecture [3]. Some adopted bit-parallel approach to decode multiple symbols in one clock cycle [4]. And others used look-up-tab techniques to achieve high performance [5]-[7]. Tree-based architectures have good flexibility, but they usually cannot decode one symbol at a time to achieve high frequency. Parallel methods can check more than one symbol in a clock, however, it takes a long period (so system clock frequency is restrained). So there is a strong need for a VLD for high-performance real-time applications. With scheme of mapping two-bit tree onto memory, paper [3] presents a tree-based VLD which can decode one symbol in 1.54 cycles and work at 100 MHz. In [4], the architecture has to take 110ns to analyze 32 bits which contain an average 3.5 symbols. Nowadays, most designers choose the third type of architecture. Paper [5]-[6] proposed look-up-table based schemes which can decode one symbol per clock and work at a high frequency. But both of them use CMOS technology to achieve a high working frequency and not suitable for real-time application based on FPGA. Besides, these schemes cannot always keep a constant throughput when decoding a codeword astride two registers.

In this paper, we propose a high speed VLD architecture for HD video decoder based on FPGA implementation. It can perform Fixed Length Decode, unsigned or signed k-th Exp-Golomb Decode, CA-2D-VLD, de-stuffing and pre-fetching bitstream. According to AVS VLD tables, we devised several special types of table for fast computing. These tables enable us to extract a codeword and update table number and Golomb Code rank at one clock. What's more, the special shifting scheme of bitstream shifter can avoid searching two registers for a codeword. Thus it can always keep a constant throughput of one codeword per clock. Besides, we adopt an efficient way to save block RAM for VLD tables, with certain overhead on the logic.

The rest sections are organized as follows. Section Ⅱ briefly explains AVS entropy coding algorithms, including k-th Exp-Golomb Decode and CA-2D-VLD. We describe the whole proposed architecture in section Ⅲ and present the

simulation and implementation results in section Ⅳ. Then we make a conclusion and list future work in the final section.

II. ALGORITHMS INTRODUCTION

A. K-TH Exp-Golomb Decode

Exp-Golomb Code is a variable length code with regular construction. A typical structure of k-th Exp-Golomb Code is shown as following.

$$codeword = \underbrace{\overbrace{0...0}^{m}1\overbrace{X_{n-1}X_{n-2}X_0}^{n}}_{L}$$

$$n = m + k, L = m + n + 1 = 2m + k + 1$$

The codeword contains m bits of "0", a bit "1" and n bits random value. L is the length of the codeword by bits. There are two equations signify the numerical relationship between m, n, k, and L.

The last n bits consist of the suffix of codeword and carry effective information. We define a variable INFO to represent the value of last n bits.

$$INFO = \sum_{i=0}^{n-1} x_i \cdot 2^{i-1}$$

Table I presents the mapping relation between codeword and codenum.

B. CA-2D-VLD

In AVS, CA-2D-VLC, a combination of k-th Exp-Golomb Code and 2D-VLC, is designed to encode these residual pairs. Each run-level pair is assigned a CodeNum according to a suitable table. Then the CodeNum is coded with signed k-th Exp-Golomb Code.

There are about four steps in decoding a coefficient.

a) *Check the bitstream register bit by bit and find the first '1", then we can calculate the length of current codeword for the k is known (at the very beginning, each block has its own initial value of table number and golomb rank). Update the value of bitstream register immediately for the next codeword if we need pipeline operation.*

b) *Perform signed k-th Golomb Decode with inputs of codeword and k, and we can get the corresponding codenum.*

c) *According the value of codenum we can decide whether it is a normal codenum or an ESCAPE one. For normal ones, we look up VLD tables to get run-level pairs. For ESCAPE ones, carrying out an escape function to obtain the value of run, and then wait for the next codenum to calculate the value of level.*

d) *Update the value of table number and k according the value of level if the level is obtained in last cycle.*

TABLE I. STRUCTURE OF K-TH GOLOMB DECODE

rank	code word	value range	codenum
0	1	1	$2^m + INFO - 2^0$
	0 1 x_0	2～3	
	0 0 1 x_1 x_0	4～7	
	……	……	
1	1 x_0	2～3	$2^{m+1} + INFO - 2^1$
	0 1 x_1 x_0	4～7	
	0 0 1 x_2 x_1 x_0	8～15	
	……	……	
2	1 x_1 x_0	4～7	$2^{m+2} + INFO - 2^2$
	0 1 x_2 x_1 x_0	8～15	
	0 0 1 x_3 x_2 x_1 x_0	16～31	
	……	……	
3	1 x_2 x_1 x_0	8～15	$2^{m+3} + INFO - 2^3$
	0 1 x_3 x_2 x_1 x_0	16～31	
	0 0 1 x_4 x_3 x_2 x_1 x_0	32～63	
	……	……	

Apparently, the whole process is complicated and time consuming, and most of all it is a serial operation in nature and is difficult to make it pipelined. In most of architecture it usually becomes the critical path, for the k is context-based and the codeword boundary cannot be determined until the previous codeword has been decoded.

III. PROPOSED ARCHITECTURE

The VLD module is the critical part of the whole AVS decoder. It can support the decoding of Fixed Length Code, unsigned or signed k-th Exp-Golomb Code, and CA-2D-VLC. Meanwhile it provides functions such as de-stuffing and pre-fetching for input bitstream.

A. Overview of Proposed VLD Architecture

Our VLD architecture is shown in Fig.1. From system level, it gets some basic parameters and commands from embedded CPU via AHB BUS and feeds back states of decoding and decoded information to CPU. There is a MB syntax FIFO

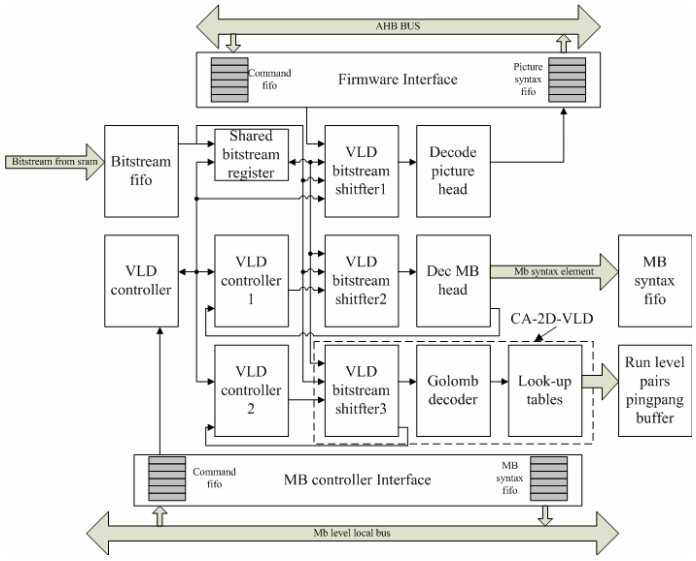


Figure 1. Overview of VLD

between VLD and MB Controller. After decoding syntax elements, VLD push these syntax elements into the FIFO and MB Controller distributes them to other modules in the later stage.

We designed three controller modules, for decoding process is too irregular. VLD controller is responsible for the whole control of the decoding flow. VLD controller1 is in charge of controlling the decoding of syntax elements in MB header. While VLD controller2 schedule the decoding of coefficients in every block. For modules such as decoding picture header, decode MB header, Golomb decoder, they function as the name tells, including decoding Fixed Length Code and k-th Exp-Golomb Code. There are three significant VLD bitstream shifters in our design. They are responsible for extracting codwords from bitstream registers. Especially, VLD bitstream shifter3 can extract one codeword per clock. Module Look-up Tables mainly executes looking up VLD tables or escape-function and output run-level pairs. CA-2D-VLD comprises VLD Bitsream Shifter3, Exp-Golomb Decode and Look-up Tables.

B. Design of CA-2D-VLD

In Fig.2, we give a comparison between traditional structure of CA-2D-VLD (fig.2.a) and our proposed structure (fig.3.b). In fig.2.b, yellow figures such as RA, RB, RC, Length_B are variables triggered by the rising edge of the clock. Blocks in cyan are logic modules triggered by the same clock too. Gray blocks are elaborated tables for above variables, of which each has a series tables. These tables are combinational logics circuits which enable us to update the value of corresponding variable in one clock at a high speed.

For VLD bitstream shifter3, we use three 64-bit registers (RA, RB and RC) for extracting a codeword. The total bits of RC are always available for checking the first "1" from the MSB every clock. RB is design to compensate the lost bits of RC for current codeword. RC is responsible to keep data form

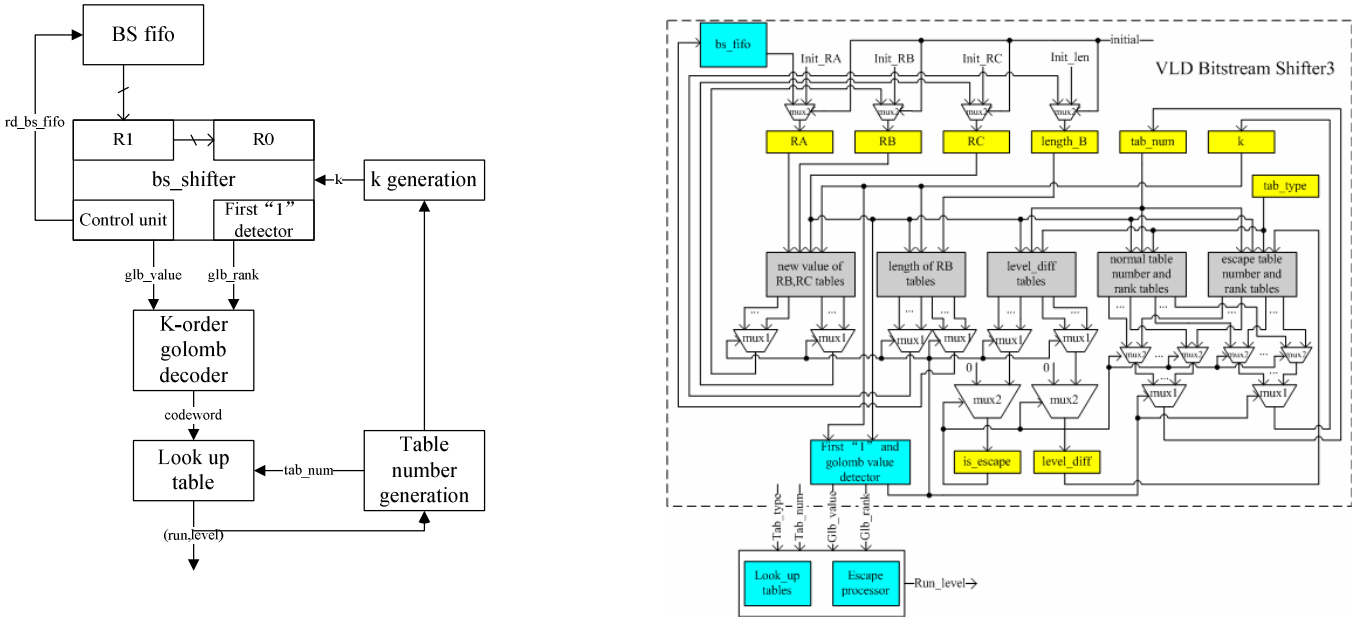
FIFO and transfer its value to RB if the length of valid bits in RB is not enough for shifting. The essential of the shifter is to make sure all of bits in RC are available and valid. So in this way, we can always extract a codeword from RC every clock without being interrupted by boundary of registers.

Let's see an example of the decoding process. Assuming register RC with value of 16 bits string "00111010…" (actual length of RC is 64 bits), k with value of 2 and table of VLC3_INTRA, we can obtain the first codeword is "0011101" and its corresponding value of codenum is 25. Then look up table VLC3_INTRA and find codenum 25 is mapped to run-level (0, 8). Finally we know the level 8 is equal with the threshold of VLC5_INTRA, so we update the value of table number and k for the next codeword. It's not difficult to find that there is a mapping relation between codeword 29 ("0011101") and next table number and k. Actually, only when codenums are less than 59 will we have a mapping relation with run-level pairs. So we can list these codenums and their mapping relation and make tables for each variable.

C. Scheme for Saving Block RAM

In AVS there are 19 tables for CA-2D-VLC and 2 tables for mapping syntax element CBP to codenum. We adopted special schemes for the look-up to reduce the number of RAMs needed for coding these tables.

A block RAM in Xilinx FPGA V5 has a capacity of 18K bits. So it really wasteful for using two such tables to represent these CBP tables(each of them is 6×64 bits). After careful analysis, we find that only one CBP table is used at any given time. So we can merge these two tables into one table and increase the ram address by one bit.



(a) (b)

Figure 2. (a)Traditional structure of CA-2D-VLD, (b)Proposed structure of CA-2D-VLD

TABLE II. TRIMMED TABLE OF VLD

codenum	run	level
0	0	1
1	0	-1
2	EOB	
3	0	2
4	0	-2
5	1	1
6	1	-1

As for these 19 tables, we use special scheme to reduce the cost of block RAM too. According to AVS standard, CA-2D-VLC adopts signed k-th Exp-Golomb Code. Generally, when we make these VLD tables, we have to list all possible values bellow 59. Take codenum "5" and "6" for example in Table Ⅱ. Their corresponding run-level pairs are (1, 1) and (1, -1). Actually we can only list "5" in Ⅱ and derive the output of "6". That means we can make use of the parity of the codenum to reduce the cost of tabulation. Theoretically, the cost of bits can be reduced by 50%. Additional overhead is present because codenum EOB disturbs the regularity of the look-up.

Experiments show general design of table VLC0_INTRA consumes one Block RAM and the proposed design uses only 5 LUTs. So these schemes are really practical.

IV. SIMULATION RESULTS AND IMPLEMENTATION PERFORMANCE

Our proposed architecture of VLD is described in Verilog HDL, simulated with Modelsim SE, and implemented using Xilinx FPGA Vertex5 VLX330.

A. Simulation Results

Using our design, we can decode all of the syntax elements of MB header within 100 cycles (can be further improved) and decode one coefficient in a clock. We have tested the proposed architecture with several standard sequences. Results are shown in Table Ⅲ.

B. Implementation Performance

With the novel architecture of VLD Bitstream Shifter3, the Decoder can function at frequency of 150 MHz. Actually, the synthetic report shows the estimated frequency can reach 190 MHz and the critical path is less than 6.5ns after Place & Route. The design consumes about 9K LUTs (4.5% of total LUTs), 3K Registersn (1.5% of total Registers) and 1 Block Ram (those special tables for accelerating CA-2D-VLC and VLD tables are synthesized into LUTs). Comparisons are listed in Table IV.

In order to better understand about the relation between the gain of frequency and cost of logic cells. We implemented our design using the same FPGA with [8]. Report shows we nearly use two times as many LUTs as in [8] and gain about two times the frequency of [8]. But we use less Block Rams.

TABLE III. SIMULATION RESULTS

Sequence	QP	tested frames	average CodeNums per MB	average cycles per MB
forman cif	28	50	27.5	148.2
city 720p	28	30	21	140.7
raven 1080P	15	30	47	170.1
fireworks 1080P	28	10	147.6	271.6
fireworks 1080P	15	10	257.3	380.5
riverbed 1080P	15	10	110	237.3

TABLE IV. IMPLEMENTATION PERFORMANCE

	[5]	[6]	[7]	[8]	Proposed
technology	0.18μm cmos	0.18μm cmos	FPGA1	FPGA2	FPGA3
function	avs vld	avs vld	mpeg-2 vld	avs vld	avs vld
freqiemcy (MHz)	166	125	55	75	153
throughput (sym/cycle)	1	1	1	<1	1
Gate/LUTS	15KG	2.8KG	1KL	5KL	9KL
ROM/RAM	4Kb	6.5Kb	-	-	1Kb

Note: FPGA1: Altera EP20K 200 EFC4840-2x; FPGA2: Xilinx Virtex4 XC4VLX80; FPGA3: Xilinx Virtex5 XC5VL330

V. CONCLUSION AND FUTURE WORK

In all, we design an efficient architecture of VLD for real-time HD video decoders based on FPGA. It can perform Fixed Length Decode, unsigned or signed k-th Exp-Golomb Decode, CA-2D-VLD, de-stuffing and pre-fetching bitstream. Also the ideal for high speed VLD architecture can be easily adapted to other standards. Future work need to be done to reduce the cost of hardware resources and support multiple standards.

REFERENCES

[1] Audio Video Coding Standard Workgroup of China (AVS), Advanced Coding of Audio and Video - Part 2: Video, December 2004.

[2] ISO/IEC IS 13818, General Coding of Moving Picture and Associated Audio Information, 1994.

[3] Bai-Jue Shieh, Yew-San Lee and Chen-yi Lee, "A High Throughput Variable Length Decoder with Modified Memory Based Architecture," in ISCAS vol.2, pp. 486-489.

[4] Nikara, J.; Vassiliadis, S.; Takala, J.; Sima, M.; Liuha, P., "Parallel multiple-symbol variable-length decoding," IEEE Trans. Computer Design. VLSI in Computers and Processors, 2002, pp. 126-131.

[5] Bin Sheng, Wen Gao, Don Xie, et al., "An efficient VLSI architecture of VLD for AVS HDTV decoder," IEEE Trans. Consumer Electronics, Vol. 52, No. 2, May, 2006.

[6] Li, L. Yu, and J. Dong, "A decoder architecture for advanced video coding standard," SPIE Proc. of Visual Communications and Image Processing, pp. 2299-2308, 2005.

[7] Chen Guanghua, Ma Shiwei, Li Min, et al, "A fast variable length decoder with optimized lookup tables on FPGA," Proc. 7th International Conference on Solid-State and Integrated Circuits Technology, vol.3 pp. 1649-1652, 18-21 Oct. 2004.

[8] Zheng Si, Zhu Lan-juan, Liu Pei-lin, "Design of an AVS HD-video Decoder," Video Engineering (in China), No.12,Vol 31, pp 26-30, 2007.