ENHANCED ARCHITECTURAL SUPPORT FOR VARIABLE-LENGTH DECODING

Mohanarajah Sinnathamby

ViXS Systems
245 Consumers Road, Suite 301
Toronto, Ontario, Canada M2J 1R3

Subramania Sudharsanan and Naraig Manjikian

Dept. of Electrical and Computer Engineering
Queen's University
Kingston, Ontario, Canada K7L 3N6

ABSTRACT

This paper proposes a new architecture for efficient variable-length decoding (VLD) of entropy-coded data for multimedia applications on general-purpose processors. It improves on earlier proposals for low-complexity performance-enhancing hardware structures that exploit prefix/suffix properties of variable-length codes for common multimedia formats [1]. The enhanced architecture is compared to the previous architectures in terms of complexity and operating speed for FPGA implementation, and also in terms of area requirements, power consumption, and operating speed for a 0.18-μm ASIC fabrication process. Simulation results are reported for a pipelined processor with caches executing MPEG-4 software where VLD performance is doubled by incorporating the proposed architecture.

1. INTRODUCTION

The ubiquity of multimedia data is leading to the inclusion of performance-enhancing hardware support for encoding and decoding such data on general-purpose and embedded processors. Although instructions for bit-level parallelism can improve performance for many aspects of multimedia decoding, the variable-length decoding (VLD) portion has inherent serial characteristics. VLD on specialized hardware that is specific to a particular multimedia format has seen significant improvements since early work in this area [2]. This specialized approach cannot easily be extended to general-purpose processors that need the capability to decode multiple different formats. VLD on general-purpose processors has only seen modest gains through certain architectural improvements, even though it may account for up to 30% of the decoding time in a given application [3]. Recent work has augmented an existing media processor with programmable logic for custom VLD acceleration [4], but there are inherent chip-area penalties with this approach. Instead, we have previously proposed the incorporation of flexible instruction extensions for VLD acceleration with modest implementation complexity that are applicable to general-purpose processors [1].

Multimedia data is typically compressed using lossy transform coding followed by lossless entropy coding [5]. The latter commonly uses modified Huffman coding with fixed or dynamic codeword tables. The first column of Table 1 gives sample codewords for chroma block patterns in an MPEG-4 video [5]. The codewords in such a table can be grouped by their prefix of the leading number of zeros (LNZ) to enable efficient variable-length decoding. Once a codeword is classified into its LNZ group, the remaining codeword suffix can be an index into a table that contains the decoded information [1, 3, 6]. Software-based MPEG decoders have used this property for symbol-by-symbol decoding, rather than slower bit-by-bit decoding [3, 6]. Although this property has been used in hardware accelerators for a specific multimedia codec [2], flexible and efficient mechanisms using the same property to support different multimedia codecs in general-purpose processors are lacking.

We have recently proposed two low-complexity hardware architectures that exploit codeword prefix/suffix characteristics in order to enhance VLD performance in general-purpose processors [1]. Our memory-based (MB) decoding architecture uses a small intermediate memory in the processor with a number of entries equal to the number of distinct groups in a codeword table having the same LNZ count. Each entry provides the suffix length for the group and a base address in the main memory where the group-related codeword table is located. Our Single Fixed-Length Suffix (SFLS) architecture, on the other hand, does not use a small memory in the processor and hence has reduced hardware complexity. Instead, it uses the maximum suffix length across an entire codeword table for indexing all group tables in memory. As a consequence, it requires larger group tables than the memory-based architecture.

This paper proposes an enhanced Multiple Fixed-Length Suffix (MFLS) architecture for incorporation into general-purpose processors. It exploits prefix/suffix properties of variable-length entropy-coded data, similar to our previous architectures [1], but seeks to balance hardware complexity and memory requirements. The remainder of this paper describes the proposed architecture, presents FPGA and ASIC synthesis results, and summarizes performance results from instruction-level simulation.

2. THE MULTIPLE FIXED-LENGTH SUFFIX ARCHITECTURE

The Multiple Fixed-Length Suffix (MFLS) architecture consists of combinational logic and an associated group control register. It would be implemented in a general-purpose processor and used by a special instruction to support variable-length decoding. The combinational logic performs the LNZ count, group selection, bit-shift, and arithmetic operations that generate the table index for variable-length decoding. The latter three operations, in particular, depend on the contents of the group control register that is configured by standard control register access instructions prior to executing VLD code. The maximum number of groups supported by a hardware implementation of the MFLS architecture is fixed. Up to that maximum number, the actual grouping of codewords into subsets with sequential LNZ counts, as explained below, is at the discretion of the multimedia programmer.

2.1. Definition of Groups

Let $L - 1$ represent the maximum LNZ count for a collection of variable-length codewords. In our previous proposals [1], the memory-based (MB) architecture would require L codeword tables, whereas

Table 1. Prefix properties in MPEG-4 Video Table B-7

Codeword	LNZ	Group	s_{ℓ_m}	S_m	$offset_{\ell_m^{min}}$
`1`	0		0		
`010`	1		1		
`011`	1	m=0	1	1	0
`0010`	2		1		
`0011`	2		1		
`0001 00`	3		2		
`0001 01`	3		2		
`0001 1`	3		1		
`0000 100`	4		2		
`0000 101`	4		2		
`0000 110`	4		2		
`0000 111`	4		2		
`0000 0100`	5	m=1	2	2	6
`0000 011`	5		1		
`0000 0101`	5		2		
`0000 0010 0`	6		2		
`0000 0010 1`	6		2		
`0000 0011`	6		1		
`0000 0001 0`	7		1		
`0000 0001 1`	7	m=2	1	1	22
`0000 0000 1`	8		0		

the Single Fixed-Length Suffix (SFLS) architecture would require only one codeword table. The MFLS architecture that is proposed in this paper is a compromise between the two previous architectures in that it may use at most M codeword tables, where $1 < M \leq L$, and this value of M represents the maximum number of groups supported in an MFLS hardware implementation. The programmer may then define up to M groups of codewords in the software.

Each codeword group m that is identified by the programmer for the MFLS architecture reflects a sequence $\{\ell_m^{min}, \ell_m^{min}+1, \ldots, \ell_m^{max}\}$ of LNZ counts for that group, where ℓ_m^{min} is the minimum LNZ count and ℓ_m^{max} is the maximum LNZ count for the group. Let s_{ℓ_m} denote the suffix length for the subset of codewords in a group m with the LNZ count ℓ_m. The suffix is the portion of the original codeword that remains after the leading number of of zeros and the first non-zero bit are excluded. The maximum suffix length for the entire group is then given by $S_m = \max(s_{\ell_m^{min}}, s_{\ell_m^{min}+1}, \ldots, s_{\ell_m^{max}})$. From these definitions, the address offset in a group table for a codeword with LNZ value of ℓ that is contained in group m is given by

$$\begin{aligned} offset_\ell &= 2^{S_m} \cdot (\ell - \ell_m^{min}) + \sum_{i=0}^{m-1} 2^{S_i} \cdot (\ell_i^{max} - \ell_i^{min} + 1) \\ &= 2^{S_m} \cdot (\ell - \ell_m^{min}) + offset_{\ell_m^{min}}. \end{aligned}$$

The total number of memory entries to store all of the codewords is then $\sum_{m=0}^{M-1} 2^{S_m} \cdot (\ell_m^{max} - \ell_m^{min} + 1)$.

The application of the above definitions is illustrated for the variable-length codewords given in Table 1. Inspection of the codewords suggests that three groups are appropriate. Codewords with an LNZ count ranging from 0 to 2 are assigned to group 0, whose minimum LNZ value is $\ell_0^{min} = 0$. Codewords with an LNZ count ranging from 3 to 6 are assigned to group 1, whose minimum LNZ value is $\ell_1^{min} = 3$. Finally, codewords with an LNZ count of 7 or more are assigned to group 2, whose minimum LNZ value is $\ell_2^{min} = 7$. The maximum suffix lengths for each of the three groups are $S_0 = 1$, $S_1 = 2$, and $S_2 = 1$. The group offset values are provided in Table 1.

2.2. VLD Instruction Format

To utilize the MFLS architecture in a general-purpose processor implementation, the following instruction is proposed for a typical three-operand instruction set architecture:

```
    vldecode rdest, rsrc1, imm
```

The instruction uses the left-aligned codeword in register `rsrc1` and returns the index in the VLC table identified by the `imm` field in `rdest`. The internal group control register that is used by this new instruction consists of group minimum LNZ values ℓ_m^{min}, right-shift amounts $W - S_m$, where W is the maximum suffix length supported, and offset values corresponding to each group m. The right-shift amount $W - S_m$ aligns the suffix bits to form the index for the codeword table entry. Precomputed offset values must be used in order to reduce storage requirements for a VLC table. Multiple group control registers could support multiple VLC tables, and the `imm` field in the instruction would select the intended one.

2.3. 3FLS Architecture

Figure 1 provides the details of an MFLS architecture with $M = 3$. For commonly-used MPEG-4 look-up tables, Section 5 will show how the choice of $M = 3$ results in total MFLS table storage requirements that are significantly less than the requirements for the SFLS architecture and moderately more than the MB architecture. A larger value of M would further reduce the total MFLS storage requirements, but it would also increase the hardware complexity and the size of the group control register.

The proposed instruction format and the group control register contents are depicted at the top of Figure 1. For each group m as defined by the programmer, the group control register contains the offset, the right-shift amount $W - S_m$, and the minimum LNZ count ℓ_m^{min}. The exception is group 0, where the offset value and the minimum LNZ count are often zero, hence these fields are omitted from the register (the right-shift amount is still required, however). Standard control register access instructions can be used to set or obtain the contents of the group control register. The combinational logic in the remainder of Figure 1 includes a block to determine the LNZ count for the input codeword in register `rsrc1`, and then various blocks to use the LNZ count in order to select the group, the offset, and the shift amount. These selection blocks use multiplexers with inputs from the group control register.

The logic in Figure 1 uses the bit field from processor register `rsrc1` as input to the LNZ count block and the left shifter. The most significant bit from the output of the left shifter is the '1' bit that follows the leading zeros, hence it is ignored. The next 12 most-significant bits from the left shifter are concatenated with the output of the 4-bit subtract unit that computes $\ell - \ell_m^{min}$. The combined 16 bits are used as the input to the 16-bit right shifter that produces the codeword offset within the group. This codeword offset is then added to the group offset in order to obtain the required index in the table. The architecture in Figure 1 assumes a maximum codeword length of 20 bits. Because of the actual maximum lengths of 17 bits for MPEG-2 and 13 bits for MPEG-4, there is additional capacity for any new codes in the future. Furthermore, the maximum suffix length, W, for MPEG-2 and MPEG-4 is 12 bits, and this property is also exploited in the architecture.

The size of the group control register in an MFLS implementation depends on the number of bits needed for the grouping information. A possible allocation is 12 offset bits, 4 right-shift bits, and 4 bits for the minimum LNZ value for each group. The index

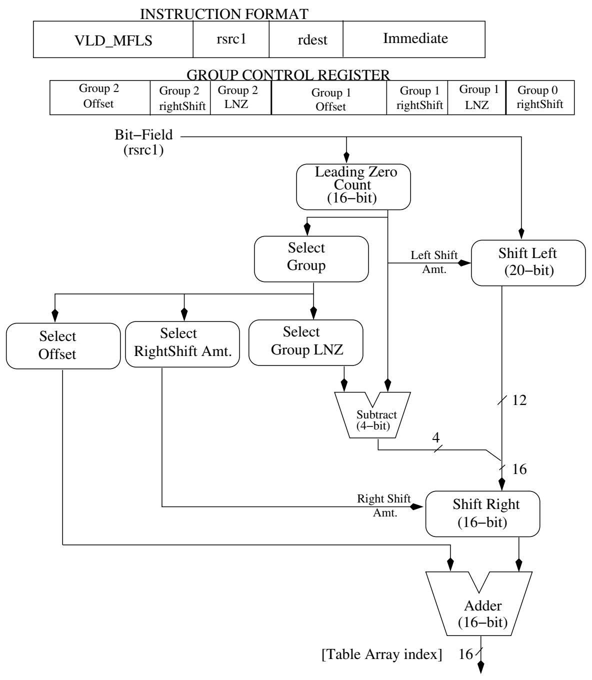

Fig. 1. 3-Fixed-Length Suffix Architecture

range supported with the above bit allocation is sufficient for many common multimedia formats. With the above bit allocation, the minimum required size of one group control register for a maximum of M groups is $20 \times (M-1)+4$ bits.

For generality, the proposed `vldecode` instruction generates only the table index in the destination register `rdest`. This index value must then be added to the base address of the appropriate table in memory in order to retrieve the decoded information. Automatically performing the final calculation using a base address register could affect the cycle time.

The method provided in Figure 1 can be extended for $M = 4$ or higher with ease. For most cases, the VLC table size is a non-increasing function of M. Increasing M will, however, add to the hardware complexity of the selection blocks and it will also increase the size of the group control register. The memory-based architecture introduced in our previous work [1] can be thought of as an MFLS implementation with all possible leading number of zeros or $M = 16$. Similarly, the single fixed length suffix (SFLS) architecture can be thought of as an MFLS implementation with $M = 1$ and without any group control register.

3. FPGA AND ASIC SYNTHESIS RESULTS

The MFLS architecture with $M = 3$ was synthesized for FPGA and ASIC implementation, along with the MB and SFLS architectures from our earlier work [1]. Section 2.1 explained the differences between the three approaches, specifically how the MFLS architecture is a compromise between the previous MB and SFLS architectures. The designs were implemented in VHDL and used vendor-supplied shifter and adder components to maximize performance. The 16-bit leading-zero-count block is a hierarchical radix-4 implementation. A 44-bit group control register was used in the MFLS implementation; 4 bits were used for each LNZ and right-shift field, and 12 bits for each offset field.

Table 2. FPGA synthesis results

	MB	SFLS	3FLS
Num. Logic Elems.	200	119	156
Critical Path Delay (ns)	12.96	10.40	12.98
Max. Freq. (MHz)	77	96	77

Table 3. ASIC synthesis results

	MB	SFLS	3FLS
Total Area (μm^2)	27170	3496	6984
Power (mW)	19.00	4.78	9.37
Max. Freq. (MHz)	117	246	114

The target FPGA is an Altera Stratix EP1S40 chip with synthesis performed by the Altera Quartus II software. Table 2 summarizes the synthesis results for all three architectures. The MB results exclude the memory, which is implemented in a predefined on-chip RAM block and not in logic elements. The 3FLS architecture has a logic complexity that is between the other two architectures, and it has a frequency of operation that is the same as the MB architecture.

For ASIC synthesis, we used Synopsys tools with 0.18 μm TSMC generic libraries. The design was optimized by using the DesignWare library components for adders with fast carry look-ahead and for shifters. The operating voltage of the design is 1.6 volts. The results are summarized in Table 3. The area and power results for the 3FLS architecture are better than the results for the MB architecture (the MB results include the area and power for a 320-bit latch-based asynchronous RAM). The operating speeds are comparable, however. The 3FLS architecture would therefore be suited for embedded systems with area/power constraints. For higher performance, even more specialized or advanced library cells could be employed for integration with processors that would be used for decoding applications. Further optimizations could include pipelining the combinational logic in Figure 1 if its propagation delay exceeds the critical path through the execute (EX) stage of the processor with which the logic is integrated.

4. PERFORMANCE RESULTS FROM SIMULATION

Performance results were obtained by simulating the execution of software for MPEG-4 variable-length decoding on a model of a general-purpose pipelined processor with and without the decoding support of the MFLS architecture. The simulations were performed with tools provided by Tensilica, Inc., for their configurable Xtensa processor core [7]. Using the Tensilica Instruction Extension (TIE) language, the proposed `vldecode` instruction from Section 2 was added to the instruction set, along with other instructions to set or obtain the contents of the group control register with $M = 3$. The simulations were initially configured for a processor with 32-kbyte, 4-way-associative data and instruction caches using 32-byte blocks. The processor pipeline and the memory hierarchy were modeled, with a data cache miss penalty of 14 cycles and an instruction cache miss penalty of 13 cycles.

The simulated MPEG-4 code is the MoMuSys MPEG-4 reference software in the C language [5]. The variable-length decoding portion of the reference software was modified to use our new instruction; the Tensilica tools allow the use of special C functions to represent the operation of a new instruction, from which the compiler generates executable machine code with the new instruction. The simulated performance results are reported only for computations re-

Table 4. Group information for VLC tables in MPEG-4

	LNZ_{min}	Right Shift	Offset
Group 0	-	5	-
Group 1	3	7	96
Group 2	6	4	480

Table 5. Performance results for MPEG-4 VLD

Video	Orig. (Mcycles)	3FLS (Mcycles)	Speedup
Coastguard	431	201	2.14x
Waterfall	180	84	2.14x
Mobile	771	361	2.14x

lated to the frequently-used MPEG-4 intra- and inter-transform coefficient tables B-16 and B-17, which were modified appropriately for the new `vldecode` instruction. Each modified table for 3FLS has 528 entries as compared to 205 entries for the original tables. The settings of the group control register with $M = 3$ for both tables are provided in Table 4.

The results from decoding 100 frames (352×288, 30 fps) in three commonly-used MPEG-4 test video sequences are provided in Table 5. Use of the MFLS architecture in the processor that executes the modified software led to a doubling of performance over the original code on the base processor. This result was achieved with just one group control register.

For further insight, additional simulations were conducted with different cache configurations. Table 6 summarizes results for the Coastguard video sequence. With a smaller cache size, the performance of the modified software using the MFLS architecture is less sensitive to the cache associativity than the original software on the base processor.

5. MEMORY EFFICIENCY ANALYSIS

We also analyzed the codeword table sizes for the reference MPEG-4 decoder software that was used in the simulation experiments in order to characterize the memory required for VLC tables using different methods. The results are provided in Table 7. The first column lists the tables used in MPEG-4 video coding. Table B-12 is the motion vector table. Tables B-16 and B-17 are the intra- and inter-transform coefficient tables, respectively. These three tables are the most frequently-accessed tables in the decoder software. The second column lists the number of codewords in the original tables. The last three columns indicate the number of entries in the modified tables for the three VLD architectures being compared in this section. The total memory requirements for all tables under each method are also provided (assuming 4 bytes per entry). The 3FLS architecture requires less than half of the storage of the SFLS architecture.

6. CONCLUSION

The multiple fixed-length suffix (MFLS) architecture proposed in this paper provides a generic programmable solution for accelerating variable-length decoding in general-purpose processors for various multimedia formats including MPEG-1/2/4, H.261/3/4, and JPEG. The hardware extension for MFLS consists of a modest amount of combinational logic and a group control register. The parameter M for MFLS defines the maximum number of codeword groups that a hardware implementation will support; larger values of M imply a larger group control register and increased hardware complexity.

Table 6. Performance results for different cache configurations

Cache Size	Orig. (Mcycles)	3FLS (Mcycles)
32KB-4way	431	201
32KB-direct	458	208
16KB-4way	458	221
16KB-direct	490	221

Table 7. MPEG-4 VLC table sizes

Table	Orig.	3FLS	SFLS	MB
B-12	65	134	352	87
B-16	205	528	1152	350
B-17	205	528	1152	350
Total entries	475	1190	2656	787
Total memory	**1.90KB**	**4.76KB**	**10.62KB**	**3.15KB**

Compared to our earlier architecture proposals, the MFLS architecture results in lower memory requirements than the single fixed-length suffix (SFLS) architecture and reduced hardware complexity with respect to the memory-based (MB) architecture, as reflected in synthesis results for FPGA and ASIC implementation. Simulation results using MPEG-4 reference software demonstrate that the MFLS architecture with $M = 3$ can double performance in the computations related to the intra- and inter-transform coefficient tables.

7. ACKNOWLEDGMENTS

The authors are grateful for the support provided by the Natural Sciences and Engineering Research Council of Canada, the Canadian Microelectronics Corporation, Communications and Information Technology Ontario, Sun Microsystems, Inc., Tensilica, Inc., and Queen's University.

8. REFERENCES

[1] S. Sudharsanan and M. Sinnathamby, "Support for variable length decode on embedded processors," in *Proc. of the Workshop on Media and Signal Processors for Embedded Systems and SoCs*, 2004, pp. 33–40.

[2] S. M. Lei and M. T. Sun, "An entropy coding system for digital HDTV applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1, no. 1, pp. 147–155, March 1991.

[3] S. Sriram and C.-Y. Hung, "MPEG-2 video decoding on the TMS320C6X DSP architecture," in *Proc. IEEE Asilomar Conf. on Signals, Systems and Computers*, 1998, pp. 1735–1739.

[4] M. Sima, S. D. Cotofana, S. Vassiliadis, J. T. J. van Eijndhoven, and K. A. Vissers, "MPEG-compliant entropy decoding on FPGA-augmented TriMedia-CPU64," in *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, 2002, pp. 261–270.

[5] "Information technology - coding of audio-visual objects: Visual ISO/IEC 14496-2," March 1998, N2202, Committee draft.

[6] C. Fogg, "Survey of software and hardware VLC architectures," in *Proceedings of SPIE on Image and Video Compression*, 1994, vol. 2186, pp. 29–37.

[7] R. E. Gonzalez, "Xtensa: a configurable and extensible processor," *IEEE Micro*, vol. 20, no. 2, pp. 60–70, March/April 2000.