High-Throughput Data Compressor Designs Using Content Addressable Memory*

Ren-Yang Yang and Chen-Yi Lee

Dept. of Electronics Engineering, National Chiao Tung University

1001, University Road, Hsinchu 300, Taiwan, ROC

Tel: 886-35-712121 ext.3286; Fax: 886-35-724361; Email: cylee@cc.nctu.edu.tw

Abstract

This paper presents a novel VLSI architecture for high-speed data compressor designs which implement the well-known LZ77 algorithm. The architecture mainly consists of three units, namely content addressable memory, match logic, and output stage. The content address memory generates a set of hit signals which identify those positions whose symbols in a specified window are the same as input symbol. These hit signals are then passed to the match logic which determines one matched stream and its match length and location in the window to form the kernel of compressed data. These two items are then passed to the output stage for packetization before sent out. By trading off hardware complexity and compression ratio, 2KB window size and adjustable maximum match length are considered in our proto-type VLSI chip. Simulation results show that, based on a 0.8 μ m CMOS process technology, clock speed up to 50MHz can be achieved. This implies that the developing data compressor chip can handle many real-life applications such as in video coding and high-speed data storage systems.

I Introduction

Since Lempel and Ziv [1] published the well-known LZ77 lossless data compression algorithm in 1977, many different versions have been developed. A good survey of these compression algorithms can be found in [2]. In principle, compression ratio, instead of algorithm complexity, is the major issue of these algorithms in the development phase. However when real-time requirements are demanded, trade-off between algorithm complexity and achievable compression ratio has to be taken into account seriously. Fortunately, state-of-the-art VLSI technology offers great advantages in system integration to overcome such complexity. Some research reports on such hardware implementation can be found in the literature [3,4,5,6]. Among these hardware solutions, different realization approaches have been exploited such as content addressable memory or CAM approach [3,5], array processor approach [4], and RISC approach [6].

In this paper, we present a VLSI architecture for single chip implementation of a modified LZ77 algorithm. The architecture is achieved by exploiting partitioning and pipelining techniques based on the CAM approach. In Section II, we first briefly describe the modified LZ77 algorithm and then put some efforts on selection of window size and maximum match length. The ASIC architecture will be discussed in detail in Section III, where hierarchical design strategy as well as partitioning and pipelining techniques are exploited to improve clock speed. In Section IV, we present some evaluation data on the proposed architecture and provide some comparisons with those hardware solutions mentioned earlier.

II The Modified LZ77 Compression Algorithm

The LZ77 algorithm can be briefly illustrated as in Fig.1(a) which contains a window to buffer a certain amount of continuous symbols. For each input symbol, the hit signal will be propagated to next symbol for the purpose of stream matching. Then a codeword consisting of match length and start position will be sent out. However, in some cases when

matched stream	(0, match length, start address) / (1, symbol)
"Th"	(0,2,0)
"a"	(1,'a')
"t"	(1,'t')
" is a "	(0,6,0)

(b) code word

Fig.1 Illustration of the LZ77 algorithm

* Work supported by the National Science Council of Taiwan, ROC, under Grant NSC-83-0404-E-009-031.

the matched stream is less than 2 symbols, we do not gain compression ratio. Thus in our modified version of the LZ77 algorithm, we define that input stream will be replaced by codeword only when its match length is more than 1. Otherwise source symbol together with an identification (ID) will be sent. This is shown in Fig.1(b).

Fig.2 test file size is more than 200KB

Since the window size needs a lot of memory space in hardware realization, one has to trade off algorithm complexity and achievable compression ratio. To find an optimal solution, we define two parameters--namely window size S_w and maximum match length L_m . Fig.2 shows the simulation results for one text test files and one image test file with different (S_w, L_m) pair. By adjusting these two parameters and simulating several test files as well as trading off hardware complexity and application requirements, we defined the codeword length is 16 bits and (S_w, L_m) is (2KB, 32).

III High-Throughput VLSI Architecture

The functional block diagram is shown in Fig.3. It mainly consists of three blocks. The content addressable memory (CAM) acts as a dynamic moving window to partially store previous symbols, and in the mean time, to output hit signals indicating those locations whose symbols are identical to current input symbol. These hits signals are then passed to the match logic (ML) to produce three output information such as match length, physical position, and synchronization. These information items together with the current input symbol are then processed at the output stage (OS) to produce compressed data which will be sent out. In the following, we first discuss the details of each block and then present some strategy to overcome critical path so that clock speed can be enhanced.

Fig.3 Block diagram of the LZ77 encoder

Content Addressable Memory Design

The basic structure of this unit is given in Fig.4. Since only the hit signal is needed, each CAM bit-cell can be realized on 9 transistors [7]. However, the address generation for the CAM should be taken into account to optimize area and timing. For encoding purpose, input symbols are cyclically stored and then compared. This implies that ring counter can be exploited. For decoding purpose, start position should be first determined from received compressed data and cannot be produced efficiently by the ring counter. Thus random access address generator is proposed here.

Fig.4 Structure of Content Addressable Memory

Fig.5 Structure of Match Logic

Fig.6 A unit match cell

Matching Logic Design

The ML can further be partitioned into 3 sub-units as shown in Fig.5.

Match Cells These cells are designed to (1) detect the hit signals between input sequences and windowed symbols and to (2) conditionally propagate hit signal for stream matching. A unit match cell is shown in Fig.6. It consists of 2 delay elements, one multiplexer, and one AND gate. The top delay element can be preset in the initial phase to indicate that all windowed symbols are candidates. Then this delay element continuously reports the match signal from left match cell to indicate the match permission of current match cell. The bottom delay element stores the hit signal from left cell and is needed only when maximum match length is detected or not any match signals are asserted (i.e. no more longer streams are detected). The selection is governed by the mode control signal.

Length Generator This sub-unit is designed to (1) calculate the match length, (2) limit the maximum match length in stream matching, and (3) generate a sync signal to inform the output stage unit to accept the produced data.

The match signals are first ORed to detect if the global match signal exists for current input symbol. This signal is to produce mode and sync signals as well as to control length counter for calculating matched stream length. Also, the accumulated length is compared with a adjustable maximum length to generate the sync signal. Detailed structure of this sub-unit is given in Fig.7.

Fig.7 Structure of length generator

Priority Generator This sub-unit is to generate the physical address of matched stream. Since multiple match signals may appear simultaneously at match cells, and only one match is needed to produce the corresponding address in the CAM, we use priority generation scheme [8] to reach this goal. That is match signal from the low-order address has high priority than those from the high-order addresses.

Output Stage Design

This circuit is to packetize the compressed data according to the sync signal. To improve the compression ratio, the match length and start position will be assembled only when the match length is greater than 2. Otherwise input symbols together with an ID code will be assembled. In our design, start address is to be sent out and can be calculated by subtracting length from physical address.

Strategy to Improve Clock Speed

We first locate the critical path of the architecture design and then use partitioning and pipelining strategies to improve speed. The critical path can be identified from CAM, ML, and to OS. Since the window size of 2KB is selected to store symbols, it is necessary to partition the memory into a 64 by 32 structure as shown in Fig.8. Here each row contains 32 symbols whose output match signals are then ORed to generate a row match signal. These 64 row match signals are again ORed to generate the final match signal exploited by other units. In the meantime, these 64 row match signals are sent to a row priority generator and encoder to produce the row address of the matched position. Also the output from the row priority generator is sent to a tri-state buffer which selects a matched row whose 32 match signals are sent to the column priority generator to generate the column address as

shown in Fig.8(b). After this row-column partitioning, we find that the speed can be improved. In addition, the layout becomes more feasible in physical design.

Fig.8 Priority generator for the row and column address
Note that in (a) the CAM is now reconfigured to 64 by 32 and pipeline section can be included to improve speed (b) Shows that a set of tri-state buffers are added for row selection

We then consider the iterative bound in order to insert pipeline register. As above-mentioned, the longest path is from CAM, ML, to OS. However, the recursive loop is only detected from CAM to ML which continuously determines the match signals. Since the detection of these matched streams are sequential and cannot be partitioned separately, pipeline insertion does not improve throughput rate. Also we have partitioned the CAM into a 64 by 32 memory structure so that timing constraints can be encountered. To enhance the clock speed, we can insert pipeline section at both row and column priority generation blocks as shown in Fig.8. Here pipeline registers are inserted at the output of row priority generator. In addition, a set of tri-state buffers are added at each row to select the matched row for column address generation. Then these selected match signals are pipelined at the input of the column priority generator. According to this arrangement, only 64+32 pipeline registers are needed. If we distribute pipeline registers in another way, say inserted at the input of row priority generator, then 64×32+64 pipeline registers are needed. Although the latter arrangement does outperform the former in speed by 5%, its hardware overhead of pipeline registers are 22 times more than that of the former. By trading off area cost and speed, the former arrangement is used in our design.

Decoding Process

Although the above mentioned architecture is derived for encoding purpose, it can also be used for decoding. When

compressed data are to be decoded, it first checks the ID code and then performs decompression. When the received ID is "1", its followed data are source symbols and can be sent out and stored in the CAM simultaneously. On the other hand, if the ID is "0", source symbols can be obtained from the CAM by decomposing the codeword into start address and match length. Note that symbols have to be read from the CAM, therefore sense amplifier is needed to improve access time.

IV Evaluation and Discussions

Here we give some comparison data in terms of speed and compression capability with those available hardware solutions mentioned in the Introduction. In [4], a systolic array is proposed to obtain speed and throughput. However, some idle cycles can be allocated in processor elements during coding, leading to much hardware overhead. Also the achievable compression ratio for real-life applications is questionable because the limit of the number of processor elements. In [6], a RISC architecture is proposed. However sample rate can only be up to a few hundreds of Kbytes per second, which cannot meet high-speed requirements. In [3,5], the CAM approach is exploited. However, the compression ratio in [3] is very low due to the limit of CAM size. The architecture proposed in [5] needs one extra cycle to load input stream from buffer into CAM, when the compression ratio is not mentioned.

In terms of speed, our proposed architecture can reach 50MHz, which also implies that data rate up to 50MSamples/s can be handled. This is sufficiently high for current applications with large volumes of data. In other words, our solution can achieve the highest throughput among the mentioned solutions. This is the first feature of our design. The achievable compression ratio lies in the range of [1.5 .. 3.0], which is competitive with the others. However, it should be mentioned here that our architecture can be reconfigured by adjustable maximum match length off chip. This is the second feature of our design.

The complete design is shown in Fig.9. The core area is 6.4mm×6.5mm, based on a 0.8μm CMOS double metal technology.

V Conclusion

In this paper, we have presented an ASIC architecture for high-throughput data compressor design using the content addressable memory. By trading off algorithm complexity and compression ratio, an optimum set for window size and match length can be determined. Then by means of design hierarchy, we obtain an efficient VLSI architecture and then achieve high-speed by exploiting partitioning and pipelining. A proto-type chip for high-speed data storage applications

has been completed based on the approach presented in this paper.

Fig.9 Chip plot of the LZ77 data compressor

References

- [1] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. Inform. Theory*, Vol. IT-23, pp. 337-343, 1977.
- [2] J.A. Storer, "Data Compression: Methods and Theory", Rockville, MD: *Computer Science Press*, 1988.
- [3] S.R. Jones, "100 Mbit/s Adaptive Data Compressor Design Using Selectively Shiftable Content Addressable Memory", *IEE Proceedings-G*, Vol. 139, No. 4, pp. 498-502, August 1992.
- [4] N. Ranganathan and S. Henriques, "High-Speed VLSI Designs for Lempel-Ziv-Based Data Compression", *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 40, No. 2, pp. 96-106, February, 1993.
- [5] W.Y. Wei, R. Tarver, J.S. Kim, and Kevin Ng, "A Single Chip Lempel-Ziv Data Compressor", in *Proc. of ISCAS'93*, pp. 1953-1955, Chicago, May, 1993.
- [6] J. Chang, H.J. Jih, and J.W. Liu, "A Lossless Data Compression Processor", in *Proc. of 4th VLSI/CAD Workshop*, pp. 134-137, Nan-Tou, Taiwan, August, 1993.
- [7] Neil H.E. Weste and Kamran Eshraghian, "Principles of CMOS VLSI Design: A Systems Perspective", *Addison Wesley*, 2nd Edition, pp.589-590, 1993.
- [8] C.Y. Lee, S.C. Juan, and W.W. Yang, "An Area-Efficient Maximum/Minimum Detection Circuit for Digital and Video Signal Processing", *Proc. of ISCAS'93*, pp. 223-226, Chicago, Illinois, May 3-6, 1993.