Highly-Parallel Table-Lookup-Coding with Scalable Architecture using Flexible Multi-Ported Content Addressable Memory

Takeshi Kumaki, Yutaka Kono, Masakatsu Ishizaki, Tetsushi Koide and Hans Jürgen Mattausch
Research Center for Nanodevices and Systems, Hiroshima University, 1-4-2, Kagamiyama,
Higashi-hiroshima, Hiroshima, 739-8527, Japan
Email: kumaki@sxsys.hiroshima-u.ac.jp

1. Introduction

Multimedia applications, requiring the capabilities of codec processing, image processing or image recognition, have spread to the end-user environment. In particular, the modern communication-network infrastructure accelerates the development of on-demand systems, digital broadcasting, mobile machinery and so on. Furthermore, security applications, for example ciphers, are developing more and more. The required basic processing operations for applications can be classified into two types, the arithmetic and the coding operations. In case of the JPEG algorithm for picture compression, the arithmetic operations mainly consist of DCT and quantization. As coding operation the JPEG algorithm uses Huffman coding. Generally, Huffman coding needs to prepare a code word table that contains the mapping information between the input symbols and the code words for encoding. Thus, Huffman coding is difficult to parallelize, because it involves essentially sequential operations requiring a large hardware amount. Presently, conventional architectures for consumer products mainly improve the arithmetic operations based on e.g. Single Instruction Multiple Data (SIMD) architectures. Therefore, the coding operation is now becoming the bottleneck operation for fast real-time applications working with multimedia and security contents.

For overcoming this coding-operation-related bottleneck, we propose an efficient parallel coding architecture using multi-ported content addressable memory as a novel architecture for high-speed and real-time coding operations. The multi-ported CAM effectively uses the previously reported Flexible Multi-ported Content Addressable Memory (FMCAM) technology [1] and enables the coding of multiple input symbols in parallel, while the hardware amount becomes lower than for conventional architectures.

2. Conventional Coding Architectures

This section discusses Huffman coding, which is a representative example for coding algorithms and is implemented in JPEG and MPEG standards for compressing multimedia contents. Especially, the encoding operation in Huffman coding is known to be difficult to implement with parallel processing [2].

General Huffman encoding has to prepare an optimized code word table according to the distribution features of the contents. Therefore, the conventional architecture, which is implemented in general Huffman encoding hardware, spends many clock cycles for processing the input data. To overcome this drawback, the so-called static Huffman encoding, which uses a fixed standardized code word table, has often been implemented by the conventional coding architectures. Most real applications use static Huffman encoding and thus the standardized coding tables have been embedded in various conventional hardware architectures, which are based on SRAMs, hard-wired logic or CAMs.

Generally, for improving the processing speed of arithmetic operations, a parallel processing approach is often used. In case of the JPEG algorithm, SIMD architectures, which consist of several Processing Elements (PEs), can handle a large number of pixel data in parallel. However, all above conventional architectures for Huffman coding require sequential operations. Therefore, these encoding architectures do not match with the SIMD architecture for executing arithmetic operations. If the conventional architectures for the encoding operation are forced to implement parallel processing, the hardware amount would increase drastically due to multiple tables and multiple comparison hardware.

3. Flexible Multi-ported Content Addressable Memory with Flexible Parallel Search Hardware

For improving the efficiency of the multi-ported CAM structure, we have been proposing a Flexible Multi-ported Content Addressable Memory (FMCAM) architecture [1], which is a novel functional memory for parallel search.

3.1. Original FMCAM Architecture

The originally proposed FMCAM architecture has p input/output ports and a common storage capacity, which is called contents-table, of 2^a reference words with d-bit word-length. As shown in Fig. 1, each port is able to receive comparison data of d-bit length and mask data of d-bit length. The corresponding output consists of a match signal of 1-bit length and a match address of a-bit length. Furthermore, asynchronous processing is allowed at each port so that the search operation can start as soon as search-request data is received, without waiting for synchronization with the other ports. Due to the parallel operation of all ports, the processing speed is p times as fast as for a conventional single-port CAM. Besides that, the FMCAM architecture applies two additional concepts for reducing the hardware resources. The first concept is a Bit-Parallel and Block-Parallel (BPBP) search instead of the a Bit-Parallel and Word-Parallel (BPWP) search, which is often applied in a conventional fully-parallel CAM. The second concept is a categorization of the stored reference words [3]. As a result, the increase of the comparator number due to the multiple ports is limited.

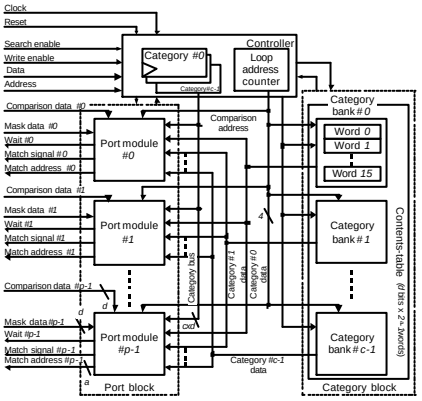

Fig. 1 Block diagram of the adapted FMCAM

3.2. Adaption of the FMCAM architecture to coding applications

In [1], the FMCAM architecture has been shown to result in effective devices, indicated by their comparatively small product of implementation area and processing time (area-time product). Therefore, if this architecture is exploited to process multiple searches, it is capable to accelerate any application, and in particular parallel coding applications, such as data compression and encryption.

In the following the novel ideas for adapting the FMCAM architecture to encoding and encryption applications are exploited.

3.2.1. Multiple/Single search mode

The original FMCAM architecture exploits the BPBP search instead of the BPWP search. Thus, it lowers the number of comparators in spite of having several ports. However, the original FMCAM architecture requires more than one clock cycles for completing a comparison task. Applications such as Huffman encoding translate each input symbol into a converted symbol. Since input and converted symbols have a one-to-one relation, the clock cycles after finding a matching symbol are wasted.

For overcoming for this problem, the adapted FMCAM can select between two search modes, namely a multiple search mode and a single search mode. The multiple search mode applies if multiple matches to the input data may exist and is equivalent to the original FMCAM comparison process. The single search mode realizes a faster comparison process for well defined single match searches. The adapted FMCAM, which is operated in the single search mode, can stop the comparison process immediately as soon as the first matching symbol is detected.

3.2.2. Counting value setting mode

For decreasing the number of comparators, the original FMCAM exploits a categorization concept [3]. The stored reference words are classified into plural categories according to a pre-defined rule. This process, which is executed during initialization and contents-table input, enables to chose a memory structure with single-port banks. To restrict each search request to one bank, the adapted FMCAM achieves multi-port capability by independent and parallel operation of these banks as in a bank-based multi-port memory. Consequently, the comparators can be located separately from the contents-table of the memory banks and a reduction of the number of comparators can be realized. However, if the categorized data does not fill the storage capacity provided for a category, a complete search through the storage space would waste a number of clock cycles. For overcoming this drawback, the adapted FMCAM allows to set the clock cycle number for the comparison process according to the application needs and can thus eliminate wasted clock cycles.

3.3. Structure of the adapted FMCAM

The block diagram of the adapted FMCAM, shown in Fig. 1, is composed of three main parts, a port block, a category block and a controller. These three parts can be operated independently of each other.

3.3.1. Port block

The block diagram of an input/output port module in the port block is shown in Fig. 2. It is able to receive comparison data and mask data, both of *d*-bit length. The output data from the port module port consists of a match signal of 1-bit length and the corresponding match address of *a*-bit length. All ports support asynchronous processing, which means that the search operation can start as soon as search-request data is received, without waiting for synchronization with other port modules. Due to the parallel operation of all ports, the processing speed becomes faster in proportion with the port number *p*.

Each port module is composed of a *d*-bit search-comparator, *c* category-comparators for *d*-bit words, a category decoder, a demultiplexer, several registers and some combinational logic. When a port module receives an input data, the category-comparators compare this input data with the present category structure data of the FMCAM to determine the category which should be searched. The category decoder translates the category-comparator results into control signals for a multiplexer, which connects the data from the searched category to the search-comparator. Thus, the search-comparator is enabled to compare the input data with the relevant reference data of the FMCAM. At the same time, the first comparison address generated by the loop-address-counter, which is described in section 3.3.3, is memorized in a register, and becomes the starting address for the comparison process to the reference data of the relevant category. This comparison process continues until the address value from the loop-address-counter is again equal to the starting value stored in the register. Since the loop-address-counter continuously keeps counting to the next comparison address in each clock cycle and broadcasts this current address to all port modules, each port module can memorize its unique starting address independently. If the search-comparator outputs a match signal, the match address is generated by combining comparison address and category address.

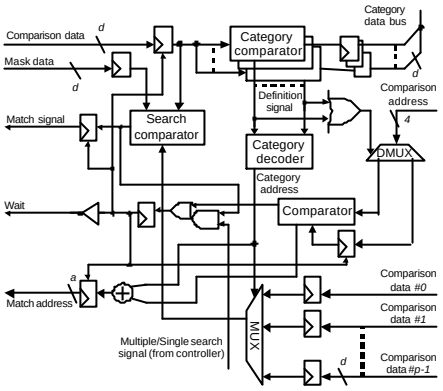

Fig. 2 Block diagram of a port module.

3.3.2. Category block

The category block consists of several category banks. Each category bank memorizes the reference data as stored words in a conventional single-port-memory bank. The reference data are broadcasted from the category banks to all port modules during FMCAM operation. As soon as each port has decided on the category of the required reference data, the broadcasted reference data from this specific category are loaded into the port module.

A conventional CAM carries out a parallel search within its database of reference words. The normal approach for realization of this CAM functionality integrates the necessary additional hardware resources such as the comparators into the memory field. This approach results in 2 important problems. On one hand, it becomes impossible to use conventional memory macros for the CAM construction, which restricts wide application of the CAM function in integrated systems. On the other hand, the introduction of multiple ports becomes difficult because the amount of additional hardware increases in propor-

tion to not only the number of reference words but also proportionally to the number of ports. Although various attempts to overcome above restrictions have been made, finding an optimum tradeoff between processing speed and hardware resources turns out to be difficult.

For overcoming above drawback, the FMCAM applies the separation concept of the CAM-specific hardware, in particular the comparators, from the memory field. The chosen realization of the comparison function within the multiple ports leads however to the necessity of data transfers form the memory part to the comparators in the port modules.

3.3.3. Controller

The controller is based on two main modules; the category-registers and the loop-address-counter. The category-registers are used to memorize the freely scalable category structure. These memorized patterns are broadcasted to each port module. The loop-address-counter implements the concept of loop counting, which enables the asynchronous parallel operation of all ports. The address-space size of the address counter is just the same as the address-space size of a category bank in the category module. The address counter sequentially generates the addresses of the stored reference words in each category bank and continues counting independent of whether search-request data is arriving at the input of a port module or not. The conventional counter solution of 1-port CAMs with BPBP construction, on the other hand, starts address generation from the first address only after receiving input data. Consequently, the conventional concept, when extended to the multi-port case, would require a synchronization time for the ports, during which arriving search requests at a specific port have to wait until presently on going searches at other ports are finished. Since the loop-address-counter automatically resumes counting from the first address after the last address has been reached, each port module is able to memorize its individual starting address in a register and can execute its search process as soon as search-request data arrives. Thus the waiting time due to the synchronization process can be removed.

4. Proposal of FMCAM-based parallel coding architecture

4.1. Architecture concept

Coding algorithms are often included in image processing and encryption applications, which mostly combine parallel and sequential processing of the data. Fig. 3-(a) shows the example of the JPEG application when implemented in a media processor. Typical media processors consist of a parallel processing block, a sequential processing block, an internal bus and so on. The parallel processing block in the JPEG application is mainly assigned to the task of DCT or data quantization. The sequential processing block is mainly used for Huffman coding. Since the parallel processing block often exploits a SIMD architecture, above arithmetic algorithms are executed effectively. However, an efficient parallel-processing implementation of Huffman coding is difficult to realize [2]. Therefore, the Huffman encoding occupies a large share of about 30% of the processing time in the JPEG algorithm. Since the sequential processing tasks depend on the sequential processing block, data has to be transferred between the parallel and sequential processing blocks across the bus, which results in a higher bus traffic and increases the frequency of bus conflicts. If the parallel processing block is upgraded to enable processing of the Huffman coding in parallel, plural Huffman tables as well as the corresponding number of processing elements have to be provided. As the result, the hardware amount increases drastically in particular due to the multiple tables.

For resolving the above bottleneck, an implementation of the adapted FMCAM near the parallel processing block is very effective, also for reducing the traffic on the bus. The corresponding block diagram is shown in Fig. 3-(b). Since the adapted FMCAM enables to combine the input and output ports of several PEs, which are implemented in the parallel processing block with SIMD architecture, and execute multiple searches in parallel, it allows to overcome the drawbacks of the conventional coding algorithm. The adapted FMCAM is used to map the input symbols to the corresponding code word addresses and to enable parallel Huffman encoding of multiple input symbols in combination with a multi-port RAM. An area-efficient bank-based multi-port RAM architecture, as proposed in [4], is very suitable for implementing the code word table.

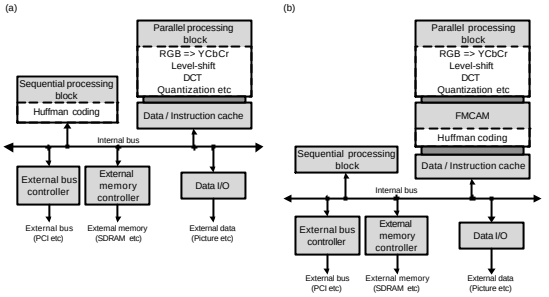

Fig. 3 Concept of a parallel coding architecture using multi-ported content addressable memory: (a) Typical media processor, (b) media processor utilizing the adapted FMCAM.

4.2. FMCAM implementation results as soft-macro

A soft-macro realization of the adapted FMCAM has been developed with verilog-HDL. For verifying the effectiveness of the adapted FMCAM, this soft-macro implementation is evaluated in this section with respect to the maximum operating frequency and area consumption, when synthesized with the Synopsys Design Compiler for a *90*nm CMOS technology. In the evaluation process the address variable *a*, the data width variable *d* and the category variable *c* are chosen as *8*, *32* and *16*, respectively. The port variable *p* is varied from *1* to *16* to determine the effect increasing port number.

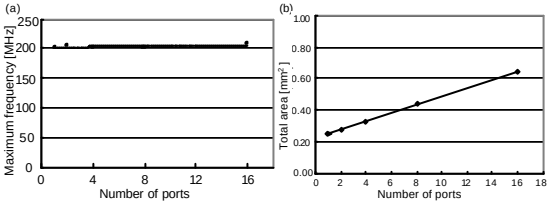

Fig. 4 Implementation results as a function of the number of ports: (a) maximum operating frequency, (b) Area consumption.

Fig. 4-(a) shows the maximum operating frequency of the synthesized FMCAMs, which is almost constant up to the large number of 16 ports. This result indicates that the adapted FMCAM has indeed the expected scalability properties to high parallelism. Due to the independent location of port block, category block and controller as well as their modules construction, the maximum operating frequency is practically not influenced by the number of ports.

Fig. 4-(b) shows the estimated total area of the adapted

FMCAM as a function of the port number. Since all memory cells are constructed from Flip-Flops in the synthesis, the memory banks become larger than in a full-custom design. Generally, the area of multi-ported architectures, such as multi-ported SRAM cells, increases with the square of the number of ports. This fact has constricted the spread of the multi-ported architectures. Since all port modules in the port block of the adapted FMCAM have a common contents-table, the area increases only linearly with just about 9% per port. Thus, FMCAM is effectively available up to large port numbers as a hardware resource for VLSI systems.

4.3. Experimental result for Huffman encoding

In this section, several experimental results of the proposed FMCAM-based architecture for Huffman encoding are reported. Four test pictures are used and shown in Fig. 5. These pictures (a) to (d) are taken of various natural motives and have several different characteristics for contents type and pixel number. Fig. 6 shows the number of clock cycles for the Huffman encoding operation with these pictures. The adapted FMCAM is compared with a conventional DSP and the original FMCAM [1]. The encoding clock cycles with the original FMCAM are almost always smaller than with the conventional DSP even in the case of 1 port. Moreover, the clock-cycle number is reducing as expected according to the increasing number of ports. The adapted FMCAM further reduces the clock-cycle number drastically due to application of the single match mode and the counting value setting mode. The average clock cycle number for a given port number is 43% smaller than for the original FMCAM. Furthermore, in case of picture (d) and for 16 ports, the clock cycle number with the adapted FMCAM is 93% smaller than with the conventional DSP.

Fig. 5 Test pictures.

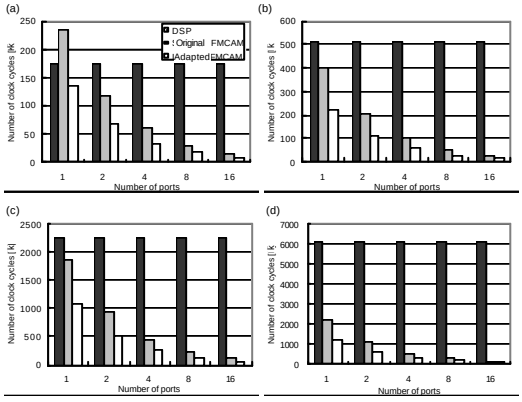

Fig. 6 Comparison of clock cycles for Huffman encoding.

Table 1 shows the processing efficiency expressed in Mega Operations Per Second (MOPS) per mm^2. The value of MOPS/mm^2 for FMCAM is a function of the number of ports, which changes from 1 to 16. The counted operations are the comparison tasks in the Huffman-encoding process. For processing multiple data, several conventional DSPs may be used in parallel. Maximum possible frequencies of the both architectures are almost equal to the each other. The maximum frequency constraint condition of the adapted FMCAM is applied 200MHz. All architecture MOPS become double previous values according to the number of ports. It seems that the MOPS of the conventional DSP are better than that of the FMCAM. However the FMCAM architectures can use the hardware resources with better efficiency. Thus, the area consumption remains smaller. Consequently, the MOPS/mm^2 value of the adapted FMCAM are superior to other architectures. Especially, in case of 16 ports, the ability of the adapted FMCAM is 1.7 times better than the value of the original FMCAM and can achieve up to 3.8 times larger than conventional DSPs.

As a result, the adapted FMCAM can utilize efficiently the comparison operation per area for parallel coding applications. The adapted FMCAM is a suitable solution for the sequential processing block, and can additionally realize to avoid the conflicts on the internal bus between the encoding data and other signals. Thus, the adapted FMCAM architecture is a promising solution for real-time multimedia and ciphers applications.

TABLE I COMPARISON OF PROCESSING EFFICIENCY

Ports	Maximum frequency		Comparison operation [MOPS]			Total area [mm^2]		MOPS / mm^2		
	Original/ Adapted FMCAM	Parallel DSPs	Original FMCAM	Adapted FMCAM	Parallel DSPs	Original/ Adapted FMCAM	Parallel DSPs	Original FMCAM	Adapted FMCAM	Parallel DSPs
1	200	200	10.55	18.58	25.60	0.26	0.21	41	72	122
2	205	200	21.61	38.11	51.20	0.28	0.42	78	137	122
4	201	200	42.26	74.61	102.40	0.33	0.84	127	224	122
8	202	200	85.18	149.14	204.80	0.44	1.68	195	342	122
16	207	200	174.11	302.92	409.60	0.64	3.36	271	472	122

5. Conclusion

In this paper, parallel coding architecture using flexible multi-ported content addressable memory is proposed. This architecture realizes fast coding for multimedia and cipher applications using the adapted FMCAM. For Huffman encoding in the JPEG application, the clock cycle number of the adapted FMCAM is 93% smaller than for a conventional DSP. The efficiency in MOPS/mm^2 of the adapted FMCAM is up to 3.8 times higher than for conventional parallel operated DSPs.

Consequently, the adapted FMCAM is a very effective architecture for ASIC implementation with many real-time coding application possibilities, which can be implemented with small area consumption.

Acknowledgment

Part of this work has been supported by the 21st century COE program, Ministry of Education, Culture, Sports, Science and Technology, Japanese government and a Grant-in-Aid for JSPS Fellows, 175303, 2005.

References

[1] T. Kumaki, K. Iwai and T. Kurokawa, "A flexible multi-port content addressable memory," *IEICE Trans. Inf. & Syst.*, vol. J87-D-I, no. 1, pp. 12-21, Jan., 2004.

[2] T. Miyazaki and I. Kuroda, "Video codec implementation on programmable processors," *IEICE Trans.*, vol. J83-A, no. 12, pp. 1339-1348, Dec., 2000.

[3] K. J. Schultz and P. G. Gulak, "Architectures for large-capacity CAMs," *INTEGRATION the VLSI Journal*, vol. 18, pp. 151-172, June, 1995.

[4] H. Mattausch, K. Kishi and T. Gyoten, "Area-efficient multi-port SRAMs for on-chip data-storage with high random-access bandwidth and large storage capacity," *IEICE Trans. Electron.*, vol. E84-C, no. 3, pp. 410-417, Mar., 2001.