IEICE TRANS. INF. & SYST., VOL.E90–D, NO.1 JANUARY 2007

PAPER

Scalable FPGA/ASIC Implementation Architecture for Parallel Table-Lookup-Coding Using Multi-Ported Content Addressable Memory

Takeshi KUMAKI[†a)], ***Member***, **Yutaka KONO**[†], **Masakatsu ISHIZAKI**[†], ***Nonmembers***, **Tetsushi KOIDE**[†], ***and*** **Hans Jürgen MATTAUSCH**[†], ***Members***

SUMMARY This paper presents a scalable FPGA/ASIC implementation architecture for high-speed parallel table-lookup-coding using multi-ported content addressable memory, aiming at facilitating effective table-lookup-coding solutions. The multi-ported CAM adopts a Flexible Multi-ported Content Addressable Memory (FMCAM) technology, which represents an effective parallel processing architecture and was previously reported in [1]. To achieve a high-speed parallel table-lookup-coding solution, FMCAM is improved by additional schemes for a single search mode and counting value setting mode, so that it permits fast parallel table-lookup-coding operations. Evaluation results for Huffman encoding within the JPEG application show that a synthesized semi-custom ASIC implementation of the proposed architecture can already reduce the required clock-cycle number by 93% in comparison to a conventional DSP. Furthermore, the performance per area unit, measured in MOPS/mm^2, can be improved by a factor of 3.8 in comparison to parallel operated DSPs. Consequently, the proposed architecture is very suitable for FPGA/ASIC implementation, and is a promising solution for small area integrated realization of real-time table-lookup-coding applications.

key words: *multiport, content addressable memory, CAM, parallel processing, SIMD, categorization, bit parallel block parallel, table-lookup-coding, Huffman coding*

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 equipment and so on. Furthermore, security applications, for example ciphers, are developing more and more.

The required basic processing operations for the above applications can be classified into two types, namely the arithmetic and the coding operations. In the case of the JPEG algorithm for picture compression, the arithmetic operations mainly consist of the translation from R, G and B to Y, Cb and Cr representation, Discrete Cosine Transformation (DCT) and quantization. Since each Minimum Coded Unit (MCU), which includes 8×8 pixels for the JPEG algorithm, is processed independently, the above arithmetic operations are suitable for parallel processing architectures. As coding operation the JPEG algorithm uses Huffman coding [2], [3]. Generally, Huffman coding is based on the table-lookup-coding method and 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, each 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 table-lookup-coding architecture for FPGA/ASIC implementation 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], [4]–[6] and enables the table-lookup-coding of multiple input symbols in parallel, while the hardware amount becomes lower than for conventional architectures.

2. Conventional Table-Lookup-Coding Architectures

The table-lookup-coding operation is implemented in many ASICs for consumer products and is also a field of on-going research. For verifying the effect of the proposed architecture in Sect. 4.3, this section discusses Huffman coding, which is a representative example for table-lookup-coding algorithms and is implemented in JPEG and MPEG standards for compressing multimedia contents such as sound and video, as well as in ZIP and LHA standards for compressing data files. Especially, the encoding operation in Huffman coding is known to be difficult to implement with parallel processing [2], [3], [7].

Generally, Huffman encoding, which uses a fixed standardized code word table, has often been implemented by the conventional table-lookup-coding architectures. Most real applications use Huffman encoding and the corresponding standardized coding tables [8], [9] have been embedded

Manuscript received May 22, 2006.
Manuscript revised August 28, 2006.

[†]The authors are with Research Center for Nanodevices and Systems, Hiroshima University, Higashi-hiroshima-shi, 739–8527 Japan.
a) E-mail: kumaki@sxsys.hiroshima-u.ac.jp

DOI: 10.1093/ietisy/e90–d.1.346

in various conventional hardware architectures, which are based on SRAMs, hard-wired logic or CAMs.

The standardized code word table, which is implemented in a single port SRAM, is also often used in DSP-based systems [7], [10], [11]. However, the encoding operation needs a large number of clock cycles for finding the Huffman code word in the code word table, because many read operations and comparison operations have to be executed sequentially during the searching process.

The hard-wired logic solution [12], [13], which uses specific hardware for each application, often constructs the code word table by an internal AND-OR array, and is unable to update or modify the code words in this table during encoding. Therefore, it cannot be considered suitable for changing application-requirement specifications.

A previously reported single-port CAM approach [14], [15] is able to substantially improve the encoding speed. Since the CAM carries out a parallel search within the database of reference words, the processing time for the necessary comparison function is much faster than in conventional software implementations, which are executed by a processor, as well as hardware implementations, which are based on SRAM because these implementations operate sequentially on the database for carrying out the comparison process.

3. Flexible Multi-Ported Content Addressable Memory with 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], [4]–[6], as shown in Fig. 1, which is a novel functional memory for parallel search.

3.1 Original FMCAM Architecture

The originally proposed FMCAM architecture [1] 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 [16], [17] 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 [18], [19]. As a result, the increase of the comparator number due to the multiple ports are limited.

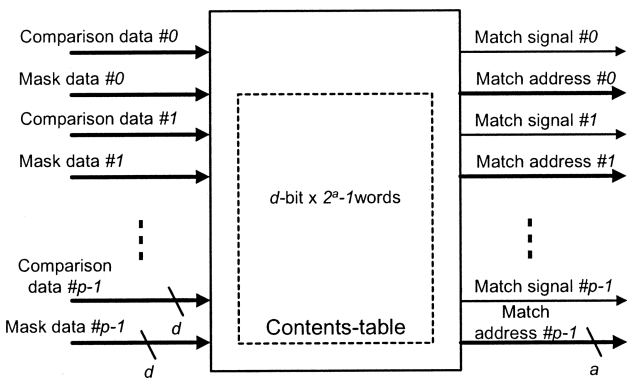

Fig. 1 Input/output configuration overview of the FMCAM architecture.

3.2 Adaptation of the FMCAM Architecture to Table-Lookup-Coding Applications

In Ref. [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 most table-lookup-coding applications, and in particular easily parallelizable table-lookup-coding applications, such as data compression and encryption.

In the following the three novel ideas for adapting the FMCAM architecture to encoding and encryption applications are explained. The corresponding block diagram is shown in Fig. 2.

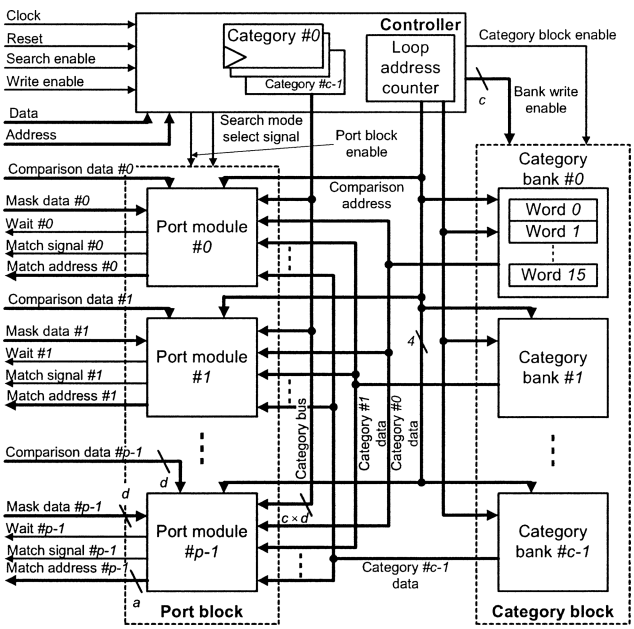

Fig. 2 Detailed block diagram of the adapted FMCAM.

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 cycle 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. Figure 3 shows the difference of waveforms for the search process with both searching modes, which are mainly executed in the port modules of the port block as shown in Figs. 2 and 5. The multiple search mode (Fig. 3 (a)) is equivalent to the original FMCAM comparison process and applied if multiple matches to the input data may exist. Since the number of clock cycles in the comparison process is fixed, as in conventional CAM architectures applying BPBP searches, the original FMCAM normally wastes several clock cycles in comparison operations after finding the last matching data. The single search mode (Fig. 3 (b)) 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 after the first matching symbol is detected. As a result, the search time of the adapted FMCAM becomes shorter than that of the original FMCAM.

3.2.2 Counting Value Setting Mode

For decreasing the number of comparators, the original FMCAM exploits a categorization concept [18], [19]. 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 choose a memory structure with single-port banks. While restricting 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-ported 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. In the illustration of Fig. 4 (a), a category bank has four invalid data. Thus, four clock cycles are unnecessarily executed after receiving comparison data. For overcoming this drawback, the adapted FMCAM can adjust the loop-address-counter condition in the controller, shown in Fig. 2 and described in Sect. 3.3.3, and thus allows to set the clock cycle number for the comparison process according to the application needs. This is illustrated in Fig. 4 (b), where the loop-address-counter continuously keeps counting just twelve clock cycles, so that the adapted FMCAM can eliminate the four wasted clock cycles used in the comparison operations with the invalid data.

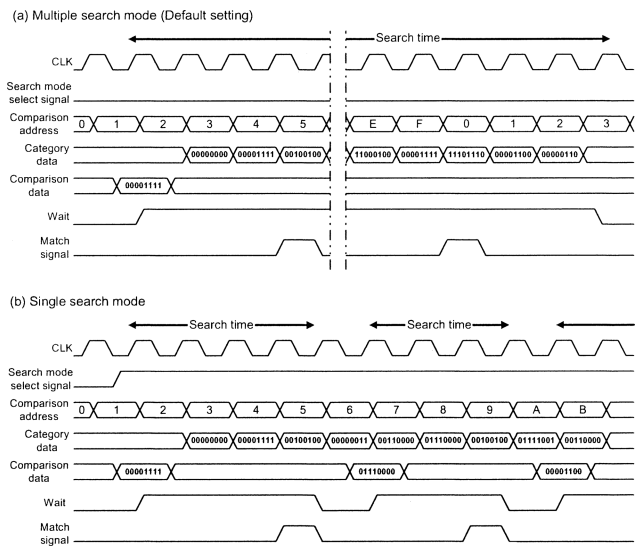

Fig. 3 Two search waveforms of the adapted FMCAM: (a) Multiple search mode, applied if multiple matches may exist, (b) Single search mode, applied if the search problem has a single well defined match.

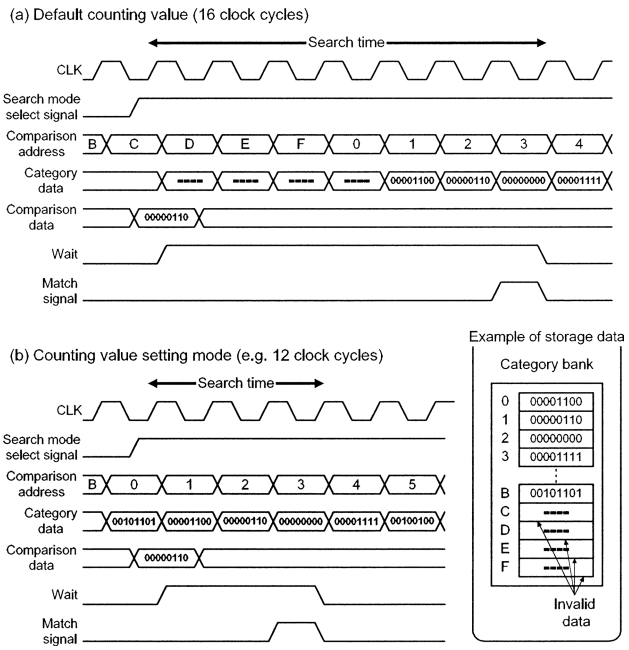

Fig. 4 Two address counting waveforms of the adapted FMCAM: (a) Default counting mode, (b) Counting value setting mode.

3.2.3 Scalability of the Categorization Structure

Conventional CAM architectures [18], [19], which implement a categorization are unable to change the magnitude of the category's capacity. Thus, the capacity of a category may become too small or too large in the case of certain applications, which have a different distribution of the stored contents. For extending the capability to all possible applications, the adapted FMCAM uses a freely scalable categorization structure, which is mainly realized in the category block of Fig. 2. The adapted FMCAM can combine plural category resources, which are needed to store the reference data of a large category into one new category and operate the comparison process accordingly. While the number of comparison clock cycles will increase in the case of combining some smaller categories, it is still possible to execute a comparatively fast search if the user can select the single search mode. Furthermore, the definition of the category pattern and the position of category bits in the complete entry can be modified flexibly by use of the adapted FMCAM.

3.3 Structure of the Adapted FMCAM

The block diagram of the adapted FMCAM, shown in Fig. 2, 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. A more detailed description of their hardware structures is given in the following sections.

3.3.1 Port Block

The port block is mainly constructed from p port modules corresponding to the designed port number p of the FMCAM. The block diagram of an input/output port module in the port block is shown in Fig. 5. It is able to receive comparison data and mask data, both of d-bit length. The output data from the port module 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.

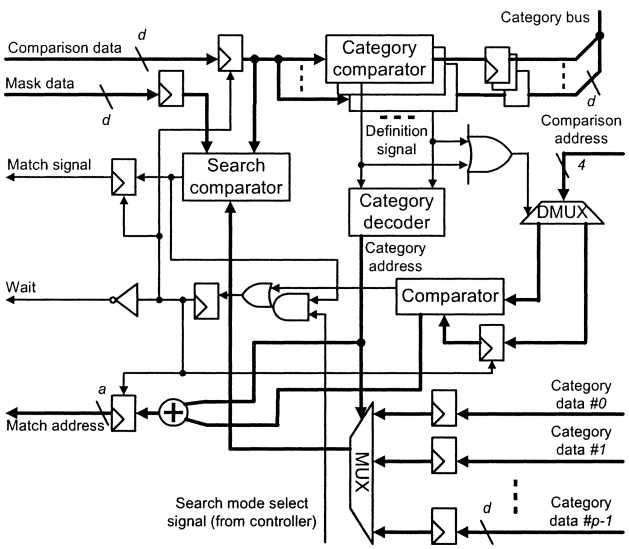

Fig. 5 Block diagram of a port module of the adapted FMCAM.

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 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.

Normally, the adapted FMCAM is operated in the multiple search mode and the search-comparator enables the output of several match signals. In this case, the match addresses are generated by combining the matching comparison address and the category address. On the other hand, if a search mode select signal, which is sent from the controller, rises to high condition, the adapted FMCAM changes to single search mode operation. Then, as soon as the first matching symbol is detected by the search-comparator, the port module is reset and made ready to receive a new comparison data.

3.3.2 Category Block

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 proportion 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 [20], [21], finding an optimum tradeoff between processing speed and hardware resources turn 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. On the other hand, big advantages are obtained, namely that conventional memory macros can be used for data storage in FPGA/ASIC implementations and that the number of necessary comparators can be substantially reduced. Furthermore, the design time for a VLSI implementation of the FMCAM becomes shorter than that of conventional CAM designs, because semi-custom design methods can be applied without the penalty of a largely increased chip area.

The category block consists of several category banks. Each category bank receives a bank write enable signal and 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.

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 category-structure patterns are broadcasted to each port module. The loop-address-counter implements the concept of circular counting, which enables the asynchronous parallel comparison 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 an Adapted FMCAM-Based High-Speed Parallel Table-Lookup-Coding Architecture

Coding algorithms are often included in image processing and encryption applications, which mostly combine parallel and sequential processing of the data. Typical media processors consist of a parallel processing block and a sequential processing block [22]. In the example of the JPEG application, the parallel processing block is mainly assigned to the tasks of Discrete Cosine Transformation (DCT) and data quantization, while the sequential processing block is mainly used for Huffman coding. Since the parallel processing block often exploits a SIMD architecture, above arithmetic algorithms for DCT and data quantization are executed effectively. However, an efficient parallel-processing implementation of Huffman coding is difficult to realize [2], [3]. Therefore, the Huffman encoding occupies normally a large share of about 30% of the processing time in the JPEG algorithm [7], [23]. Since the sequential processing tasks depend on the sequential processing block, data has to be transferred between the parallel and sequential processing blocks across a bus, which results in 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. Since the adapted FMCAM enables to combine the input and output ports of several processing elements (PEs), which are implemented in the parallel processing block with SIMD architecture, and executes multiple searches in parallel, it allows to overcome the drawbacks of the conventional table-lookup-coding algorithm.

4.1 Procedure of the Adapted FMCAM-Based High-speed Parallel Table-Lookup-Coding

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. The adapted FMCAM is exploited to implement the input symbol table and the multi-port RAM stores the code word table. An area-efficient bank-based multi-port RAM architecture, as proposed in [24], is very suitable for implementing the code word table. The principle concept is explained in Fig. 6. When the adapted FMCAM receives multiple input symbols, these symbols are compared in parallel with stored symbols. Then each output port of the adapted FMCAM sends the determined matching address to the corresponding port of the multi-port RAM and finally the code words are outputted.

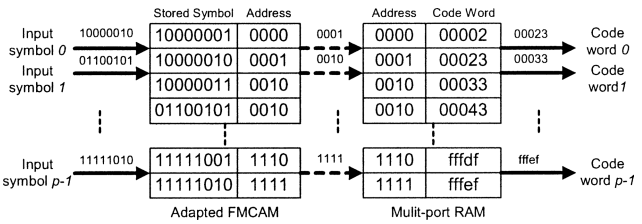

Fig. 6 The proposed solution of parallel table-lookup-coding.

Of particular importance for the effectiveness of the concept is the fact that the hardware amount of the adapted FMCAM can be expected to be far less than that of a parallel-CAM concept, operating several CAMs in parallel [1]. Furtermore, the adapted FMCAM can perform fast parallel comparison with stored symbols and realize a reduction of the comparison clock cycles by implementing the multiple/single search mode and the counting value setting mode of Figs. 3 and 4. Detailed comparison results are reported in Sect. 4.3.

4.2 Adapted FMCAM Implementation Results

A soft-macro realization of the adapted FMCAM has been developed with Verilog-HDL. For verifying the effectiveness of the adapted FMCAM, FPGA and ASIC implementation results for the Verilog-HDL soft-macro are evaluated in this section. For FPGA implementation of the adapted FMCAM, the ISE Foudation 7.1i tool and Synplify Pro 8.1 are applied for synthesis and fitting to the target FPGA device Xilinx, XC4VLX160. In this way maximum operating frequency and total equivalent gate count are determined. For ASIC implementation, the estimation results of maximum operating frequency and total area consumption are evaluated, when synthesized with the Synopsys Design Compiler for a 90 nm CMOS technology. FPGA and ASIC results are shown in Figs. 7 and 8, respectively. In our 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 of increasing port numbers.

Figure 7 (a) shows the maximum operating frequency after synthesis, Placement and Routing (P&R) for the FPGA case, while Fig. 8 (a) shows the maximum operating frequency of the synthesized FMCAMs for the ASIC case. In both cases the maximum operating frequency 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 for FPGA and ASIC implementation. Due to the independent location of port block, category block and controller as well as their modular construction, the maximum operating frequency is practically not influenced by the number of ports. In the FPGA evaluation, the P&R maximum frequencies could only be determined up to 8 ports due to the limited FPGA hardware resources. The extrapolation to 16 ports is therefore just indicated by a dashed line.

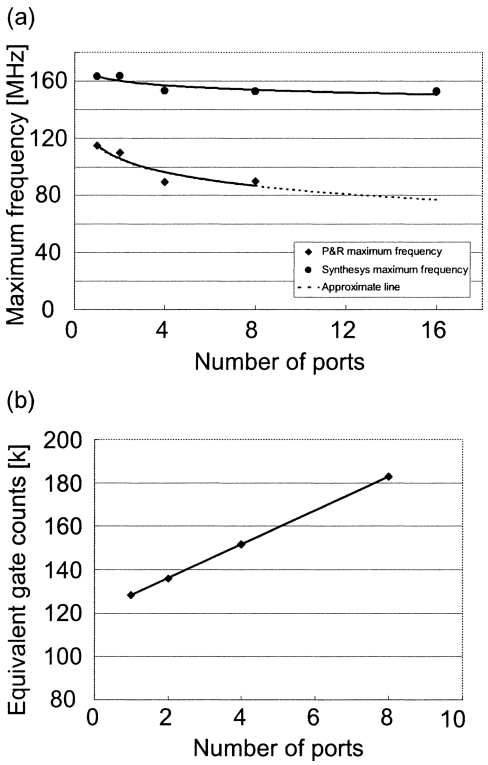

Fig. 7 FMCAM implementation results for the FPGA case (Xilinx XC4VLX160) as a function of the number of ports: (a) Maximum operating frequency, (b) Total equivalent gate counts.

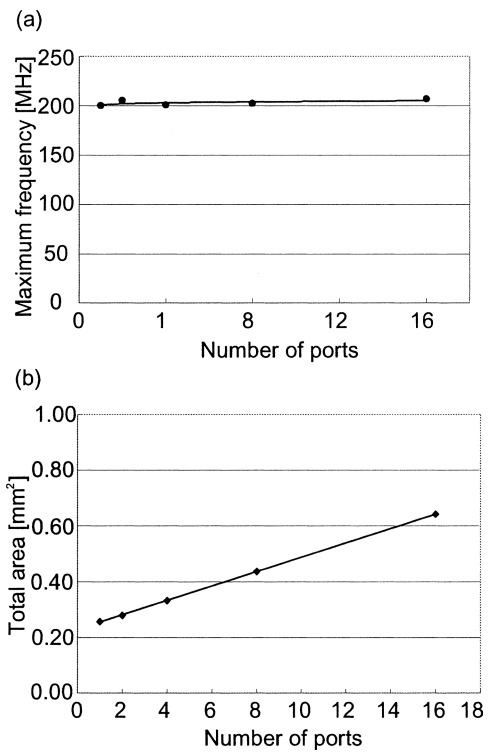

Fig. 8 FMCAM implementation results for the ASIC case (90 nm CMOS technology) as a function of the number of ports: (a) Maximum operating frequency, (b) Area consumption.

Figure 7 (b) shows the total equivalent gate counts for implementing the adapted FMCAM in the FPGA case, while Fig. 8 (b) shows the estimated total area of the ASIC implementation of the adapted FMCAM. The data combines total cell area with virtual interconnect area and are automatically calculated by the Synopsys Design Compiler, 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 restricted 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 systems implemented as FPGA and ASIC. From above results, it is concluded that the adapted FMCAM is a very effective architecture not only for FPGAs but also for ASICs.

4.3 Experimental Results for Huffman Encoding

In this section, several experimental results of the proposed adapted FMCAM-based architecture for Huffman encoding is reported. Four test pictures are used and shown in Fig. 9. These pictures (a) to (d) are taken of various natural motives and have several different characteristics for contents type and pixel number. Figure 10 shows the number of clock cycles for the Huffman encoding operation with these pictures. The adapted FMCAM is compared with a conventional 16-bit DSP, which has a 2-way VLIW (Very Long Instruction Word) architecture, 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 adapted FMCAM with a given port number is 43% smaller than for the original FMCAM. Furthermore, in the case of picture (d) and for 16 ports, the clock cycle number with the adapted FMCAM is 93% smaller than with the conventional DSP.

Table 1 shows the average of encoding clock cycles per one comparison task from 1 port to 16 ports. The adapted FMCAM realizes fast parallel search due to single search mode and counting value setting mode regardless of the BPBP search. Thus, the adapted FMCAM can be compatible with the requirements of restricting the amount of hardware increases when realizing fast parallel search by multiple ports.

Table 2 shows the processing efficiency expressed in Mega Operations Per Second (MOPS) per mm^2. The performance in 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 fair judgment between the adapted FMCAM and the conventional DSP, above parameters deal with the results of Fig. 8. For processing multiple data, several conventional DSPs are used in parallel. Since the maximum possible frequencies of both architectures are almost equal to the each other, the maximum frequency constraint condition of the adapted FMCAM is fixed to 200 MHz. There-

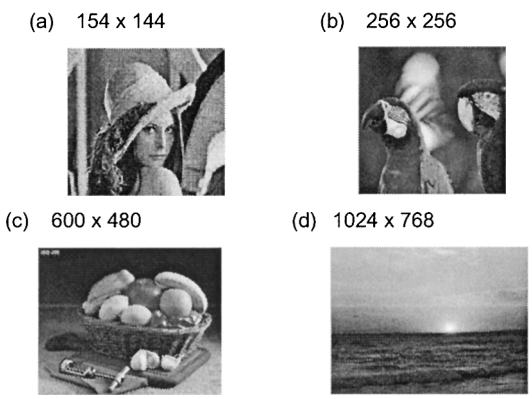

Fig. 9 Test pictures.

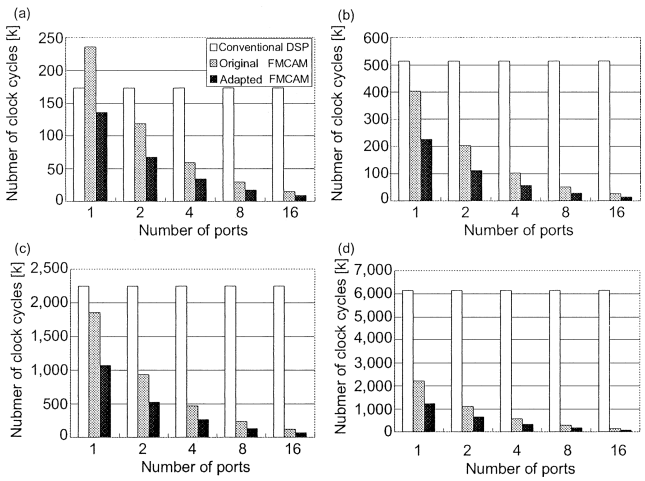

Fig. 10 Comparison of Huffman encoding clock cycles for FMCAM and parallel DSP solutions.

Table 1 Average of Huffman encoding clock cycles per comparison operation.

Number of ports	Average of encoding clock cycles	
	Original FMCAM	Adapted FMCAM
1	19.00	10.79
2	9.50	5.39
4	4.75	2.69
8	2.38	1.36
16	1.19	0.68

Table 2 Comparison of Huffman-coding processing capability for FMCAM (original, adapted) and parallel DSP solutions.

Number of ports	Maximum frequency [MHz]		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

fore, all MOPS values become double if the number of ports is doubled. It seems that the MOPS performance of the parallel conventional DSPs is somewhat better than that of the FMCAM. However, the FMCAM architecture can use the hardware resources even in a synthesized design with much better efficiency. Thus, the additional area consumption remains smaller. Consequently, the MOPS/mm^2 values of the adapted FMCAM are superior to the other architectures. Especially, in the case of 16 ports, the performance in MOPS/mm^2 of the adapted FMCAM is 1.7 times better than that of the original FMCAM and can achieve an up to 3.8 times larger value than for conventional DSP.

As a result, the adapted FMCAM can realize an area-efficient solution for high-speed parallel table-lookup-coding applications. The adapted FMCAM is therefore suitable for replacing the sequential processing block, and can additionally realize a traffic decrease on the internal bus, thus avoiding bus conflicts between the encoding data and other signals. Consequently, the adapted FMCAM architecture is a promising solution for real-time multimedia and ciphers applications.

5. Conclusion

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

Consequently, the adapted FMCAM is a very effective architecture for FPGA and ASIC implementation with many real-time table-lookup-coding application possibilities, because it operates parallel, is scalable and 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. (Japanese Edition), 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. Fundamentals (Japanese Edition), vol.J83-A, no.12, pp.1339–1348, Dec. 2000.
[3] http://edevice.fujitsu.com/fj/MARCOM/find/19-1j/pdf/j19-1-1.pdf
[4] T. Kumaki, K. Iwai, and T. Kurokawa, "A proposal of MFMCAM and its applications," Proc. ITC-CSCC2002, vol.1, pp.224–227, July 2002.
[5] T. Kumaki, K. Iwai, and T. Kurokawa, "A proposal of improved multi-functional multi-port content addressable memory," Forum on Information Technology 2002 (FIT2002), vol.1, no.C-9, pp.205–206, Sept. 2002.
[6] T. Kumaki, Y. Kuroda, T. Koide, H.J. Mattausch, H. Noda, K. Dosaka, K. Arimoto, and K. Saito, "Multi-port CAM based VLSI architecture for Huffman coding with real-time optimizede code word table," Proc. IEEE International Midwest Symposium on Circuits And Systems (MWSCAS'05), pp.55–58, Aug. 2005.
[7] M. Anderson and P. Karlström, "Parallel JPEG processing with a hardware accelerated DSP processor," Examensarbete utfört i Datorteknik vid Tekniska Högskolan I Linköping, May 2004.
[8] ISO/IEC 10918.
[9] ISO/IEC 13818-2.
[10] S.J. Lee, K.H. Yang, J.S. Song, and C.W. Lee, "An efficient memory allocation scheme for huffman coding of multiple sources," Signal Process., Image Commun., vol.14, pp.311–323, March 1997.
[11] CS6100 Motion JPEG Encoder, http://www.amphion.com /cs6100.html
[12] S.M. Lei and M.T. Sun, "An entropy coding system for digital HDTV applications," IEEE Trans. Circuits Syst. Video Technol., vol.1, no.1, pp.147–155, March 1991.
[13] H. Fujiwara, T. Sakaguchi, R. Saito, and M. Maruyama, "Development of VLSI of entropy coding for digital video," IEICE Trans. Commun. (Japanese Edition), vol.J76-B-I, no.12, pp.998–1007, Dec. 1993.
[14] "Data compression with MUSIC CAMs," MUSIC Semiconductors

Application Note AN-N6, Nov. 1998.
[15] R. Saha, "Content-addressable memory speeds up lossless corporation," Electronic Design Online, MOSAID Technologies, Sept. 2003.
[16] K. Kobayashi, K. Tamaru, H. Yasuura, and H. Onodera, "A bit-parallel block-parallel functional memory type parallel processor architecture," IEICE Trans. Electron., vol.E76-C, no.7, pp.1151–1158, July 1993.
[17] M. Defossez, "Content addressable memory (CAM) in ATM applications," Xilinx Application Note XAPP 202, Feb. 2000.
[18] M. Motomura, J. Toyoura, K. Harata, H. Ooka, H. Yamada, and T. Enomoto, "A 1.2-million transistor, 33-MHz, 20-b dictionary search processor (DISP) ULSI with a 160-kb CAM," IEEE J. Solid-State Circuits, vol.25, no.5, pp.828–834, Oct. 1990.
[19] K.J. Schultz and P.G. Gulak, "Architectures for large-capacity CAMs," INTEGRATION the VLSI Jounal, vol.18, pp.151–172, June 1995.
[20] K.J. Schultz and P.G. Gulak, "Fully parallel integrated CAM/RAM using preclassification to enable large capacities," IEEE J. Solid-State Circuits, vol.31, no.5, pp.689–699, May 1996.
[21] M. Hariyama and M. Kameyama, "Collision detection VLSI processor for highly - safe intelligent vehicles using a multiport content-addressable memory," Interdisciplinary Information Sciences, vol.5, no.2, pp.109–116, 1999.
[22] K. Kojima, K. Nishioka, A. Kawaguchi, K. Hosogi, and N. Komoda, "A system-on-chip for real-time software digital media processing," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J87-D-I, no.12, pp.1051–1059, Dec. 2004.
[23] J. Redford, Parallelizing JPEG, ChipWrights Inc., April 2003.
[24] H.J. Mattausch, K. Kishi, and T. Gyohten, "Area-efficient multi-port SRAMs for on-chip data-storage with high random-access bandwith and large strage capacity," IEICE Trans. Electron., vol.E84-C, no.3, pp.410–417, March 2001.

Takeshi Kumaki received a B.S. in 1998 from the Department of mathmatics, Faculty of Science, National Defence Academy, and completed the first half of the M.E. program in Information Mathematics in 2003. From 2003 to 2004, he was affiliated with the Air Force Electric Experimentation Group. Since 2003 he joined the Research Center for Nanodevices and Systems (RCNS), Hiroshima University, Japan, where he has engaged in the system design and architecture research. He is interested in content addressable memory and its applications, SIMD processing, and also in FPGA.

Yutaka Kono received the B.E. degree in Electronic Engineering from Oita National College of Technology, Oita, Japan in 2005. Now, he has joined Hiroshima University, Hiroshima, Japan since 2005. He has research interests in SIMD Architectures and CAM.

Masakatsu Ishizaki was born in Tochigi, Japan in 1983. He received his B.S. degree in Electronic Engineering from Hiroshima University, Japan in 2006. Now he is continuing his Masters course in the Graduate School of Advanced Sciences of Matter in the same university. He has research interests in SIMD Architectures and CAM.

Tetsushi Koide received the B.E. degree in Physical Electronics, the M.E. and the Ph.D. degrees in Systems Engineering from Hiroshima University in 1990, 1992, and 1998, respectively. He was a Research Associate and an Associate Professor in the Faculty of Engineering at Hiroshima University in 1992–1999 and 1999, respectively. From 1999 he was with the VLSI Design and Education Center (VDEC), The University of Tokyo as an Associate Professor. Since 2001 he has been an Associate Professor in the Research Center for Nanodevices and Systems, Hiroshima University. His research interests include system design and architecture issues for memory-based systems, real-time image processing, VLSI CAD/DA, genetic algorithms, and combinatorial optimization. Dr. Koide is a member of the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, and the Information Processing Society of Japan.

Hans Jürgen Mattausch received the Dipl. Phys. degree from the University of Dortmund, Dortmund, Germany, in 1977, and the Dr. rer. nat. degree from the University of Stuttgart, Stuttgart, Germany, in 1981. In 1982 he joined the Research Laboratories of Siemens AG in Munich, Germany, where he was involved in the development of MOS technology as well as the design of memory and telecommunication circuits. From 1990 he led a research group on MOS-technology based power semiconductor devices, which included device design, modeling and packaging. In 1995 he joined the Siemens Semiconductor Group as Manager of the Department for Product Analysis and Improvement in the Chip Card IC Division. Since 1996 he is with Hiroshima University, Higashi-Hiroshima, Japan, where he is presently a Professor at the Research Center for Nanodevices and Systems. His present interests are circuit design and device model issues related to effective utilization of nanodevices and nanotechnology. Dr. Mattausch is a senior member of IEEE.