

A HIGH PERFORMANCE JPEG2000 ARCHITECTURE

Kishore Andra^{}, Chaitali Chakrabarti^{*} and Tinku Acharya^{*+}*

^{*} Arizona State University, Tempe, Arizona, 85287, USA

⁺ Intel Corporation, Chandler, Arizona, 85226, USA

ABSTRACT

JPEG2000 is an upcoming compression standard for still images that has a feature set well tuned for diverse data dissemination required in the present wireless and internet age. These features are possible due to adaptation of discrete wavelet transform, intra-subband bit plane coding and binary arithmetic coding. All the three algorithms are complex and require substantial number of memory accesses. In this paper we propose a system level architecture capable of encoding and decoding using the JPEG2000 core algorithm. The key components include dedicated architectures for wavelet, bit plane and arithmetic coders and memory interfacing between the coders. The system architecture has been implemented in VHDL and its performance evaluated for a set of images. The estimated area of the architecture, in 0.18 μ technology, is 3 mm square and the estimated frequency of operation is 200 Mhz.

1 INTRODUCTION

The differences in the computing power, bandwidth and memory of wireless and wired devices, as well as emergence of diverse imaging application requirements, have made resolution scalability and quality scalability essential in today's still image compression standards. Although these properties can be attained with present JPEG, they cannot be achieved in a single bit stream [1]. To overcome these drawbacks, the upcoming still image compression standard JPEG2000 has been designed [2]. Error resilience, manipulation of images in compressed domain, acceptable performance even at very low bit rates (~ 0.1 bpp), region of interest coding, lossy and lossless performance using same coder, non-iterative rate control etc., are some of the other important features of the JPEG2000 standard. All these features are possible due to adaptation of the Discrete Wavelet Transform (DWT) and intra-subband entropy coding along the bit planes using a combination of Bit Plane Coder (BPC) and Binary Arithmetic Coder (BAC) in the core algorithm.

All the three core blocks namely, DWT, BPC and BAC blocks, are computational as well as memory intensive. The DWT algorithm is a typical "DSP algorithm" with a small set of arithmetic operations performed continuously with symmetrical data access (read) and generation (write) pattern. These properties makes it amenable for implementation using DSP processors and media processors or even dedicated hardware [3]-[6]. In contrast to DWT, both BPC and BAC are control intensive (i.e. contain substantial branching conditions) with few arithmetic operations and are performed bit plane wise. As a result, they cannot be efficiently implemented on DSP or media processors. Even though inherent parallelization is present in the

entropy coding algorithm, the parallel paths are complex, data dependent and are defined only at run time. Further, since the JPEG2000 kernel will be a part of applications like digital cameras, scanners, printers, wireless devices with multimedia capabilities etc., it is important that the kernel be area, time and power efficient. Thus we conclude that while DWT can be implemented by DSP or media processors but specialized implementations are needed for the BPC and BAC coders. Recently, Analog devices has introduced a JPEG2000 co-processor [7] further supporting the hardware implementation paradigm.

In this paper, we propose an integrated architecture to implement the encoding and decoding for the JPEG2000 coder. The architecture primarily consists of three modules: the DWT module, the BPC module and the BAC module. The modules interface with each other via memory and buffers. The DWT module is capable of performing (5,3) filter in the lossless mode and (9,7) filter in the lossy mode on a 8 bit input data. Three pairs of BPC and BAC modules are used to reduce the time required for entropy coding. The architecture has been implemented in VHDL and its performance has been evaluated. The estimated area of the architecture, in 0.18 μ technology, is 3 mm square and the estimated frequency of operation is 200 Mhz.

The rest of the paper is organized as follows. In section 2 we describe the JPEG2000 Part I coder in brief. The proposed system level architecture is discussed in section 3. The performance of the architecture is discussed in section 4 and the paper is concluded in section 5.

2 JPEG2000 BASICS

During encoding, an image is split into rectangular structures called tiles. The tiles are coded separately as if they are different images. The encoding steps are summarized below. For more details, please refer to [2]. Decoding is symmetric to encoding and can be achieved by performing the encoding steps in the reverse order.

Wavelet Transform: In the first step, the Discrete Wavelet Transform (DWT) is applied on the tile to decompose it into a number of wavelet subbands at different levels and resolutions. Recently, a new methodology called lifting [8] has been proposed to perform the DWT. Lifting enables DWT to be computed using a series of banded matrix multiplications. In Part I of the JPEG2000 standard, lifting based implementation of the (5,3) filter (requires 4 banded matrix multiplications) is prescribed for lossless encoding and that of the (9,7) filter (requires 10 banded matrix multiplications) for lossy encoding.

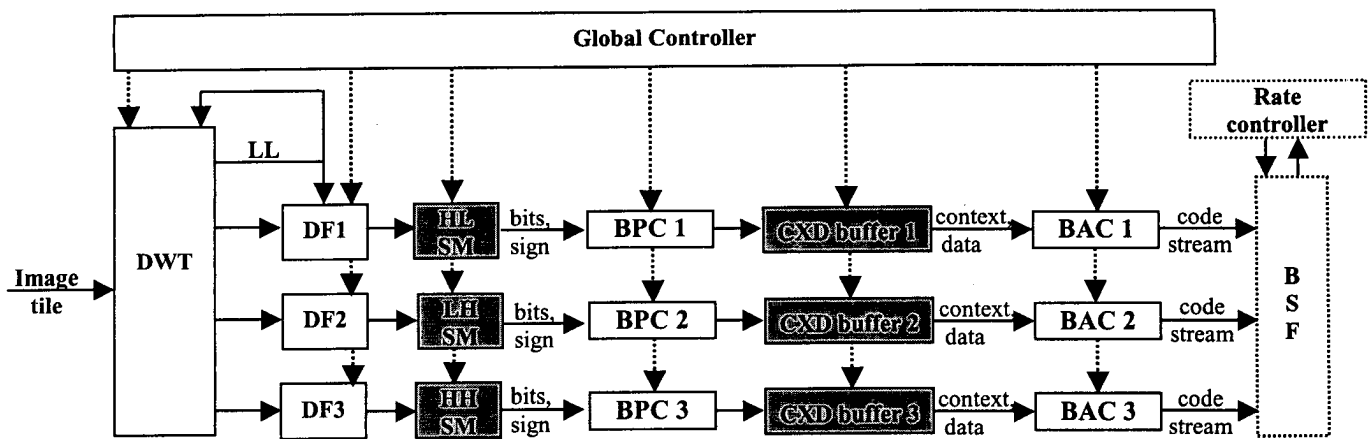


Fig.1 Proposed architecture for JPEG2000 encoder

Quantization: The wavelet coefficients in each subband are scalar quantized if lossy compression is required. The quantization step size is calculated based on the dynamic range of the subband values and constants defined in the standard.

Bit Plane Coding (BPC): The quantized subbands are divided into code blocks. The code blocks are entropy coded along the bit planes using the combination of an embedded Bit Plane Coding (BPC) and the Binary Arithmetic Coding (BAC). In JPEG2000, the Embedded Block Coding with Optimized Truncation (EBCOT) algorithm [9] has been adopted to implement the BPC. This algorithm exploits the symmetries and redundancies within and across the bit planes. It generates the input to the BAC block based on statistics (state information bits that are maintained across the bit planes) of the data coded previously.

Binary Arithmetic Coding (BAC): The BPC outputs are entropy coded using BAC to generate the code stream. The MQ coder, which is a derivative of the Q coder [10], has been proposed to implement the BAC. The algorithm is multiplication free. Predetermined probability values are supplied by the standard and are stored in a look up table. The adaptation state machine is also supplied by the standard.

File formatting and layer formation: For each of the code blocks, distortion for a fixed number of bit rates and code size are calculated by a suitable rate control mechanism. The final bit stream is formed based on the available bit rate by means of "layers" which contain incremental contribution from each code block. So even though neither the required resolution nor the required rate is known while encoding, the best possible image of required resolution is generated for a given bit rate.

3 PROPOSED SYSTEMS ARCHITECTURE FOR JPEG2000

Here, we propose a systems architecture capable of performing the coding process described in the previous section. The input to the architecture is an image tile and the outputs are three code streams (one for each subband). The block diagram of the proposed architecture is shown in Fig.1.

The architecture consists of a DWT module, three pairs of BPC and BAC modules and three Data Formatters (DF). It also consists of (i) three Subband Memory (SM) blocks between the

DWT coder and three BPC coders to store the code blocks formed from the subband data and (ii) three CXD buffers between the BPC and BAC modules to store the context and symbol pairs generated by the BPC module. A global controller is present to control the interactions between all these blocks. The data flow of the architecture is as follows : DWT is applied on the image tile to generate the three high frequency subbands (HL, LH, HH) and one low frequency subband (LL) at each level. The LL subband data is used by the DWT module to compute the next level of decomposition while the other 3 subbands are entropy coded. The subband data is quantized (if required) and broken up into rectangular structures called code blocks. The code blocks are then entropy coded, independently.

Code blocks are written into the Subband Memory (SM) blocks. Each BPC reads the data from the corresponding SM and writes the context-data pairs into the corresponding CXD buffer. BAC reads from the CXD buffer and generates the code stream for each code block. At the last level, the LL subband is entropy coded using the HL entropy coder pair. The code stream generated is supplied to the Bit Stream Formation (BSF) tool to form the final bit stream based on the resolution and quality needed. This process is controlled by a rate controller. The proposed architecture does not handle the division of the image into tiles, the rate controlling or the BSF tool; a host processor with minimal software or an ASIC can perform these functions.

The entropy coding of JPEG2000 takes inordinately long time. For instance, to entropy code a $N \times N$ code block, with one bit position being coded in each cycle, $N \times N \times 16 \times 3$ cycles are required. This is because the internal precision is 16 bits for lossless performance [4], and BPC performs the coding in 3 passes [9]. On top of this, the BAC requires at least two table lookups and two additions to code one bit [2]. Note that while the number of computations can be significantly reduced if the by pass mode proposed in [2] is used, the entropy coder still requires a few million cycles while the DWT requires only about 300 thousand cycles to code a 128×128 block to 5 levels.

Fortunately, the time required for encoding can be reduced if multiple hardware modules are provided since the code blocks are entropy coded independently. For instance, for the case where the DWT coder and the entropy coder work in sync (i.e. while the DWT coder operates on level i , the entropy coder operates on coefficients of level $i-1$), at the most $4 \times 3 + 1$

hardware modules are needed at the first level. This is because in the code block structure that we have considered each subband is split into 4 code blocks at the first level and the whole subband forms a code block in the rest of the levels. In such a scheme, during entropy coding of higher levels, most of the modules would be under utilized. For instance, during computation of levels 2,3 and 4, only three out of thirteen modules are needed. Each hardware module costs ~6000 gates + memory interface. So the choice of the number of hardware modules is clearly a balance between the time constraint and the area constraint.

In our design, we chose 3 hardware modules – one for the HH subband, one for the LH subband and one for the HL and LL subbands. This makes the memory interface between the DWT coder and the entropy coder easier to handle and at the same time reduces the computation time by a factor of 3.

The decoder architecture is similar to the encoder architecture with data flow in the opposite direction. The BSF tool is replaced by a Code Stream Formation tool. The CXD buffer is replaced by a single register to hold the context. This is because the bit plane decoding cannot proceed before the data is obtained from the binary arithmetic decoder. The by-pass mode helps in accelerating the decoding process significantly.

Next we briefly describe the different architectural components.

3.1 Discrete Wavelet Transform module

The architecture performs lifting based DWT/IDWT for the (5,3) and (9,7) filters. The transform is computed in row-column fashion one level at a time (i.e. with no interleaving between the levels). Symmetric extension is used at the boundaries. The architecture generates an output from a lifting step every cycle. The data path width has been determined to be 16 bits [4] and consists of two adders, a 4-level pipelined multiplier and a shifter. A memory block of size $N \times N$, with a read and a write port and a controller are also present. The processor reads in the data from the memory block and writes it back after the transform. The controller generates the input/output signals for both the processor and the memory modules. An “in-place” addressing scheme has been used. The data flow remains the same for both DWT and IDWT. The 4 processor architecture proposed in [3] was not used here because the entropy coder following the DWT module is unable to cope up with such high data rates. If T_a , T_s and T_m are the delays of the adder, shifter and multiplier respectively then the total time required to calculate one level of transform on a $N \times N$ block is $-(2 * T_a + T_s + 1 + N * N / 2) * 4$ for the (5,3) filter and $(2 * T_a + T_m + 1 + N * N / 2) * 8 + (T_m + 1 + N * N / 2)$ for the (9,7) filter.

3.2 Data formatter

The DWT module works with 2's complement data to simplify the addition and multiplication operations. But the BPC module requires the data in sign-magnitude format. The Data Formatter (DF) carries out this conversion in the encoder. Further, DF also determines the most significant bit plane (i.e. the first bit plane which contains a '1') of each code block. The BPC starts coding from the significant bit plane. Quantization, if needed, can be performed by DF. In the decoder, DF performs the conversion from sign-magnitude form to 2's complement form. The significant bit plane value for each code block is supplied by the encoder. The bit planes from the 15th bit plane to the significant

plane are filled with zeros by the DF. Inverse quantization, if needed, is performed by DF.

3.3 Subband Memory (SM)

The data formatters convert the 2's complement data to sign-magnitude form and write the data to the subband memory blocks. The bit plane coders read the data bits and sign bit from the SM blocks along the strips. We have designed a novel memory structure that can handle word in-bit out format combined with the strip structure required for the BPC.

3.4 Bit Plane Coding (BPC) module

The architecture carries out the BPC based on the EBCOT algorithm and generates the context-data bit pairs that are coded by the BAC module. The EBCOT algorithm is carried out in 3 non-overlapping passes, with part of code block being coded in each pass. The encoder architecture consists of (i) combinational logic blocks that transform the state information into input to the BAC module, (ii) 3 memory blocks (of size 32×4 for the proposed architecture) to hold the state information bits, (iii) 5 registers (of various sizes and functionality) to hold the state and magnitude bits, and (iv) a 24-state controller to control all the blocks (please refer to [11] for a detailed discussion of the state diagrams). Based on the pass being performed, appropriate context-data bit pair is selected using a mux. The decoder architecture is very similar to the encoder architecture.

3.5 CXD buffer

The CXD buffer is a FIFO with a read and a write port. Each entry contains a context data bit pair (6 bits). The length of the buffer needs to be as large as possible to account for speed difference between the BPC and BAC coders. We have used 128 entry CXD buffers; the number of entries was determined with experimentation. The global controller uses two pointers to keep track of the FIFO stack.

3.6 Binary Arithmetic Coding (BAC) module

The architecture to implement the BAC module is based on the MQ coder. The architecture consists of a (i) 16 bit adder (ii) registers (various sizes and functionality) (iii) a logic block that helps in the adaptation process (iv) two memory blocks to perform the table look up operations and (v) a controller. The control state machine of encoder (36 states) and decoder (28 states) are different but the rest of the architecture remains the same. While encoding, the context-data bit pair is read from the CXD buffer and code is generated by the module. While decoding, code and context are supplied to the module and data bit is generated. It should be observed that while encoding BPC and BAC can work independently but during decoding BPC has to follow BAC. To speed up the entropy coder, the by-pass mode, proposed in [2]. The proposed architecture is capable of performing the by-pass mode.

4 PERFORMANCE OF THE PROPOSED SYSTEMS ARCHITECTURE

We have conducted the performance analysis of the proposed architecture with four images (baboon, barbara, fish and elaine) of size 512×512 . The input to the architecture is a 128×128 image tile. DWT is carried out to 5 levels. The maximum size of the code block is fixed at 32×32 . After the first level of encoding, each of the 3 subbands are of size 64×64 and are split into 4 code blocks each of size 32×32 . For the rest of the levels whole subband is treated as a code block since the subband is of size 32×32 or smaller. The number of cycles (in millions)

required to encode the images with and without bypass mode for (5,3) and (9,7) filters are given in Table 1. It can be seen that the speed up with by pass mode is around 15% for both the filters.

Table 1 Cycles (in millions) required to encode with and without by pass mode

	(5,3) filter			(9,7) filter		
	Reg.	bypass	%diff	Reg.	bypass	%diff
baboon	7.812	6.646	14.93	7.887	6.687	15.22
barbara	6.879	5.936	13.71	6.944	5.930	14.61
fish	5.665	4.998	11.79	5.789	5.020	13.29
elaine	7.005	6.019	14.07	7.059	6.096	13.65

For decoding, the code stream for the code blocks and attributes (size and number of bit planes) of each code block are supplied to the architecture. Number of cycles (in millions) required for the decoder are given in Table 2. Note that the number of cycles required for decoding is significantly higher than that required by encoding. This is expected as while encoding BPC and BAC coders work independently most of the time due to the CXD buffer, while decoding however the coders are not independent. The speed up due to by pass mode is around 25% which is significant.

Table 2 Cycles (in millions) required to decode with and without by pass mode

	(5,3) filter			(9,7) filter		
	Reg.	bypass	%diff	Reg.	bypass	%diff
baboon	12.050	8.922	25.96	13.001	9.727	25.18
barbara	10.942	8.104	25.94	11.864	8.880	25.15
fish	9.262	7.376	20.37	9.970	8.053	19.23
elaine	10.974	8.369	23.74	11.998	8.053	32.88

The system architecture has been implemented in VHDL. We have synthesized the data path units in DWT coder, BPC encoder and decoder and BAC encoder and decoder. The preliminary gate counts (in 2 input nand gate equivalents) of the modules and the memory required by each module is given in Table 3. The estimated area of the architecture, assuming the control is 20% of data path area in case of DWT, in 0.18 μ technology is 3 mm square and the estimated operation frequency is 200 Mhz.

Table 3 Hardware requirement of the proposed architecture

	Datapath	Memory(bits)
DWT	3500	128x128x16
SM	-	(32x8x64)x3
BPC	2000 (encoder/decoder)	(32x4x3)x3
CXD	-	(128x6)x3
BAC	3200 (encoder)/2500 (decoder)	(19x7+47x16)x3

5 CONCLUSION

In this paper, we have proposed a systems architecture to perform the new JPEG2000 standard compression and decompression of images. The architecture consists of modules to implement the DWT, BPC and BAC algorithms and interfacing memory structures. The BPC and BAC modules are implemented by three sets of computation engines. Such a structure was necessary to compensate for the high

computational requirements of these two modules. The system level architecture has been implemented in VHDL.

Table 4 Differences between ADV JP2000 and the proposed architecture

	ADV JP2000	Proposed architecture
Tile size	256x256	128x128
Code block size	64x64	32x32
Wavelet filters	(5,3) lossy/lossless	(5,3) and (9,7)
Levels of DWT	3	5
Entropy coder pairs	1 (assumed from figure)	3

To the best of our knowledge, the only other JPEG2000 architecture is the JPEG co-processor, ADV-JP2000, by Analog Devices [7]. There are several differences between the two architectures some of which have been listed in Table 4. It is stated that the by pass mode has no effect on the coding speed for ADV. This is quite surprising since in our implementation by pass mode speeds up encoding by around 15% and decoding by 25%.

6 REFERENCES

- [1] J. L. Mitchell and W. B. Pennebaker, "JPEG still image data compression standard," Van Nostrand Reinhold, 1993.
- [2] JPEG2000 Final Committee Draft (FCD), "JPEG2000 Committee Drafts" <http://www.jpeg.org/CDs15444.htm>
- [3] K. Andra, C. Chakrabarti and T. Acharya, "A VLSI architecture for lifting based wavelet transform", SiPS 2000, 70-79, 2001.
- [4] K. Andra, C. Chakrabarti and T. Acharya, "An efficient implementation of a set of lifting based wavelet filters", Appears in proceedings of ICASSP 2001.
- [5] W. Jiang and A. Ortega, "Lifting factorization-based discrete wavelet transform architecture design", IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, 651-657, May 2001
- [6] M. Ferretti and D. Rizzo, "A parallel architecture for the 2-D discrete wavelet transform with integer lifting scheme", Journal of VLSI Signal Processing, vol. 28, 165-185, July 2001.
- [7] "Analog products -ADV-JP2000", <http://products.analog.com/products/info.asp?product=ADV%2DJP2000>
- [8] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting schemes", The J. of Fourier Analysis and Applications, vol. 4, 247-269, 1998.
- [9] D. Taubman, "High performance scalable image compression with EBCOT", IEEE Transactions on Image Processing, vol.9, July 2000.
- [10] J. L. Mitchell and W. B. Pennebaker, "Software implementations of the Q-coder," IBM J. of Res. Develop, vol. 32, No. 6, pp. 753- 774, Nov. 1988.
- [11] K. Andra, T. Acharya and C. Chakrabarti "Efficient VLSI implementation of bit plane coder of JPEG2000", Appears in proceedings of SPIE Applications of digital image processing XXIV, vol. 4472.