A HIGH SPEED BIT PLANE CODER FOR JPEG 2000 AND IT'S FPGA IMPLEMENTATION

Kishor Sarawadekar and Swapna Banerjee

Department of E & ECE, I.I.T. Kharagpur

ABSTRACT

In this paper an optimized architecture of bit plane coder for Embedded Block Coding with Optimal Truncation (EBCOT) algorithm targeting its FPGA implementation is proposed. Although several speed up techniques exist, we present architecture whose performance is improved based on detailed analysis of data path used to obtain context windows. Multiplexer based coding style is adapted to utilize the resources optimally. The proposed design works at 67 MHz after post placement and routing on Xilinx XC2V1000 device. Even though 18 bit planes are used, the implementation results show that the consumption of logic resources in terms of LUTs, slices and flip-flop slices have reduced drastically compared to that of reported designs [1, 2, 3, 4, 5 and 6]. Moreover, power consumption is also moderate.

Index Terms- JPEG2000, EBCOT, Bit Plane Coder

1. INTRODUCTION

JPEG 2000 standard is designed to meet the needs of variety of applications such as multimedia, medical imaging etc. It is the only standard which supports lossy as well as lossless image compression [7]. The Discrete Wavelet Transform (DWT) and EBCOT algorithms used in this standard are computation and memory intensive. Hence dedicated and efficient hardware implementation of JPEG 2000 coding standard is need of the time.

The block diagram of the JPEG 2000 coder is shown in Figure 1. It has three main processing phases: pre-processing, quantization and entropy coding. During pre-processing an image is partitioned into a number of tiles. All samples are level shifted to make their distribution symmetric about zero. Then wavelet transform is applied to obtain sequence of wavelet coefficients. These coefficients are quantized, if required, and stored into code block memory. EBCOT is a two-tier coder. Tier 1 is a context based adaptive arithmetic coder and Tier 2 performs layered bit stream formation.

FPGA based EBCOT tier-1 architecture which can encode more than one bit per clock cycle is discussed in [1]. However, temporary buffer memory can be eliminated. Sample Skipping and Group of Sample Skipping architecture [2] reduces overall image encoding time at the time at the cost of redundant bit access. The Dynamic Significant State Restoring technique [3] avoids use of state variable memory as it is possible to reconstruct state variables. However, data width used is too high. A three level high speed power efficient architecture [4] uses extra hardware in the data path to achieve multi-level parallelism. Another low power architecture is described in [5]. A Bit Plane Coder (BPC) which generates up to 10 context data pairs in a single clock is presented in [6].

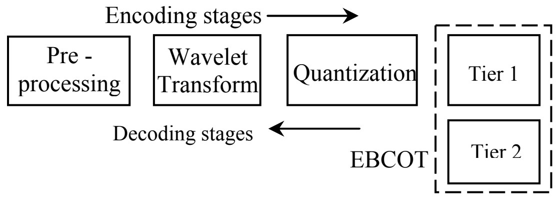

Figure 1. JPEG 2000 block diagram

In this paper, BPC data path is analyzed which leads to optimum and regular hardware. To exploit FPGA resources optimally, multiplexer based coding style is used which contributes to improve overall system performance. The BPC operation is briefed in Section 2. Detailed data path analysis is done in Section 3. In Section 4, proposed bit plane architecture is discussed. The detailed data path architecture is explained in Section 5. Implementation results are given in Section 6 and the paper is concluded in Section 7.

2. BIT PLANE CODER

BPC is the first stage in tier1 of the EBCOT algorithm. It generates contexts and decision based on quantization indexes grouped in code-blocks. The wavelet coefficients are converted from two's complement format to sign-magnitude format and stored in a Code Block (CB) memory as binary numbers [1]. Each binary number can be seen as a sequence of bits with position index from N to 1, where N is number of bits and the binary number has N bits. Bits those are at the same significant position compose a bit plane.

Within bit plane, every four rows form a stripe. JPEG 2000 standard has two types of scanning methods: regular and vertical causal, to scan the stripe data. In a bit plane, scanning order is stripe by stripe from top to bottom. In every stripe, data is scanned bit by bit from top to bottom and column by column from left to right.

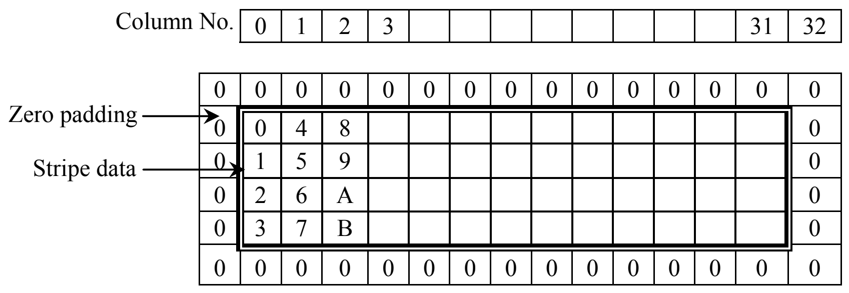

Figure 2. The stripe surrounded with zero padding

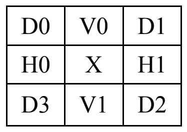

Figure 3. Scanned element and its neighbors

Table 1. Neighbors necessary to process elements starting form first column

Frame	D0	V0	D1	H0	X	H1	D3	V1	D2
1	0	0	0	0	0	4	0	1	5
2	0	0	4	0	1	5	0	2	6
3	0	1	5	0	2	6	0	3	7
4	0	2	6	0	3	7	0	0	0
5	0	0	0	0	4	8	1	5	9
6	0	4	8	1	5	9	2	6	A

Table 2. Frame wise neighbors and their location in SIPO

Element No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Element labeled in SIPO	0	0	0	K	L	M	N	P	Q	R	S	T	U	V	W	0	0	0
Stripe data	0	0	0	0	0	4	0	1	5	0	2	6	0	3	7	0	0	0

There are three primitive coding passes – Significance Propagation Pass (SPP), Magnitude Refinement Pass (MRP), and Cleanup Pass (CUP). These passes are applied on each bit plane of a CB except the most significant bit plane, on which only CUP is applied. Due to the initial condition none of the most significant bit plane bits will be coded in the SPP and MRP. Rest of the bit planes will be coded sequentially using SPP, MRP and CUP [7].

3. DATA PATH ANALYSIS

In vertical causal scanning mode each stripe is considered as a separate entity. In this mode stripe is padded with zero as shown in Figure 2. The BPC processing starts from the 0th row and column, then it moves down to 1st row and so on. After processing 3rd row, next element to be processed is the 0th row of next column to right and process repeats until it reaches the end of the stripe. The operating principle reveals that it is a sliding window architecture. As window slides the neighborhood positions change. First 6 data elements and their corresponding context frames, for the stripe shown in Figure 2, are listed in Table 1. The convention used for identifying neighbors of a frame is shown in Figure 3.

For analysis purpose, Table 1 contents can be rearranged as shown in Table 2. Comparing rows of Table 1 with the last row of Table 2, it is observed that elements numbered 1 to 9 represent the first frame of the stripe shown in Figure 2. Similarly elements 4 to 12 form second frame, elements 7 to 15 form third frame and elements 10 to 18 form last frame of the stripe given in Figure 2. The first and last three elements of Table 2 are labeled as '0' because all through they remain zero. Hence there is no need to implement hardware for it. So it is possible to implement data path by using four 3 bit Serial In Parallel Out (SIPO) registers. Complete data path is discussed in Section 5.

4. PROPOSED ARCHITECTURE

The proposed BPC architecture is shown in Figure 4. The DWT engine used produces 17 bit coefficients which are converted into sign magnitude format and stored in the CB memory. Hence in the proposed architecture 32x32x18 size CB is used. While writing data in CB, sign bit information is separated and stored in a 32x32 bit sign memory. The state variables (σ,σ'and η) required at the time of primitive coding are stored in separate 32x32 bit memory planes.

To process any magnitude bit, one has to read its sign and state variables information along with eight neighbors. This information is produced with the help of data path 1 and 2. Based on run time conditions generated, BPC controller decides which pass is to run, selects a particular coding primitive and stores context data in the context buffer.

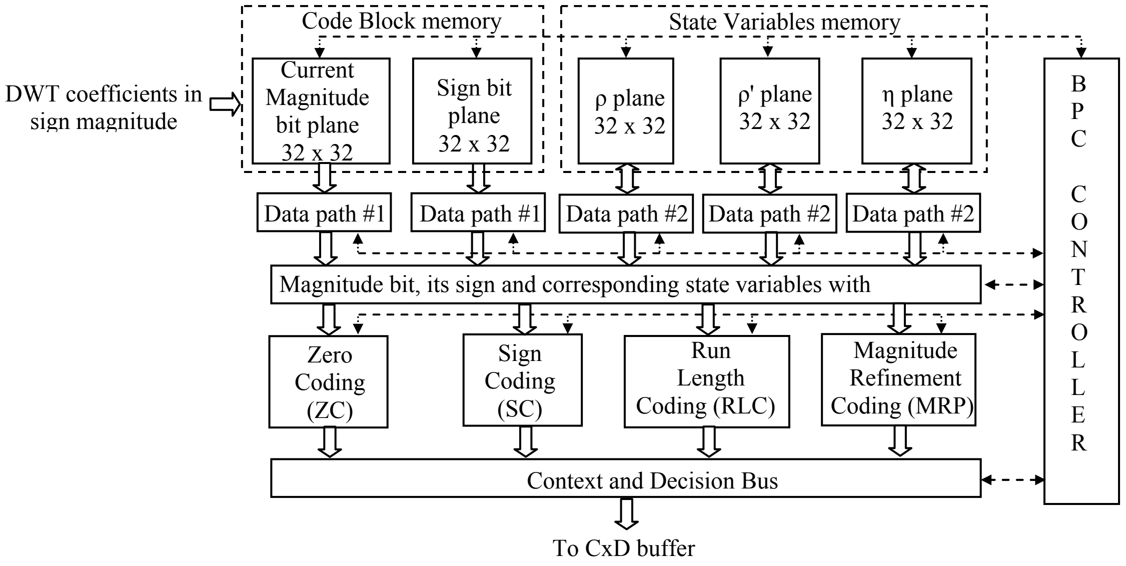

Figure 4. Proposed architecture for BPC

5. DATA PATH ARCHITECTURE

In Figure 4 two distinct data paths are marked. These data paths read data from a selected stripe. However, after processing, sign and magnitude bits will not change so there is no need of write operation whereas based on certain conditions state variables update and new values have to be registered for further processing. Therefore data path 2 slightly differs from data path 1. Data path analysis in Section 3 has shown that it is possible to obtain current context window with the help of four 3 bits SIPO, a mod 5 row counter and nine 4:1 multiplexers as shown in Figure 5.

BPC processes one magnitude bit at a time. However, to reduce number of memory read/write cycles we have chosen to read/write one column data in single cycle. Data just read from a column is stored in the M, Q, T and W elements of the SIPO. The column data which will be processed in current cycle is present in L, P, S and V elements of the SIPO. The column data which is processed in earlier cycle is present in remaining SIPO elements. Thus, four SIPO holds 12 data bits corresponding to 3 columns. These bits are fed to nine 4:1 multiplexers which outputs 9 bits at a time forming the current context window. A mod 5 counter is used to select a particular row to be processed as well as to control the select lines of multiplexer chain as shown in Figure 5. First 4 states of this counter correspond to the row in a column whereas the fifth state is used to pre-fetch next data column and store updated variables.

In data path 2 only one extra 4:1 decoder is required. The decoder outputs are fed to the asynchronous set input of middle (i.e. L, P, S and V) flip-flop in the SIPO register. Whenever a state variable updates, its value is forced at the decoder output setting the flip-flop. The mod 5 counter controls this decoder too.

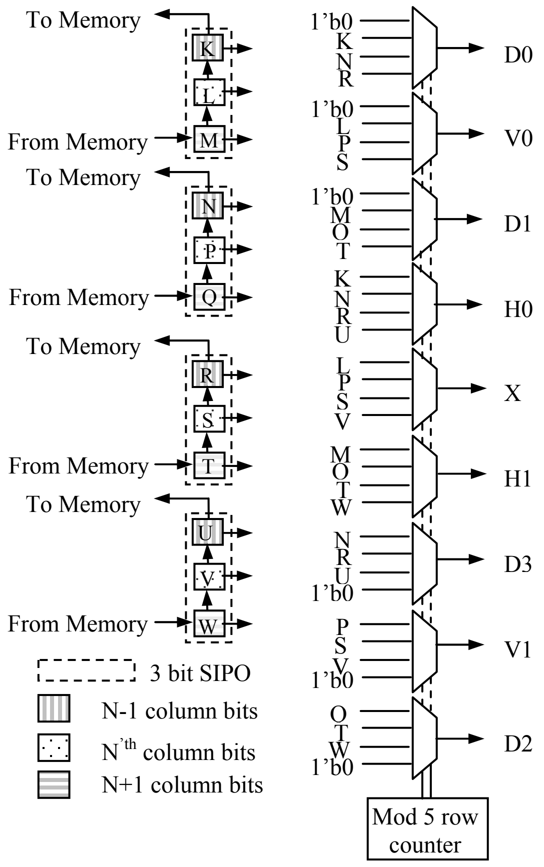

Figure 5. Detailed data path Architecture

5. RESULTS

The proposed architecture is coded using Verilog HDL. The design has been implemented using Xilinx Foundation Series ISE 7.1i version. The target device is XC2V1000-5BG575. Design is simulated using Modelsim XE simulator.

Table 3. BPC implementation results

Parameters	Gangadhar et al. [1]	Proposed Architecture
No. of bit planes	8	18
Frequency of operation (MHz)	50	67
No. of 4 input LUTs used	4,430	2,149
Total Slices used	3,024	2,488
Total flip flop slices used	1,200	105
Both designs implemented using XC2V1000		

Table 4. Comparison with other designs

Architecture	No. of bit planes	CLK (MHz)	Gate count + memory	Power (mW)	Technology used
[2]	8	50.00	19,000 + 13kb	115	TSMC 0.35um 1P4M
[3]	10	66.00	~ 8,500	---	Altera Stratix
[4]	---	75.00	5,200 + 6Kb	410	AMS 0.35um CMOS
[5]	16	43.47	27,069	22	Altera EPXA10DDR
[6]	16	51.70	631	---	APEX20KE
Proposed	18	67.28	12,952+ 21 Kb	330	XC2V1000

The implementation design summary is listed in Table 3. The analysis of data path confirms that it is possible to speed up BPC design with optimum hardware. Gangadhar et al. have used 8 bit planes whereas in the proposed design 18 bit planes are used and for the same device the consumption of logic resources in terms of LUTs, slices, and flip flop slices have reduced significantly at the same time system performance has improved. It is possible because the architecture is more regular and multiplexer based coding style is adapted wherever possible such that the device is utilized optimally.

The proposed architecture is compared with various other similar designs and summary is given in Table 4. Although the platforms are different, proposed design operates on 18 bit planes and shows significant reduction in hardware with improved speed. Memory requirement is higher because of higher number of bit planes. Power consumption is measured using Xilinx XPower tool and moderate power consumption is observed.

7. CONCLUSION

In this paper, an optimized BPC architecture for EBCOT algorithm is proposed. Though this architecture processes bits in serial manner, it speeds up operating frequency because of optimized data path design and appropriate CB data handling technique. The design is coded with multiplexer based approach wherever possible such that the device utilization is optimal. The estimated working frequency is 67 MHz with Xilinx XC2V1000 target device. The maximum operating frequency shoots up to 82 MHz when implemented on Virtex-II Pro family of devices. With this speed, the proposed architecture may encode on an average 27.33 Msamples/s which is equivalent to encoding 89 gray scale VGA frames per second. In addition to this, the consumption of logic resources in terms of LUTs, slices, and flip flop slices is much less. Hence high throughput requirement of real-time systems, like medical imagery requiring lossless compression to digital transmission of images through communication networks, may be met. With slight modification in BPC controller, performance can be further improved by processing multiple stripes concurrently.

ACKNOWLEDGEMENT

The authors would like to thank GE Healthcare India Pvt. Ltd. for supporting this research work.

REFERENCES

[1] Manjunath Gangadhar, Dinesh Bhatia, "FPGA Based EBCOT Architecture for JPEG 2000," Microprocessors and Microsystems, Vol.29, pp. 363 – 373, November, 2005.

[2] Chung - Jr. Lian, Kua - Fu Chen, Hong - Hui Chen, and Liang - Chen, "Analysis and Architecture Design of Block Coding Engine for EBCOT in JPEG 2000," IEEE Transactions on Circuits and Systems for Video Technology, Vol.1, No.3, pp. 219 - 230, March, 2003.

[3] Grzegorz Pastuzak, "A High-Performance Architecture for Embedded Block Coding in JPEG 2000," IEEE Transactions Circuits and Systems for Video Technology, Vol. 15, No. 9, pp. 1182–1191, September, 2005.

[4] Yijun Li, Magdy Bayoumi, "A Three-Level Parallel High-Speed Low-Power Architecture for EBCOT of JPEG2000," IEEE Transactions Circuits and Systems for Video Technology, Vol. 16, No. 9, pp. 1153–1163, September, 2006.

[5] Tien-Wie Hsieh and Youn-Long Lin, "A Low-Power and High Performance EBCOT Architecture of JPEG2000 Encoding," IEEE Symposium on Circuits and Systems, Vol. 1, pp. 773–776, May, 2002.

[6] Amit Kumar Gupta, David Taubman and Saied Nooshabadi, "High-Speed VLSI Architecture for Bit Plane Encoder of JPEG2000," 47'th IEEE Midwest Symposium on Circuits and Systems, Vol. 2, pp. 25-28, 2004.

[7] K. Andra, C. Chakrabarti, and T. Acharya, "A High Performance JPEG2000 Architecture," IEEE Transactions Circuits and Systems for Video Technology, Vol. 13, No. 3, pp. 209–218, March, 2003.