EFFICIENT VLSI ARCHITECTURE FOR BIT PLANE ENCODER OF JPEG 2000

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. After place and route on Xilinx XC2VP30 the proposed design operates at 82 MHz which is capable of encoding 720p (HDTV 1280 x 720, 4:2:2) pictures at nearly 44 frames per second. Even though 14 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 and 5].

Index Terms- JPEG 2000, EBCOT, Bit Plane Coder

1. INTRODUCTION

JPEG 2000 standard has been developed to meet the demand for efficient, flexible and interactive image representations. It is the only standard which supports lossy as well as lossless image compression [6]. The key algorithms involved, Discrete Wavelet Transform (DWT) and EBCOT are computation and memory intensive. Therefore, 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. The DWT first decomposes the image into spatial frequency subbands which generates sequence of wavelet coefficients in 2’s complement format. These coefficients are quantized, if required, and converted to sign-magnitude format prior to storing them in Code Block (CB) memory [1] based which context modeling generates the contexts. The MQ coder performs entropy coding and compresses the data. Finally, the compressed data is packed into a feature rich code stream by using post compression rate distortion optimization algorithm.

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 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. A Bit Plane Coder (BPC) which generates up to 10 context data pairs in a single clock is presented in [5].

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. VLSI design of the 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 stored in CB memory. Bits in each number are segregated and bits with same binary weight are stored in separate memory forming a bit plane. Within bit plane, every four rows form a stripe. JPEG 2000 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.

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 (MSB) plane, on which only CUP is applied. Due to the initial condition none of the MSB plane bits will be coded in
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 sliding window architecture. As window slides the neighborhood positions change. Neighborhood window of first 6 data elements 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 are rearranged as shown in Table 2. Comparing rows of Table 1 with stripe data 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 discussed in Section 5.

4. PROPOSED ARCHITECTURE

The proposed BPC architecture is shown in Figure 4. The DWT engine used produces 13 bit coefficients which are formatted and stored in the CB memory. Hence the proposed architecture has CB size 32x32x14. During CB write operation sign bit information is separated and stored in sign plane. The state variables (σ, σ’ and η) are stored in state variable planes. Neighborhood windows necessary to process any magnitude bit are generated using 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.

5. VLSI ARCHITECTURE

5.1. Data Path Architecture
In Figure 4 two distinct data paths are marked. These data paths read data from a selected stripe. Since sign and magnitude bits do not change these planes are not updated. 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 neighbors window with the help of four 3 bits SIPO, a mod 5 row counter and nine 4:1 multiplexers. In Figure 5, data path 2 is depicted.

BPC processes one magnitude bit at a time. In order to reduce memory access and hence power consumption, data read/write is performed once for each column. To process ith column, i+1th column is read and stored in SIPO elements marked M, Q, T and W. Data for i and i-1th column will be present respectively in the L, P, S, V and K, N, R, U elements of the SIPO. This way four SIPO holds 12 data bits corresponding to 3 columns. These bits are fed to nine 4:1 multiplexers who outputs 9 bits at a time forming the current neighbors 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 which is not present in data path 1. 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 any state variable updates, its value is forced at the decoder output setting the flip-flop asynchronously.

5.2. Zero Coding Architecture
Out of 19 contexts listed in JPEG 2000 standard first 9 (i.e. 0-8) are reserved for ZC primitive. Contexts table for LL and LH band is same whereas for HL and HH subbands table is different. Entries in this table represent summation of H, V and D neighbors which are less than or equal to 4. One can simplify these tables by re-writing them in binary format while doing so represent 4 with 2'b11. Context table simplified in this manner, for LL band is given in Table 3. On FPGA this table is directly mapped using multiplexers.

<table>
<thead>
<tr>
<th>H1</th>
<th>H0</th>
<th>V1</th>
<th>V0</th>
<th>D1</th>
<th>D0</th>
<th>CX</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Here v_L, v_P, v_S and v_V are current magnitude bit at row 0, 1, 2 & 3 respectively.

5.3. RLC Architecture
In RLC current magnitude column bits are used to generate ‘decision’ bits as shown in Figure 6. When RLC is enabled and if none of the current magnitude bit is set, ‘Inc_clm’ signal goes high and current column is skipped. However, if any magnitude bit is high, the 2 bit counter starts so that on each clock 1 context is generated. In counter state 3, row
counter in data path is reloaded with ‘lsb D’ and ‘msb D’ data so that sign for first non zero magnitude bit is coded.

6. 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. The implementation design summary is listed in Table 4. 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 14 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 5. Although the platforms are different, proposed design operates on 14 bit planes and shows significant reduction in hardware with improved speed. Memory requirement is higher because of higher number of bit planes. To handle the boundary conditions as per design requirements, we preferred to code entire CB and state variables memory rather than using internal block RAM. Power consumption is measured using Xilinx XPower tool and moderate power consumption is observed.

7. CONCLUSION

The proposed architecture encodes all passes sequentially. To use available resources optimally, multiplexer based design approach is adopted. By optimized data path design and appropriate CB data handling technique, the design operates at 67 MHz on Xilinx XC2V1000 device. After place and route on Vertex-II Pro family devices maximum operating frequency achieved is 82 MHz which is capable of encoding HDTV 1280 x 720 4:2:2 44.48 frames per second in real time. 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., Bangalore and Department of Information Technology, Govt. of India, for supporting this research work.

REFERENCES