Area Efficient, High-speed VLSI Design for Ebcot Block Coder in JPEG 2000

Kishor Sarawadekar
Department of E & ECE
Indian Institute of Technology
Kharagpur, India
e-mail: to.skishor@gmail.com

Swapna Banerjee
Department of E & ECE
Indian Institute of Technology
Kharagpur, India
e-mail: swapna@ece.iitkgp.ernet.in

Abstract—With the growth in multimedia technology, demand for high-speed real-time image compression system has also increased. JPEG 2000 standard is developed to cater such application requirements. However, the sequential execution of the bit plane coder (BPC) used in this standard consumes more clock cycles. To improve the performance of the BPC, a new concurrent context modeling technique is proposed in this paper. To study number of contexts generated in each clock cycle, analysis is carried out on five ISO grayscale images with size 512 x 512. The study revealed that about 58% of time more than 4 contexts are generated in one clock. Therefore, a new concurrent context coding architecture is proposed in this paper. It is implemented on Stratix FPGA and the hardware requirement is reduced significantly, compared to similar architectures. Moreover, number of clock cycles required to encode a bit plane is reduced by 10% and it is minimum 2.5 times faster than the similar designs in existence. This design operates at 164.47 MHz, which makes it compatible for encoding HDTV 1920 x 1080 4:2:2 at 39 frames per second.

Keywords— JPEG 2000; Bit Plane Coder; EBCOT; VLSI Architecture

I. INTRODUCTION

JPEG 2000 [1] is a popular image-coding standard, which is most widely used in applications like medical imagery, printing, digital photography, remote sensing etc. It has very rich set of features such as region of interest coding, error resilience, random code stream access etc. All these features are supported in a single compressed bit stream. Discrete wavelet transform (DWT) and embedded block coding with optimized truncation (EBCOT) [2] are the key algorithms used in this standard [1]. Computational complexities of both the algorithms are very high. To overcome this problem, a high speed DWT architecture is presented in our earlier work [3] and in this paper, a high-performance context modeling EBCOT architecture is proposed.

![Figure 1. Block diagram of EBCOT encoder](image-url)

Fig. 1 shows the functional block diagram of EBCOT encoder having Tier-1 (block coder) and Tier-2. The block coder is further partitioned into bit plane coder (BPC) and MQ coder. The input to BPC is quantized DWT coefficients, which are stored in code block (CB) memory. BPC produces a context and decision (CxD) pair for each bit in the CB memory. The MQ coder processes these pairs and produces a compressed bit stream. Finally, as per the user’s requirement, Tier-2 organizes this bit stream and generates compressed data. The Dynamic Significant State Restoring technique [4] avoids use of state variable memory, as it is possible to reconstruct state variables. However, it uses high data width and needs additional coefficients surrounding the scan square to process a sample. A three level high speed power efficient architecture [5] uses extra hardware in the data path to achieve multi-level parallelism. Memory efficient parallel architecture [6] eliminates need for all the state variable memories. In this architecture, three clock cycles are wasted whenever run length coding (RLC) skip a column. BPC architecture proposed in [7] processes one stripe column in one clock and generates up to 10 CxD pairs at a time. However, to arrange these CxD pairs according to their generation sequence additional Arrange-Contexts block is necessary. All these architectures [4]-[7] do not support 1920 x 1080 p25 scanning format and are unsuitable for HDTV application.

This paper presents an analysis of CxD pairs generation using concurrent samples coding technique. Based on this analysis, a new BPC architecture is proposed which can produce up to 10 CxD pairs in a clock. With this method need for one of the state variables memory has also reduced. The rest of the paper is organized as follows. Section II provides brief overview of BPC. Analysis of concurrent contexts modeling is given in Section III. The proposed BPC architecture and its implementation details are discussed in Section IV. Experimental results and comparison of performances are given in Section V. Finally, brief conclusions are drawn in Section VI.
### TABLE I. PROBABILITY OF NUMBER OF CxD PAIRS GENERATED PER CLOCK CYCLE

<table>
<thead>
<tr>
<th>Image</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>1280</td>
<td>0</td>
<td>0</td>
<td>323812</td>
<td>391595</td>
<td>279708</td>
<td>119133</td>
<td>33160</td>
<td>5211</td>
<td>1500</td>
<td>1282176</td>
</tr>
<tr>
<td>Baboon</td>
<td>106509</td>
<td>0</td>
<td>0</td>
<td>641096</td>
<td>531860</td>
<td>285750</td>
<td>109564</td>
<td>28040</td>
<td>4779</td>
<td>1080</td>
<td>1708678</td>
</tr>
<tr>
<td>Boat</td>
<td>120964</td>
<td>0</td>
<td>0</td>
<td>426452</td>
<td>458945</td>
<td>284172</td>
<td>114478</td>
<td>28666</td>
<td>4581</td>
<td>1550</td>
<td>1439798</td>
</tr>
<tr>
<td>Peppers</td>
<td>148539</td>
<td>0</td>
<td>0</td>
<td>375744</td>
<td>434875</td>
<td>283824</td>
<td>118223</td>
<td>30040</td>
<td>4662</td>
<td>1640</td>
<td>1397547</td>
</tr>
<tr>
<td>Barbara</td>
<td>116878</td>
<td>0</td>
<td>0</td>
<td>476512</td>
<td>453035</td>
<td>286032</td>
<td>117915</td>
<td>32568</td>
<td>5598</td>
<td>2090</td>
<td>1490628</td>
</tr>
<tr>
<td>Average</td>
<td>6.39</td>
<td>0</td>
<td>0</td>
<td>36.35</td>
<td>30.99</td>
<td>17.41</td>
<td>6.72</td>
<td>1.75</td>
<td>0.30</td>
<td>0.09</td>
<td>100</td>
</tr>
</tbody>
</table>

II. OVERVIEW OF BPC CODER

The quantized DWT coefficients are converted into sign magnitude format and stored in CB memory. These coefficients are encoded by examining each bit-plane starting from most significant bit-plane to least significant bit-plane. In order to reduce the memory requirement each bit plane is further partitioned into stripes. Each stripe can be treated as a matrix of 4 rows and 32 columns. To examine each coefficient 3 passes viz. Clean-up Pass (CUP), Significant Propagation Pass (SPP) and Magnitude Refinement Pass (MRP) are used and CxD pairs are generated. To determine the CxD value, these passes use 4 different coding primitives: Zero Coding (ZC), Sign Coding (SC), Run Length Coding (RLC) and Magnitude Refinement Coding (MRC). The coding state of each sample is determined by using state variables $\sigma$, $\sigma'$ and $\eta$. The SPP pass uses ZC and SC and updates $\sigma$ variable. MRP uses only MRC and updates $\sigma'$ state. The CUP uses RLC, ZC and SC primitives and it may also update $\eta$ state variable. Of the 4 primitives, ZC and MRC generate contexts with the help of 8 adjacent neighbors’ $\sigma$ state. SC uses 4 neighbors’ (left, right, top and bottom) sign and their $\sigma$ states. When $\sigma$ state of all the samples in a given column and their neighbors is equal to 0, that particular column is skipped after applying RLC only on the first sample.

III. ANALYSIS OF CONCURRENT CONTEXT MODELLING

To improve the computational efficiency concurrent sample coding methodology is studied and statistical analysis is performed on five standard test images — Lena, Baboon, Boat, Peppers and Barbara. During encoding procedure, if all magnitude bits, for a column under examination, are high with its own and neighboring $\sigma$ states equal to 0, maximum 10 CxD pairs are generated. These pairs include 3 RLC contexts, 3 ZC contexts and 4 SC contexts. Similarly, when none of the magnitude bit for a column under examination is high and its own and eight neighborhood $\sigma$ states are also equal to 0, only 1 CxD pair is generated using RLC primitive. Moreover, the magnitude column being examined is skipped. Table 1 shows the probability of number of CxD pairs generated per clock cycle. Interestingly, it is observed that 2 or 3 CxD pairs are never generated. It is so because at a time 4 samples are coded and if a magnitude column is not skipped, minimum 4 CxD pairs get generated. Similarly, count value generated by CxD pair 1 represents how many magnitude columns are skipped while encoding an image. From Table 1, it is also clear that occurrence of 4 CxD pair is higher because each sample is coded in any one of the passes. Therefore, time required to encode these samples can be reduced by processing 4 samples concurrently.

### Figure 2. Proposed VLSI architecture of the BPC

IV. PROPOSED BPC CODER ARCHITECTURE

The proposed VLSI architecture of the BPC module is illustrated in Fig. 2. The memory controller selects a bit plane to be encoded and its corresponding sign plane, reads sample magnitude, sign and the state variables data. Since magnitude and sign data are never updated, these planes are implemented using single port RAM (SPRAM). The state memories are implemented using dual port RAM (DPRAM). Once bit plane is selected, the stripe controller selects a stripe to be processed. While processing first column of every stripe, its left side neighboring column is assumed to be zero. Similarly, while processing last column of every stripe, its right side neighboring column is assumed to be...
zero. These boundary conditions are handled by the boundary handler. In order to process all magnitude bits concurrently, entire magnitude column \((M)\) and the corresponding sign bits column \((S)\) are read in one clock. In addition, four sign neighbors, eight \(\sigma\) neighbors and four \(\sigma'\) values of the column to be processed are read which form reference windows. Based on the values in these windows, the pass detector determines the coding pass and primitive to be run on the selected samples, which is detailed in Section IV-A. The coding primitives encode each sample concurrently and generate 1 to 10 CxD pairs simultaneously. The buffer controller reorders these CxD pairs and stores them in the respective pass buffers.

A. Concurrent Pass Detection Logic

During concurrent samples coding the passes are not run sequentially and entire encoding results are dependent on pass detection logic. To accomplish this, depending on the sample’s magnitude bit recent change in \(\sigma\) state is predicted. The pseudo code for this module is given below where \(v[i]\) is the magnitude of the \(i\)th sample, \(\sigma[i]\) is the state value, \(P[i]\) is pass applied and logical OR operation is denoted by \(|\|\).

For sample at \(i = 0\):

if (current \(\sigma\) column and all its neighbors == 0) then
  \(P[0] = CUP\)
else if (\(\sigma[0]\)) then
  \(P[0] = MRP\)
else
  \(P[0] = CUP\)
end if

For sample at \(i = 1\):

if (\(P[0] == CUP\)) then
  if (\(v[0]\)) then
    \(P[1] = SPP\)
  else if (\(v[1]\)) then
    \(P[1] = CUP\)
  else
    skip sample \(i = 1\)
  end if
else if (\(\sigma[1]\)) then
  \(P[1] = MRP\)
else
  \(P[1] = SPP\)
end if

For sample at \(i = 2\):

if (\(P[0] == CUP\)) then
  if (\(v[0] || v[1]\)) then
    \(P[2] = SPP\)
  else if (\(v[2]\)) then
    \(P[2] = CUP\)
  else
    skip sample \(i = 2\)
  end if
else if (\(\sigma[2]\)) then
  \(P[2] = MRP\)
else
  \(P[2] = SPP\)
end if

For sample at \(i = 3\):

if (\(P[0] == CUP\)) then
  if (\(v[0] || v[1] || v[2]\)) then
    \(P[3] = SPP\)
  else if (\(v[3]\)) then
    \(P[3] = CUP\)
  else
    skip sample \(i = 3\) % all samples skipped
  end if
else if (\(\sigma[3]\)) then
  \(P[3] = MRP\)
else
  \(P[3] = SPP\)
end if

During CUP some or all of the magnitude bits may be skipped. The first if clause demonstrates technique to detect the same and it also describes \(\sigma\) prediction method. The entire pseudo code is implemented using combinational logic.
**TABLE II. COMPARISON WITH SIMILAR BPC ARCHITECTURES**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Startix</td>
<td>ASIC</td>
<td>–</td>
<td>Startix</td>
<td>Startix</td>
</tr>
<tr>
<td>Clk (MHz)</td>
<td>86.9</td>
<td>75</td>
<td>100</td>
<td>66</td>
<td>164.47</td>
</tr>
<tr>
<td># of LE</td>
<td>817</td>
<td>43k</td>
<td>4.7k</td>
<td>8.5k</td>
<td>690</td>
</tr>
<tr>
<td># of Clk/bp</td>
<td>–</td>
<td>1659</td>
<td>1642</td>
<td>1202</td>
<td>1088</td>
</tr>
<tr>
<td>Encoding time (uS)</td>
<td>–</td>
<td>22.12</td>
<td>16.42</td>
<td>18.21</td>
<td>6.61</td>
</tr>
</tbody>
</table>

**VI. CONCLUSION**

This paper presents high performance VLSI architecture for the BPC in JPEG 2000. Analysis of concurrent contexts modeling shows that the probability of generating more than four CxD pairs generation is 58%, which implies that the time required to encode the symbols individually is reduced. The implementation results demonstrate that the proposed architecture encodes a bit plane 2.5 times faster than the existing similar architectures with less hardware requirement. The architecture operates at 164.47 MHz Thus, it is capable of encoding HDTV 1920 x 1080 4:2:2 at 39 frames per second. As the encoding speed is higher, using this encoder lossless compression in the domain of multi-spectral imaging, bio-medical imaging, satellite imagery etc. is possible in real time. Moreover, it is expected that the ASIC implementation of this design will outperform all the existing designs.

**ACKNOWLEDGMENT**

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

**REFERENCES**


