Agnieszka Dąbrowska-Boruch*, **, Kazimierz Wiatr*, **

Implementation of FCT Transformation in JPEG-XR Standard in Programmable Devices

1. Introduction

One of the newest image compression standards is the JPEG-XR (ITU-T T.832 [1]) standard developed by Microsoft. Microsoft started working on a new compression standard in 2006. This standard has to substitute so far used JPEG standard. Requirement of the new algorithm was to achieve a similar performance to the JPEG2000 standard in terms of compressed image quality and in term of compressed image file size. The new algorithm was known under the working name: Windows Media Photo (WMP) or HD Photo, but during process of standardization was renamed to JPEG-XR [3, 4, 6].

A ITU-T T.832 standard offers possibility of lossless compression as well as lossy compression. There is ability to support input data stored in formats: integer (UINT, SINT), fixed point and floating point (IEEE 754). A single pixel can be represented in the case of RBG images with 8 or 16 bits per colour component for the integer format and 16 or 32 bits per colour component for fixed point and floating-point format. JPEG-XR can represent 65536 colour tones for a single colour component using 48 bits per pixel. It means that the JPEG-XR can represent $2{,}8 \cdot 10^{14}$ colours. It can be used a lossy compression mode as well as a lossless compression mode for 8 or 16 bits per each colour component. It is allowed to use only a lossy compression for 32 bits per colour elements.

The JPEG-XR can process RGB, YUV, CMYK, monochromatic images. Figure 1 shows a block diagram of JPEG-XR standard compression.

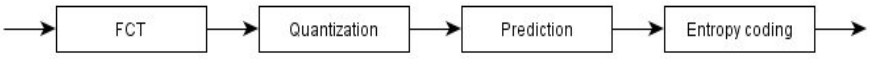

Fig.1. Block diagram of JPEG-XR standard compression

* AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, Institute of Automatics, al. A. Mickiewicza 30, 30-059 Krakow, Poland
** AGH University of Science and Technology, ACC CYFRONET AGH, Krakow

The smallest processing unit during compression and decompression according to JPEG-XR standard is a block, like in the JPEG standard. The JPEG-XR block (Fig. 2) has a structure of 4×4 pixels in contrast to the JPEG's structure of 8×8 pixels.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Fig. 2. Block structure according to JPEG-XR standard

A macroblock (Fig. 3) is a greater processing object than the block. The macroblock consist of 16 blocks in the configuration of 4×4 pixels.

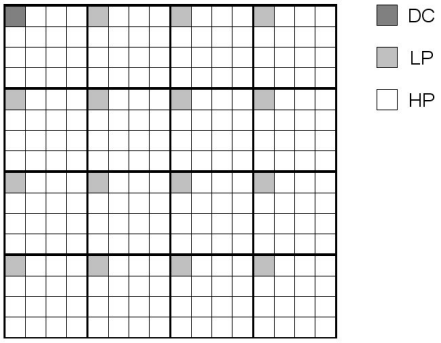

Fig. 3. Macroblock structure according to JPEG-XR standard

There is only one element marked as DC in the single macroblock. It is the coefficient of the first block marked as 0 shown in Figure 2. Zero coefficients of other blocks are marked as LP (*Low-Pass*). There are 15 such LP elements in the single macroblock. Other coefficients (white colour on Fig. 3) are identified as HP (*High-Pass*) coefficents. There are 240 HP coefficients in the single macroblock.

2. Forward Core Transform

One of elements of the ITU-T T.832 compression process is a FCT (Forward Core Transform). The Forward Core Transform in JPEG-XR has a similar function as the Discrete Cosine Transform in JPEG. It is responsible for conversion of image data into the frequency domain representation. The FCT is a transform that provides lossless compression as the discrete cosine transform used in JPEG standard. The advantage of the FCT is

less computational complexity in comparison to the DCT [7]. The Forward Core Transform is performed in two stages. Each block of macroblock is subjected to Hadamard transform in the first stage. 2×2 Hadamard transform (T2×2h) (1) is used four times in each blocks processing. Correlation between pixels in block (Fig. 2) is shown in Figure 4.

$$T_H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{1}$$

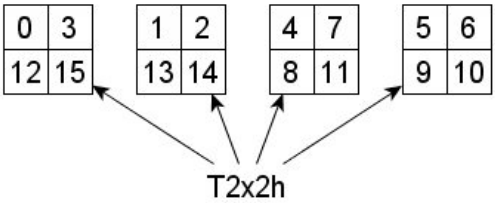

Fig. 4. Correlation between pixels in the first stage of FCT

All DC and LP coefficients of macroblock (Fig. 3) each of blocks processed in the first stage are grouped into a new block. The rule of grouping is shown in Figure 5.

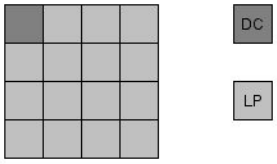

Fig. 5. Coefficients of macroblock processed in the second stage of FCT

Then, the block is divided into 4 groups. Each group contains 2×2 coefficients obtained after the first stage FCT. Figure 6 shows the method of numbering of block coefficients and division into groups.

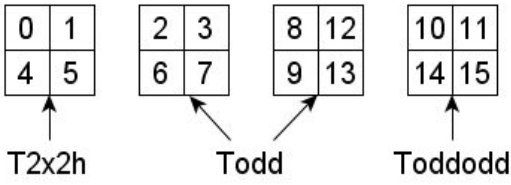

Fig. 6. Correlation between pixels in the second stage of FCT

$$T_R = \begin{bmatrix} -\cos(\pi/8) & \sin(\pi/8) \\ \sin(\pi/8) & \cos(\pi/8) \end{bmatrix} \tag{2}$$

A Todd transform and a Toddodd transform are used in the second stage of FCT. The Todd transform uses the one-dimensional rotation T_R (2). The Toddodd transform uses the two-dimensional rotation.

Some hardware architectures have been proposed in [2] and [5]. The attained throughput of the FCT in [2] is approximately 2.7 pixels per clock step.

3. Implementation results

The first stage of Forward Core Transform in processing of image compression compatible with JPEG-XR standard requires T2×2h transform implementation. A pseudocode of the two-dimensional Hadamard's transform is following:

```
T2x2h(iCoeff[ ], valRound) {
 iCoeff[0] += iCoeff[3]
 iCoeff[1] −= iCoeff[2]
 valT1 = ((iCoeff[0] − iCoeff[1] + valRound) >> 1)
 valT2 = iCoeff[2]
 iCoeff[2] = valT1 − iCoeff[3]
 iCoeff[3] = valT1 − valT2
 iCoeff[0] −= iCoeff[3]
 iCoeff[1] += iCoeff[2]}
```

where *iCoeff[x]* are input coefficients (pixels in the first stage of FCT or coefficients obtained after the first stage of FCT that are input data in the second stage of FCT) and a *valRound* value is equal 0 in the first stage of FCT and equal 1 in the second stage of FCT.

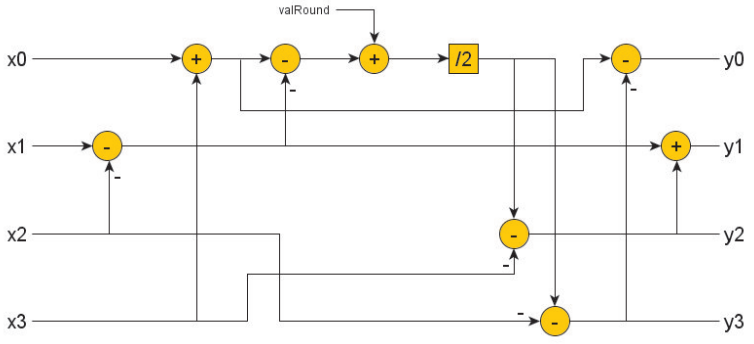

Fig. 7. Graphical diagram of data flow in T2×2h

Figure 7 shows a graphical diagram of data flow of the two-dimensional Hadamard transform. A hardware implementation of the single T2×2h transform uses 90 Slice blocks in Xilinx V5LX110 device at the 16-bit input data. This 90 Slice blocks corresponds with 14739 equivalent gates. This occupancy of the device contains 273 flip-flop blocks and 224 LUT blocks.

Another of the implemented hardware blocks of FCT is the block of one-dimensional rotation Todd. The pseudocode (specified in ITU-T T.832 recommendation) defining the Todd is as follows:

```
Todd(iCoeff[ ]) {
 iCoeff[1] −= iCoeff[2]
 iCoeff[0] += iCoeff[3]
 iCoeff[2] += ((iCoeff[1] + 1) >> 1)
 iCoeff[3] = ((iCoeff[0] + 1) >> 1) − iCoeff[3]
 iCoeff[1] −= ((3* iCoeff[0] + 4) >> 3)
 iCoeff[0] += ((3* iCoeff[1] + 4) >> 3)
 iCoeff[3] −= ((3* iCoeff[2] + 4) >> 3)
 iCoeff[2] += ((3* iCoeff[3] + 4) >> 3)
 iCoeff[3] += (iCoeff[1] >> 1)
 iCoeff[2] −= ((iCoeff[0] + 1) >> 1)
 iCoeff[1] −= iCoeff[3]
 iCoeff[0] += iCoeff[2]}
```

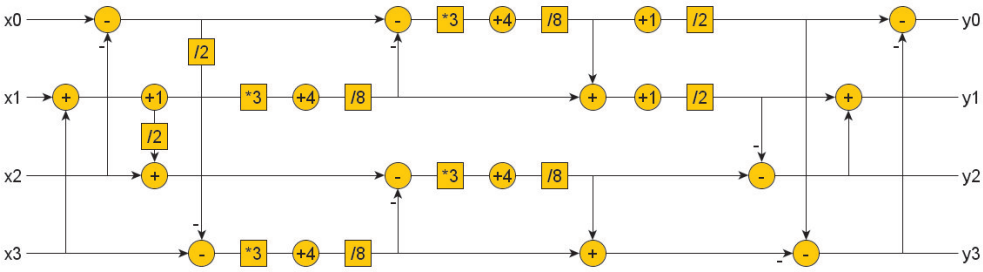

Fig. 8. Graphical diagram of data flow in Todd

Figure 8 shows a graphical diagram of data flow in Todd.

The hardware implemented Todd transform compatible with above-mentioned pseudocode needs 6 clock steps to determine the output values. The Todd module uses 137 slice blocks (1%), therein 379 Flip-Flop blocks and 332 LUT blocks. This 6 clock steps needed to obtain output values causes necessity of using additional registers to extension

T2×2h transform processing of 2 clock steps. The modification of pseudocode was made in order to time reduction to 4 clock steps. The modified pseudocode for the Todd transform is following:

```
Todd(iCoeff[ ]) {
 valT1=iCoeff[1]
 iCoeff[1] −= iCoeff[2]
 iCoeff[2]=(iCoieff[2] + valT1+1)>>1
 valT2=iCoeff[0]
 iCoeff[0] += iCoeff[3]
 iCoeff[3]=(valT2-iCoeff[3]+1)>>1
 iCoeff[1] −= ((3* iCoeff[0] + 4) >> 3)
 iCoeff[0] += ((3* iCoeff[1] + 4) >> 3)
 iCoeff[3] −= ((3* iCoeff[2] + 4) >> 3)
 iCoeff[2] += ((3* iCoeff[3] + 4) >> 3)
 valT3=iCoeff[3]
 iCoeff[3] += (iCoeff[1] >> 1)
 iCoeff[1] =( iCoeff[1]>>1) − valT3
 valT4=iCoeff[2]
 iCoeff[2] −= ((iCoeff[0] + 1) >> 1)
 iCoeff[0] = valT4 + ((iCoeff[0] − 1)>>1)}
```

The execution of operations from first six pseudocode lines takes only one clock step of implemented algorithms. The operations of the next four pseudocode lines require another two clock steps. The operations of last six pseudocode lines require only one clock step. The Todd transform needs only four clock steps for processing thanks to made modification. A hardware implementation of such modified Todd transform uses 112 (1%) slice blocks of V5LX110 device. This device utilization contains 275 Flip-Flop blocks and 331 LUT blocks.

The last one of the blocks of the second stage of FCT is the two-dimensional rotation Toddodd. The pseudocode of Toddodd transform specified in the recommendation of the JPEG-XR standard is following:

```
Toddodd(iCoeff[ ]) {
 iCoeff[1] = –iCoeff[1]
 iCoeff[2] = –iCoeff[2]
 iCoeff[3] += iCoeff[0]
 iCoeff[2] −= iCoeff[1]
 iCoeff[0] −= (valT1 = iCoeff[3] >> 1)
 iCoeff[1] += (valT2 = iCoeff[2] >> 1)
 iCoeff[0] += ((iCoeff[1] * 3 + 4) >> 3)
 iCoeff[1] −= ((iCoeff[0] * 3 + 3) >> 2)
```

*iCoeff[0] += ((iCoeff[1] * 3 + 3) >> 3)*
iCoeff[1] −= valT2
iCoeff[0] += valT1
iCoeff[2] += iCoeff[1]
iCoeff[3] −= iCoeff[0]}

Output values are received after 7 clock steps in case of the implemented block of Toddodd in accordance with the JPEG-XR standard.

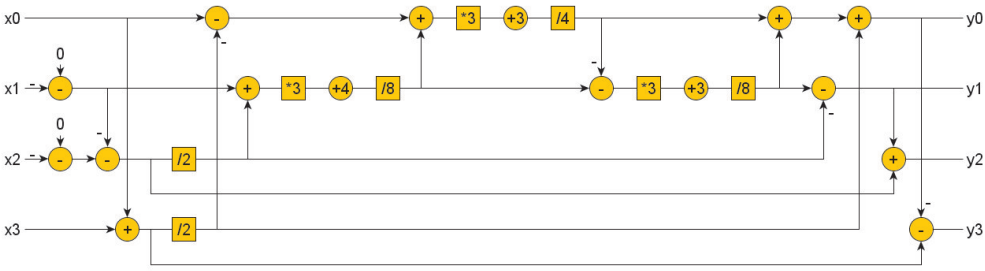

Fig. 9. Graphical diagram of data flow in Toddodd

Figure 9 shows a graphical diagram of data flow in Toddodd.

This Toddodd block uses 123 slice blocks of V5LX110, therein 356 Flip-Flop blocks and 331 LUT blocks. We can use only 4 clock steps to calculate all output values for Toddodd transform like in case of the Todd transform. A pseudocode for this solution is following:

Toddodd(iCoeff[ ]) {
valT1 = iCoeff[3]
iCoeff[3] += iCoeff[0]
valT2 = iCoeff[2]
valT3 = iCoeff[2] >> 1
iCoeff[2] = iCoeff[1] − iCoeff[2]
*iCoeff[0] = (iCoeff[0]*8 − valT1*8 − iCoeff[1]*3 − valT2*3 + 8)>>4*
*iCoeff[1] −= ((iCoeff[0] * 3 + 3) >> 2)*
*iCoeff[0] += ((iCoeff[1] * 3 + 3) >> 3)*
iCoeff[1] −= valT3
valT4 = iCoeff[0]
iCoeff[0] += iCoeff[3]>>1
iCoeff[2] += iCoeff[1]
iCoeff[3] = (iCoeff[3]>>1) − valT4 }

An execution of operation from first six lines of the pseudocode of the implemented Toddodd transform uses only one clock step. Operations from next two lines require two clock steps. Operations from the last five lines of the pseudocode require only one clock

step for execution. Results are received after 4 clock steps. Designed Toddodd block uses 103 slice blocks of Xilinx V5LX110 (therein 224 Flip-Flop blocks and 317 LUT blocks).

The block of the first stage of FCT in accordance with JPEG-XR standard recommendation uses above–mentioned transforms. This block consists of four T2×2h transform blocks. The block of the first stage of the Forward Core Transform implemented in V5LX110 uses 318 slice blocks (therein 828 Flip-Flop blocks and 904 LUT blocks) with 16-bit input data. This block can operate at a frequency of 374 MHz. The attained throughput of the first stage FCT is 16 pixels per clock step.

The implementation of the second stage of FCT uses more resources than the first stage of FCT. The implementation of the second stage uses 433 slice blocks (2% resources of V5LX110) therein 1065 Flip-Flop blocks and 1309 LUT blocks. The second stage block of FCT transform can operate at a frequency of 254 MHz.

4. Summary

The advantage of implemented blocks of the first and the second stage in accordance with the recommendation of the JPEG-XR standard is possibility of simultaneous processing of 16 input values. The processing of a single block of the macroblock takes 4 clock steps in the first stage of the Forward Core Transform. The processing of DC and LP coefficients received from the first stage of FCT in the second stage of FCT takes also four clocks steps. Subsequent output values can appear in each clock step in cases of both stages. Modifications in pseudocodes defined by the ITU-T T.832 recommendation don't result in discrepancies between output values received from implemented blocks and output values received from original pseudocodes.

Table 1
Implementation parameters of FCT transformations compatible with JPEG-XR standard

	FCT stage	
	1st	2nd
SLICE	318	433
LUT	904	1309
FF	828	1065
Frequency [MHz]	374	254
Power [W]	1,123 (leakage 1,109)	1,125 (leakage 1,109)

The implementation of the first stage of the Forward Core Transform uses 318 slice blocks in Xilinx V5LX110 device. The implementation of the second stage of FCT takes 433 slice blocks. Other parameters of implementation are presented in Table 1.

Acknowledgements

A research financed from NCBiR financial means as part of Synat project (SP/I/1/77065/10)

References

[1] ITU-T T.832 – Information technology – JPEG-XR image coding system – Image coding specification.
[2] Guanghui Ren, Xiaokai Hu, Gangyi Wang, Jinglong Wu, *An Optimized Pipelined Architecture of LBT for JPEG-XR Encoding*. Information Technology Journal, 9 (7), 2010, 1376–1382.
[3] Perra C., Giusto D., *An image browsing application based on JPEG-XR*. CBMI 2008, 396–401.
[4] Dufaux F., Sullivan G.J., Ebrahimi T., *The JPEG XR Image Coding Standard*. IEEE Signal Processing Magazine, Nov. 2009, 195–199, 204.
[5] Ching-Yen Chien, Sheng-Chieh Huang, Chia-Ho Pan, Ce-Min Fang, Liang-Gee Chen, *Pipelined Arithmetic Encoder Design for Lossless JPEG XR Encoder*. ISCE2009, 144–147.
[6] Maalouf A., Larabi M.-Ch., *Enhancing the Intra-Prediction in JPEG-XR by Using Edge Information*. Proc. of 2009 Fifth International Conference on Signal Image Technology and Internet Based Systems, 138–143.
[7] Dąbrowska A., Wiatr K., *Modyfikacja algorytmów transformacji FDCT dla potrzeb kompresji obrazów implementowanej w układach FPGA*. Kwartalnik Elektroniki i Telekomunikacji, 2006, t. 52, z. 4, 629–641.