14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

RECONFIGURABLE ARCHITECTURE OF AVC/H.264 INTEGER TRANSFORM

Adam Łuczak and Marta Stępniewska

Division of Multimedia Telecommunications and Radioelectronics
Poznan University of Technology
Piotrowo 3a, 60-965 Poznań, Poland

phone: + (48) 61 6652071, fax: +(48) 61 6652292, email: {aluczak, mstep}@multimedia.edu.pl
web: www.multimedia.edu.pl

ABSTRACT

The paper presents an original reconfigurable architecture of inverse integer transformation for H.264/AVC decoder. Proposed design can perform integer 4x4, 8x8 and Hadamard inverse transform including inverse quantization process as well. The design exploits pipelined architecture and supports FPGA devices. Simulation result indicates that proposed structure is characterized by low implementation cost and high efficiency. Final synthesis and test has been made for Xilinx Virtex family devices.

1. INTRODUCTION

H.264/AVC is the newest and the most efficient video compression standard which offers wide compression tool set and advanced algorithms of stream encoding. It was designed as a solution in area of broadcast services, transmission in local networks, streaming services etc [1]. Compression tool set includes weighted prediction, quarter-sample accurate motion compensation, flexible scaling algorithm and efficient small block integer transform.

The first issue of AVC standard [2] introduced integer 4x4, and hadarmard transform only, and user-defined weight scale was unavailable. The latest issue (ver. 2.0 of AVC/H.264 standard) introduces new tools that are useful in high resolution profiles/levels (e.g. HD-TV). Among new tools there are user-defined weight scale matrix and 8x8 integer transform. The mentioned tools support efficient compression of high resolution video streams and can be used in profiles known as Fidelity Range Extensions (commonly referred as FRExt). Unfortunately new features increase coder/decoder complexity, especially its hardware implementations.

Therefore in order to cover all profiles and levels without to high hardware overhead new design of reconfigurable inverse transformation is needed.

2. IMPLEMENTAION GOALS

AVC transform implementation issues have been discussed in [4][5]. Also in [6] the 2-bit serial architecture for 4x4 integer transform has been proposed. Given solutions are appropriate for Baseline profile decoder that should be able to make only inverse integer transform and Hadamard transform. For the highest profiles and levels more efficient design is needed. Moreover, new design should support ASIC and especially FPGA devices. It means that proposed architecture should utilize efficiently available memories (2KB blocks for Xilinx FPGA family), uses only simple logic functions (available 4 input LUTs only) and design should use narrow busses and exploits local connections mostly.

Efficiency of the created structure should be adjusted to other modules needs. Implemented circuit should not work too fast because it generates tie-ups. Such wait states leads to conclusion that the considered structure should be smaller and slower. On the other hand, it should be fast enough to produce in real-time luminance and chrominance samples even for high resolution image sequences.

Therefore, main goal is to implement reconfigurable design as smaller (in logic area sense) as possible with enough efficiency to process transform coefficients in real time without any wait states.

3. INVERSE QUANTIZATION

H.264/AVC standard (version 3.0) introduces flexible and efficient scaling algorithm. The scaling formula can be defined in general as follows:

$$sample = \left(f_{i,j} \cdot W_{i,j} \cdot N_{qp\_rem,i,j}\right) << (qp\_per - QS), \quad (1)$$

where:

- i, j – sample position,
- $f_{i,j}$ – transform coefficient,
- $W_{i,j}$ – optionally user-defined weight scale,
- $N_{qp\_rem,i,j}$ – norm adjust – defined by AVC standard,
- qp_rem – reminder of quantization parameter division by 6,
- qp_per - result of quantization parameter division by 6,
- QS – constant dependent on transform kind (is equal 4 or 6).

Scaling factors $N_{qp\_rem,i,j}$ are constant and are defined by H.264/AVC standard, for each transform independently. The weight scale matrix can be defined by user in FRExt profiles only. For other profiles it is implicitly equal to 16. The scaling formula basically stays the same for different block sizes (4x4 andd 8x8). The differences come from sample position in scaled block that can be 4x4 or 8x8 size. This requires different scaling tables marked as $W_{i,j}$ and $N_{qp\_rem,i,j}$, that are the same size as scaled block.

4. INVERSE INTEGER TRANSFORMATION

AVC transforms were designed as easy to implement in software and especially in hardware. Integer transform is an DCT approximation and has similar efficiency in compression applications. At first the 4x4 transform has been introduced (AVC/H.264 ver. 1.0) and then to increase compression in high resolution video sequences 8x8 transform has been added. AVC transforms are used in intra-frame and inter-frame prediction mode. Division operations are rounded down (floor operation) that the problems with rounding have been eliminated. Moreover, 16-bit accuracy of arithmetic is enough to perform all calculations. Introduced integer transform allows using only shifters and adders instead of multiplications.

$$DCT_{4x4} = \begin{pmatrix} 1 & 1 & 1 & 0.5 \\ 1 & 0.5 & -1 & -1 \\ 1 & -0.5 & -1 & 1 \\ 1 & -1 & 1 & -0.5 \end{pmatrix} \cdot \begin{pmatrix} c_{00} & c_{01} & c_{02} & c_{30} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 0.5 \\ 1 & 0.5 & -1 & -1 \\ 1 & -0.5 & -1 & 1 \\ 1 & -1 & 1 & -0.5 \end{pmatrix}^T \quad (2)$$

Equation 2 shows integer 4x4 inverse transform matrix, it is assumed that all divisions are defined as shift right. Given transform matrices do not contain scaling coefficients which are defined as $N_{qp\_rem,i,j}$ tables in quantization/dequantization process. Such separation of scaling factors simplifies transform matrices greatly.

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} c_{00} & c_{01} & c_{02} & c_{30} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}^T \quad (3)$$

Equation 3 shows Hadamard inverse transform which is used in Intra 16x16 prediction mode as a part of hierarchical transform. To improve compression efficiency in intra prediction mode at plain surface the hierarchical transform has been introduced. At first the 4x4 transform is performed on 16 blocks of the whole macroblock. Then the DC coefficients of each 4x4 transform are gathered into new 4x4 block and Hadamard transform is calculated. Moreover the 4x4 block with DC coefficients is scaled differently than other blocks.

The 8x8 integer transform has been added to AVC standard for better compression of high resolution video sequences. It may be defined as a combination of two separable 1-D 8-point transform computations: horizontally and vertically. Each one can be defined as two 1-D 4-point transforms: first one same as 1-D of 4x4 integer transform and the second one as modified 1-D 4-point integer transform. The equation 4 gives definition of 8x8 transform as a sum of two operations: the first one is performed on odd samples and the latter is a computation carried out on even samples. The results are gathered and combined together according to equation 4 with given matrices at equation 5.

\

$$T_{8x8} = \left[q1 \cdot \begin{pmatrix} 0 & 1 & -1 & 1.5 \\ -1 & -1.5 & 0 & 1 \\ 1 & 0 & 1.5 & 1 \\ -1.5 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & \frac{-1}{4} \\ 0 & 1 & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & -1 & 0 \\ \frac{1}{4} & 1 & 0 & 1 \end{pmatrix} \cdot q1^T + w1 \cdot \begin{pmatrix} 1 & 1 & 1 & 0.5 \\ 1 & 0.5 & -1 & -1 \\ 1 & -0.5 & -1 & 1 \\ 1 & -1 & 1 & -0.5 \end{pmatrix} \cdot w1^T \right] \cdot M \quad (4)$$

Where matrices q, w and M can be defined as follows:

$$q1 = \begin{pmatrix} 0&0&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \\ 0&0&0&1 \end{pmatrix} \quad w1 = \begin{pmatrix} 1&0&0&0 \\ 0&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{pmatrix} \quad M = \begin{pmatrix} 1&0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&-1&0 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&1&-1&0&0&0 \\ 0&0&0&1&1&0&0&0 \\ 0&0&1&0&0&-1&0&0 \\ 0&1&0&0&0&0&1&0 \\ 1&0&0&0&0&0&0&-1 \end{pmatrix} \quad (5)$$

Calculations performed on even samples are the same as 1-D integer transform (it can be noticed when comparing equation 2 and 4). Odd samples require more calculations to carry out and these operations cannot be described as simple conversion of integer -D integer transform.

Given approach allows decomposing 8x8 transform into two 4x4 transforms and designing reconfigurable device performing calculations for all AVC transforms.

5. PROPOSED ARCHITECTURE

A structure consists of two pipelines marked at figure 2 as top and bottom is proposed. Such decomposition is necessary to calculate 8x8 transform. Each pipeline consists of inverse quantization, horizontal transform, samples mixer, and vertical transform and second sample mixer. Proposed circuit performs computations of scaling and transforms defined in AVC standard in all profiles. Modules have been implemented using parallel architecture using 16 bit arithmetic accuracy.

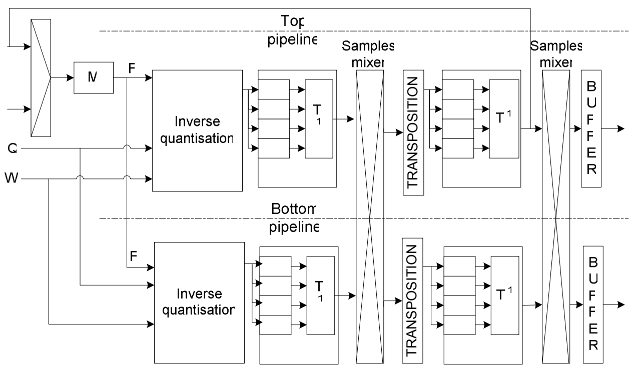

Figure 1 – Proposed architecture of reconfigurable block of integer inverse transformation

a) 4x4 integer transform - The samples appear at the input of inverse quantization and after scaling they are put into 1-D 4-point horizontal transform. In both pipelines the calculations are being made for two 4-point sets at one time. The sample mixer is transparent and samples are rearranged in transposition block. Next the vertical transformation is calculated and samples are written into a buffer.

b) Hadamard transform - The calculations are mostly the same as 4x4 integer transform. The main differences are: the inverse quantization block is transparent, the optional shifts (in 1-D 4 point transform) are disabled and the samples are not written into a sample buffer but into the input memory using feedback loop.

c) 8x8 transform - The 8-point line is put from memory into the inverse quantization module in both pipelines. Then it is scaled and 1-D 8-point transform is calculated by both pipelines together. Top pipeline performs calculations on even samples and the bottom one carries out calculations on odd transform samples. Next samples are added in mixer samples module. After samples rearrangement in transposition module the vertical 8-point transform is calculated and result is written into output buffer.

5.1 Inverse quantization

The inverse quantization process, as shown on equation 1, requires 3 multiplications to obtain the scaled sample. The last multiplication realises variable shift, and allows designing flexible circuit structure. It is also worth mentioning that multipliers are easily synthesized as optimized modules on FPGA devices (e.g. on Xilinx and Altera).

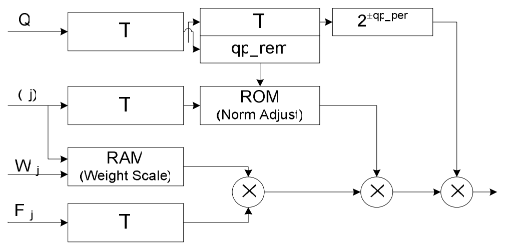

Figure 2 – Proposed structure of inverse quantization module

Quantization parameter is divided by 6 or 4 (transform size dependently), the result of this marked as *qp_per* and the reminder is referred as *qp_rem*. The first value defines size and direction of shift which is performed by the last of multipliers, according to equation 6.

$$y = \begin{cases} x << qp\_per - QS, & qp\_per \geq Q\_THR \\ x >> QS - qp\_per, & qp\_per < Q\_THR \end{cases} \qquad (6)$$

the variable shift is carried out as multiplication:

$$y = \left(x * 2^{qp\_per}\right) >> 6 \qquad (7)$$

The variable shift defined as multiplication by two to the power of *qp_per* shifted by 6 bits. Constant shift does not require any logic: it is an operation made on wires.

Value marked as *qp_rem* is used to index ROM with defined by AVC standard norm adjust values. A user can define own scaling tables putting $W_{i,j}$ values into weight scale memory. As mentioned, weight is implicitly equal 16. Thanks to pipelining no additional control data are necessary to manage the calculations.

5.2 Inverse integer transformation

Figure 3 shows AVC transforms implementation straight from defined in standard formula.

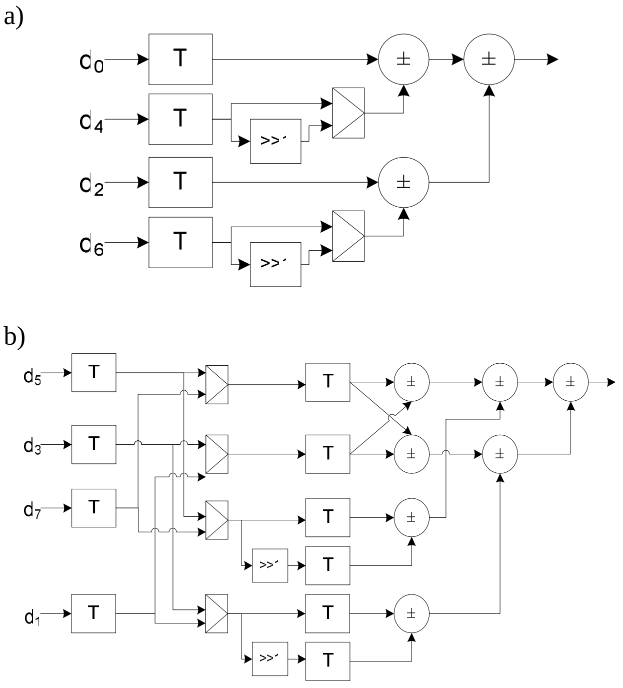

Figure 3 – The computational trees of inverse integer transform, for a) 4 points of 1-D 4-(even)point integer transform and for b) 4 points of 1-D 4-(odd)point integer transform.

Figure 3a) shows integer DCT transform or part of calculations required to perform on 1-D 8-point even transform samples. Figure 3b) shows computation that need to be carried out on 1-D 8-point odd transform samples. Simple (direct) implementation gives complex and big (in area sense) structure.

Therefore, the authors propose original solution of inverse integer transform block. The proposed circuit has been designed using parallel architecture with two dependent and reconfigurable pipelines. The first one is designed to carry out calculations to obtain result of 4x4 integer transform or perform computations on even 8x8 transform samples.

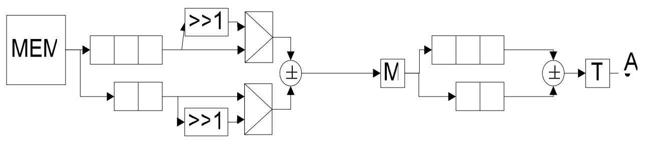

Figure 4 – Structure of computational block of 1-D 4-point inverse integer transform

a) 1-D 4 point integer transform for even 8x8 samples or 4x4 transform – Top pipeline

Transform module implemented in top pipeline is showed on figure 4. The scaling module puts samples into memory. If suitable sample block is fulfilled the calculations may start. Samples are taken from memory one by one and put into one of two shift registers. After loading of all samples the first sum is made and sample is put into small memory to rearrange samples. Next samples are read and put at the input of one of shift registers. After sum, they are buffered in register for one clock tick.

b) 1-D 4 point integer transform for even 8x8 samples or 4x4 transform – Bottom pipeline

The second processing path is similar to the first one. It is designed to carry out calculations both 1-D 4-point and 8-point transform samples. The module has been shown at figure 5. The circuit has two outputs marked as B_1 and B_2. The first one is the output of integer 1-D 4-point transform or the first part of even samples calculations. The B_2 output gives a result of the second part of computation carried out on odd 8 transform samples. The first sum (fig.5 - grey marked part) performs integer multiplication by 1.5.

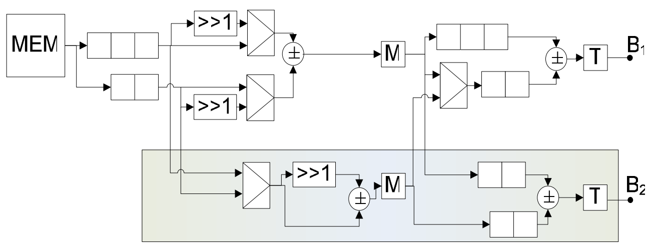

Figure 5 – Structure of computational block of 1-D 4-pel integer transform and odd samples of 8-point integer transform.

c) sample mixer

Samples mixer was introduced in order to finish computations of 1-D 8-point transform. In the case of 1-D 4-point transforms mode (also Hadamard) this module pass the results unchanged to the next block (of vertical transform).

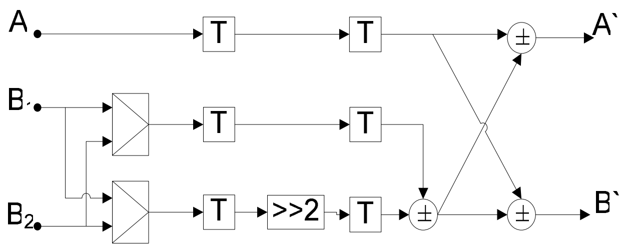

Figure 6 – Samples mixing – 8x8 transform only

The advantage of such a structure is the possibility of performing both 1-D 4-point and 8-point transform using one module. Moreover, circuit arrangement is regular and easy to implement. Transform module produces always two transform samples per clock tick. It is also worth mentioning that marked part of the circuit shown at figure 5 is used only when 1-D 8-point transform is being calculated.

Moreover, management unit for described structure is quite easy to implement. For example, transposition is made by simple bits' rearrangement in sample position without using any logic. Most of the control data, which are passed along the processing path, can be derived form sample position.

6. SIMULATION AND PERFOMANCE RESULTS

The presented architecture has been implemented with Verilog hardware description language. Simulations and test with real data from reference encoder indicate that implemented design works properly.

Figure 7 shows timing analysis of both (top and bottom) pipeline performance.

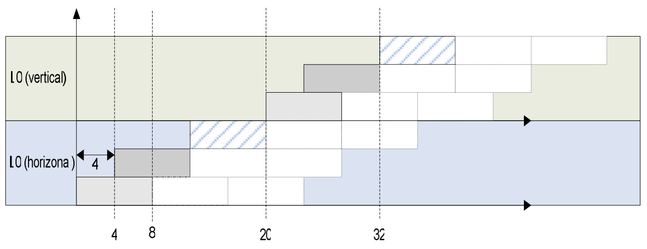

Figure 7 – Timing diagram

Two full inverse transforms are done during 48 clock tics. Samples can be loaded into the pipes without any wait-states. It means that design throughput is 2 samples per clock and delay of result is 48 clock ticks.

7. CONCLUSIONS

Original efficient architecture of (3 types of) inverse integer transform has been proposed.

Proposed architecture is fully pipelined and enables performing computation of AVC integer transforms both for 4:2:0 and 4:2:2 resolution systems, producing 2 samples per clock tick. It means that inverse transformation process can be done with clock frequency close to the sampling frequency (13.5 MHz for SDTV resolution 720x576).

Moreover, resulting structure of inverse transform block is compact and suitable for FPGA devices what was proved by synthesis results.

Efficiency measured by calculating cost factor (working-time/wait-time and utilization of design structure) is about 90% for 4x4 inverse transform (fig.5 greyed part inactive only) and 100% for 8x8 inverse transform. Such result of a device utilizing is excellent.

8. ACKNOWLEDGMENT

The work was supported by the public founds as a research project.

REFERENCES

[1] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra, *Overview of the H.264/avc video coding standard*. IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, pp. 560–576, July 2003.

[2] ISO/IEC 14496 10 *Advanced Video Coding*, 3rd ed., ISO/IEC JTC1/SC29/WG11, Redmond, July 2003.

[3] JVT of ISO/IEC MPEG & ITU-T VCEG *Text of ISO/IEC 14496 10 Advanced Video Coding 3rd Edition,* ISO/IEC JTC1/SC29/WG11, Redmond, July 2004.

[4] A. M. Patiño, M. M. Peiró, F. Ballester, and G. Payá, *2D-DCT on FPGA by polynomial transformation in two dimensions*, ISCAS 2004

[5] R. C. Kordasiewicz, S. Shirani, "*ASIC and FPGA implementations of H.264 DCT AND quantization blocks*", 2005

[6] P. Garstecki and A. Łuczak, *A flexible architecture for image reconstruction in h.264/avc decoders,* in Proc. of ECCTD 2005, Cork,Ireland, August 2005.