High-Performance VLSI Architecture for the Microsoft® HD Photo Image Compression Algorithm

Orlando HERNANDEZ and Graham APGAR
Department Electrical and Computer Engineering, The College of New Jersey
Ewing, New Jersey 08628-0718, USA

ABSTRACT

The Microsoft HD Photo image format attempts to improve on the JPEG standard by providing improvements in image quality, less artifacts, and higher compression ratios. The algorithm uses a hierarchy of spatial-to-frequency transforms, which requires 4 independent operators to be applied serially to the image data. These operators consist of two hierarchical applications of a pre-filtering transform, which attempts to eliminate blocking artifacts when the image is decoded, and two applications of the Photo Core Transform, a Hadamard-based frequency transform which is very computationally efficient. This architecture represents a VLSI implementation suitable for performing the full transform in a high-performance application.

Keywords: Microsoft HD Photo; JPEG; Image Compression Architectures; FPGA Design; Specialized Architectures; VLSI Design

1. INTRODUCTION

Spatial-to-frequency transforms based on Fourier analysis have proven to be one of the most effective methods of image compression. JPEG is perhaps the most well-known for taking advantage of this. In such a compression process, image data is converted into luma-chroma color-space, sub-sampled, and broken into fixed-size blocks. Each block undergoes a 2-dimensional in-place transform, which extracts the frequency components of the data. Lower-frequency components are quantized to reduce the size of the image, as the human eye is less sensitive to this information. In the post-transformed format, the data is in a desirable state for lossless entropy coding, which completes the encoding process. While JPEG has remained the preferred format for lossy compression since the mid-1990s, the standard has changed relatively little since then, and many possible improvements have been identified.

HD Photo Format

Microsoft created the HD Photo format [1], [2] as a proprietary alternative to JPEG, which makes use of many new algorithmic improvements to mitigate JPEG's shortcomings. As with JPEG, the format's images are encoded in blocks of pixels. One common problem with such an approach is that, after lossy compression, the image will show artifacts at the edges of these blocks. HD Photo attempts to reduce these by using a pre-filtering technique, called the Photo Overlap Transform (POT), which operates on the edges between blocks before the core transform occurs. The frequency transform used by the HD encoder, called the Photo Core Transform (PCT) is also unique. It is based on the Hadamard transform, a discrete analog to the Fourier transform. This transform is implemented as a series of operators which require only a small number of mostly trivial operations per pixel, as the PCT coefficients, unlike in Fourier analysis, are always 1 or -1.

Fig. 1 shows the spatial partitioning of image data used for the format. Note that the image width and height, in pixels, must be quantized to be a multiple of 16.

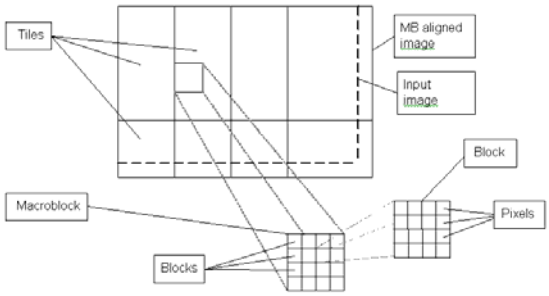

Fig. 1. Image data sub-divisions

Fig. 2 shows the structure of image data in HD Photo format. The raw data in spatial mode is broken into macro blocks, which are groups of 16x16 blocks with each block being 4x4 pixels. For each macro block, image data is organized into a frequency domain DC coefficient representing the average luma-chroma value, a set of 16 low-pass coefficients (1 for each block), and a number of "flex-bits", which are used in entropy coding.

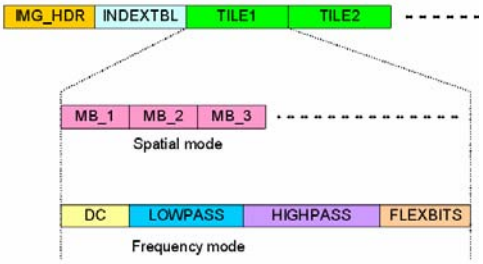

Fig. 2. Structure of HD Photo image data

HD Photo Transforms

HD Photo uses a reversible lapped bi-orthogonal transform [3], which is implemented as the concatenation of the PCT and POT operators. The POT, which is designed to reduce blocking artifacts in highly compressed images, is optional. The transform as a whole is fully reversible, allowing for a perfect reconstruction of an image provided no quantization takes place. Thus, HD Photo is capable of lossy or lossless encoding.

The compression algorithm used in HD Photo is computationally efficient, and is designed for high performance encoding and decoding while minimizing system resource

requirements. The core compression transform requires at most 3 non-trivial (multiply plus addition) and 7 trivial (addition or shift) operations per pixel (with no divisions) at the highest quality level. In the highest performance mode, only one non-trivial and four trivial operations per pixel are required.

The encoding algorithm can be seen in the block diagram of Fig. 3. Note the two concatenated stages. Each box represents an operation performed on a set of 8 or 16 pixels.

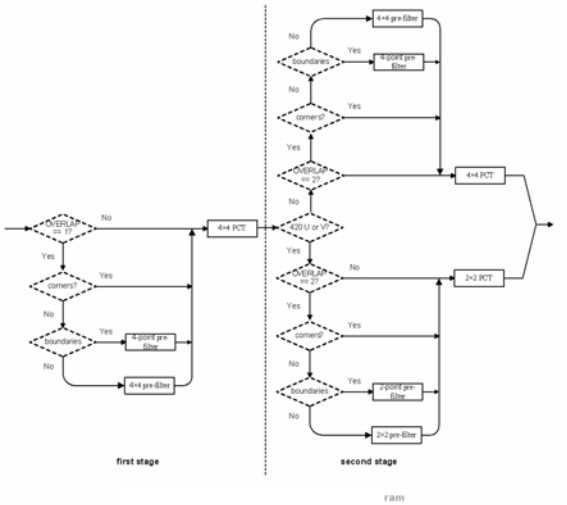

Fig. 3. Flow diagram of encoder transformation algorithm

The fact that the POT and PCT operators do not act on the same data is problematic from an architectural standpoint. The dual nature of the transform yields added complexity to the process, which this architecture attempts to handle efficiently. Complicating things further is the fact that the transformation occurs at two levels; that is, the image is pre-filtered, frequency-transformed, pre-filtered a second time, and then transformed a second time, all in place. At each level, the POT/PCT operates on different data as well. How these complexities affect the proposed architecture will be discussed in the following section.

Several characteristics of the HD Photo transforms make them apt for hardware implementation. First, as discussed earlier, the Hadamard-based PCT makes use of minimal multiplications and no division, using all integer arithmetic. Each addend or coefficient is a small integer requiring a low bit width. In addition, the transform operators are broken into lifting steps, which can be translated to hardware easily. Lifting is a mathematical technique of breaking a transformation, represented by a matrix operator, into a series of steps by factoring the filter. A single lifting step is illustrated below in Fig. 4. Here, a, b, c, and d represent the non-transformed pixels, while A, B, C, and D are the transformed pixels.

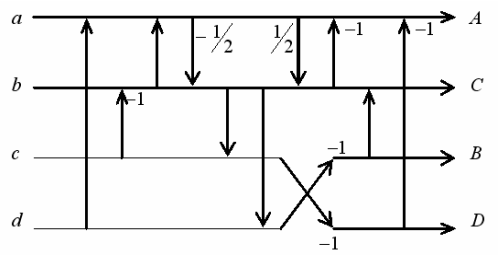

Fig. 4. Example of a lifting step as a signal flow diagram

2. RECENT RELATED WORK

In [4], the authors describe the HD Photo format's capabilities, including supported pixel formats, color formats, lossless/lossy compression (using the same algorithm), etc. The format also allows for progressive decoding, allowing a single part of the bit stream to be decoded to produce a thumbnail of the image, or alternatively allowing easy size reduction of the bit stream after the transformation process.

Malvar [5] introduces several new lapped transforms – the lapped biorthogonal transform (LBT) (used by HD Photo) is of interest, as well as the hierarchical LBT. The other two transforms are aimed at audio encoding and are not of interest. Malvar discusses how low bit-rate encoding of images using block transforms leads to blocking artifacts – discontinuities across block boundaries – because inter-block correlations are not accounted for by the traditional block transform. To alleviate this, the lapped transform basis functions extend beyond block boundaries and decay to near zero at their boundaries. These transforms take into account spatial coherency across block boundaries. However, the transformed coefficients are contained within the size of the block. In other words, the size of the encoded image is not larger, but the encoding process is slightly more complex and involves extra computations. Malvar defines a fast-computable LBT transform as a flow graph and compares the transform coding gain achieved with other transformation approaches.

In this paper [6], the authors present a method for implementing lapped transforms using pre and post-filtering at block boundaries. Through manipulation of LT poly-phase matrix operators, the authors show how the pre-and post operators, together with a core transform (in this case a DCT) effectively form a lapped transform. The advantage of this approach is that the lapped transform exploitation of inter-block correlation may be implemented as independent processes outside the core transform (a key feature of HD Photo). In addition, the authors discuss how post-filtering along block boundaries can reduce blocking and ringing artifacts.

3. PROPOSED ARCHITECTURE

The proposed architecture makes use of three pipelines to achieve the necessary first-level transformations and three combinational logic blocks to perform second-level transformations. Each pipeline stage or combinational block performs part of the transformation using arithmetic derived from signal flow diagrams such as the one in Fig. 4.

The architecture also uses an efficient approach to caching. When a transform is finished, the data is not written back to the main memory, but instead stored in a temporary cache for fast access by the following pipeline.

A block diagram of the system can be seen in Fig. 5. The top module is interfaced with the main memory, and contains a main controller and input and output routers for each group of pipelines and combinational blocks. The main controller contains a sub-unit, which calculates constants required for controlling the routers. The router modules serve to route information to and from their respective caches, memories, and operational blocks. The PCT level 1 transform is achieved using three parallel pipelines, one for groups of center pixels and two for edges. Between each set of pipelines or routers, there is a cache module, which serves as a buffer between operations.

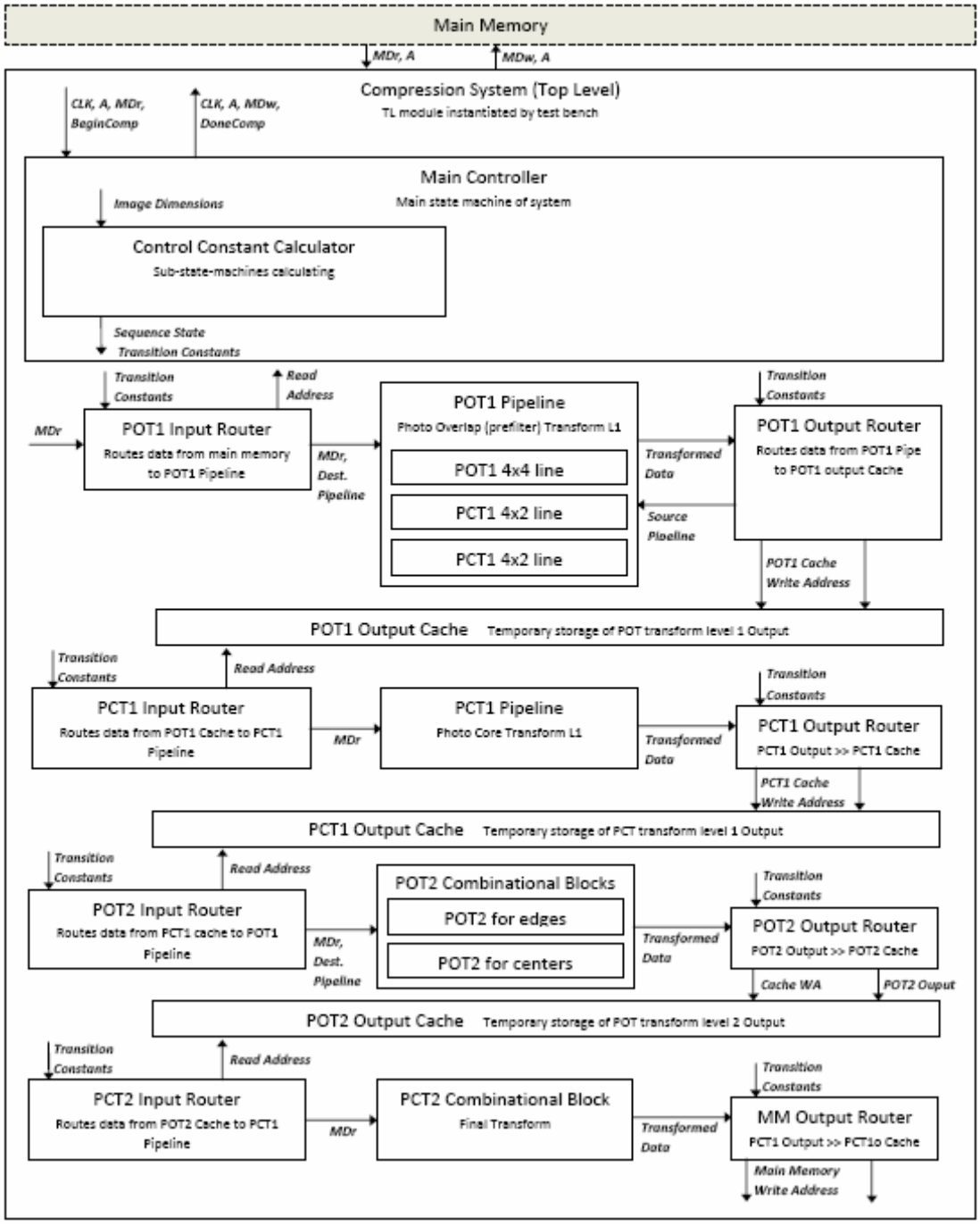

Fig. 5. Block diagram of system

4. ARCHITECTURAL DETAILS

This section discusses the contents and function of each module in detail. Where necessary, schematics and Verilog code are presented.

Main Controller

The main controller handles only the basic state of the system. There are only three such states, one for initialization following a request for transformation, one for activating the components during a transformation, and an 'off' state. During initialization, the routers are reset and a group of calculations is performed using the control-constant calculator. The calculated data includes the quantized width and height of the image – the dimensions of the image quantized to align with 16x16 macro blocks, and the number of horizontal and vertical 4x4 blocks in the image data.

The resulting values are dubbed control-constants, as they are loaded into each router at the start of a transformation. They affect the transitional logic of the state machines within the routers, which will be discussed later.

Input/Output Routers

The IO Routers in the system perform the task of calculating read and write memory addresses, reading and writing from memory, a pipeline, or a cache. The sequence of addresses necessary for each router is unique and is determined by the dimensions of the image. In order to understand these sequences, an image must be considered as it is broken into blocks, and examined by the ordering of operations performed block-by-block. Fig. 6 shows a hypothetical set of image data composed of a single macro block containing 16 blocks. Pixels are labeled according to their row and column in hexadecimal. To the right and bottom, it can be seen that the quantized image over-extends the real edges.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	CD	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B1	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

Fig. 6. Example of image data across a macro block

The level 1 POT is performed on groups of 4x4 pixels straddling blocks or on 2x4 groups at the four edges of the image. Thus the pixels {22, 23, 24, 25, 32, 33, 34, 35, 42, 43, 44, 45, , 52, 53, 54, 55} would be subject to the POT4x4 operator. The blocks themselves represent the domain of the level 1 PCT transform. After the first level transformation, the macro block will look as shown in Fig. 7. At this point, the HP (high pass) coefficients have been calculated. In order to compute the LP coefficients, we must do a second level transform, which acts on only the upper-left coefficients of each block. Finally, the DC coefficient of the entire macro block is designated as the upper-left coefficient of the LP block.

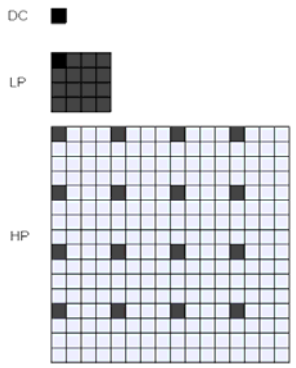

Fig. 7. Frequency coefficients of first and second level transforms

Consider the necessary function of the POT level 1 input router. This module must fetch all data to be transformed by the POT operator from the main memory where the image resides. In this pipelined, approach, 16 pixels are fetched at once, sending them into the appropriate POT1 pipeline. Starting with the top edge of the image, refer to the example in Fig. 7. For edges, the necessary transformation operator is the POT2x4, which will act first on the group {02, 03, 04, 05, 12, 13, 14, 15}. However, since the POT1 pipeline has a bandwidth of 16 pixels, two of these can be performed in parallel. Thus, the next top-edge group, {06, 07, 08, 09, 16, 17, 18, 19} can be fetched. The two groups are transformed in parallel in the two POT1 2x4 pipelines shown in Fig. 5. In the following cycles, the process continues moving rightwards and fetching the next two top-edge groups until the right edge of the image is reached. Next, the left and right edge of the image, group {20, 30, 40, 50, 21, 31, 41, 51, 2C, 3C, 4C, 5C, CD, 3D, 4D, 5D}, is transformed. After this, the inner-pixels shared by the first and second row of blocks, beginning with {22, 23, 24, 25, 32, 33, 34, 35, 42, 43, 44, 45, 52, 53, 54, 55}, are transformed, and the process continues moving right 1 block per cycle. This sequence of left-right edge and center is repeated until the bottom of the image is reached. At this point, as with the top, the two POT2x4 transforms are performed in parallel from left to right, which completes the pre-filtering stage of the transform.

To implement this module, it must be taken into account that the 16 addresses fetched per cycle are not adjacent and they each must acquired individually. Thus, this router, as with all routing modules in the design, operates on a x16 clock, filling its output buffer between main clock cycles. With this in mind, consider that for each sub-cycle, a read address RA to the main memory will be provided, and a write address WA to the POT1 pipeline. At the beginning of each 1/16th clock cycle, the current read address is transformed - that is, a memory offset is added to it to bring it to the next necessary location. There are a limited number of these transformations, and they are dependent on the dimensions of the image. This is why they are calculated at the start of the transformation process.

Because the process is regular, each block or half-block will always use the same sequence of transformations. Thus, the data acquisition process can be divided into a number of finite states or sequence states.

The size of the image can be a variable and the system's routers can easily adjust its state machines to read and write the correct data between memories and transform logic.

Now, consider the output router of the POT1 pipeline as seen in Fig. 5. This module is responsible for taking data from one of two pipeline sources and writing it to the POT1 output cache. Because cache is being written, there are certain points of flexibility available in the design. If memory were being written back directly, the memory would have to be shared with the input router, which would be simultaneously trying to read while a write was taking place. This is not an issue in this design, so similarly to the input router; the output router operates on a x16 clock, writing 16 pixel values per cycle to the cache.

A second point of flexibility in this design is that the transformed data can be organized in any way within the cache. Intuitively, it makes sense to organize it in a fashion that will facilitate easy retrieval for the PCT1 input router. Since the PCT is performed always on one block at a time, it was chosen to organize the POT1 cache in blocks. This organization is shown in Fig. 8.

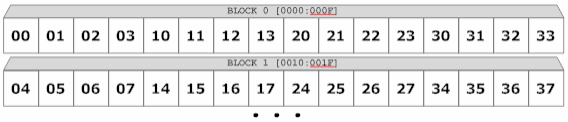

Fig. 8. Organization of POT L1 Cache

Where blocks in the original image data were non-continuous, as they existed across two dimensions, here they have been linearized, so that the following PCT operation can occur with minimal complexity in the routing. However, there is some extra complexity involved in how this cache must be filled. For instance, after the first two top-edge POTs occur, the outgoing

data belongs to several different blocks. The transformed pixels 02, 03, 12, 13 must be written to the block 0 space, addresses 2, 3, 6, 7 respectively. This can be seen in the pixel labels in Fig. 6. If the pre-filtering of the whole top edge yields 4 coefficients of blocks 0 and 3, and 8 coefficients of blocks 1 and 2, then pixels {00, 01, 10, 11} and {0E, 0F, 1E, 1F} are never passed through the pre-filter pipeline and thus must be directly written from memory to the POT1 cache. Eventually, as the pre-filtering operation is continued, entire blocks will become "filled in" in the cache. When this occurs, the PCT level 1 input router can begin fetching the data in these blocks and passing them through the PCT1 pipeline. Once a block enters the PCT pipeline, the data in that section of cache can be overwritten. This will be discussed later.

The remaining routing blocks work in a similar fashion, with the final block writing data back to its original location in the main memory.

Operational Pipeline Blocks

Now let us look at the pipelines performing the operations. Each stage of the PCT and POT pipelines is a lifting step based on signal flow diagrams such as the one in Fig. 4. That step, for example, occurs in the first pipeline stage of the POT L1 4x4 pipeline. Four of these circuits operate in parallel to transform all 16 pixels, completing the stage.

Architectural diagrams for the POT and PCT pipelines can be found in Fig. 9 and Fig. 10. The second level transforms, POT2 and PCT2, are not broken into lifting steps and thus they must be implemented as combinational blocks. To ensure that these do not lengthen the critical path time of the system, the routing blocks at the input and output of these modules send data through only once per 2 clock cycles.

Cache Size

The fact that it the POT and PCT require the image to be broken up in separate ways means that caching residual values from these operations requires more space. The first PCT block cannot be fetched until the top and left-right states of the POT have been completed. It is only when the cache receives the output of the first POT4x4 center transform that the PCT pipeline can begin operation. Once a block enters the PCT pipeline, the data in that section of cache can be overwritten. What this all implies is that the width of the image determines the amount of cache space necessary. Specifically, the cache size must be

$$S_{cache} = \left(W_{iQuant} \times 4 + 32\right) \times bitdepth. \quad (1)$$

Where W_{iQuant} is the width of the image quantized to the macro block. Thus, a 256x256 image with a bit depth of eight would require 1056 Bytes of cache, which is 1.6% of the total image data size.

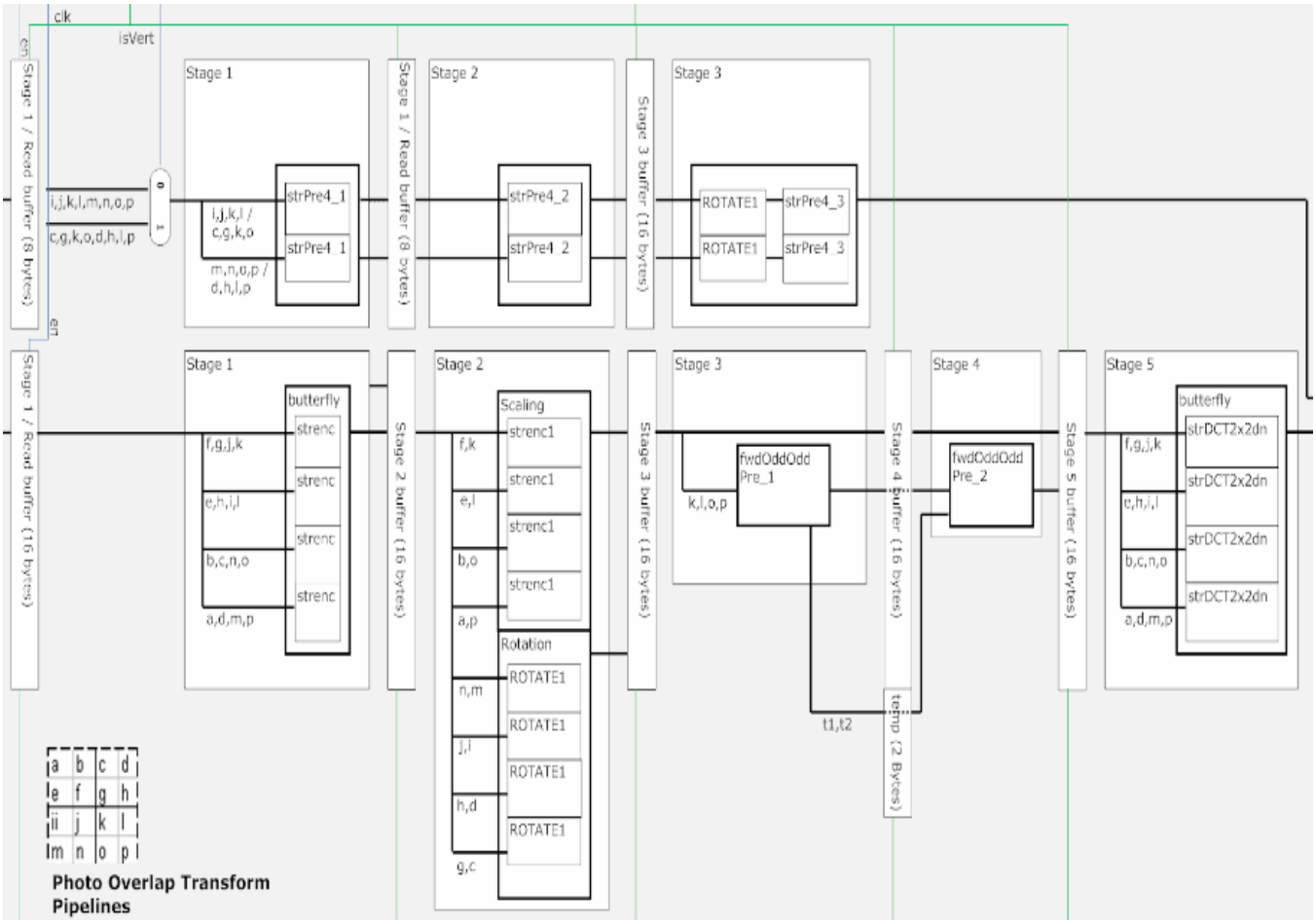

Fig. 9. Photo Overlap Transform Pipeline Block Diagram

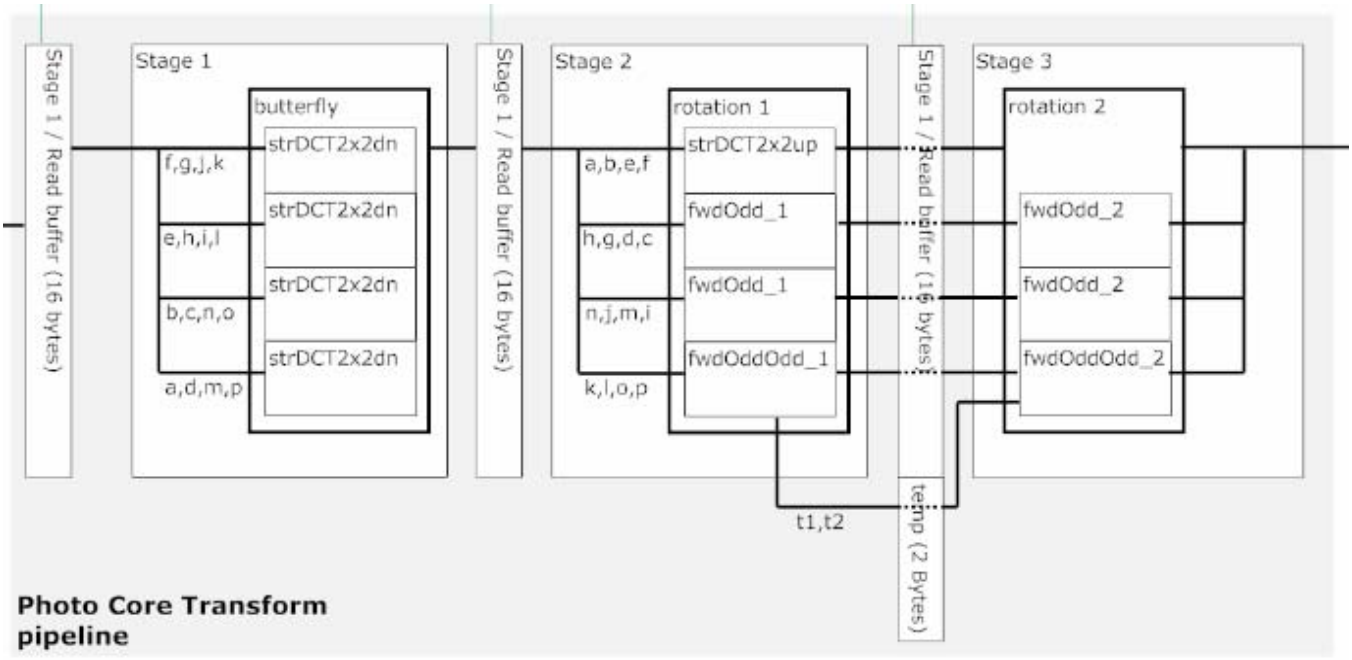

Fig. 10. Photo Core Transform Pipeline Block Diagram

5. CONCLUSIONS

The architecture was implemented for a Xilinx FPGA and simulated behaviorally. The results were compared against an open-source Microsoft HD Photo sample encoder and were verified to be consistent. The architecture's pipelined approach dramatically increases the throughput over the software implementation, and parallelization of lifting steps in the pipelines further improves speed. Finally, caches with independent controllers prevent wasteful read/write cycles to and from the main memory. The added complexity of the caches requires more overhead but also offers more flexibility in terms of how data is structured within the system.

This architecture, while a suitable approach to the lapped bi-orthogonal 2-stage transform used in HD Photo, does not perform all encoding steps. Color conversion, sub sampling entropy-coding are all well-suited to hardware implementation as well, so the proposed architecture could be used as part of a larger VLSI system for full encoding and decoding. These issues are part of our ongoing research, where the tradeoffs of entropy coding via hardware or software executed by an embedded processor in a Hardware/Software co design approach are investigated.

6. REFERENCES

[1] S. Srinivasan, C. Tu, Zhi Zhou, D. Ray, S. Regunathan and G. J. Sullivan (Microsoft Corporation), **An Introduction to the HD Photo Technical Design**, JPEG document WG1 N4183.

[2] Microsoft Corp., **HD Photo Specification and HD Photo Device Porting Kit 1.0**, available at http://www.microsoft.com/windows/windowsmedia/forpros/wmphoto/, Dec. 2006.

[3] H. S. Malvar and D. H. Staelin, **Reduction of Blocking Effects in Image Coding with a Lapped Orthogonal Transform**, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1988), New York, NY, Apr. 1988, pp. 781-784.

[4] S. Srinivasan, C. Tu, S. L. Regunathan, R. A. Rossi, Jr., and G. J. Sullivan, **HD Photo: A New Image Coding Technology for Digital Photography**, Applications of Digital Image Processing, Proceedings of SPIE, vol. 6696, San Diego, CA USA (August 2007).

[5] H. S. Malvar, **Biorthogonal and Nonuniform Lapped Transforms for Transform Coding with Reduced Blocking and Ringing Artifacts**, IEEE Trans. Signal Processing, Apr. 1998, pp. 1043-1053.

[6] T. D. Tran, J. Liang, and C. Tu, **Lapped Transform Via Time-Domain Pre- and Post- Filtering**, IEEE Trans. on Signal Processing, June 2003, pp. 1557-1571.