A High Speed Shift-Invariant Wavelet Transform Chip for Video Compression

Henry Y.H. Chuang, David P. Birch, Li-Chang Liu, Jong-Chih Chien, Steve P. Levitan, and C. C. Li
Departments of Computer Science and Electrical Engineering, University of Pittsburgh
chuang@cs.pitt.edu,dpbst18@pitt.edu ,{lilst4,jocst4}@pitt.edu,steve@ee.pitt.edu,ccl@ee.pitt.edu

Abstract

Wavelet-based video compression can provide improved codec and bit rates. The shift-variance problem of the discrete wavelet transform on image sequences, however, may cause large errors in motion estimation in the wavelet domain and thus degrade its performance of video compression. To make the wavelet transform of an image shift-invariant, a large amount of additional computation is required. For this reason a high-speed hardware implementation is necessary. This paper is focused on the VLSI design and implementation of a highly parallel shift-invariant wavelet transform chip. The RTL design, synthesis, simulations, and layout have been completed, and OKI Semiconductor, Inc. is fabricating the chip.

1. Introduction

The wavelet transform is well known for providing highly efficient data compression, as manifested by its adaptation in the JPEG 2000 standard. For video compression applications, one option is to apply the wavelet transform to each frame of a video sequence yielding multiresolution subband images, and then utilize motion estimation and compensation in the wavelet domain for encoding. However, the discrete wavelet transform (DWT) is subject to shift variance due to the dyadic downsampling used in its successive decompositions, i.e., the discrete wavelet transform of a displaced object is different from its wavelet transform before the displacement. This may cause large errors in estimating the object's motion vectors by block matching. Chui and his colleagues [1] devised a means to achieve shift invariance by using a minimal (and relatively prime) rate of a spline interpolation (n=3) which gives two interpolated points between a pair of successive data points in the 1-dimensional case. Three data channels are organized for their individual wavelet transforms with specific input spatial relations at each level of the transformation and output placements; their combination gives an almost shift-invariant wavelet transform. For 2-dimensional images, the above process can be performed horizontally first, and then the result is similarly processed vertically. This has been used in studies of video compression [2]. But the interpolation causes a nine-fold increase in the amount of computation and storage for the output. Extending their idea, we reformulate the problem without interpolation by sampling the 1-dimensional data into 3 data channels under the hypothesis that point 2 (assigned to channel 2) and point 3 (assigned to channel 3) are approximately spline interpolations of their two neighboring points (point 1 and point 4) that are assigned to channel 1. For the 2-dimensional images, we consider the contiguous 3x3 blocks of data as being made up of 9 channels of data under the same hypothesis as stated above, both horizontally and vertically. In doing so, an almost shift-invariant wavelet transform (SWT) can also be obtained.

The SWT is very computation intensive. Due to high degree of regularity in the DWT algorithms, many VLSI parallel processors have been proposed to speed up the computation [3,4,5,6]; most of them use systolic array architectures. We have designed a highly parallel and pipelined SWT chip for wavelet based video compression. This paper is focused on its VLSI design and implementation. After explaining the shift-invariant wavelet transform in Section 2, we describe the architecture of the chip in Section 3. The major implementation issues are discussed in Section 4.

2. Shift-Invariant Wavelet Transform (SWT)

Let us consider the first row of an image, the data points are assigned to Channel 1, Channel 2, and Channel 3 in succession repeatedly, as shown schematically in Figure 1. They are designated by $\{ c_0^1(m) \}$, $\{ c_0^2(m) \}$, and $\{ c_0^3(m) \}$ respectively, where the superscript denotes the channel number and m denotes the data index. Channel 1 data undergo the regular discrete wavelet transform at all resolution levels j:

$$c_j^1(m) = \sum_k \tilde{h}(k) c_{j-1}^1(k-2m) \qquad d_j^1(m) = \sum_k \tilde{g}(k) c_{j-1}^1(k-2m)$$

Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'02)
0-7695-1486-3/02 $17.00 © 2002 IEEE

where $c_j^1(m)$'s denote scaling coefficients and $d_j^1(m)$'s wavelet coefficients. Channel 3 data undergo the regular transform to the decomposition level 1:

$$c_1^3(m) = \sum_k \tilde{h}(k) c_0^3(k-2m) \qquad d_1^3(m) = \sum_k \tilde{g}(k) c_0^3(k-2m)$$

Channel 2 data sequence needs an advance by one unit, then undergoes the regular wavelet transform to level 1:

$$c_1^2(m) = \sum_k \tilde{h}(k) c_0^2(k-2m-1) \quad d_1^2(m) = \sum_k \tilde{g}(k) c_0^2(k-2m-1)$$

With the initial sample being skipped, its output sampling will be, in effect, delayed by two units; this results in its output samples being placed respectively after each point of Channel 3 when three channel outputs are combined at level 1. So, the roles of Channel 2 and Channel 3 are interchanged in the wavelet decomposition from level 1 to level 2, and the role interchange repeats in successive levels of decomposition.

In the second row of the image shown in Figure 2, the data are assigned to channels 4, 5, and 6 in succession and repeatly; the third row data are assigned to channels 7, 8, and 9 in similar fashion. Channel 2, Channel 5, and Channel 8 data need to be advanced row-wise in the horizontal processing to level 1. For vertical processing to level 1, Channels 4, 5, and 6 need to be advanced column-wise. Then the role of Channels 4, 5, and 6 will be interchanged with the role of Channels 7, 8, and 9, and so forth. In the next three rows, the same process described above is repeated to complete the computation for the shift-invariant wavelet transform.

We choose the biorthogonal wavelet BIOR(2,2) in our VLSI design because it has short support and its symmetric filter coefficients have dyadic rational values enabling the simple arithmetic computation of inner-products. The low-pass decomposition filter $\tilde{h}(k)$ has five coefficients (1/8) {-1, 2, 6, 2, -1}, and the high-pass filter $\tilde{g}(k)$ has three coefficients (1/2) {-1, 2, -1}. We consider three levels of wavelet decompositions.

To smooth the boundary processing in decomposition, data extension is provided by symmetric flip-over at the image borders on the left, right, top and bottom. Since the longer filter has five coefficients, a flip-over of four points at each border at each level of decomposition results in sixteen points of flip-over at each border of a row (column) of the original channel data. If the filter index starts from zero instead of a negative number, the left boundary and the top boundary will need fewer points of flip-over. However, to take care of the effect of the delayed sampling that is used at certain stages in every channel except Channel 1 in order to obtain the complete reconstruction at the beginning of a row or a column, an additional 2-point flip-over is used at the left side and on the top. Overall, we use 16-point flip-over on the left and on the top, and 20-point flip-over on the right and on the bottom. Furthermore, to isolate the trailing edge effect in systolic processing between rows at each decomposition level, twelve zero-paddings are added to the end of each row of the original channel data.

The above gives a general description of the logic and organization of the SWT as well as the need of data preparations for decomposition. The VLSI architecture will be presented in the next section.

3. Architecture

The SWT chip is intended to be the front-end processor for a wavelet-based MPEG-4 video compression system. Figure 3 shows the system block diagram, with only the portion involving the SWT, motion estimation and prediction error computations shown in detail. Figure 4 shows the architecture of a three-channel SWT chip, where each of the three channels is a linear pipeline. The on-board processor RISC PE controls the operation of the chip. In every pipeline cycle, the PE fetches a set of pixels from RAM1 for each channel and initiates the cycle that advances computation in the pipeline of every channel by one stage. While the logic computation inside the chip is synchronous with an external clock, the pipeline operation is asynchronous for easier control. The Cycle Ready signal from the chip provides the necessary handshaking. The on-chip DMA controllers output the results of the wavelet transform to RAM2 cycle by cycle.

Although all 9 channels of SWT can be on a chip, we have implemented only 3 channels on the chip due partially to our gate count limitation. To compute a shift-invariant wavelet transform, it is necessary to use the chip 3 times or to operate 3 chips in parallel. However, because the computation time and the IO time of the 3-channel chip are nearly equal based on contemporary RAM speeds, the whole SWT computation using a single 3-channel chip would only be slightly slower than that on a 9-channel chip. Because a 9-channel chip would produce 3 times more output data and require the PE to input 3 times more input data per cycle, the IO time of the 9-channel chip would be 3 times more than that of the 3 channel chip. Since all channels compute in parallel on their individual linear pipelines, the IO and the computation in the 9-channel chip would be grossly unbalanced. Of course, as faster RAM's become available it would be more advantageous to build a 9-channel chip. The 3-bit chip-select signal from the PE and the chip select controller in the chip determine which 3 channels the chip operates.

Each channel computes 3 levels of DWT, and a row processor followed by a column processor computes one level of transformation. The differences in row-wise

advances among the channels are realized by registers between levels, and the differences in column-wise advances are realized by shifting the filter coefficients in the column processors. The row processors in the three levels use the same linear systolic array architecture shown in Figure 5. However, since data rates are reduced by one-half row-wise and column-wise in successive levels, the number of systolic arrays (SA) employed and the systolic cycle frequencies are also halved in successive levels. The first level systolic cycle is also the pipeline cycle. Each stage of the linear systolic array compute the MAC (multiplication and accumulation) of two pixels (c_0,c_1) and two high-pass coefficients (g_0,g_1) as well as two low-pass coefficients (h_0,h_1). Since the longer filter of BIOR (2,2) has 5 coefficients, the systolic array has 3 cells consisting 3 stages of the pipeline. The pair of pixels and the pair of partial sums L (low-pass) and H (high-pass) propagate downstream, with the pair of pixel streams moving at one half the speed of the pair of partial sum streams (realized by the two registers $\bar{c}_0, \bar{c}_1$ acting as one unit delays). Each column processor, as a stage of the pipeline, is composed of 4 parallel arrays; two Hc's for the low-pass filter and two Gc's for the high-pass filter. The Hc array of the first level is shown in Figure 6. The adders accumulate sums of products, which are either stored in shift register buffers to be retrieved and added to other products later in the case of incomplete sums, or to be output to RAM2 through an output bus and a DMA, in the case of complete sums. The column processors in the 3 levels have similar architecture, with minor changes in the higher levels for the smaller number of inputs and slower data rates.

Each of the 9 pixels in a 3 by 3 block of the image is assigned to a channel, as shown in Figure 2. Each first level systolic array processes one-third row of pixels, two pixels at a time from the corresponding locations in the two neighboring 3 by 3 blocks. The two systolic arrays (SA) in the first level of a channel process, in a cycle, the 4 pixels located at the corresponding locations in the four neighboring 3 by 3 blocks which form a 6 by 6 block. Therefore, in each cycle, the 3 channels of the chip process 12 pixels, 6 from a row and 6 from another row two rows apart. Because the SWT starts with row processing, the 12 pixels for the next pipeline cycle come from the same two rows. So, the image is scanned 2 rows at a time in a 6-row partition. After finishing the two rows, image scan starts from the beginning of the corresponding two rows in the next 6-row partition. Actually, because of the different advances required in the first level of SWT, the starting points of the 9 channels spread over 4 neighboring 3 by 3 blocks as shown in Figure 7. Different advances among the channels, in the second and third levels, are realized by delays between levels and shifting of filter coefficients in the column processors.

The chip was designed for HDTV (high definition TV) images of size 1248X960. So, the size of input data array for a single channel is 416X320. Adding the 16 flip-over points on the left and on the top as well as the 20 flip-over points and 12 zero-padding points on the right and on the bottom, the input array for a channel is 464X368. In order to shorten the shift-registers required in the column processors, the input array is turned 90 degree. Namely, the input array is considered to be 368X464.

For higher speed and lower transistor count, integer arithmetic is used throughout the chip. However, the accuracy of the transformation is not sacrificed. This is accomplished by progressively increasing the number of data bits from a DWT level to the next to maintain the necessary precision. The linear pipeline architecture makes this possible.

Because the chip is designed for the specific application of wavelet-based video compression, we have chosen an architecture that can be optimized for the application.[6] Compared to other VLSI architectures for 2-D discrete wavelet transform such as those based on folding or the recursive pyramid algorithm [3,4,5], this architecture is less flexible. However, it has the advantages of higher speed, simpler control, higher precision, and fewer transistors needed. It is faster because several rows can be processed in parallel by the multiple systolic arrays in the first level of DWT, each systolic array processes 2 pixels at a time, and there is no feedback to slow down the flow in the pipeline. The control is simple because it is a linear pipeline and thus complex routing is not required. High precision can be maintained without using floating-point arithmetic because necessary numbers of data bits can be used in various stages of the linear pipeline. Finally, it uses fewer transistors because partial sums of products instead of the products themselves are stored in the buffers of level 1, and smaller multipliers can be used for multiplying with the smaller high-pass filter coefficients in the biorthogonal DWT.

4. Implementation

We first optimize the logic for the MACs since they are the most numerous components besides registers. Due to the use of the BIOR (2,2) filters, each of the MACs can be reduced to at most 2 adders. This is because the coefficients of the BIOR (2,2) filters are numbers that are easily multiplied by one or two left shift of the data rather than the need for a multiplier unit. The largest MAC multiplies the coefficient 6/8, which is accomplished by a left shift of one bit plus a left shift of two bits followed by a second adder to accumulate the result to the partial sum of other products. A division by 8 is unnecessary because

it is the common denominator of all coefficients in the filter.

The next major components to be optimized are the long and costly shift-registers in the column processors. In level one where there are multiple inputs (in a partition) to a column processor, saving partial sums of products is better than saving input data because fewer shift-registers are needed. Because the first two coefficients of the high-pass and low-pass filters are identical, the partial sums involving only the two coefficients are identical also. Consequently, the number of shift registers buffers required in level 1 can be further reduced by 2. In level two and three, however, there is only one input per partition to a column processor, and so saving input data till all inputs needed to compute a complete sum have arrived would result in fewer shift-registers. Also, because the data saved are before the MAC operations in the column processor, the bit widths of data saved in the shift-registers are smaller. Because the longer filter has 5 coefficients, four shift-registers each are needed to store the high-pass inputs and the low-pass inputs. Figure 8 shows the block diagrams for the first level and second level of a SWT channel.

Due to the linear nature of the pipeline, little control is needed to maintain the functionality of the data path. More controls are needed for synchronizing with the PE, the chip select control unit, and the output DMA controllers. Synchronization with the PE is necessary because the PE and the SWT chip operate asynchronously, and handshaking is required to make sure that input and output of data in the SWT chip do not get out of phase. After six inputs from RAM1, a handshake between the PE and the SWT chip prepares them for the next batch of 6 input data in a systolic cycle. This handshake also is used to initialize the systolic cycles, where each handshake makes a transition of the internally generated systolic clocks. This handshake also controls the input bus to distribute the data among the three channels. The chip select control, set by the three chip-select pins, changes the mode of the chip from one set of three channels to the next set. This control is essential because the three modes differ in when the DMA controllers are active as well as the shifting of filter coefficients (due to different advances required) in the column processors of levels two and three. This control is not needed in level one because the advances required for this level are accomplished in the scan pattern of the input image. The last main control for the chip is the output DMA control. The first task of each of the three DMA controllers is to poll the data bus on the outputs of the three channels. It controls which addresses on the bus are valid by waiting for the time when the valid data arrive at the bus, and then reads those addresses. This is necessary because of the differences between the times needed to produce a high-pass and a low-pass complete sum. The high-pass output requires only three coefficients to compute and so it is available earlier than the low-pass output which requires five coefficients. This timing is also different in the different chip modes due to the differences in the vertical (column-wise) advances in levels two and three. The next task of the DMA control is to re-integrate the outputs from the three channels into a wavelet coefficient image in RAM2. This is done by saving an initial address for each of the subbands, which is increased by 3 as data on a row are being put out and then the stride is added when a new row of output data begins to be put out.

We first verify the correctness of the architecture by carrying out functional simulation in Matlab. We then complete logic design, RTL coding in VHDL, synthesis, and gate-level simulation. The design was then handed over to OKI Semiconductor, Inc for layout and fabrication. The IC technology used is 0.25 micron with 3 metal layers, and the package is Ball Grid Array with 256 pins. (less than 200 pins used). The total gate count is about 625K. With a 16 ns clock, the chip can process 27 frames (with flip-over and zero-padding) per second. Figure 9 shows the chip layout.

5. Conclusion

The shift-invariant wavelet transform has a potential for improved video compression. We have designed a highly parallel and pipelined chip for this very computation intensive transformation. The chip is being fabricated by OKI Semiconductor, Inc. We have also designed the second chip in the block diagram, i.e., the motion estimation chip, and plan to build a complete wavelet based video compression system in the future.

Acknowledgement

This work is supported by a grant from Pittsburgh Digital Greenhouse in collaboration with OKI Semiconductor, Inc. We would like to thank Fred Samandari, Anna Ling, and Alan Zhang of OKI Semiconductor Inc. for their constant help during the last phase of the design.

6. References

[1] C. Chui, X. Shi, and A. Chen, "An Oversampled Frame Algorithm for Real-time Implementation and Applications," *Proc. SPIE Conf. Wavelet Applications*, vol. 2242, pp. 272-301, 1994.

[2] M. A. Al-Mohimeed and C. C. Li, " Motion Estimation and Compression Based on Almost Shift-Invariant Wavelet Transform for Image Sequence

Coding," *Intern. Jour. of Imaging Syst. Technol*. Vol. 9, pp. 214-229, 1998.

[3] M. Vishwanath, R. M. Owen, and M. J. Irwin, "VLSI Architectures for the Discrete Wavelet Transform," *IEEE Trans. on Circuits and Systems*, Vol. 42, PP. 305-316, 1995.

[4] A. Grzeszczak, M. K. Mandal, S. Panchanathan, and T. Yeap, " VLSI Implementation of Discrete Wavelet Transform", *IEEE Trans. on VLSI Systems*, Vol. 4, No. 4, pp. 421-433, Dec. 1996.

[5] K. K. Parhi, and T. Nishitani, "VLSI Architectures for Discrete Wavelet Transform", *IEEE Trans. on VLSI Systems*, Vol. 1, No. 2, PP. 191-202, June 1993.

[6] H. Y. H. Chuang and L. Chen, "VLSI Architecture for Fast 2D Discrete Orthonormal Wavelet Transform", *Journal of VLSI Signal Processing*, Vol. 10, pp.225-236, 1995.

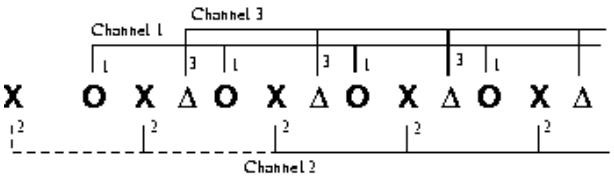

Figure 1: One-dimensional 3-Channel Input for Decomposition

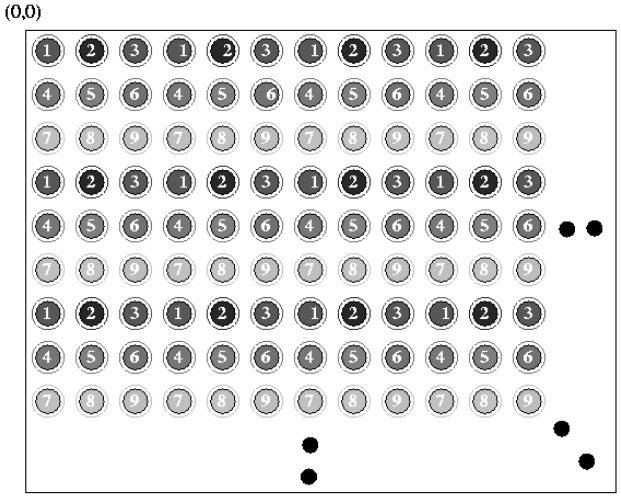

Figure 2: An image formulated as 9-channel data for shift invariant wavelet transform

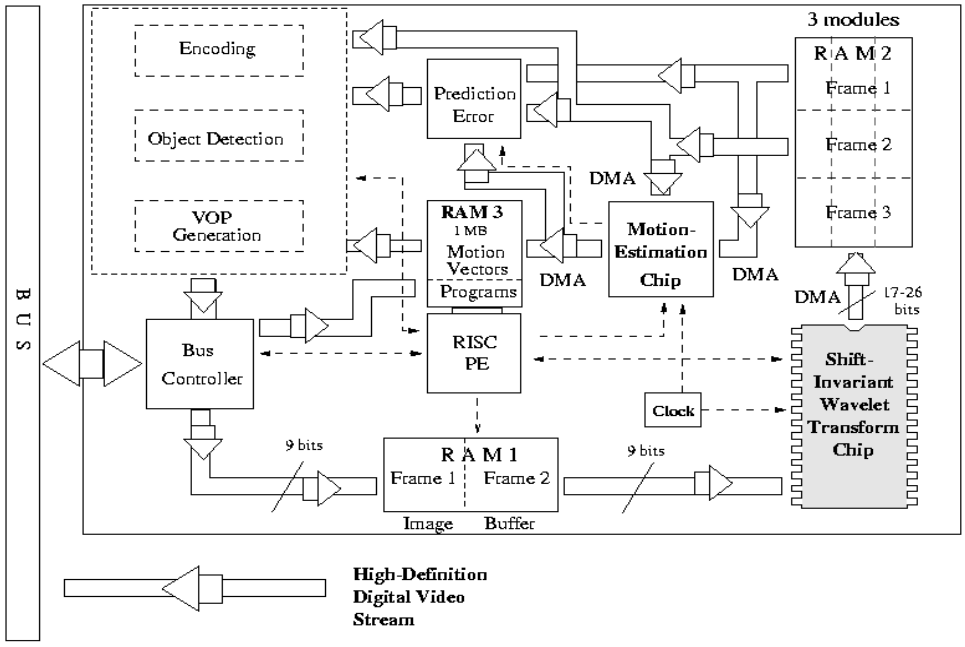

Figure 3: System Block Diagram

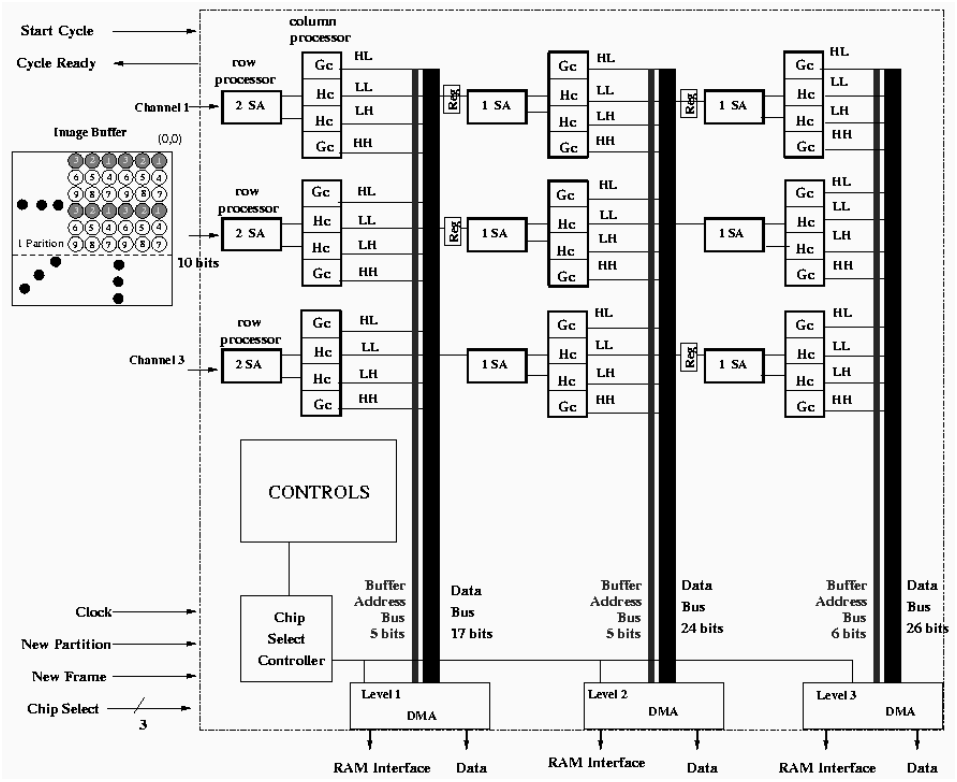

Figure 4: Three-Channel SWT Chip

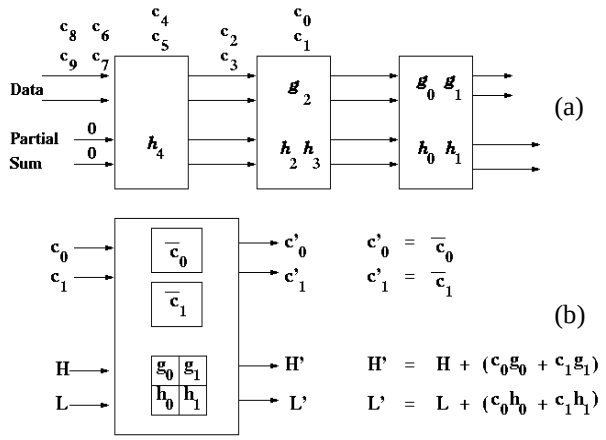

Figure 5: Row Processor: (a) Systolic Array (SA) and (b) Array Cell

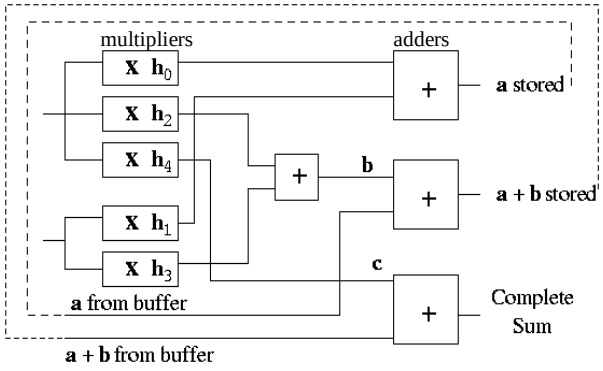

Fig. 6: Column Processor: Parallel Array Hc (low-pass)

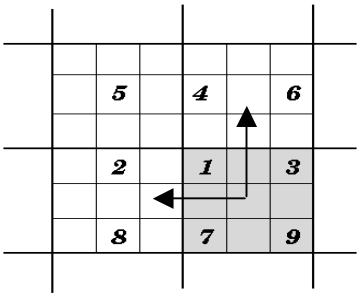

Figure 7: Relative Starting Positions of Input Scanning for SWT Channels

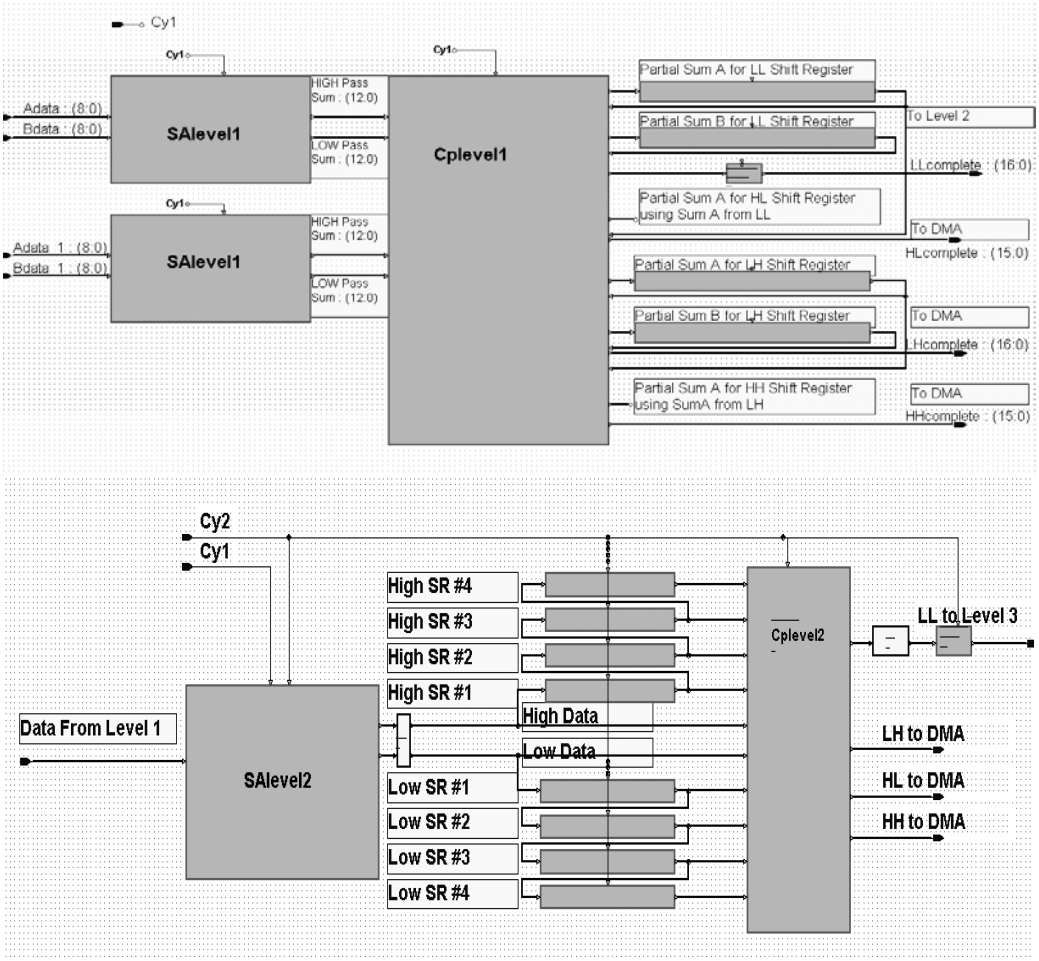

Figure 8: Block diagrams for the (a) first level and (b) second level of a SWT channel

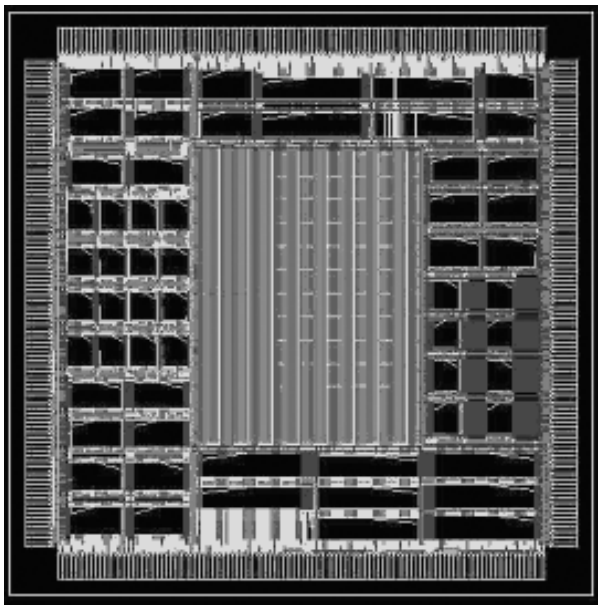

Figure 9: SWT Chip Layout