AN FPGA IMPLEMENTATION OF HIERARCHICAL MOTION ESTIMATION FOR EMBEDDED OBJECT TRACKING

Michael McErlean

Institute for System Level Integration
The Alba Campus, Livingston, EH54 7EG, Tel: - +44(0)845 2266508
Michael.McErlean@sli-institute.ac.uk,

***Abstract* – This paper presents the hardware implementation of an algorithm developed to provide automatic motion detection and object tracking functionality embedded within intelligent CCTV systems. The implementation is targeted at an Altera Stratix FPGA making full use of the dedicated DSP resource. The Altera Nios embedded processor provides a platform for the tracking control loop and generic Pan Tilt Zoom camera interface. This paper details the explicit functional stages of the algorithm that lend themselves to an optimised pipelined hardware implementation. This implementation provides maximum data throughput, providing real-time operation of the described algorithm, and enables a moving camera to track a moving object in real time.**

***Keywords* – Wavelet Decomposition, Phase Correlation, Motion Detection, Motion Estimation, Object Tracking, FPGA, DSP, CCTV.**

I. INTRODUCTION

Current research has identified reconfigurable computing platforms as a key technology in the implementation of computationally demanding real time image processing functionality. Wong *et. al.* [1] describe the implementation of existing computer vision applications on FPGA based hardware to achieve real time performance.

This paper describes a similar application with the goal of embedding tracking intelligence on the camera at the perimeter of the CCTV network. A system is described based on Hierarchical Block Based Phase Correlation (HBBPC) [2]; an algorithm developed specifically for hardware implementation that allows a Pan Tilt Zoom (PTZ) camera to automatically detect and track a moving object within its field of view. This includes the ability to both pan and tilt the camera in order to centralise the target during tracking. The system comprises a hierarchical motion estimation algorithm integrated within a software control loop allowing a PTZ camera to perform automatic tracking of subjects. The HBBPC motion estimation algorithm is described briefly here but the main focus is on the hardware implementation of this algorithm and the development of the generic camera PTZ control system.

The remaining part of this paper is organised as follows, Section II outlines the hierarchical block based motion estimation algorithm. Section III details the design process involved in realising the algorithm in hardware targeted at an Altera Stratix FPGA. Section IV details the development of the software control loop in the embedded Nios processor. Finally Sections V and VI present a discussion of results and conclusions.

II. HIERARCHICAL BLOCK BASED PHASE CORRELATION

This distinguished hardware oriented tracking approach uses a motion estimation technique based on a thoroughly proven technique called Phase Correlation (PC) [3-12]. Initial experimentation and prior literature indicate that PC provides a robust and very accurate means of determining the relative motion between two images. This technique has also proven robust in handling changes in orientation and illumination and is capable of producing sub-pixel motion estimation measurements. With PC the inter-frame motion is measured in the frequency domain as a phase shift. The result is an array of real values, termed a correlation surface, that contain peaks at coordinates corresponding to the time domain shift between images.

Let g_0 and g_1 be the two images which differ by a displacement of (x_0, y_0), the two images are related as follows:

$$g_1(x, y) = g_0(x - x_0, y - y_0) \tag{1}$$

Their Fourier transform is then related as follows:

$$G_1(\xi,\eta) = e^{-j2\pi(\xi x_0 + \eta y_0)} \cdot G_0(\xi,\eta) \tag{2}$$

The inverse Fourier transform of the phase correlation equation (3) produces a correlation surface.

$$\frac{G_0(\xi,\eta).G_1^{*}(\xi,\eta)}{|G_0(\xi,\eta).G_1^{*}(\xi,\eta)|} = e^{j2\pi(\xi x_0 + \eta y_0)} \tag{3}$$

If the two images are identical but shifted, the result will be an impulse at (x_0, y_0) which represents the translational displacement between the two images.

In terms of an object tracking application PC makes it possible to find a true match for an object's inter-frame motion as opposed to the best match results produced by the common motion estimation technique of block matching, this has been discussed in [4]. However there are limitations when applying the PC technique to object tracking. Firstly the size of the tracked object within the image has an impact on sensitivity to motion. This is due to the direct relationship between the size of an object within the scene and the amplitude of its correlation peak. Secondly, PC is capable of identifying multiple motion content between images but does not indicate where within the scene the object exists. The resulting correlation surface records motion information not positional information.

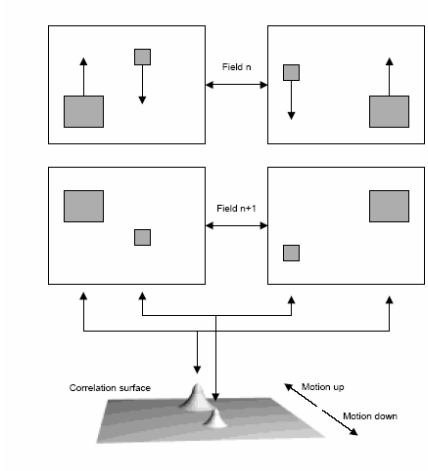

Fig.1 Phase correlation measures a relative motion between images and is dependent on object size.

These limitations are illustrated in Fig.1 where two sets of images contain the same motion and produce the same correlation surface, the larger object producing a larger correlation peak than the smaller object, but the location of the objects in the two image sets are different.

By applying a block based PC technique (BBPC) localised motion content is determined within constrained regions of an image solving the problem of determining where within the image the moving object exists. Again there are tradeoffs with a block based approach, smaller blocks require less hardware resource but limit the measurable displacement of the object; larger blocks risk the problem of the background dominating the correlation process.

The use of a hierarchical algorithm, when performing BBPC, provides the capability to handle much larger object motion than that of standard block based techniques. The algorithm proposed in [2] uses a discrete wavelet transform (DWT) decomposition to provide a number of resolution representations of each image, feeding into a fixed block size implementation of PC. Fig.2 illustrates the process flow. Motion vectors produced using BBPC at lower resolution levels will be to the nearest sub-sampled pixel value. However, by using interpolation, the effective resolution of the correlation map can be increased in the area around the detected peak to identify the actual pixel or even sub-pixel motion. Also this hierarchical technique allows the block based algorithm to target specific local motion at higher levels whilst handling global motion at lower levels. This hierarchical approach solves the problem of background dominance in the correlation process by using a smaller block size in conjunction with multiple image resolution levels.

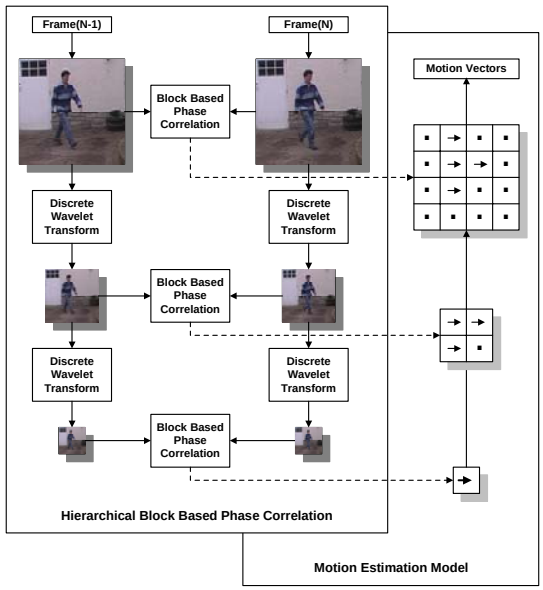

Fig.2 Hierarchical Block Based Phase Correlation

New information at block edges, due to object motion, creates noise in the output of the BBPC process due to spectral leakage. To reduce this noise a raised cosine windowing function is applied to each block at the input to the BBPC process. The windowing process attenuates the perimeter signal frequencies thus reducing spectral leakage. While this improves the results of the BBPC it limits the effective area of correlation and thus reduces the maximum measurable velocity of a tracked object within the block.

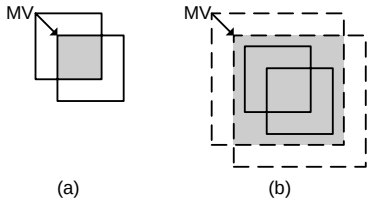

Fig.3 Correlation area using (a) 16*16 and (b) 32*32 blocks.

By using a larger block size of 32*32 pixel elements, a larger overlapping area is provided for correlation (Fig.3) and the windowing function has a negligible impact on the area inside the central 16*16 block.

III. FPGA HARDWARE IMPLEMENTATION

This hardware implementation of the HBBPC algorithm makes use of a three level hierarchy; this provides multiple resolution motion information using a fixed BBPC pipeline in hardware. This hierarchy comprises two functional blocks (Fig.2) the DWT function and the BBPC Pipeline.

The hardware implementation of the DWT is based on the lifting scheme implementation of the Haar transform. This produces multiple resolution data sets which are used for the BBPC process. The use of the lifting scheme provides an algorithm for producing sub-sampled low frequency estimates using a simple differencing and averaging filter bank [13-15].

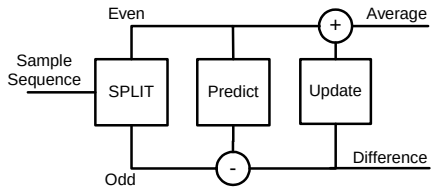

Fig.4 DWT Lifting Scheme.

Fig.4 illustrates the lifting scheme forward transform. The predict stage assumes that neighbouring pixel values are correlated and so the even elements are used to predict the odd elements, resulting in a difference coefficient. In this lifting scheme implementation of the Haar transform the update stage replaces the even element with the average of the odd/even pair. The current implementation buffers only the resulting average data at each level of the hierarchy and these buffers provide the previous and current frame data to the BBPC pipeline block. In order to save on memory resources the high frequency difference components at each level are not stored in this implementation.

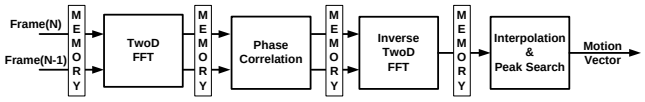

Fig.5 Block Based Phase Correlation Pipeline.

The hardware implementation of the BBPC algorithm represents a direct translation of the mathematics described previously. This pipelined implementation provides a trade off between resource usage and system performance. As a result each stage of the pipeline is buffered, utilising the FPGA BlockRAM resource [16], allowing for the pipeline stages to be executed in parallel thus increasing the performance of the system. The hardware implementation of the BBPC pipeline comprises three main functional blocks (fig.5):

- A Two Dimensional Fast Fourier Transform (2DFFT) of the windowed image segments.
- Extraction of the normalised phase difference information using CORDIC functions.
- An Inverse 2DFFT of the array of normalised phase difference coefficients.

The current and reference image data at the input to the 2DFFT block represents two real data sets. These data properties are exploited in this hardware implementation to perform the simultaneous 2DFFT transform of two real signals by combining them to form a complex input to the 2DFFT transform [17]. The alternative being to perform an individual 2DFFT operation on each of the two data sets, either doubling the processing time or the hardware resource required. For two real data sets *h* and *g* the complex data set *y* is formed by combining the inputs into a complex form as a complex input to the FFT as follows:

$$y(m,n) = h(m,n) + jg(m,n) \tag{4}$$

The individual 2DFFT transform coefficients are then separated at the output using the symmetry inherent in the FFT as follows:

$$H(m,n) = \left[\frac{R(m,n)}{2} + \frac{R(M-m,N-n)}{2}\right] + j\left[\frac{I(m,n)}{2} - \frac{I(M-m,N-n)}{2}\right] \tag{5}$$

$$G(m,n) = \left[\frac{I(m,n)}{2} + \frac{I(M-m,N-n)}{2}\right] - j\left[\frac{R(m,n)}{2} - \frac{R(M-m,N-n)}{2}\right] \tag{6}$$

for data in the ranges $m = 0,1....M-1$, $n = 0,1.....N-1$

The hardware implementation of the 2DFFT algorithm is an in-place decimation in time radix 2 FFT [18]. The trade off between accuracy and resource usage is achieved using a normalised implementation of a signed two's complement fixed point arithmetic engine which implements the radix two butterfly. Fixed point arithmetic is used throughout the hardware implementation for all arithmetic functions. This implementation allows the fixed point accuracy to be adapted and the hardware can be configured for any power of two, N point FFT. By using separate predefined complex twiddle coefficients in ROM for the normalised forward and reverse FFT transforms the same hardware block is instantiated in both cases. The DSP resource in the Altera Stratix device provides a bank of four multipliers and two add/sub blocks for dedicated complex multiply operations of fixed data sizes [19][20].

Measurement of the normalised phase difference between images is the next stage in the pipeline. The phase correlation function identifies the phase shift between corresponding frequency coefficients by subtracting the phase of the previous frequency coefficient from that of

the current. In the complex plane this is done by a complex multiply of the current component with the complex conjugate of the reference component. In order to normalise the phase difference and eliminate the impact from the magnitude product of the complex multiply the hardware implementation uses two CORDIC blocks (fig.6) [21–24]. The first CORDIC block in the system converts each complex coefficient to polar form, allowing separation of the phase and magnitude parts of the complex coefficient. The second CORDIC stage converts the normalised polar form back to complex form providing the normalised phase difference. The advantage of the CORDIC algorithm is that its implementation uses simple add, subtract, shift register and comparator hardware eliminating the need for multipliers.

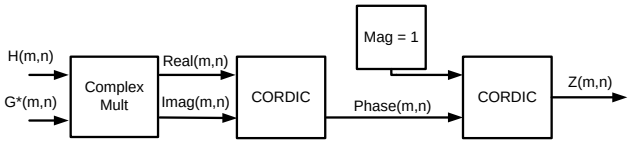

Fig.6 Normalised Phase Difference Using CORDIC

IV. Software Control Loop

The Altera Nios 32bit soft core processor [25] is hosted in the FPGA as a means of providing software control to the hardware system for motion estimation. The camera control loop is also implemented in software and the Nios handles the updates of the tracker position and camera control commands based on the motion information provided by the hardware. Status and control registers provide an interface to hardware by the software control loop and the motion vectors produced by the hardware are used to update the location of the current region of interest. The object mask block in fig.7 is not part of the hierarchical algorithm but has been used in this hardware implementation as a means of reducing the amount of memory required; the software provides position information which allows the hardware to buffer a region of interest from each frame rather than the entire frames information. This also allows the user to define a region of interest such as a doorway where motion detection is applied.

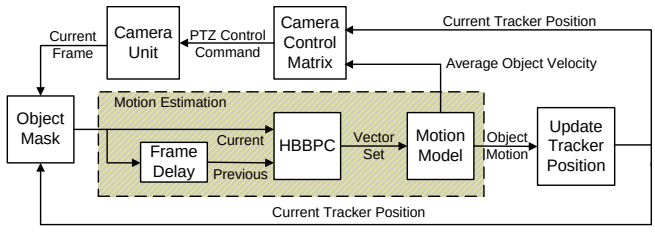

Fig.7 Automatic Object Tracking System.

Serial camera control commands issued via a UART provide control of the speed and direction of the camera during tracking. A generic command matrix interfaces the 9 camera control commands generated by the system to the camera specific instruction format which is stored in a software array (Fig.8). This array can be configured remotely allowing multiple camera protocols to be supported and interfaced to the tracking hardware. From a preset position the hardware measures the motion content between frames and so the initial state of the system is in motion detection. When motion is detected the hardware begins to track that motion and the system moves into an object tracking state. The tracking position is then continually updated using the motion data produced by the hardware. Tracking will continue as long as the object remains within the sensitive region and within the field of view of the camera.

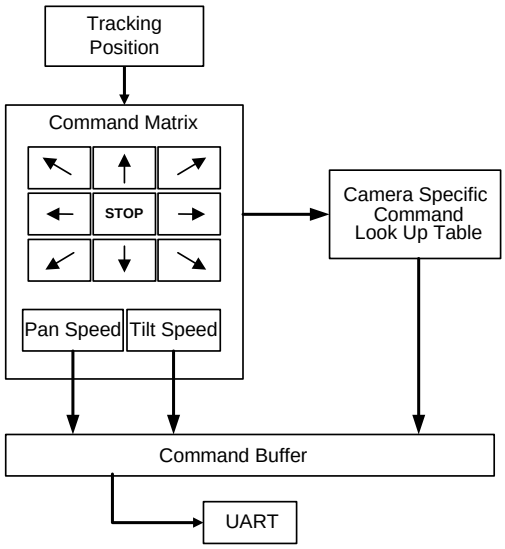

Fig.8 Generic Camera Control.

The positional information of the tracker is also used to control the speed and direction of the camera. Depending on the velocity and direction of the tracked object a control command is issued to the camera that attempts to compensate for this motion by moving the camera. The aim of the camera control algorithm is to maintain the tracked object as close to the centre of the scene as possible. The control itself is based on the spatial position of the tracker. Speed control is done spatially; the distance of the tracker from the centre of the scene has a direct mapping to the speed at which the camera moves in that direction. Small motion vectors signify a low velocity in a given direction whereas larger motion vectors signal a larger velocity and require faster motion by the camera to compensate.

V. RESULTS

The following section presents results obtained from the system, the simulation results identify the system performance in terms of the number of blocks that can be processed in a fixed time interval, providing a metric on the processing capability of the system. The fitter resource results obtained from the synthesis tool during placement identify the hardware resource requirements for the current hardware configuration. From these results it can be seen that the system has a high processing rate but at the cost of memory resource due to the pipelined implementation. The autonomous object tracking performance is illustrated via captured images from the system during operation.

Simulation of the current hardware implementation of the BBPC Pipeline shows that the 2DFFT blocks are capable of processing a 32x32 block every 220us. This is equivalent to a total of 5120 butterfly operations per block with internal block RAM accesses running at 108MHz. The FFT blocks are currently the limiting factor in the data rate of the pipeline as they represent the slowest stage in the process. From the simulated data rate the current pipeline is capable of processing up to 4540 blocks per second with an initial 4 stage pipeline delay.

It is clear from the fitter information provided by the Quartus II tools, shown in Table 1 below, that memory usage is the major limitation in the current implementation. The pipelined approach to implementing the algorithm increases the processing ability but at the cost of extra memory resource requirements.

Functional Block	Logic Elements	Memory (bits)	DSP (9 bit elements)
2DFFT	640	4608	8
CORDIC	3774	70	0
BBPC Pipeline	6238	287 814	28
DWT (inc reference frame buffers)	396	344 064	0
HBBPC Block	6735	631 878	28
Nios Processor	5486	100 352	0
Overall System Requirements	12221	732 230	28

Table.1 FPGA Fitter Resource Results

The following image sequences taken from captured video footage illustrate the operation of the autonomous object tracking algorithm. In each image the tracked region is highlighted by a bounding box and as the position of the bounding box moves away from the centre of the image the control system adjusts the camera position to compensate for this motion.

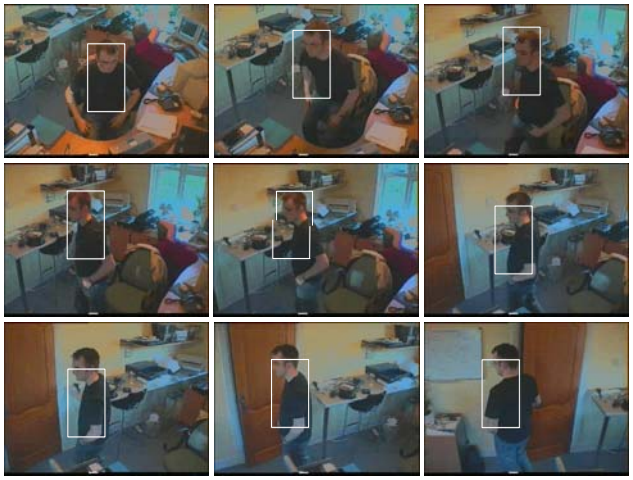

Fig.9 Object Tracking Sequence.

In the first sequence, Fig.9 above, the tracker follows the subject as he proceeds to leave the room. As the subject stands up and moves toward the door the camera automatically tilts upwards and pans to the left to maintain him in the field of view.

In the second sequence, Fig.10 below, the system is initially in a motion detection state sensitive within the marked area. As the subject moves into the sensitive area the system detects motion and begins to track the subject; again panning the camera to the left to maintain the subject in the field of view.

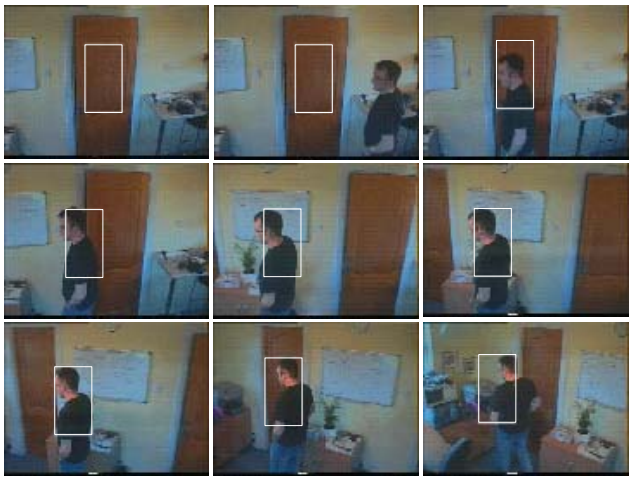

Fig.10 Object Detection and Tracking Sequence.

VI. CONCLUSIONS

This pipelined hardware implementation of the HBBPC algorithm provides the ability to perform motion detection and object tracking at full video rate in real time with cooperative cctv camera motion control. The design of the system allows for a number of BBPC pipelines to be

instantiated in parallel depending on hardware resource availability and the amount of processing required per frame period. The current implementation utilises a single pipeline but depending on the amount of processing required the motion estimation can be performed at a reduced frame rate if necessary. Within the pipeline itself the CORDIC stages are currently pipelined but could be implemented as a single iterative stage. This would reduce the hardware resource required by only having a single stage as opposed to the current 16 stage pipeline. This optimization would reduce the hardware requirements but at the sacrifice of the overall performance of the system but this could be compensated for by increasing the process clock frequency.

The software control loop implemented using the embedded Nios processor provides a flexible system for control allowing interfacing to any generic PTZ CCTV camera.

The next stage of development is a prediction based motion probability model using Kalman filters that will provide the ability to handle partial or total temporary occlusion during the tracking process.

ACKNOWLEDGMENTS

Fig.1 was used with permission from Snell & Wilcox, and sourced from "The Engineer's Guide to Motion Compensation".

This work is supported as part of EngD funding provided by the Engineering and Physical Sciences Research Council (EPSRC) and has been carried out in conjunction with:

Salent Technologies Ltd
The Rowan House
Wyndford Brae
Philpstoun
Nr. Linlithgow
West Lothian
EH49 6RN, U.K.
http://www.salent.co.uk

REFERENCES

[1] Wong, S.G.; Jasiunas, M.; Kearney, D., "Towards a reconfigurable tracking system," Field Programmable Logic and Applications, 2005. International Conference on , vol., no.pp. 456- 462, 24-26 Aug. 2005
[2] McErlean M, "Hierarchical Motion Estimation for Embedded Object Tracking", 6th IEEE Int. Symp. On Sig. Processing and Info. Technology, 2006.
[3] Kuglin C. and Hines D, "The phase correlation image alignment method", Proc. Of the IEEE Int. Con. On Cybernetics and Soc., 1975, pp.163-165.
[4] Thomas G, "Television motion measurement for DATV and other applications", 1987 BBC Research Department, Research Report 1987/11.
[5] Hill L. and Vlachos T, "Motion measurement using shape adaptive phase correlation", Electronics Letters., 2001, Vol. 37 No. 25.
[6] Hill L, Vlachos T, "On the estimation of global motion using phase correlation for broadcast applications", Image Processing and Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465) , Volume: 2 , 13-15 July 1999 Pages:721 - 725 vol.2
[7] Argyvriou, V.; Vlachos, T., "Motion estimation using quad-tree phase correlation," Image Processing, 2005. ICIP 2005. IEEE International Conference on , vol.1, no.pp. I- 1081-4, 11-14 Sept. 2005.
[8] Erturk S, "Digital image stabilization with sub-image phase correlation based global motion estimation", Consumer Electronics, IEEE Transactions on, Volume: 49, Issue: 4, Nov. 2003 Pages:1320 – 1325
[9] Wu S.F, Fernando G.M.X, "Comparative study of two motion estimation techniques", Image Processing and its Applications, 1989, Third International Conference on , 18-20 Jul 1989 Pages:305 – 309
[10] Kumar E, Biswas M, Nguyen T.Q., **"**Global motion estimation in frequency and spatial domain"**,** Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on, Volume: 3 , 17-21 May 2004 Pages:iii - 333-6 vol.3
[11] Dabner S.C.; "Motion estimation for HDTV", Image Processing for HDTV, IEE Colloquium on , 26 Oct 1989 Pages:2/1 - 2/4
[12] Watkinson J, "The Engineer's Guide to Motion Compensation", Snell & Wilcox, 1994
[13] W. Sweldens, "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions", Wavelet Applications in Signal and Image Processing III, 1995
[14] W. Sweldens and P. Schroder, "Building your own wavelets at home", 1996, Pages 15-87
[15] A. Jensen and A. la Cour-Harbo, 'Ripples in Mathematics: The Discrete Wavelet Transform', *Springer*, 2001
[16] http://www.altera.com/literature/hb/stx/ch_3_vol_2.pdf
[17] E. Oran Brigham, 1974, "The Fast Fourier Transform", Prentice-Hall.
[18] Rabiner, LR, and Gold, B, 1975, "Theory and Application of Digital Signal Processing," Prentice-Hall.
[19] http://www.altera.com/literature/hb/stx/ch_6_vol_2.pdf
[20] http://www.altera.com/literature/hb/stx/ch_7_vol_2.pdf
[21] Ray Andraka, "A survey of CORDIC algorithms for FPGAs**"** FPGA '98. Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, Feb. 22-24, 1998, Monterey, CA. pp191-200.
[22] Volder, J, "Binary computation algorithms for coordinate rotation and function generation," Convair Report IAR- 1 148 Aeroelectrics Group, June 1956.
[23] Volder, J, "The CORDIC Trigonometric Computing Technique," IRE Trans. Electronic Computing, Vol EC-8, pp330-334 Sept 1959.
[24] M. E. Frerking, Van Nostrand Reinhold , "Digital Signal Processing in Communication Systems", 1994. ISBN 0-442-01616-6.
[25] http://www.altera.com/literature/lit-nio.jsp