Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 512746, 8 pages
doi:10.1155/2008/512746

Research Article

VLSI Implementation of Hybrid Wave-Pipelined 2D DWT Using Lifting Scheme

G. Seetharaman, B. Venkataramani, and G. Lakshminarayanan

Department of ECE, National Institute of Technology, Tiruchirapalli 620015, India

Correspondence should be addressed to G. Seetharaman, gsraman@nitt.edu

Received 24 July 2007; Revised 4 March 2008; Accepted 5 June 2008

Recommended by Tsutomu Sasao

A novel approach is proposed in this paper for the implementation of 2D DWT using hybrid wave-pipelining (WP). A digital circuit may be operated at a higher frequency by using either pipelining or WP. Pipelining requires additional registers and it results in more area, power dissipation and clock routing complexity. Wave-pipelining does not have any of these disadvantages but requires complex trial and error procedure for tuning the clock period and clock skew between input and output registers. In this paper, a hybrid scheme is proposed to get the benefits of both pipelining and WP techniques. In this paper, two automation schemes are proposed for the implementation of 2D DWT using hybrid WP on both Xilinx, San Jose, CA, USA and Altera FPGAs. In the first scheme, Built-in self-test (BIST) approach is used to choose the clock skew and clock period for I/O registers between the wave-pipelined blocks. In the second approach, an on-chip soft-core processor is used to choose the clock skew and clock period. The results for the hybrid WP are compared with nonpipelined and pipelined approaches. From the implementation results, the hybrid WP scheme requires the same area but faster than the nonpipelined scheme by a factor of 1.25–1.39. The pipelined scheme is faster than the hybrid scheme by a factor of 1.15–1.39 at the cost of an increase in the number of registers by a factor of 1.78–2.73, increase in the number of LEs by a factor of 1.11–1.32 and it increases the clock routing complexity.

1. INTRODUCTION

Field-programmable gate arrays (FPGAs) have grown enormously in their complexity and can encompass all the major functional elements of a complete end product into a single chip [1]. An FPGA-based system on chip can contain one or more processors, memories, dedicated components for accelerating critical tasks and interfaces to various peripherals. Development tools for the FPGAs, the Altera, San Jose, CA, USA system-on-programmable-chip (SOPC) builder, enable the integration of intellectual proprietary (IP) cores for common DSP functions and user-designed custom blocks with the softcore processors Nios II. The availability of on-chip dedicated multipliers, softcore/hardcore processors and IP cores make the FPGAs to be an ideal platform for the implementation of area as well as speed intensive image processing applications such as discrete cosine transform (DCT) and discrete wavelet transform (DWT) [2].

Joint Pictures experts Group 2000 (JPEG2000) is a recently standardized image compression algorithm that provides significant enhancements over the existing JPEG standard. JPEG2000 differs from widely used compression standards in that it relies on DWT and uses embedded bit plane coding of the wavelet coefficients. DWT has been traditionally implemented using convolution or FIR filter bank structures. These structures require both a large number of arithmetic computations and a large memory for storage, which are not desirable for high-speed/low-power image processing applications.

A new multiplier algorithm denoted as Baugh-Wooley pipelined constant coefficient multiplier (BW-PKCM) is proposed and used for the study and comparison of distributed arithmetic algorithm(DAA) and lifting schemes on FPGAs in [3]. For the computation of 2D DWT, 2's complement multiplications are required. In the literature BW method [4] has been studied with carry save, carry ripple, and serial parallel algorithms. These schemes are inefficient in speed, area, or both when one of the operand is fixed. For an N-bit number, conventional 2's complement multiplier (C2CM) requires [N-1/4] arrays of 4-inputs LUTs. But sign extension

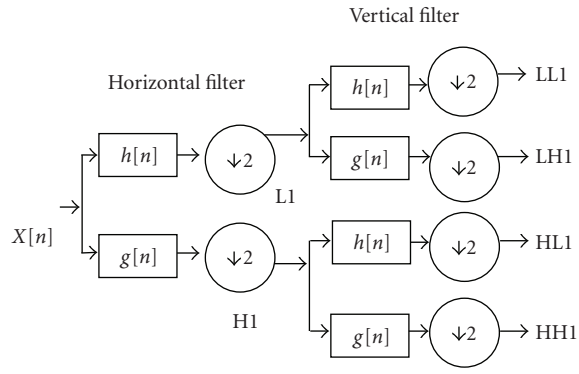

Figure 1: Subband decomposition for one-level 2D DWT.

and BW methods require [N/4] arrays of 4-inputs LUTs. The size of the array is equal to the number of product bits. The 2's complement block and control logic increases the number of LUT arrays area and multiplication time for the C2CM. However, for the sign extension and BW, the number of LUT array may be the same as that required for the first scheme. The lifting scheme with BWPKCM requires 4% less area but has the same speed compared to that using distributed arithmetic algorithm with sign extension scheme. The implementation details are available with [3]. In 2D DWT, filter coefficients are constant. Hence, BW-PKCM which combines the pipelined KCM with Baugh-Wooley multiplication algorithm is used in this paper.

The operating frequency of the 2D DWT may be increased, if it is implemented using either pipelining or WP. Pipelining results in the highest operating frequency but has number of disadvantages such as increased area, power dissipation, and clock routing complexity. WP has been proposed as one of the techniques for overcoming these limitations. WP results in increase in the speed and reduction in the clock routing complexity. The proposed hybrid scheme is aimed at combining the advantages of both pipelining and wave-pipelining.

The organization of the rest of the paper is as follows: In Section 2, the previous work on 2D DWT and the design of wave-pipelined (WP) lifting blocks on FPGAs are described. In Section 3, the previous work related to WP and the challenges involved in the design of WP circuits are described. In Section 4, automation schemes for WP circuits are presented. In Section 5, BIST approaches for the implementation of WP circuits are discussed and the implementation results are presented. In section 6, SOC approaches for the implementation of WP circuits are discussed and the implementation results are presented. Section 7 summarizes the conclusions.

2. REVIEW OF PREVIOUS WORK ON 2D DWT

The lifting-based implementation of one-level 2D DWT may be computed using filter banks as shown in Figure 1. The input samples $x(n)$ are passed through the 2 stages of analysis filters. They are first processed by the low-pass ($h[n]$) and high-pass ($g[n]$) horizontal filters and are sub

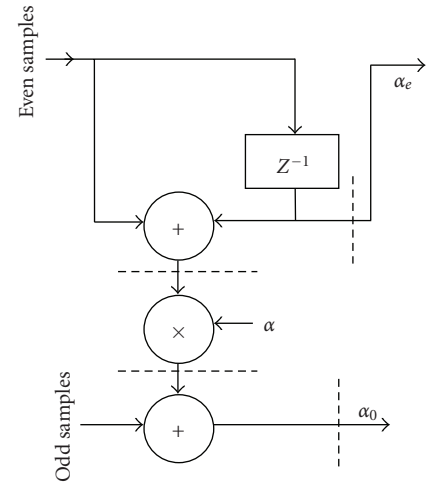

Figure 2: α block.

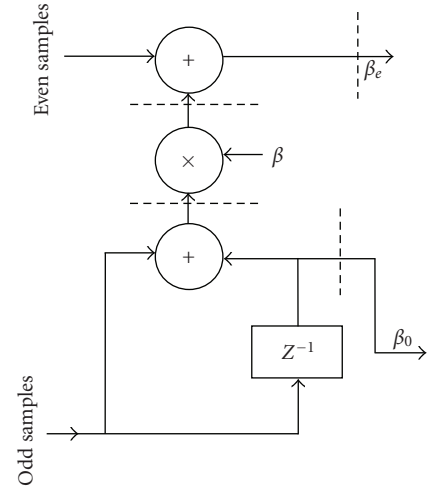

Figure 3: β block.

sampled by two. Subsequently, the outputs (L1, H1) are processed by low-pass and high-pass vertical filters. The lifting scheme uses a polyphase structure for the analysis filter [5]. The main feature of the lifting scheme requires a far fewer computations compared to the convolution-based DWT. Also the computational complexity can be reduced by 50%. As a result, lifting-based implementation provides an efficient way to compute wavelet transforms [6].

In the lifting scheme, the odd and even input samples are processed by five lifting blocks $(\alpha, \beta, \gamma, \delta, \xi(\xi_1, \xi_2))$ in cascade. ξ_1, ξ_2 are scaling blocks.

Details of α and β blocks are shown in Figures 2 and 3. γ and δ blocks are obtained by replacing the constants α, β with γ, δ. The following modifications are proposed in the lifting scheme [3].

(i) In Figure 2, since the output from one block is fed as input to the next block, the maximum rate at which the input can be fed to the system depends on the sum of the

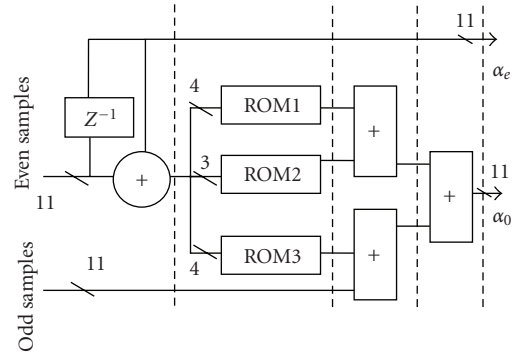

Figure 4: α block using BW-PKCM.

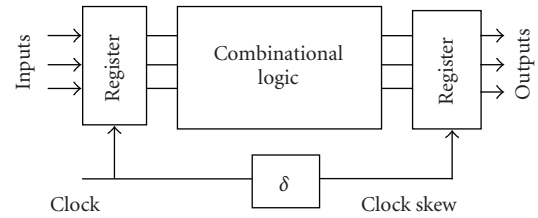

Figure 5: A combinational logic circuit with input and output registers.

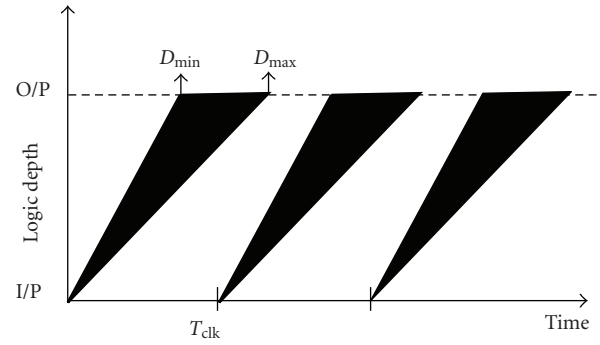

Figure 6: Temporal/spatial diagram of data flow through the combinational logic circuit.

delays in all the four stages. The speed is increased in [5], by introducing pipelining at the points indicated by dotted lines in Figure 2. In this case, the input rate is determined by the largest delay among all the four blocks.

(ii) The delay in the individual stages is reduced further by using constant coefficient multiplier (KCM).

KCM uses a ROM for finding the product of a constant and a variable. The variable is fed as address to the ROM which contains the products corresponding to all possible combinations of the operands. When the ROM is implemented using 4 input look-up tables (LUTs), a number of stages of LUTs and adders are required to find the product. The speed of the KCM can be increased by introducing the pipelining registers at the outputs of ROMs and adders.

The detailed diagram of the α block implemented using BW-PKCM is shown in Figure 4. The same scheme can be adopted for the $\beta, \gamma, \delta, \xi_1, \xi_2$ blocks. The dotted line indicates points where registers may be inserted for pipelining. For wave-pipelining all the stages are directly connected without registers. The registers are used only at the inputs and outputs. In hybrid wave-pipelining, registers are used between adjacent lifting blocks and the individual lifting blocks are connected without registers.

2.1. Overlapping scheme for block 2D DWT

In the overlapping scheme, the image block is formed such that a number of pixels overlapped between adjacent blocks along the vertical and horizontal direction are equal to the order of the filter. For example, for the 9/7 biorthogonal filter used for the 2D DWT, the number of overlap pixels should be equal to 4 on the left and 4 on the right between horizontal blocks. Similarly, the number of pixel overlap between vertical blocks should be equal to 4 on the top and 4 on the bottom. For the blocks on the boundary, overlapping needs to be done only on the nonboundary edge.

3. BACKGROUND ON WAVE-PIPELINING

The concept of wave-pipelining has been described in a number of previous works [4, 7, 8]. To illustrate the concept of wave-pipelining, graphical representation [7] of the data flow through combinational logic is used. Figure 5 shows the combinational logic with wave-pipelining circuit surrounded by edge triggered input and output registers [7]. Figure 6 gives the associated timing diagram [7]. In Figure 6, the shaded regions bounded by the maximum and minimum delays through the logic ($D_{\max}$ and $D_{\min}$) depict the flow of data through the combinational logic and the variations in the logic block with time. The nonshaded areas depict the stable duration of the logic.

In the conventional system, the output register is clocked in the nonshaded region and the minimum clock period, T_{clk}, is chosen to be greater than $D_{\max}$. In the wave-pipelined system, the clock period is chosen to be $(D_{\max} - D_{\min})$ + clocking overheads such as set-up time and hold time. In Figure 5, δ denotes clock skew between the input and output register. To ensure correct operation, the skew should be adjusted such that the active clock edge occurs in the stable period. To maximize the frequency of operation of the wave-pipelined system, the difference $(D_{\max} - D_{\min})$ is minimized by equalizing the path delays. However, the stable period decreases with the increase in the logic depth. By adjusting the latching instant at the output register to lie in the stable period, the wave-pipelined circuit has to be made to work properly. But, for large logic depths, there may not be any stable period. Hence, adjusting the latching instant by itself may not be adequate for storing the correct result at the output register. For such cases, the clock period has to be increased to increase the stable period. Equalization of path delays, adjustment of the clock period and clock skew are the three tasks carried out for maximizing the operating speed of the wave-pipelined circuit. All the three tasks require the

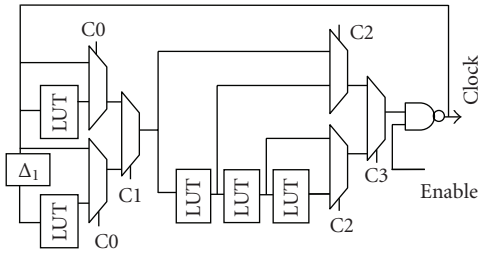

Figure 7: Programmable clock generator.

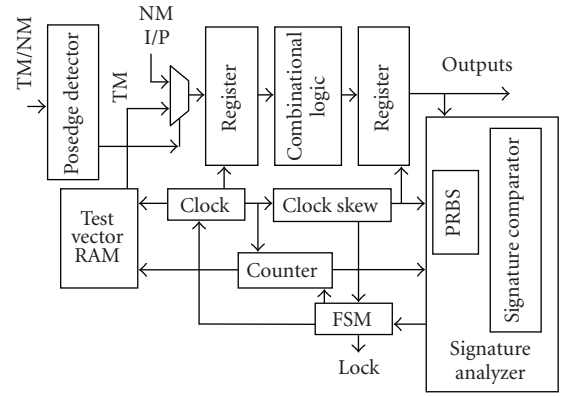

Figure 8: Self tuned wave-pipelined circuit.

delays to be measured and altered if required. These tasks are carried out manually in [9, 10].

For Xilinx FPGAs, the physical design editor referred to as FPGA editor may be used for measuring and altering the delays. Using this feature, the implementation of wave-pipelined circuits on Xilinx FPGAs is considered in [10]. The wave-pipelined circuit designed using the FPGA editor may be tested using simulation. However, the simulation is inadequate for testing due to the difference between the actual delays and the delays calculated by the FPGA editor. This is because the FPGA editor considers only the worst-case delays and the actual delays may be significantly different due to fabrication variations. This difference becomes important as the logic depth of the circuit increases. Hence, the design has to be downloaded to the actual FPGA and its operation has to be checked by feeding the test data [10]. If correct results are not obtained, delays are altered and the design is downloaded for testing again. A number of iterations of place and route, simulation, downloading, and testing in the actual device may be required till the correct results are obtained. The design of wave-pipelined circuit in this fashion requires human intervention and is time consuming. Automation of the above three tasks are considered in the next section.

4. AUTOMATION SCHEMES FOR WAVE-PIPELINED CIRCUITS

To maximize the operating speed of the wave-pipelined circuit, the equalization of the path delays is considered first. This cannot be completely automated as the commercially available synthesis tools do not support the specification of interconnect delays. However, the difference in path delays can be minimized by specifying the physical location of logic cells (slices) or logic elements used for the implementation, through either the user constraints file (UCF) or the logic lock feature supported by the FPGA CAD tools [11, 12]. UCF approach is proposed for Xilinx FPGAs in [10]. The logic lock feature is adopted for the Altera FPGAs in this paper. The adjustment of the clock skew and clock period can be automated by adopting programmability. The programmable clock and clock skew generator may be implemented in the FPGAs. Figure 7 gives the circuit diagram of a clock generation scheme which consists of a delay block and an inverter.

The actual clock period depends on the interconnect delay. The select input of the multiplexer is varied with either a processor or a finite state machine (FSM) to achieve different clock frequencies. The wave-pipelined circuit using the programmable clock and skew generator can be operated at a higher frequency than that can be achieved using the commercially available synthesis tools which use $D_{\max}$ for fixing the operating frequency. The automation may be carried out using either off-chip processor or on-chip processor. The off-chip processor is used when the FPGA is used as a coprocessor or hardware accelerator for a main processor or microcontroller.

Since off-chip communication between the FPGA and a processor is bound to be slower than on-chip communication, in order to minimize the time required for adjustment of the parameters of the wave-pipelined circuit (clock frequency and skew), the BIST approach using design for testability [13, 14] technique, is proposed for this case. In the SOC approach, a processor is assumed to be available on-chip and it is used for adjustment of the parameters of the wave-pipelined circuit.

4.1. BIST approach for wave-pipelined circuit

The automation can be carried out by including two blocks to the basic wave-pipelined circuit: a finite state machine (FSM) and a self test circuit. The FSM systematically varies the clock skew and clock period till the wave-pipelined circuit operates satisfactorily. The self test circuit is used for testing the correctness of the operation.

The block diagram of a wave-pipelined circuit with BIST is given in Figure 8. This is obtained by including the test vector RAM, a signature analyzer, programmable clock and clock skew generators and FSM blocks in the circuit of Figure 5. The signature analyzer consists of a pseudorandom binary sequence (PRBS) generator-based signature generator and a comparator.

4.1.1. FSM block

The FSM block generates the control signal to choose between the normal mode and the self test mode and this is applied to the select input of multiplexer. In the self test mode, the FSM systematically varies the clock skews and

clock periods. For each clock frequency and skew, the self test circuit generates the test inputs, applies them, generates the signature, compares it with the expected result, and finally generates a flag indicating the match. The FSM progresses with the testing till the frequency at which the DUT works for at least 3 or more skew values is found. The operating skew value is chosen to be the middle value so that the DUT would reliably work even if the delays change due to environmental conditions.

4.1.2. Signature generator

For testing the correctness of the circuit, N test vectors may be fed one after another and the N outputs obtained should be compared with the expected outputs. In order to minimize the number of comparisons, a unique signature is generated out of the N outputs and it is compared with the signature corresponding to the expected outputs. The signature generator consists of a pseudorandom binary sequence (PRBS) generator with multiple data input [13]. The successive output of the output register is XOR'ed with the state of the PRBS to generate the next state.

4.1.3. Programmable clock generator

The circuit diagram of the programmable clock generator is shown in Figure 7. Programmable clock skew generator may be implemented using only delay blocks. The clock generator is implemented using only the LUTs and interconnects (nets) and is proposed for the first time in [10]. The interconnects are manually chosen using the FPGA layout editor in [10]. Programmable feature is proposed for the first time in this paper. In this case, the interconnect delays are selected using the multiplexer. The number of possible interconnect delays is restricted to minimize the overheads due to the additional LUTs required for the introduction of the delay and the multiplexers. Hence, only a coarse variation in the delay values can be achieved. In Figure 7, inputs C0–C3 are the programmable select inputs, which determine the actual clock frequency.

The operating frequency of the wave-pipelined circuit is expected to lie between that of nonpipelined circuit and pipelined circuit. Hence, the minimum and maximum frequency of the clock generator should correspond to the maximum operating frequencies of the nonpipelined circuit and pipelined circuits, respectively. The approximate values of the clock periods of these circuits for the implementation of the β block on FPGA are 5.6 nanoseconds and 7.4 nanoseconds, respectively. The values of $D_{\max}$, $D_{\min}$ for the α block are 15.302 nanoseconds and 7.34 nanoseconds, respectively. The programmable clock and skew generator are designed such that the clock period can be varied from 8.4 nanoseconds to 20.6 nanoseconds in steps of 0.8 nanoseconds and skew can be varied from 12.3 nanoseconds to 26.2 nanoseconds in steps of 0.9 nanoseconds approximately. The same exercise is carried out for β, γ, and δ blocks using the synthesis report. A single clock generator is used for all the four blocks. Separate skew generators are used for each of the four blocks. By varying the select line through FSM or processor, different clock periods and skew values are achieved.

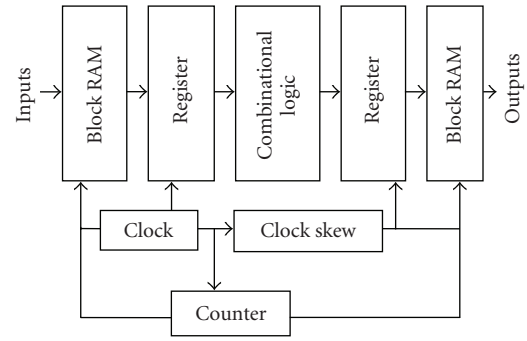

Figure 9: SOC approach for wave-pipelined circuit.

4.1.4. Test vector generation

In principle, the number of test vectors required for an M input combinational logic circuit is 2^M. If the value of M is small, exhaustive testing of the circuit may be carried out by generating the test inputs through an M-bit counter and checking the signature after the counter completes one full cycle. However, some of the inputs may contribute more to $D_{\max}$ than the others. For higher order circuits, exhaustive testing would require a large testing time. In this case, a set of random vectors may be used for testing the wave-pipelined circuits.

4.2. SOC approach for wave-pipelined circuits

The block diagram of a wave-pipelined circuit which is tuned using the SOC approach is shown in Figure 9. It consists of programmable clock, clock skew generator, and block RAMs for storing the inputs and output vectors of the wave-pipelined circuit.

During normal operation, the block RAM contains the array of data to be processed. In the test mode, the block RAM contains the test data. During the testing mode, the processor writes the test vectors into block RAM, systematically applies the select inputs for the clock generator and clock skew blocks and uploads the results stored into the output block RAM for each combination of select inputs. It then checks the results with the expected results.

4.2.1. Implementation of programmable clock and clock skew generator on SOC

The programmable clock and clock skew generator may be implemented in the custom block using circuit given in Figure 7. In this case, the LUTs are replaced by logic elements (LEs). It may be noted that an external clock may be multiplied by an arbitrary number using the Altera mega core function altclklock. However, the multiplication factor has to be specified at the synthesis time and hence the clock frequency cannot be dynamically altered as in the scheme given in Figure 7. The select inputs for the clock as well as

Table 1: Area and speed performance of one-level forward 2D DWT for 32×32 subimages.

Lifting scheme	Slices used	Speed (MHz)	Number of registers
Nonpipelined	836	54.45	611
Pipelined	1110	87.54	1670
Hybrid WP-P	836 [188]*	75.75	611 [85]*

*denotes additional overhead for testing WP circuits.

Table 2: Area and speed performance of one-level forward 2D DWT for 32×32 subimages.

Lifting scheme	Slices used	Speed (MHz)	Number of registers	Power at normalized frequency (mw/Hz)
Nonpipelined	703	117.83	375	—
Pipelined	782	203.92	671	158.97
Hybrid WP-P	703 [30]*	147.5	375[8]*	179.58

skew blocks and the data inputs to the wave-pipelined circuit may be applied and varied through the on-chip processor.

5. IMPLEMENTATION RESULTS OF 2D DWT USING BIST APPROACH

A 128×128 imagewith 8 bits per pixel is used for testing the three schemes. The 2D DWT scheme is implemented using the lifting blocks with 9/7 biorthogonal filters and BW-KCM multipliers. The lifting multiplier constants $(\alpha, \beta, \gamma, \delta, \xi)$ are assumed to be of 8 bits each and the input samples are assumed to be of 11 bits. For 2D DWT, image block of size 32×32 is assumed. The one-level 2D DWT is implemented on Xilinx Spartan II FPGA using BIST approach. A personal computer (PC) is used for the realization of the FSM. The interface used between PC and FPGA is same as that described in [10].

5.1. Implementation results on 2D DWT using Spartan-II XC2S100PQ208-5

The implementation of one-level 2D DWT for image block of size 32×32 is carried out for lifting scheme using Spartan-IIXC2S100PQ208-5. For the hybrid wave-pipelined circuit, the number of logic elements, number of registers, and maximum operating frequency are computed and the results are given in Table 1. Overheads required for the wave-pipelined circuits are also shown in Table 1.

From Table 1, it may be concluded that for the lifting scheme, the method using hybrid WP-P BW-KCM is faster than nonpipelined BW-KCM by a factor of 1.4 and requires the same area. The pipelined BW-PKCM is in turn faster than the hybrid WP-P BW-KCM by a factor of 1.2 and this is achieved with the increase in the number of registers by a factor of 2.73 and the increase in the number of slices by a factor of 1.32.

6. IMPLEMENTATION RESULTS OF 2D DWT USING SOC APPROACH

The BIST approach requires a number of overheads such as FSM, signature generator and test vector RAM [14]. Instead of using a dedicated circuit such as BIST, a processor may be used to carryout the tuning and retuning tasks [15]. The tasks performed in software use the on-chip processor. The hardware block may use wave-pipelining and it may be retuned by the on-chip processor periodically. Hence, the retuning task may be time shared with the other tasks performed by the processor.

The block diagram of a wave-pipelined 2D DWT is implemented along with the Nios II soft-core processor and the former is added as the custom block to the Nios II using SOPC builder. The program to be executed by the Nios II is written in C/C++ and the custom block is invoked as a function in the C/C++ program. A C++ program is written to read and write from the block RAM in the custom block. When the C program is run, it systematically varies the "select" inputs for the clock and clock skew blocks, and uploads the content of the output block RAM. It compares this with the expected results. The clock and skew are adjusted till the match occurs for at least three consecutive clock skews. The operating clock and clock skew of the wave-pipelined circuit is fixed at the middle value and from now on, the custom block works without any intervention from the Nios II processor. Only when retuning is required, the Nios II processor interacts with the custom block.

6.1. Implementation results on 2D DWT using Cyclone-II EP2C35F672C6

For the hybrid wave-pipelined circuit, the number of logic elements, number of registers, maximum operating frequency, and power dissipated are computed and the results are given in Table 2. Overheads required for the wave-pipelined circuits are also shown in Table 2. To reduce the hardware complexity the same horizontal filter is used, instead of vertical filter for computing LL1.

From Table 2, it may be concluded that for the lifting scheme, the method using the hybrid WP-P BW-KCM is faster than nonpipelined BW-KCM by a factor of 1.25. The scheme with Baugh-Wooley pipelined constant coefficient multiplier is in turn faster than the hybrid WP-P BW-KCM by a factor of 1.38 and this is achieved with the increase in

Input image (a)

LL1 component using BW- hybridWPK CM (b)

Figure 10: LL1 component compared with input image.

the number of registers by a factor of 1.78 and the increase in number of LEs by a factor of 1.11.

In order to assess the superiority of hybrid wave-pipelining with regard to power dissipation, both hybrid wave-pipelined and pipelined circuits are operated at the same frequency (corresponding to the maximum operating frequency of the hybrid wave-pipelined circuit) and the power dissipated for the two approaches are also given in Table 2. From this Table 2, it may be noted that the pipelined circuit dissipates 1.5% more power than hybrid wave-pipelined 2D DWT. If the overheads required for hybrid wave-pipelined 2D DWT are also considered, then the pipelined 2D DWT dissipates 12.9% less power than hybrid wave-pipelined 2D DWT.

6.2. Validation of the scheme for 2D DWT

To verify the correctness and efficacy of the schemes proposed for the computation of 2D DWT, Lena image of size 128×128 with blocks (subimages) of size 32×32 pixels is used for the computation of the 2D DWT. The Lena image shown in Figure 10 is obtained by compressing the image dimension by a factor of 4 along both dimensions. Overlap of 4 pixels is used between the adjacent blocks. Totally 36 image blocks are used for the 128×128 image. This is carried out by hardware approach using FPGA. For storing the image input, outputs of the horizontal filter and the outputs of the vertical filters, the block RAMs are configured suitably.

The one-level 2D DWT is computed using the above scheme for all the 36 image blocks and merged suitably. The LL1 component of the image is shown in Figure 10. From these figures, it may be concluded that the LL1 components obtained through the FPGA implementation match well with the original image. The PSNR value is computed for the image obtained using BW-hybrid WPKCM is 28.22.

7. CONCLUSION

In this paper, two automation schemes are proposed for the implementation of the 9/7 biorthogonal filters using hybrid WP-P constant coefficient multiplier with Baugh-Wooley multiplication algorithm. The 9/7 biorthogonal filters are implemented on both Xilinx and Altera devices with the following three multipliers: BW-PKCM, BW-KCM, and hybrid WP-P BW-KCM. From the implementation results, it is verified that hybrid WP-P BW-KCM is faster than nonpipelined BW-KCM by a factor of 1.25–1.39. The scheme with BW-PKCM is in turn faster than the hybrid WP-P BW-KCM by a factor of 1.15–1.39 and this is achieved at the cost ofincrease in the number of registers by a factor of 1.78–2.73 and increase in the number of LEs by a factor of 1.11–1.32. The hybrid wave-pipelined 2D DWT dissipates 12.9% more power than pipelined 2D DWT. Because one of the challenges in the design of FPGA-based wave-pipelined circuits is the nonavailability of accurate models for the interconnects and the temperature dependence of their delays. In the absence of these models, the wave-pipelined circuits can only be operated at moderate speeds. The implementation of the automation schemes on the ASICs can lead to better performance firstly because area is not wasted by unwanted registers and secondly because models are available for interconnects in the literature. The work on the computation of two-level 2D DWT using Xilinx and Altera FPGAs and the automation schemes for the ASIC implementation of one-level 2D DWT are under progress.

REFERENCES

[1] G. Martin and H. Chang, "System-on-chip design," in *Proceedings of the 4th International Conference on ASIC (ASICON'01)*, pp. 12–17, Shanghai, China, October 2001.

[2] B. A. Draper, J. R. Beveridge, A. P. W. Bohm, C. Ross, and M. Chawathe, "Accelerated image processing on FPGAs," *IEEE*

Transactions on Image Processing, vol. 12, no. 12, pp. 1543–1551, 2003.

[3] G. Lakshminarayanan, B. Venkataramani, J. S. Kumar, A. K. Yousuf, and G. Sriram, "Design and FPGA implementation of image block encoders with 2D-DWT," in *Proceedings of IEEE Conference on Convergent Technologies for Asia-Pacific Region (TENCON '03)*, vol. 3, pp. 1015–1019, Bangalore, India, October 2003.

[4] K. K. Parhi, *VLSI Signal Processing Systems*, John Wiley & Sons, New York, NY, USA, 1999.

[5] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[6] T. Aharya and P.-S. Tsai, *JPEG2000 Standard for Image Compression Concepts, Algorithms and VLSI Architectures*, John Wiley & Sons, New York, NY, USA, 2005.

[7] C. T. Gray, W. Liu, and R. K. Cavin III, *Wave-Pipelining: Theory and CMOS Implementation*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

[8] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: a tutorial and research survey," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 3, pp. 464–474, 1998.

[9] E. I. Boemo, S. Lopez-Buedo, and J. M. Meneses, "Wave pipelines via look-up tables," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '96)*, vol. 4, pp. 185–188, Atlanta, Ga, USA, May 1996.

[10] G. Lakshminarayanan and B. Venkataramani, "Optimization techniques for FPGA-based wave-pipelined DSP blocks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 7, pp. 783–793, 2005.

[11] Altera documentation library, Altera Corporation, San Jose, Calif, USA, 2003.

[12] Xilinx documentation library, Xilinx Corporation, San Jose, Calif, USA.

[13] M. J. S. Smith, *Application Specific Integrated Circuits*, Pearson Education Asia, Singapore, 2003.

[14] G. Seetharaman, B. Venkataramani, and G. Lakshminarayanan, "Design and FPGA implementation of self tuned wave-pipelined filters," *IETE Journal of Research*, vol. 52, no. 4, pp. 281–286, 2006.

[15] G. Seetharaman and B. Venkataramani, "SOC implementation of wave-pipelined circuits," in *Proceedings of IEEE International Conference on Field-Programmable Technology (ICFPT '07)*, pp. 9–16, Kitakyushu, Japan, December 2007.