Journal of Computer Science 2 (9): 746-753, 2006
ISSN 1549-3636

Design of an IP Library for the Synthesis of Image Compression Systems

Chokri Souani, Ihsen Gazzah and Kamel Besbes
Laboratory of Microelectronics and Instrumentation μEI, Faculty of Sciences of Monastir,
Boulevard de l'environnement, 5019 Monastir, Tunisia

Abstract: To design a hardware system we may use generated Intellectual Property (IP) cores from a library with optimized parameters. The IP parameters are area minimization, memory use optimization and speed acceleration. The designed system depends on such IP parameters. In this paper, we propose the design of an IP library optimized with different constraints. The created IPs may be used to implement optimized hardware architecture for discrete wavelet transform. Here, we take as a realization example the one dimension Discrete Wavelet Transform (DWT) with multi-resolution-level decomposition. We show that it is possible to design a hardware system with adjusted constraints. Moreover, it is possible to merge different structures. The design results in very flexible system architectures.

Key words: Intellectual Property, compression, Discrete Wavelet Transform, Architecture

INTRODUCTION

The increasing demand for electronic devices that can handle multimedia contents, and the expansion of wireless communication networks have set new challenges to the embedded system modeling, design and verification tasks. Applications involve more and more complex data and functionalities while their architectural realizations have to cope with heavy constraints such as high integration density, high reconfigurability, real-time performance and low power consumption.

The classical approach for designing a complex system is based on a top-down refinement flow where an initial abstract specification of the application is progressively and hierarchically decomposed into interacting subsystems. Such a design flow provides a wide freedom of choice for partitioning a system into subsystems and mapping subsystems into hardware or software, so that a wide variety of architectural instances can be proposed for a given application. Because applications are getting more complex, an increasing number of design decisions must be taken before reaching an architectural realization. As a result, at the beginning of the design flow, it is more and more difficult to foresee which design decisions will actually yield an architecture that satisfies the application requirements.

In order to speed up the design and verification of a complex system, intensive re-use of pre-designed, pre-verified hardware and software components—known as virtual components (VCs) or intellectual property (IP) cores—has already become an unavoidable practice [1].

Industrial groups such as the virtual socket interface alliance (VSIA) [2] have issued a set of recommendations and standards in order to facilitate the interaction between IP providers and users (IP integrators) and make the re-use flow more reliable. The VSIA recognizes three abstraction levels for IP synthesizable models: ''hard'' IP cores are provided as a fixed-technology placed-and routed layout of the component; ''firm'' IP cores are provided as an optimized gate-level netlist targeting a given technology and can be inserted into the system synthesis flow at the physical synthesis stage; ''soft'' IP cores are described at the register-transfer level (RTL) and are technology independent. The latter are inserted into the system synthesis flow at the RTL and logic synthesis stage.

The resulting hardware system may impose some constraints for a given application. The IPs to be used must be the most flexible as possible, that is, it may let the designer choose the architecture from a large number of proposed ones. It must be possible to merge architectures in order to get an optimized hardware. For some reasons we may ask for minimizing memory use, or minimizing hardware use, or optimizing the timing or other constraints.

Many authors developed IP cores which are optimized for some constraints such as the area, the clock frequency, or the memory use, varying only the generic parameters. For a given system, the user may not use memory or may use a limited quantity of memory. In other cases, the objective is to design the fastest possible system.

In this paper, we present alternatives of development in the form of IP cores for the 1D wavelet transform, which is possible to extend to a 2D wavelet design. The system's designer thus has a multitude of

Corresponding Author: Chokri SOUANI, Laboratory of Microelectronics and Instrumentation μEI, Faculty of Sciences of Monastir, Boulevard de l'environnement, 5019 Monastir, Tunisia

choices among his own criteria and he can choose a solution among some.

We use the standard 9/7 filter as an example, which is compatible with the JPEG2000 standard. We cross some of the possible solutions by the realization and the conception of IPs, aiming reliable and optimized designs.

The paper is organized as follows: next section provides some information about the wavelet based image coding standard JPEG2000 and explains several wavelet transform algorithms. In Section 3, DWT architectural models and their reusability issues are discussed. In order to overcome their flexibility limitations, we detail the possible generated architectures for designing a 1-D DWT IP core. To demonstrate the flexibility of our IP cores, results of combining the different architectures are presented. The conclusions are given in Section 4.

DWT FOR EMBEDDED IMAGING SYSTEMS

The JPEG committee has recently released its new image coding standard, JPEG 2000, which will serve as a supplement for the original JPEG standard introduced in 1992. JPEG 2000 implements a new way of compressing images based on the wavelet transform, in contrast to the discrete cosine transform (DCT) used in the original JPEG standard. The state of wavelet-based coding has improved significantly since the introduction of the original JPEG standard. Prior to JPEG 2000, wavelet-based coding was mainly of interest to a limited number of compression researchers. Since the new JPEG standard is wavelet based, a much larger audience including hardware designers, software programmers, and systems designers will be interested in wavelet-based coding. The RTL design and synthesis techniques are still well-known and widely accepted.

Our objective focuses on generating IPs for computation-intensive functions such as those involved in signal and image processing applications. First of all, we emphasize the fact that a single function (such as FIR filtering) can be declined into many versions depending on functional parameters (number of taps, cut frequency, etc.) Moreover, various architectures can be considered (MAC-based, ADD/ MULT tree, systolic) depending on the integration constraints.

Due to the variety of application profiles it supports, the JPEG2000 compression scheme—and particularly the wavelet transform investigated in this paper—is a typical case of such a complex, highly customizable function [3]: all its features can easily be captured into a single behavioral model, but many different architectures should be designed in order to fulfill the various application profiles and potential integration constraints.

The JPEG2000 standard restricts the use of the 5/3 wavelet to lossless compression and the 9/7 wavelet to lossy compression. Recursive application of the whole filtering scheme is performed over the successive approximation images until the desired resolution depth has been reached. The inverse transform relies on the dual 1-D filter banks applied in the reverse order [4]. A full introduction to wavelets is beyond the scope here but can be found elsewhere [4]. The generic form for a one-dimensional (1-D) wavelet transform is shown in figure 1.

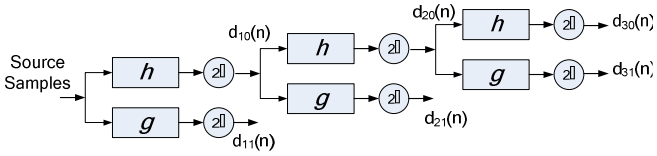

Fig. 1: A 3-level, 1-D wavelet decomposition. The coefficient notation $d_{ij}(n)$ refers to the jth frequency band (0 for low and 1 for high) of the ith level of the decomposition.

Here, a signal is passed through a lowpass and a highpass filter, h and g, respectively, then down sampled by a factor of two, constituting one level of transform. Multiple levels or "scales" of the wavelet transform are made by repeating the filtering and decimation processes on the lowpass branch outputs only. The process is typically carried out for a finite number of levels K, and the resulting coefficients, $d_{i1}(n)$, $i \in \{1,\ldots,K\}$ and $d_{K0}(n)$ are called wavelet coefficients.

The spectrum of the signal to be analyzed is recursively decomposed into an approximation subband (low frequencies) and a detail subband (high frequencies) at the immediately lower resolution [4]. In digital signal processing applications, only a partial decomposition is performed: details are extracted until a satisfyingly low resolution has been reached (e.g. with 3–5 resolution levels in most image compression applications), leaving a tiny set of low-resolution approximation coefficients after the last decomposition resolution level.

Two filter banks are supported in part I of the JPEG2000 standard [3]. The first one is Daubechies' 9/7 biorthogonal filter bank. The second one is the 5/3 biorthogonal filter bank, also known as the (2,2) biorthogonal wavelet designed by Cohen, Daubechies and Feauveau. The ''9/7'' and ''5/3'' filter banks are named after the respective number of taps in their low-pass/high-pass filters. The JPEG2000 standard restricts the use of the 5/3 wavelet to lossless compression. Besides being nearly orthogonal, the 9/7 set has been shown experimentally to give very good compression performance and has been used extensively in image compression applications [4].

Multi-level decomposition introduces another dimension in the filtering process. Performing the multi-level 1-D or 2-D DWT requires defining a scanning order taking all dimensions—space and

resolution—into account. While the choice of a wavelet filtering structure has an influence over the computational complexity of the algorithm, the scheduling of the filtering passes will have a significant impact on the memory organization. Two scheduling algorithms can be considered: the Pyramid Algorithm (PA) and the Recursive Pyramid Algorithm (RPA) [5]. The PA gives a direct implementation of the DWT using filter banks. It consists in computing the low-pass and high-pass coefficients one resolution level after the other from the highest (j = 1) to the lowest (j = J) resolution.

The RPA allows computing a low-pass/high-pass pair of coefficients at a given decomposition level as soon as enough data from the previous level have been produced.

This algorithm assumes that all low-pass samples at a given level have been computed before starting the next level.

Table 1: Estimated memory requirements for pyramid (PA) and recursive pyramid (RPA) algorithms

Scheduling algorithm	Memory size at level j	Total memory size
PA	N/2j	N/2
RPA	C	C x J

Table 1 compares the memory requirements of the two aforementioned scheduling algorithms. It can be noted that the memory required by the PA is of the same order of magnitude as the number of input samples N. It does not depend on the number of decomposition levels J. On the contrary, the memory size required by the RPA does not depend on N, but can be estimated as the product of the number of decomposition levels J with the number C of data inputs of the filter bank. In most cases, CxJ is far lower than N/2 so that the RPA requires less memory than the PA [5].

ARCHITECTURES OF THE DWT

The conception alternatives of a basic filter for the wavelet transform are numerous. Certain authors choose a single solution which they consider optimal. The IP's parameters allow making a generic filter that has a fixed architecture. We propose a set of architectures in the form of IP cores. In this paper we develop a set of possible architectures for the 9/7 filter.
To be reusable in a wide range of applications, a JPEG2000 encoder should have a high degree of flexibility in order to support the variety of application profiles (e.g. tile size, number of wavelet decomposition levels, choice of wavelet filters) and high performance (trade-off between speed, gate count, memory usage and power consumption).

Flexibility is usually addressed through software implementation where a variety of programs can be downloaded into a fixed processor-based platform. However, general-purpose processors or Digital Signal Processors have a limited degree of computation parallelism so that their performance may be insufficient in some cases. When higher computation speed is required, hardware solutions are usually preferred. A first type of hardware implementations can be denoted as ''programmable architectures'', which means that the profile parameters are provided as inputs to the component. Such architectures may be over-dimensioned in some cases and may contain useless hardware if some parameter values are never used by a given application.

Moreover due to its generality, such architecture cannot be guaranteed to work at optimal speed for all possible sets of parameter values. Due to the variety of application profiles the JPEG2000 standard supports, this kind of architecture cannot be considered. A second type of hardware implementations concerns dedicated architectures, designed and optimized for only one set of parameter values, which can be satisfactory in many embedded applications. Such architecture allows high speed performance, but it is not flexible at all so that its re-usability scope is very limited. As a consequence, a different architecture must be designed and optimized for each application profile.

Various 1-D architectural models have been proposed in the literature. Most of them are based on FIR filter banks while only a few implement the lifting scheme. Each of these architectures have strengths and weaknesses with respect to their complexity—expressed as the amount of hardware required to perform the computation, to control the evolution of the computation and to store the intermediate data—and their flexibility. Functional flexibility expresses that the architecture can be adapted by changing the functional parameters in the original specification (e.g. wavelet basis, number of decomposition levels). Structural flexibility expresses that the architecture has a periodic structure that can be easily enriched by replicating a given hardware pattern.

A 1-D DWT architecture can be based on various filtering structures using Multiplier-Accumulator (MAC) and registers: serial filters are implemented using a MAC chain whereas parallel filters use a balanced tree of adders [5]. The data path can be optimized using various techniques based either on arithmetic simplifications or on component sharing. An example of possible arithmetic simplifications is provided by linear-phase filters [6]: the symmetry of the filter coefficients allows factorization of the product expression, which will result in reduction by half of the number of multipliers. Special cases of wavelets allow other arithmetic simplifications based on exploiting simple relationships between filter coefficients [7].

DWTs give a possibility to share arithmetic operators inside a filtering pass. In fact, one can notice that only half of the samples are kept after filtering. A manipulation known as polyphase decomposition consists in splitting the operator tree into two sub-trees and computes each of them alternatively along the input sequence [6].

Architectural models at the filter level : A resolution level of decomposition consists of two filters, a low-pass and a high-pass (see figure 2). At the output of every filter the coefficients are decimated, that is one coefficient is kept for two calculated samples.

It is not necessary to calculate all the coefficients. Once, we calculate a coefficient and then we propagate the sample by shifting it through the registers chain. Thus, we have 50% of computation activity for each filter. Both filters are active simultaneously. Due to decimation, both filters are also inactive simultaneously. It is then possible to activate the first filter for the first sample. The second filter is active in the next period when samples are propagated and the first filter is not active. Therefore, we obtain 100% of activity of calculation.

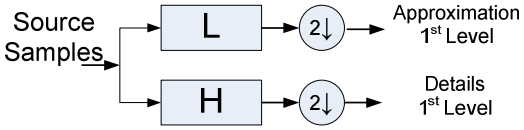

Fig. 2: Filter unit composed of the high pass (H) and the low-pass (L) filters. The symbol '2↓' means the decimation by 2.

Various architectures can be considered for the design of this decomposition level. We consider three possible architectures with the same generic parameters:

- MAC-based architecture:

The MAC unit is based on realizing a cumulative sum of the multiplied coefficients (figure 3(a)). The unit computes the inner product by means of the classical multiply/accumulate algorithm (for i=1 to n do $S_i=a_i*b_i+S_{i-1}$ end for), with A = (a_1,.., a_n), B = (b_1,…, b_n) vectors of n integer elements a_i and b_i, which are signed two's complement binary numbers. The initial result is $S_0=0$ and the final result is S_n [8], [9].

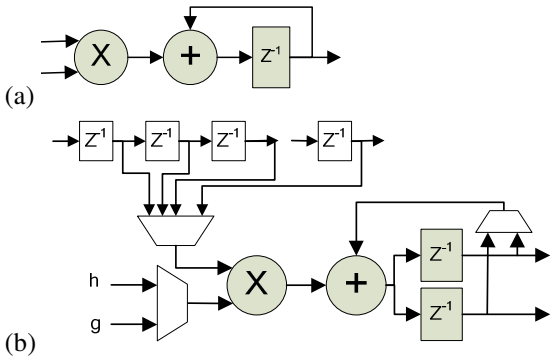

Fig. 3: (a) The simple MAC unit (b) The proposed MAC-based architecture for one decomposition level

For the MAC filter structure, we may use one FIFO pile (figure 3(b)) where the input samples are stored. The FIFO pile depth is equal to the filter taps. Because we have two unequal filter taps, the FIFO depth is equal to the maximum taps number. For the 9/7 filter the maximum filter taps is nine, and then the FIFO depth is equal to nine. A multiplexer chooses the corresponding filter coefficient. It switches for the high-pass filter coefficients (g) and the low-pass filter coefficients (h). The multiplier reads the coefficients from an addressable ROM memory. The FIFO pile is also addressed at the same time (the same address) as the memory of the filters coefficients.

The filtering unit begins by calculating the coefficients of the first filter by going through all the pile. Then, because of the decimation process, we store a new sample and we activate the second filter by going through all the pile.

In our designed architecture, the FIFO pile is not involved in the MAC-based cell. The parameters of the designed IP are the number of bits of the input samples (bus width) and the number of filter taps. The FIFO pile and the ROM memory are designed separately and have the same generic parameters.

- Semi-Systolic architecture:

It is possible to use a set of cascaded registers. Each register output is multiplied by the adequate filter coefficients. The result is then summed up to give the wavelet coefficient. One elementary cell is defined as a register with a multiplier and an adder as shown in figure 4(a). The number of cells is equal to the number of filter taps.

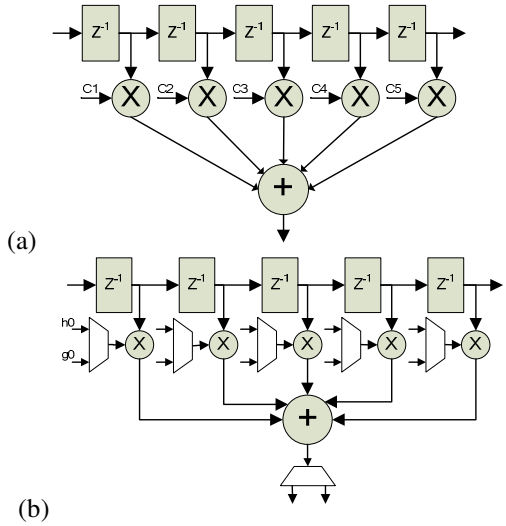

Fig. 4: (a) Example of a simple semi-systolic architecture. (b) The designed optimized semi-systolic architecture.

In this case also, and because of the decimation, it is possible to alternate the functioning of both filters. First, we store a sample and we process the coefficients of the first filter (h). Next, we store the following sample but at this time we activate the second filter (g).

Multiplexers are used to allow commuting between the coefficients of the first or the second filter (figure 4(b)).

The designed architecture involves a FIFO. The FIFO depth is equal to the maximum filter taps. The parameters of the designed IP are the number of bits of the input samples (bus width) and the number of filter taps. The filter coefficients are defined as constants and are stored into a ROM memory.

- Linear-phase architecture:

The regularity in the expressions of each filter coefficients is very suitable for mapping them into a systolic-like (semi-systolic) algorithm for an implementation using a VLSI architecture.

For the 9/7 filter it is observed that the coefficients are symmetric. Thus, it is possible to reduce the number of multipliers. Let's consider the 9 taps filter as example. Let y be the output coefficient. Then, y is expressed as $y = C_{-4} \bullet a_{-4} + \ldots + C_{-1} \bullet a_{-1} + C_0 \bullet a_0 + C_1 \bullet a_1 + \ldots C_4 \bullet a_4$ where C_i are the filter coefficients and a_i are the input samples. Because of the symmetry we have $C_{-i} = C_i = C_{-i,i}$ therefore, y may be rewritten as: $y = C_{-4,4} \bullet (a_4 + a_{-4}) + \ldots + C_{-1,1} \bullet (a_1 + a_{-1}) + C_0 \bullet a_0$.

The number of multipliers used is thus reduced from nine to five. Generally we may gain a number of multipliers in the order of $\left\lceil \frac{number\ of\ taps}{2} \right\rceil$.

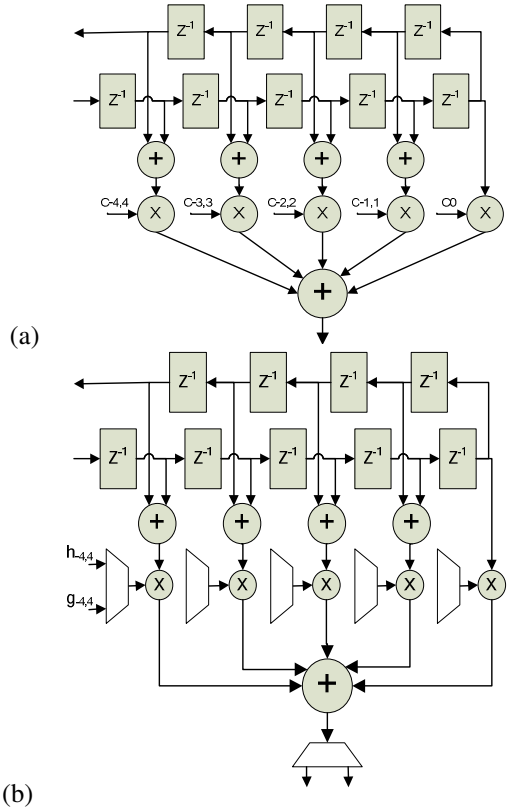

Fig. 5: (a) Semi-systolic architecture. (b) The Linear-phase like proposed architecture.

The designed architecture is represented in figure 5(b). Multiplexers are used to alternate the filter coefficients between h and g.

- Implementation results:

The proposed filter architectures are evaluated by comparing the number of multipliers, the number of adders and the number of registers used in the design. The cycle-length is another critical parameter which expresses the number of clock periods to calculate a coefficient. The MAC-based filter needs to go through all the FIFO and accumulating partial products in order to produce one coefficient. One clock period is necessary to store a sample into the FIFO and nine clocks to activate the MAC-based filter. As an example of implementation, we choose the 9/7 filter with a data bus width of 16bits.

Table 2 shows the comparative parameters for the three proposed filter architectures. The linear phase architecture uses half the number of multipliers.

Table 2: Complexity of 9/7 filter banks for various filter implementation structures

	# Mult	# Add	Cycle length
MAC-based	1	1	10
Semi-Systolic	9	8	1
Linear-Phase	5	8	1

In table 3, hardware implementation results are presented. Here, we use the Xilinx Spartan3 as a target hardware and a data bus width of 16 bits for input/output. The MAC unit is the simplest architecture but it needs ten cycles to process one coefficient. The semi-systolic architecture uses the greater number of multipliers and thus needs more hardware devices. Figure 6 depicts the total number of equivalent gates and the throughput for each filter architecture.

Table 3: Hardware implementation on FPGA

	Equivalent gates (K Gates)	Max. Freq. (MHz)	Throughput (KSamples/sec)	Ratio Thr./Eq.gates
MAC	6.7	62.6	6.3	32.1 %
Semi-systolic	38.4	63.2	63.2	57.5 %
Linear-Phase	23.7	69.1	69.1	100.0 %

The parameter 'Throughput' measures the speed of calculating the coefficients. This parameter is expressed as the 'Maximal frequency/Cycle length'. To compare the architectures we introduce the coefficient 'Throughput/equivalent gate' ratio. The normalized parameter 'Throughput/equivalent gate' is presented in figure 7.

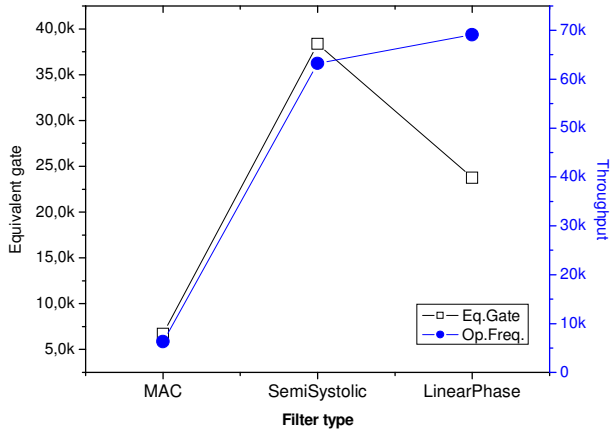

Fig. 6: Equivalent gate and throughput of the designed filters.

Based on figure 7, the linear phase filter is the most efficient because is allows 100 % of throughput per equivalent gate ratio.

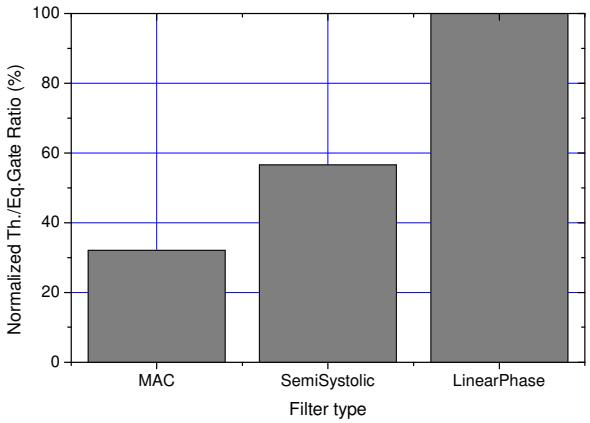

Fig. 7: The throughput per equivalent gate ratio.

Architectural models for multi-level decompositions: ''Unfolded'' architectures are a direct implementation of the 1-D DWT, that consist in cascading the filtering blocks. ''Folded'' architectures represent a high degree of component sharing consisting of re-using a single filter bank at different decomposition levels [5]. This filter bank is coupled with a ''routing network'' for storing and directing intermediate data (figure 8). In such architecture the routing algorithm will have a significant influence over the lifetime of the data produced at each level. Two such algorithms have already been presented, namely the PA and the RPA. It is shown that the latter results in more efficient memory usage. Various routing architectures are listed in [5]: multiplexer-based; bus-based (semi-systolic); with distributed control over storage cells (systolic); RAM-based; with shared registers (reduced storage).

In [6], a scalable MAC-based architecture for the 1-D DWT is presented, allowing generating wavelet filters of various lengths by simple structural extension and coefficient value setting. Structural flexibility can also be used in multi-level architectures, either unfolded by choosing the appropriate number of cascaded filter banks, or folded by relying on systolic routing networks. Such flexibility can be easily captured by RTL design methodologies (i.e. by making use of VHDL-like ''generic'' clauses for specifying parameterized architectures, and ''generate'' statements for controlling the instantiation of the hardware structures, either conditionally or iteratively).

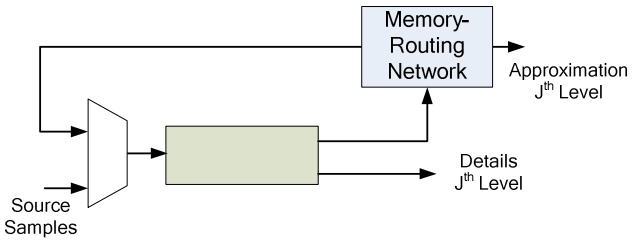

Fig. 8: Simple folded architecture

To design N-level decomposition, we my calculate the coefficients from the first level and store the results in memory. At the end of the process, we read coefficients from the memory to calculate the next resolution level coefficients.

A routing network bloc may use a memory for storing intermediate coefficients. In this case, only one filter structure is used. We may then use the MAC structure (MAC) filter, or the semi-systolic (SS) filter, or the Linear-Phase (LP) filter previously presented.

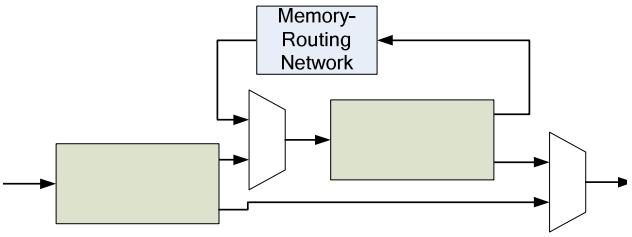

Fig. 9: Folded doubled architecture

Except for the first resolution level which operates at 100 % of activity, the second resolution level runs at 50 % of activity with respect to the first one because of the decimation. The third resolution level runs at 25 % of activity with respect to the first resolution level, and in 50 % with respect to the previous resolution level.

By combining the activity of the second resolution level with the third one, it is possible to minimize the number of filters to be used. A memory is then necessary for saving the data and temporary coefficients of the second resolution level to be used by the third resolution level. We may then use the MAC structure (MAC) filter, or the semi-systolic (SS) filter, or the Linear-Phase (LP) filter previously presented.

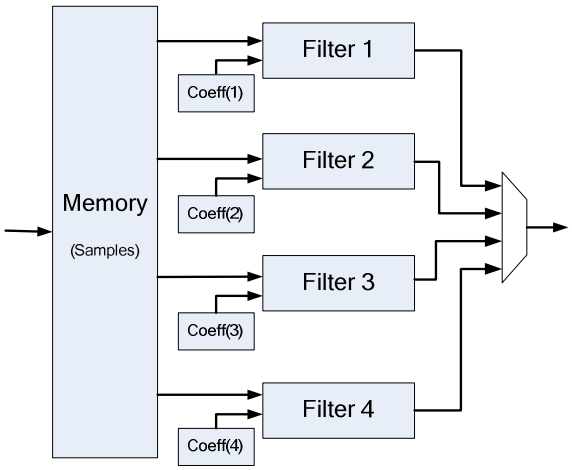

Fig. 10: The parallel architecture

In [8] and [9], an architecture based on reducing the filter bank cascade into one equivalent filter per subband is presented (figure 10). As a result, each output is associated with a different sub-sampling factor depending on the resolution of the corresponding subband. Practically, a scheduler controls the computations being performed by activating each filter at the appropriate rate. Since several filters may be active at the same time, the filtered samples are buffered so that one sample per cycle is output. Since all filters are directly connected to the source input, they can share the same memory unit. This implementation allows significant hardware savings by sharing arithmetic operators within each filter (figure 11).

Table 4: Hardware implementation on FPGA for three-level decomposition.

	# Mult			# Add		
	MAC	SS	LP	MAC	SS	LP
Folded	1	9	5	1	8	8
Folded doubled	2	18	10	2	16	16
Parallel	4	18	20	4	32	32

Table 5: Hardware implementation on FPGA for a three-level decomposition.

Wavelet Architecture		Equivalent Gate K Gates	Max. Frequency (MHz)	Throughput depending on memory
Folded	(MAC)	10.0	61.6	Yes
	(SS)	40.1	62.0	
	(LP)	26.1	68.3	
Folded doubled	(MAC)	16.4	61.3	
	(SS)	84.5	61.9	
	(LP)	52.1	69.0	
Parallel	(MAC)	30.7	40.0	No
	(SS)	172.6	45.6	
	(LP)	106.8	55.3	

Because the throughput depends on the temporary storage memory speed, the device operating frequency is that of the end bloc of the architecture. In each device namely, the folded and the folded doubled, a memory is used for storing intermediate results. Only the parallel architecture does not need intermediate memory and operates as one coefficient output at each input coefficient. The parallel architecture does not use memory to store intermediate results but uses particular addressable FIFO with a limited number of registers.

It should be noted that the parallel architecture using a semi-systolic (Parallel-SS) and the parallel architecture using a linear phase filter (Parallel-LP) allow a throughput equal to the maximum frequency.

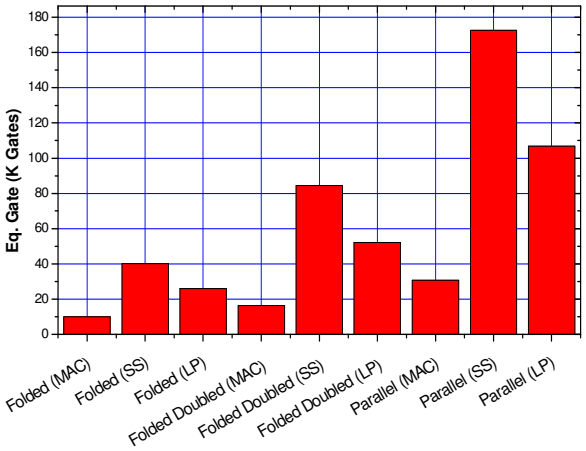

Fig. 11: Hardware implementation on FPGA.

The principle of the folded wavelet cores is to spread the computations of wavelet coefficients over multiple computation cycles. The amount of time available depends upon the desired throughput, which is linked to the folding parameters [10]. Case studies for stand-alone and cascaded silicon cores for various wavelet algorithms are reported in [10].

The designs have been captured in VHDL and are portable across a range of foundries, target technologies and are applicable to FPGA and ASIC implementations. The use of a hierarchical approach in the creation of the various described silicon generators means that tightly designed smaller blocks are used to create larger library blocks, which are in turn used to create the described circuits [10].

CONCLUSION

In this paper, a variety of architectures for designing re-usable IP cores for image processing are presented. The cores implement a 1-D discrete wavelet transform (DWT) algorithm that can be integrated in a JPEG2000-compliant image encoder. The key concepts of our methodology are re-usability—allowing accelerating the design of image compression devices within imaging systems—and customizability, allowing tighter adaptation of an IP core to the functional and system-level requirements. IP synthesis and architectural exploration are performed to allow tremendously faster architecture generation. It is

possible to customize an IP core by selecting the architecture that best suits the requirements.

The synthesis results which were shown, demonstrate the possibility to design a variety of 1-D DWT architectures for JPEG2000 with varying complexity and performance starting from elementary architectures. Our methodology is naturally oriented towards computation-intensive algorithms and can be easily generalized to most signal processing functions.

REFERENCES

1. M. Keating, P. Bricaud, 1999. Reuse methodology manual for system-on-a-chip designs. Kluwer Academic Publishers, Dordrecht, third ed.,
2. VSI Alliance, Architecture document. (http://www.vsi.org)
3. ISO/IEC JTC1/SC29, March 2000 & December 2000. JPEG2000 Part I & II Final Committee Draft, FCD154446& 1 FCD15444-2
4. Bryan E. Usevitch, 2001. A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000, IEEE Signal Processing Magazine, pp: 22- 35
5. G. Savaton, E. Casseau, E. Martin, 2006. Design of a flexible 2-D discrete wavelet transform IP core for JPEG2000 image coding in embedded imaging systems, Elseiver Signal Processing, Volume 86, Issue 7, pp: 1375-1399
6. S. Masud, J.V. McCanny, 2001. Design of silicon IP Cores for biorthogonal wavelet transforms, Journal of VLSI Signal Processing, 29, pp: 179-196
7. S. B. Pan, R. H. Park, 2002. VLSI architectures of the 1-D and 2-D discrete wavelet transforms for JPEG2000, Elseiver Signal Processing, 82(7), pp: 891-992
8. Chokri Souani, Mohamed Atri, Mohamed Abid, Kholdoun Torki, Rached Tourki, 2000. Design of new optimized architecture processor for DWT, Real-Time Imaging 6, pp: 297-312
9. Chokri Souani, Mohamed Abid, Kholdoun Torki, Rached Tourki, 2000. VLSI design of 1-D DWT architecture with parallel filters, Integration the VLSI journal 29, pp: 181-207
10. Shahid Masud, John V. McCanny, 2004. Reusable IP cores for discrete wavelet transform applications, IEEE Trans. On circuits and Systems-I, vol. 51, N° 6, pp: 1114-1124