2

A Scalable Architecture for Discrete Wavelet Transform on FPGA-Based System

Xun Zhang
Institut Superieur d'Electronique de Paris-ISEP
France

1. Introduction

In recent year, the Forward and Inverse Discrete Wavelet Transform (FDWT/IDWT) (S.Mallet, 1999) has been widely used as an alternative to the existing time-frequency representations such as DFT and DCT. It has become a powerful tool in many areas, such as image compression and analysis, texture discrimination, fractal analysis, pattern recognition and so on. The recent and future developments of high definition digital video and the diversity of the terminals had led to consider a multi-resolution codec. In this context, the FDWT/IDWT as well as the others computational functions such as Motion Estimation (ME) are required to be scalable and flexible to support rich multimedia applications and adapt to the fast changing of standards requirement. In this background, a universal, extremely scalable and flexible computational architecture which can adapt to variable workload would be more and more important and suitable for the multimedia application in the future.
In the literature, there have been several proposals devoted to the hardware implementation of FDWT/IDWT. Some proposals(M.A.Trenas et al., 2002) (et al, 2002) (Lee & Lim, 2006)(Ravasi, 2002)(P.Jamkhandi et al., 2000)(Tseng et al., 2003)addressed the importance of flexibility and proposed programmable DWT architectures based on two types: VLSI or FPGA architecture. The VLSI architectures have large limitations in terms of flexibility and scalability compared to the FPGA architectures. Even though some recent solutions proposed programmable and scalable for either variable wavelet filters(Olkkonen & T.Olkkonen, 2010) (Lee & Lim, 2006) or the structure of FDWT, they remind, in addition to their cost, dedicated to specifics algorithms and cannot be adapted to future solutions. In another hand, the existing FPGA architectural solutions are mainly ASIC like architectures and use external off-the-shelf memory components which represent a bottleneck for data access. The possibility of parallelizing the processing elements offered by FPGAs associated to a sequential access to data and bandwidth limitations do not enhance the overall computing throughput. The very powerful commercial VLIW digital signal processor obtains its performance thanks to a double data-path with a set of arithmetic and logic operators with a possibility of parallel executions and a wide execution pipeline. However, these performances are due to a high frequency working clock. Even though these DSP has a parallel but limited access to a set of instructions, the data memory access remains sequential. The performance requirement is paid by high circuit complex and power consumption. Most of work focuses on the reuse of devices likes FPGAs for different applications or different partitions of one applications.
In order to square up these needs, we propose a novel DWT architecture and implementation method. The proposed architecture can support multi-standard by reconfiguring the

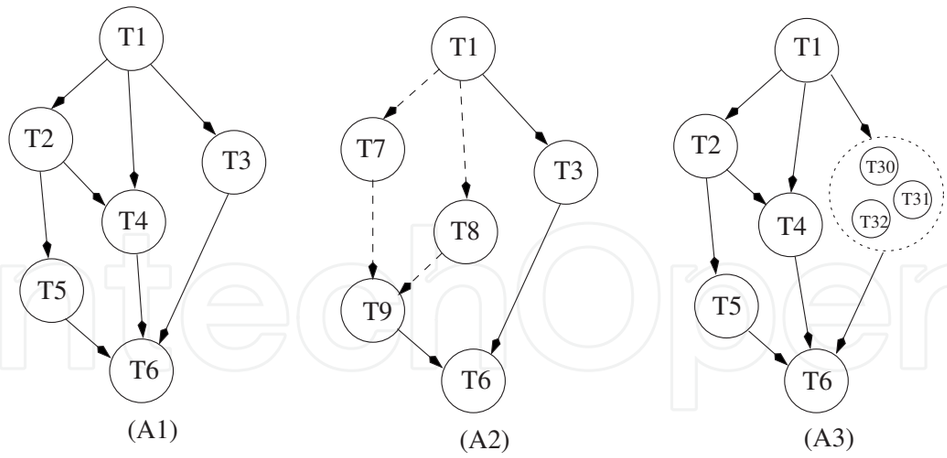

Fig. 1. Application adaptive configuration

interconnection between date memories and processing elements. Moreover, the number of processing element and its working frequency could be reconfigured dynamically. A controller plays a key role as a reconfigurable interface allowing multiple accesses to local memory, external memory through a DMA and feeding the processing element in an optimal fashion. An implementation method is developed to identify parallelism level of processing element and working frequency as well as to find out the tradeoff between power consumption and performance. In comparing with others VLSI and ASIC architecture, double size of memory can be economic in using our novel architecture.
In the following paper, we start, in Section2 by presenting a definition of adaptation in two manners: the application adaptive and task adaptive, within the system complexe context. We then give a brief overview of DWT algorithm in Section 3 where we detail a reconfigurable DWT hardware processor architecture. In order to experimentally explicit our proposed system, Section 4 focus on the detail of our proposed reconfigurable architecture which supports our DWT algorithm implementation. Section 5 focus on the implmentation, analysis and validation of system. Finally, Section 6 summarizes and concludes this work.

2. Levels of adaptation

In the multimedia computing environment, adaptation can be seen in two manners: the application adaptive and task adaptive. Following the adaptation of computing environement the different applications or different standard of one application can be switched in run-time. For example, the multimedia terminal switches it use from playing a movie to answering a video call. The task adaptive consists of the switching different versions of a task of an application, this situation can occur for instance in down scaling or up scaling situations.

2.1 Application adaptive

For a given domain, applications can be described by a set of processing tasks and sub tasks. The difference between the applications could be represented with common processing tasks and specific processing tasks. Figure 1-A2 shows an example of two applications A1 and A2 featuring common tasks (continuous lines) and specific tasks (dash lines). Switching from application A1 to application A2requires replacement of specific tasks and the communication between newly loaded tasks and common tasks. In some cases, the simultaneous execution

of two applications is required. To achieve this, different versions of specific tasks must be available.

2.2 Task adaptive

Each task of an application commonly consists of a set of sub-tasks or a set of operators depending on the complexity of task as shown in figure1-A3. To enable task adaptivity, different versions of a task for a given algorithm must be defined and characterised in terms of power, area, throughput, efficiency and other objectives. For the same task, it must be also possible to change the type of algorithm in order to adapt the application to the future standards.

In this background, the adaptability of application helps us to configure partially one part of application for adapting to a new application. The task adaptive level permits us mainly to make a small change in the task to make the application adapt to different sceneries. In this paper, we focus on the task adaptive so as to realize muti DWT processing algorithms by using partial reconfiguration technique.

3. 2-D DWT processing algorithm

A survery of 2-D DWT architecture can be referenced in the paper Olkkonen & T.Olkkonen (2010). The two dimensional ($2D$) forward discrete wavelet transform (FDWT) is a rapid decomposition in the multimedia application domain. The FDWT is computed by successive low-pass and high-pass filtering. The output of each filter is decimated that is every second value is removed halving de length of the output S.Mallet (1999). The output of each filter stage is made of transform coefficients and each filter stage represents a level of transform. The low pass result is then transformed by the same process and this is repeated until the desired level is reached. In the Inverse discrete wavelet transform (IDWT), the approximation and detail coefficients at every level are up-sampled by two, passed through the low pass and high pass filters and then added. This process is continued through the same number of levels as in the decomposition process to obtain the original signal. In this paper we will focus on the implementation of IDWT, the same approach will be applied to FDWT.

3.1 Classical processing approach

The classical approach to $2D$ decoding is to process each layer in the tree decomposition separately and to process the vertical and horizontal layers successively one after the other. The performance of this approach is strongly limited by the management of temporary data required between two successive layers and between horizontal and vertical filtering. For a $2D$ image with N rows and N columns and L levels, the amount of data to be filtered on each layer increase (for IDWT) by a factor of four from one layer to the next, and the total amount of processed data along the whole tree reconstruction process is given by the following equation:

$$D = \sum_{i=1}^{L} \frac{N \times N}{4^{i-1}} = \frac{4^L - 1}{3 \times 4^{L-1}} \times N \times N \tag{1}$$

To process a $N \times N$ image, a temporal memory of size

$$D - N \times N = (\frac{4^L - 1}{3 \times 4^{L-1}}) \times N \times N \tag{2}$$

is required. As an example, for 2 level resolution a temporal memory of 0.25 $N \times N$ size is required. For a given layer, the filtering process is achieved horizontally and vertically; thus

two read accesses and two writes accesses are necessary and the total amount of data read and written is expressed as $D_w = D_r = 2 \times D$. The memory bandwidth B, in bidirectional access case, can be considered as the production of the total amount of data processed for a frame per second (fps) $T_{df} = (D_r + D_w) \times fps$ and the number of bits N_b of a coefficient:

$$B = T_{df} \times N_b \tag{3}$$

As an example, for a gray level image of 512×512 pixels with 25 frame per second, 8 bits per pixel and 2 levels of reconstruction, a bandwidth of 260 Mb/s is required. These results illustrate the memory management problem as the main bottleneck of the classical approach.

3.2 Proposed processing approach

In order to reduce the memory size and to optimize the overall system performance, the wavelet algorithm is redesigned to exploit efficiently the inherent processing parallelism. This processing parallelism is possible if the required data is accessible in parallel, accordingly a data partitioning is used. The degree of parallelism and thus of the data partitioning will depend on the level of transformation, the number of levels and data dependency.

The proposed organization is shown in figure 2 depicting the memory fragmentation (2-a) and tasks allocation (2-b) on processing elements for two level IDWT. It is a compromise and intermediate solution between a massive parallelism and a sequential execution. The processing tasks are mainly filtering operations witch are organized and allocated to a processing element so that the among of data processed is the same. Indeed, if we consider a $W \times W$ bloc, an IDWT will be processed in three phases as shown in figure (3). In phase Φ_1 The processing element $PE1$ requires $2 \times \frac{W}{2} \times \frac{W}{2} = \frac{W \times W}{2}$ data accesses to reconstruct the LL bloc meanwhile the processing element $PE2$ can process vertically the $\frac{W \times W}{2}$ remaining data (HL and HH). In phase Φ_2, when the two processing elements terminate their executions, the LL bloc is reconstructed and the pressing element $PE2$ can resume its vertical executions on the $\frac{W \times W}{2}$ available data. In phase Φ_3, after the termination of $PE2$, data is available to process the horizontal pass on a bloc of $W \times W$data. Using $PE1$ and $PE2$ in parallel, the data processed by each PE is of $\frac{W \times W}{2}$. This architecture is scalable and can be extended to different levels of resolutions by an adequate choice of processing elements.

4. System overall architecture

With the down scaling technology, the modern chips can integrate a huge quatity of mixed grain hardware resources ranging from several hard microprocessors, hard arith- metic operators to hundred of thousand of simple gates allowing the integration of various soft cores. The prob- lem of resources management becomes then very acute especially in reconfigurable systems. In these systems, the management of reconfigurations is a very important part in the design phase due to the complexity of hardware reconfigurations and the reconfigurability needs of an aplication.

In the different proposed solutions, the two parts of reconfiguration that are reconfigurable capabilities of the hardware and the different reconfigurations possibilities of an application are not taken into account. A layered reconfiguration management approach through a hierarchical decomposition of a system will allow us to solve this problem.

The proposed adaptable architectue shown in figure 1- c, allowing the adaptation of differents applications and an application in different conditions, is organised as a set of clusters. Each cluster is designed to execute a sub-set of tasks. This clusters are parallelisable, so that the

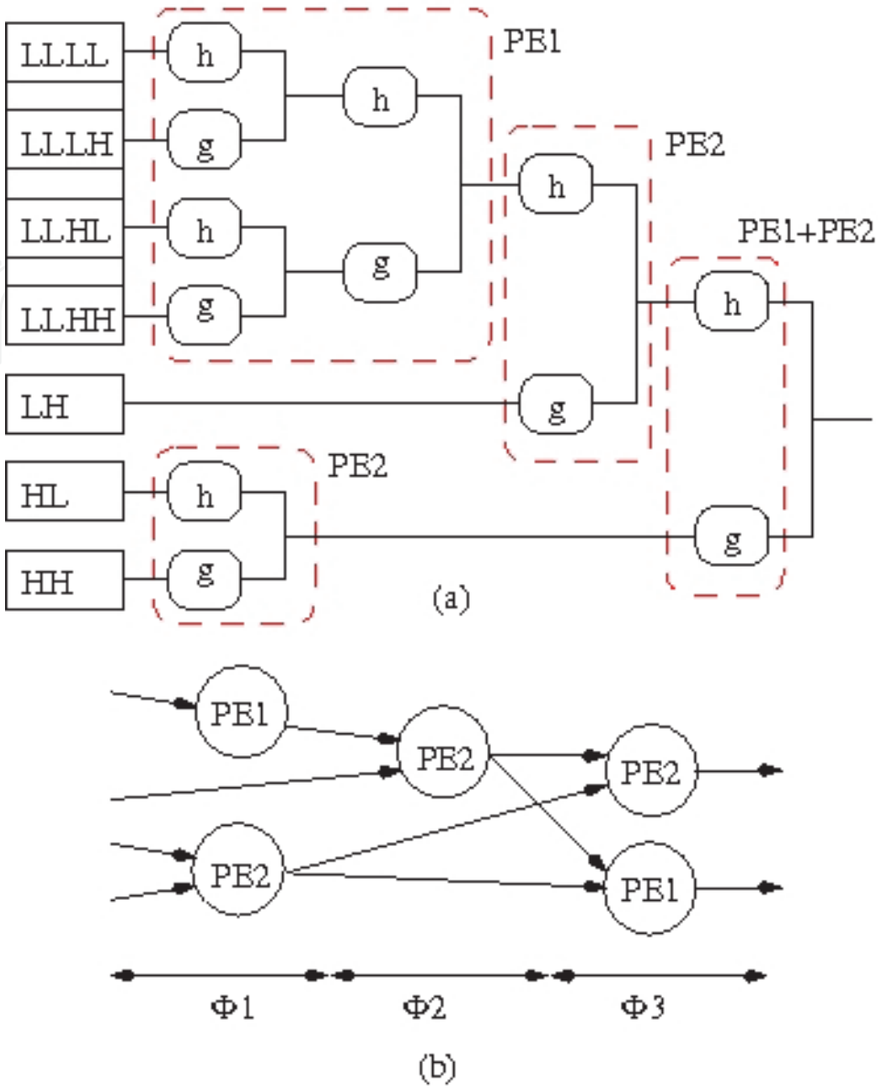

Fig. 2. Processing approach in 2*D* IDWT onto two-level(a); task allocation(b)

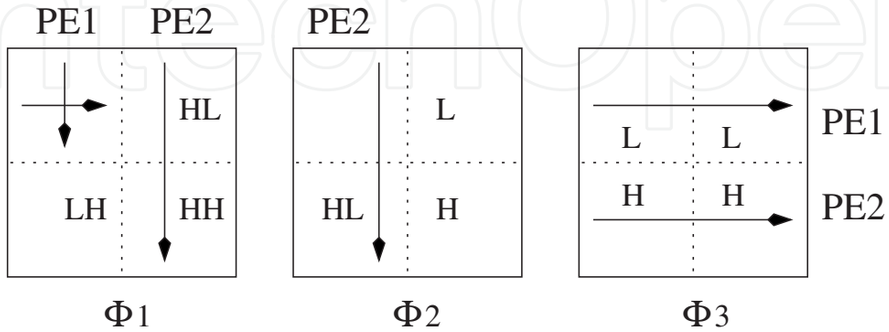

Fig. 3. 2-D IDWT Processing phases

same set of processing are performed on multiple data blocs. Each cluster is composed of an heterogeneous multiprocessor cores that allow software reuse, one or several Reconfigurable Processing Modules (RPU), a Reconfigurable Communication Module (RCM), and on chip memory. The RPM allows hardware acceleration and can be configured in a way that supports different versions of a task. The reconfigurable interface (RIF) is used to build the inter- connection between differents modules. Each RPM can be reconfigured at runtime.
Each cluster has a local configuration manager implemented in an on chip processor that controls the sequences of reconfigurations of the cluster. In this local configuration level, all clusters are configurable in parallel and independently. The reconfiguration process allocates dynamicaly to differents tasks of an application the adequate hard- ware ressources and optimal operation frequency and voltage. The presence of local configuration managers allows the acceleration of the adaptation process. To control the overall system, a global reconfiguration level is necessary. In this level, the necessary informations are managed in order to modify the global organisation of the system by configuring the communication between clusters and the elements of a cluster, allowing for instance to switch from an application to another.
The overall architectureM.Guarisco et al. (2007) is depicted in figure 4. Three memory blocks are present, while the first one and the last one repectively store original data image and deliver computed data, the second block feeds the processing elements. In addition to these three blocks, the system is composed of a reconfigurable processing unit two data organization units and control unit. This last one unit allows to connect the right memory to the right Unit at the right time. Once the memory bloc 1 is full (and as a consequence memory bloc 3 is empty, or at least, all his bytes are read or store in external memory), each memory datapath is switch allowing new picture datas to be treated. A new cycle begins, memory 3 is this time filled and datas in memory 1 are transforming.

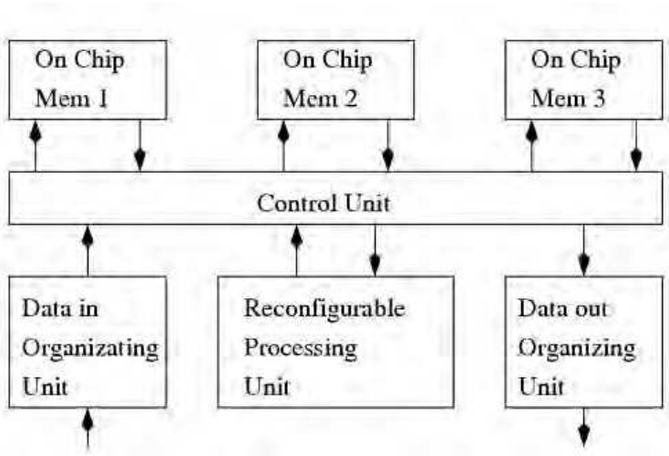

Fig. 4. Porposed DWT processing system architecture

4.1 RPU instance

The reconfigurable Processing Unit (RPU) allows the implementation of different types of wavelet filter. A filter (task) is a set of arithmetic and logic operators. A configuration of RPU consists of a type of filter or a version of a filter. For a given filter, corresponding operators can be connected by different ways in order to carry out different filter versions. These different

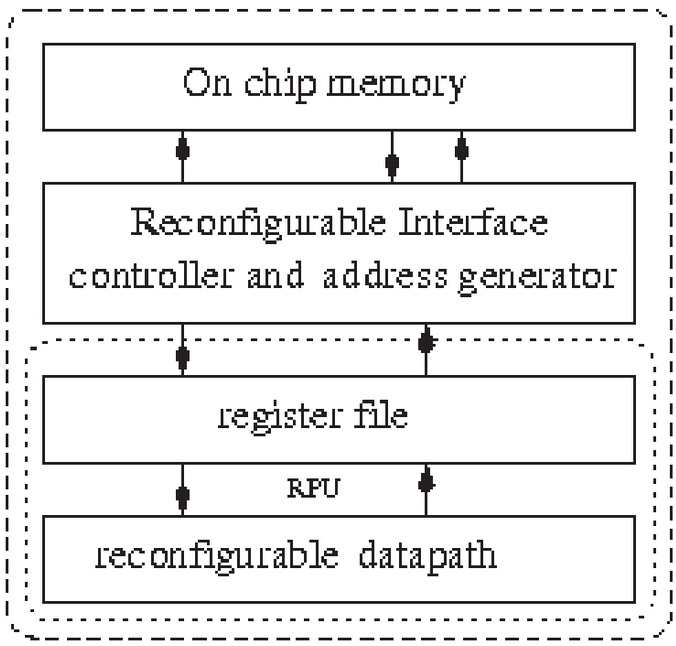

Fig. 5. General architecture of a RPU

versions can be parallel, in pipeline, sequential or an association of both methods. A possible architecture of the RPU and conction with reconfigurable interface is shown in the figure5 Chart1 lists number of computation operators needed (number of additioner, shifter, multiplier by filtering operation). We have choose two filters in order to illustrate adaptation at task level.

Filters	Additions	Shifts	Multiplications
5/3	5	2	0
2/6	5	2	0
SPB	7	4	1
9/7-M	8	2	1
2/10	7	2	2
5/11-C	10	3	0
5/11-A	10	3	0
6/14	10	3	1
SPC	8	4	2
13/7-T	10	2	2
13/7-C	10	2	2
9/7-F	12	4	4

Table 1. Different filter types of wavelet transform

Table 1 lists the number of main computational requirements (the number of additions, shifts, and multiplications per filtering operation). We choose two filters to illustrate the task adaptive level.

4.1.0.1 The $5/3$ lifting based wavelet transform

The IDWT $5/3$ lifting based wavelet transform has short filter length for both low-pass and high-pass filter. They are computed through following equations :

$$D[n] = S_0[n] - [1/4(D[n] + D[n-1]) + 1/2] \quad (4)$$

$$S[n] = D_0[n] + [1/2(S_0(n+1) + S_0[n])] \quad (5)$$

The equations for FDWT $5/3$ are given bellow:

$$D[n] = D_0[n] - [1/2(S_0(n+1) + S_0[n])]] \quad (6)$$

$$S[n] = s_0[n] + [1/4(D[n] + D[n-1]) + 1/2] \quad (7)$$

$D[n]$ is the even term and $S[n]$ is the odd term. The corresponding data flow graph(DFG) is shown in figure 6. It is composed of two partitions: odd and even. Each partition is implemented in the corresponding data path of the RPU. The register file is used to hold intermediate computation results.

4.1.0.2 The $9/7 - F$ based FDWT

The 9/7-F FDWT is an efficient approach which is computed through following equations:

$$D_1[n] = D_0[n] + [\frac{203}{128}(-S_0[n+1] - S_0[n]) + 0.5] \quad (8)$$

$$S_1[n] = S_0[n] + [\frac{217}{4096}(-D_1[n] - D_1[n-1]) + 0.5] \quad (9)$$

$$D[n] = D_1[n] + [\frac{113}{128}(D_1[n+1] + D_1[n]) + 0.5] \quad (10)$$

$$S[n] = S_1[n] + [\frac{1817}{4096}(D_1[n] + D_1[n-1]) + 0.5] \quad (11)$$

There is similarities between equations of $5/3$ filter and those of $9/7 - F$ filter which implies same similarities between the data flow graph of the two filters. It is clear that by duplicating the dataflow graph of filter $5/3$ and inserting four multipliers we obtain the data flow graph of the $9/7$ filter. Moreover, if we consider the table 1, we can see that by partially reconfiguring the $9/7$ filter we can implement all the list of the table. The reconfiguration of $9/7$ filter consists of suppressing or disconnecting unused operators and generation of an adequate control and an efficient data management.

4.2 Reconfigurable interface

The reconfigurable interface is the key element of Reconfigurable Prcessing Unit (RPU). One of its functionality is to connect together the RPU and control communication protocol between the RPU and internal memory. Th controler has to generate adresses for writing and reading operations in memory. A reconfigurable sequencer is used in order to manage the operation and communication sequence. The reconfigurable interface is composed of a three levels pipelined structure for calcul units apart from the one of the first level. Steps of pipeline are : reading (R), execution (E), and writing (W). In our bench test, two versions of interfaces holding different filters implementation are defined. The pipeline stages are :

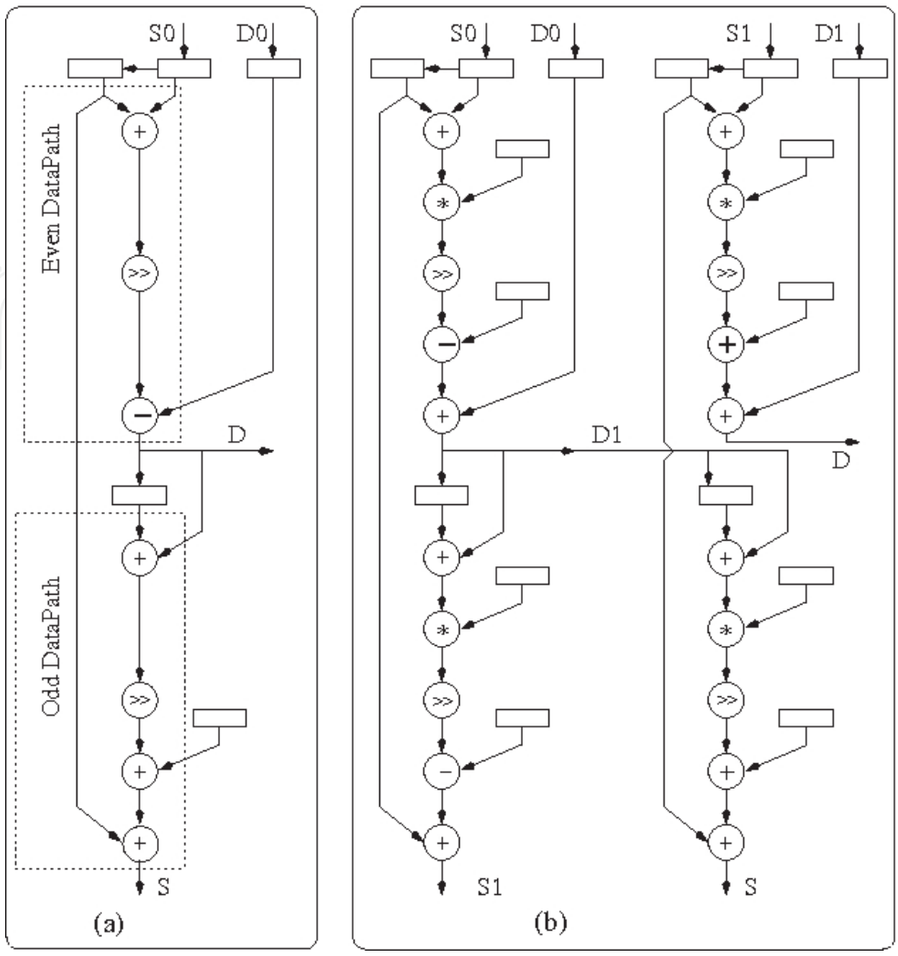

Fig. 6. IDWT 5/3 (a) and 9/7 DFG (b)

- Read (R): The source operands from the on chip memory are sent to the register file. The contro module gives an order to the reading file address generator integrated into the control module for reading the row or column resource from the memory module (SRAM) to the RPU at the address pointed to be by a read counter. Two data are read in one clock cycle.
- Execution (E): The data available in the regiter file is used bythe data-path to process in parallel the two parts of the filter. As the high pass filter part requires the previous result of low pass filter part, the execution is delayed by one clock cylce for high pass filter results. This operation is executed in one clock cycle.
- Writeback (W): The results of computation are written back to on chip memory at the address pointed to by a write counter.

The figure7 illustrates the operating mode of the three stages pipeline. Because of sequential acces to one memory bloc, the computations of the first level are performed as shown in (a) allowing the exection of three operations in one clock cycle. For the remaining porcessing, thanks to the parallel read, execute and write, six operations are executed in one clock cycle (b).

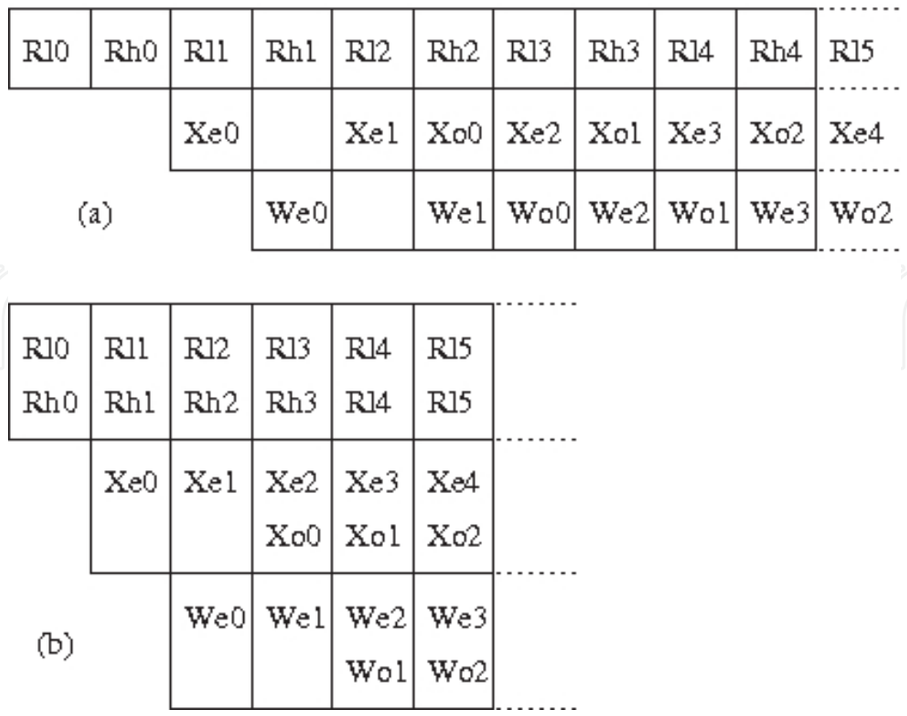

Fig. 7. Pipeline organization: special case (a) and normal case(b)

4.3 Memory access

Seemless memory is make up of several fixed size blocks. Each block is a dual port memory with simultaneous read-write access. Size of memory block correspond to image size in the first level of transformation in the case of the iDWT (inverse DWT). In our experimentation we choose an image of 32x32 pixels or bytes (we work on grey level pictures, that's why a pixel is constitued by one byte only). Because of this organisation, when the first level is proceded, the two data paths of the processing elements are sequentially feeded, that require two memory access cycles. However, for others, datas are read from (or ordered in) two different parallel memory blocks for one processing element in parallel.

4.4 Detailed operations

To explain the operating details of the system, consider an original 8x8 image as shown in figure 8-a. One of the task of this architecture is to rearrange the the pixels in the memory bloc. In order to benefit from the parallelism, Data organizing mudule arranges the pixels as shown in figure 8-b. So, due to the utilisation of memory bloc divided in four independent dual port memories, the processing controller can reach, for a given i, Si and Di which are normally two consecutive pixel in the image and those which we need to calculate at the same time the two coefficient of the DWT. If we want to calculate two new samples at each clock cycle, we have to reach two consecutive elements (Si and Di) at the same cycle.

So, in a first time, each processing element can calculate 1D-DWT in line. As we have two element, the system can compute two 1D-DWT in the same time. In a second time, the system computes the 1D-DWT, but now, in columns. Thus, we save a precious time and we can theoretically achieve an infinite number of levels. Let be T_{load}, the needed time to fill a memory bloc at the frequency of the data (it corresponds to the time of a complete reading or writing, pixel after pixel, of the whole memory bloc), we can say that the execution time of the first

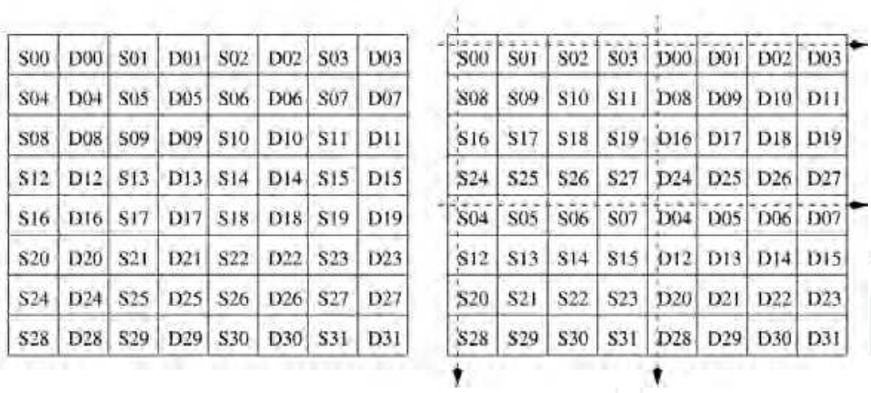

Fig. 8. Original Image(a); Reorganized Image(b)

level is $\frac{T_{load}}{2}$ (we reach two pixels at one clock cycle, so, it divides by two T_{load}, we have two PE that divides again by two Tload but we have to achieve two time the 1D-DWT. Finally the execution time is of $\frac{T_{load}}{2}$). Moreover, we know that for the next level, we need only the low frequency coefficients which represent only a quarter of the total result of the previous level. The execution time is then the result of an arithmetic suite which is represented in equation 4.

$$T_{exec}^{n} = \sum i = 1^{n} \frac{1}{4^{i-1}} \frac{T_{load}}{2} \tag{12}$$

If the number of level tends towards infinite, the execution time is then of $\frac{2*T_{load}}{3}$. A data-out unit allows getting back the DWT coefficients in an ordered way. This controller can be easily modified to adapt the structure of the data flow to the system.

4.5 Target platform

In order to demonstrate the feasibility of the proposed FDWT/IDWT architecture, we have implemented a reconfigurable architecture IDWT targeting an FPGA Xilinx from the Virtex 4 family. The virtex-4 circuit hold partial reconfiguration. Partial reconfiguration of Xilinx FPGA's is achieved using partial configuration datas Inc. (2004). The target architecture, as shown as in the figure 9, is make up of static modules (PowerPC, ICAP, BRAM, PLB Bus) and reconfigurable units(the scalable RPU and hierarchic on-chip memory). ICAP is used to achieve the partial reconfiguration through the embeded processor PowerPC. The reconfiguration datas are stocked in BRAM memory of FPGA and are loaded via ICAP.

5. Implementation results

We have modeled the architecture in HDL in the sofware suite ISE from Xilinx. The simulation results agree with our theoretical waiting. Indeed, we can perform with this architecture a very high number of levels. According to the simulation results, we can run a working frequency of 67MHz. But as we use the internal memory of an FPGA, we are limited and we can reach an image size of only 128x128 pixels. The solution consists of a small modification of the data organizing units to allow the architecture to treat macro-bloc instead of a whole picture.

Figure10 illustrates the placement and routing of one RPU on Xilinx Virtex-4 FPGA. Three mains parts of system like: reconfigurable interface(middle bloc), four registre blocs and two datapaths of IDWT algorithm. The configuration file of each is independant, which is named

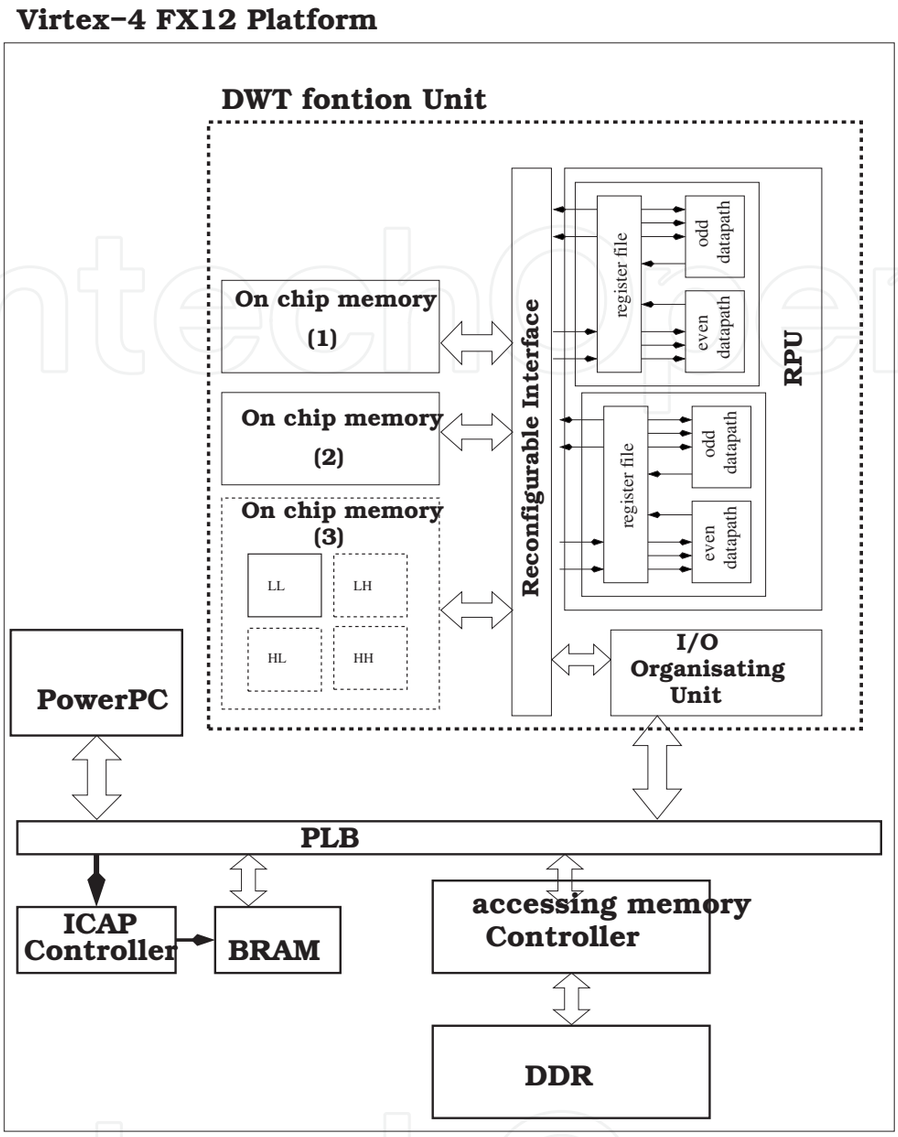

Fig. 9. Target architecture.

Partial bitstreams	RPU	bitstream size(Kbyte)	Reconfiguration time
		Ko	ms
	static part	582 Ko	21 sec (JTAG)
Partial bitstream 1	R_com_1	33Ko	0.57 ms
Partial bitstream 2	R_com_2	63Ko	0.67 ms
Partial bitstream 3	R_f_53	28Ko	0.26 ms
Partial bitstream 3	R_d_97	11Ko	0.16 ms

Table 2. Measured reconfiguration time of different bitstream files for 2-D IDWT.

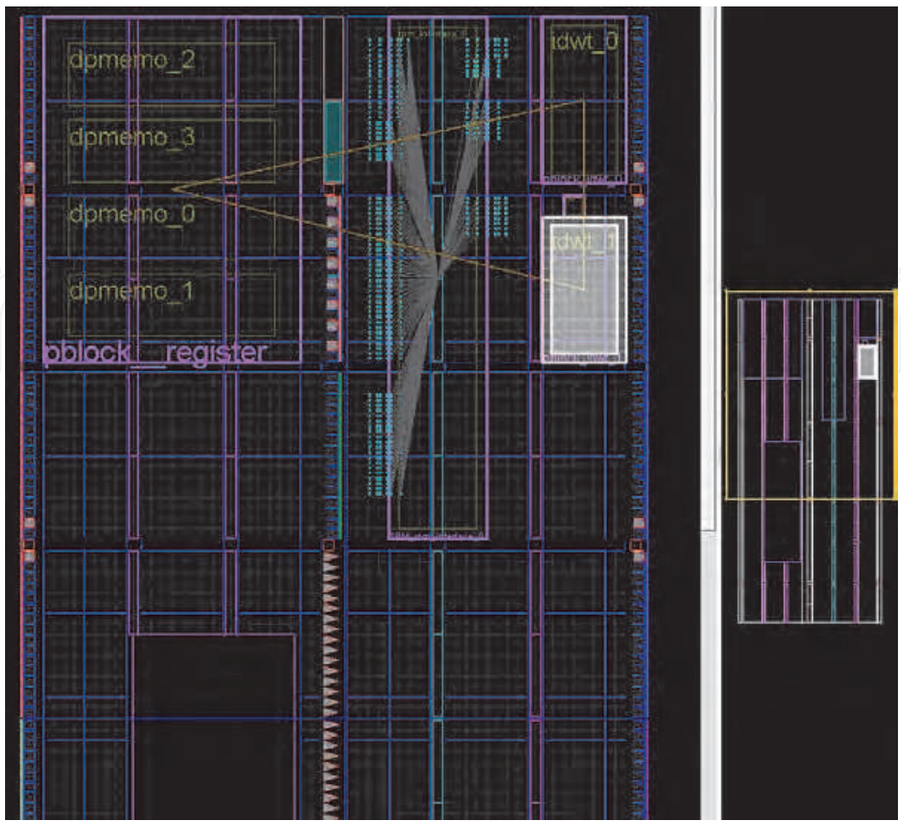

Fig. 10. placement and routing schema of DWT processing unit with Xilinx PlanAhead tool

as partial bitstreams. The different partial bitstreams are stored in the on chip BRAM. The static bitstream is loaded using cable. To measure the execution time of each partial bitstream, a free running hardware timer is used. The measurement results are in table 2. In this table, the mains modules are : the different part between filter 5/3 and 9/7 (R_d_97), 5/3 filter functional module (R_f_53), interface for 5/3 filter (R_com_1) and: communication interface for 9/7 filter (R_com_2).

The on chip PowerPC processor is used for autoconfiguration through HWICAP. As the PowerPC is an element of the system, it is used to detect external or internal events and accordingly loads automatically the adequate configuration to adapte the system to the given situation and then making the auto-adaptive. The HWICAP makes auto-configuration easier, in fact a C program on PowerPC allows the transfer of 512x32 bit blocks of the partial bitstream from the configuration memory to a fixed size buffer of the HWICAP peripheral, which the transfer from the buffer to the ICAP. The total reconfiguration time can approximated by the following equation:

$$T_{config} = T_{ICAP} + T_{BRAM} \tag{13}$$

Where T_{ICAP} is the time required to transfer configuration from the buffer to the ICAP, and T_{BRAM} is the time required to transfer data from configuration memory to the HWICAP buffer. Table 2 shows different parts of the system, the size of corresponding bitstream file and their configuration time. The system consists of a static part and reconfigurable parts ($Part_1$ and $Part_2$ are the two versions of reconfigurable communication allowing the switching between

two filters, $Part_3$ corresponds to $5/3$ filter, and $Part_4$ is the difference between $5/3$ filter and $9/7$ filter). The configuration time is measured using a free running counter (timer) incremented every system clock cycle, and capturing the start time and the end time. We see that the configuration time as expected depends linearly on the size of bitstream.

Type of architecture	Resolution	Area(mm2 for VLSI and ASIC)(CLB for FPGA)	Max frequency of operation(MHz)	Memory Requirement (KB)
Proposed architecture	32x32	153 CLBs	50	1.024
	64x64	538 CLBs	50	4.960
ASICs based(Tseng et al., 2003)	one image frame	8.796 mm2	50	2 memory frames
Zero-padding scheme (et al, 2002)	32x32	4.26mm2	50	6.99

Table 3. Implementation results

To compare the measured configuration time with the minimum possible value, the value for the reconfiguration of Virtex-4 FPGA could be obtained with this equation: $T_{config} = L/r$, where L is the length of the configuration and r is the transfer rate. As an example, for a file of 63KB size, and a clock frequency of 100 MHz as used in our experimentation, the minimum theoretical time is 0.63 ms, which is much less than 90 ms that as given in table 2. This is due to PowerPC that acts as the configuration manager in our system. Large part of time is spent to copy reconfiguration data from on chip or external memory to HWICAP buffer. The difference between the measured configuration time (0.97 ms) and the computed time (0.63 ms) is due to the imprecision of the measurement method. In fact, the capture of start and stop time is achieved using software, which tacks additional clock cycles. In table 2 we can see also that the main part of reconfiguration time is wasted for the transmission of reconfiguration files.
The reconfiguration time includes two part times: T_{bram}, the total load time for transferring the reconfiguration bitstreams from memory on chip to buffer of ICAP with package $512byte$. T_{icap}, the total configuration time through the ICAP port is grouped by the sum of configuration time for one package. Hence, the reconfiguration time is decided largely by the size of reconfiguration bit files and the number of reconfiguration bit files. The reconfiguration manager makes possible to reduce the reconfiguration time through hiding partial reconfiguration process in the execution process. It is obvious that the configuration time can be improved. A solution we are studying is based on a specific hardware reconfiguration manager capable to transfers the configuration data from on chip memory to ICAP.
Moreover, the chart 3 compare our approach with the other architecture. We observe primary two parameters based on different resolution of image. At the same working condition, the area of DWT computing module is variety according to the size of image(153CLBs for 32x32 and 538CLBs for 64x64) where including the adding of memory requirement(1,024KB for 32x32 and 4,960KB for 64x64). Thus the area of circuit can be used efficiently according to

the defnite size of image. The size of of memory requirment is scalable and thus the correcte size of memory can be configured dynamically to adapt to the requirement of bandwidth of memory. The other work shown in this table are based on ASIC(Tseng et al., 2003)and VLSI (et al, 2002). the area of circuit and the size of memory are fixed and thus the maximum size of memory must be previewered, which may lead to the urgent or surplus of memory access.
This proposed architecture features small area and low momory requirments. Processing time for a $32x32$ image blocks is $43s$ which is lower than others traditional design. Using a $64X64$ image blocks gives a good performance throughput which takes $86s$ for the transformation, for two-level 2D IDWT, which is capable to perform the image CCIR($720X576$) format image signal at 50 $frame/s$.

6. Conclusion

In this book chapter, , we have described auto-adaptive and reconfigurable hybrid architecture for F/IDWT signal processing application. Two levels of auto adaptation are defined in order to minimize the reconfiguration overhead. The application adaptive level in which different applications of a domain are classified and characterized by a set of tasks. The task adaptive level in which for a given task, a set of versions are defined and characterized for use in a situation to adapt the application to different constraints like energy, and bandwidth requirement.
The proposed architecture is a universal, scalable and flexible featuring two levels of reconfiguration in order to enable the application adaptivity and task adaptivity. We demonstrated through the case study that it can be used for any types of filters, any size of image and any level of transformation. The memory is organized as a set of independent memory blocks. Each memory block is a reconfigurable module. The high scalability of the architecture is achieved through the flexibility and ease of choosing the number of memory blocks and processing elements to match the desired resolution. The on-chip memory is used not only to hold the source image, but also to store the temporary and final result. Hence, there is no need of temporal memory. The processor has no instructions and then no decoder, in fact, the hardware reconfigurable controller plays the role of a specific set of instructions and their sequencing. For a given set of tasks, a set of configurations are generated at compile time and loaded in run time by the configuration manager via configuration memory. The prototype has been tested on FPGA developpment cart of Xilinx with 65nm CMOS technology. The prototyping chip can be reconfigured to adapte 5/3 filter or 9/7 filter. In comparing with others ASIC architecture at the same working frequency, our proposed architecture requires less memory bloc and fewer hardware resource than the others.

7. References

et al, S. (2002). Vlsi implmentation of 2-d dwt/idwt cores using 9/7-tap filter banks based on the non-expansive symmetric extension scheme, *in* IEEE (ed.), *Proceedings of the 15th International Conference on VLSI Design*, number 435 in *ASP-DAC '02*, IEEE Computer Society, Washington, DC, USA.

Inc., X. (ed.) (2004). *Two flows for partial reconfiguration: module-based or different based*, Xilinx.

Lee, S.-W. & Lim, S.-C. (2006). Vlsi design of a wavelet processing core, *in* IEEE (ed.), *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16.

M.A.Trenas, J.Lopez & E.L.Zapata (2002). A configurable arechitecture for the wavelet packet transform, *The journal of VLSI Signal Processing*, Vol. 32, pp. 151–163.

M.Guarisco, X.Zhang, H.Rabah & S.Weber (2007). An efficient implementation of scalable architecture for discrete wavelet transform on fpga, *in* IEEE (ed.), *6th IEEE Dallas Circuits and Systems workshop*, pp. 1–3.

Olkkonen, H. & T.Olkkonen, J. (2010). *Discrete Wavelet Transform Structures for VLSI Architecture Design*, intech, Hannu Olkkonen and Juuso T. Olkkonen (2010). Discrete Wavelet Transform Structures for VLSI Architecture Design, VLSI, Zhongfeng Wang (Ed.), ISBN: 978-953-307-049-0, InTech, Available from: http://www.intechopen.com/articles/show/title/discrete-wavelet-transform-structures-for-vlsi-architecture-design.

P.Jamkhandi, A.Mukherjee, K.Mukherjee & Franceschini, R. (2000). Parallel hardware/software architecture for computation of discret wavelet tranform using the recursive merge filtering algorithm, *Proceeding International Parallel Distrib. workshop*, pp. 250–256.

Ravasi, M. e. a. (2002). A scalable and programmable architecture for 2-d dwt decoding, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, pp. 671–677.

S.Mallet (1999). A theory for multi-resolution signal decomposition: The wavelet representation, number 7, IEEE, pp. 674–693.

Tseng, P.-C., Huang, C.-T. & Chen, L.-G. (2003). Reconfigurable discrete wavelet transform architecture for advancedmultimedia, *in* IEEE (ed.), *Signal Processing Systems*, number 137-141.

Discrete Wavelet Transforms - Algorithms and Applications

Edited by Prof. Hannu Olkkonen

ISBN 978-953-307-482-5
Hard cover, 296 pages
Publisher InTech
Published online 29, August, 2011
Published in print edition August, 2011

The discrete wavelet transform (DWT) algorithms have a firm position in processing of signals in several areas of research and industry. As DWT provides both octave-scale frequency and spatial timing of the analyzed signal, it is constantly used to solve and treat more and more advanced problems. The present book: Discrete Wavelet Transforms: Algorithms and Applications reviews the recent progress in discrete wavelet transform algorithms and applications. The book covers a wide range of methods (e.g. lifting, shift invariance, multi-scale analysis) for constructing DWTs. The book chapters are organized into four major parts. Part I describes the progress in hardware implementations of the DWT algorithms. Applications include multitone modulation for ADSL and equalization techniques, a scalable architecture for FPGA-implementation, lifting based algorithm for VLSI implementation, comparison between DWT and FFT based OFDM and modified SPIHT codec. Part II addresses image processing algorithms such as multiresolution approach for edge detection, low bit rate image compression, low complexity implementation of CQF wavelets and compression of multi-component images. Part III focuses watermaking DWT algorithms. Finally, Part IV describes shift invariant DWTs, DC lossless property, DWT based analysis and estimation of colored noise and an application of the wavelet Galerkin method. The chapters of the present book consist of both tutorial and highly advanced material. Therefore, the book is intended to be a reference text for graduate students and researchers to obtain state-of-the-art knowledge on specific applications.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821

IntechOpen