Design of a Two Dimensional PRSI Image Processor

Theja Tulabandhula
AVDL, IIT Kharagpur, India
INESC-ID, R. Alves Redol, 9, Lisboa, Portugal
t.theja@iitkgp.ac.in

Amit Patra,Nirmal B Chakrabarti
Advanced VLSI Design Laboratory
Indian Institute of Technology Kharagpur, India
amit.patra@ieee.org,nirmalbc@yahoo.com

Abstract

A digital processor capable of computing several two dimensional Position Rotation and Scale Invariant (PRSI) transforms on 64 x 64 pixel images is presented. The architecture is programmable to achieve the following five transforms: Two Dimensional (2D) Fast Fourier Transform (FFT), 2D Log Polar Transform (LPT), 2D Fourier Mellin Transform (FMT), 2D Analytical Fourier Mellin Transform and Phase only Correlation (POC). 1D FFT design with scale and word-length issues is also detailed. Thirty two 1-Dimensional FFTs have been reused as processing elements in the 2-Dimensional FFT unit. Scheme for matrix transpose in O(N) time is also described. The Image Processor has a word length of 16 bits each for real and imaginary parts and has been implemented on Xilinx Virtex II FPGA. It is capable of processing an image in less than 0.1ms.

1. Introduction

In some applications of image processing an objective is to recognize the object in a given image in the presence of variations. These variations can be conveniently broken down as shift, rotation and scale. Fourier Mellin Transform (FMT), analytical Fourier Mellin Transform (AFMT), Two Dimensional Fast Fourier Transform (2D FFT), Log Polar Transform (LPT) and Phase Only Correlation (POC) belong to this Position Rotation and Scale Invariant (PRSI) class of transforms which can detect the above variations. These transforms have a very important role to play in Content-based Image Retrieval (CBIR) systems [4], active vision systems [3], image registration [8] as well as image recognition [15] to name a few. Much less work has been reported on hardware optimization for the PRSI transforms as compared to others (e.g. DCT architectures for image compression).

In [16], Fourier techniques are presented as an extension to the phase correlation method as a part of a PRSI image processing scheme. In [19] an FFT based image processor design is proposed. A processor implementing POC, which is also a part of the PRSI class, using FFTs is described in [13] and [12]. The design is an LSI (ASIC) with external memory. There is no provision for LPT as well as FMT. Another 2D FFT processor in ASIC working at 4 MHz in [18] is a video-rate processor doing only 2D FFT. PRSI algorithms if implemented using this, will require additional hardware. The designs described in [13],[12] and [18] use crossbar switch for memory shifting. This is uneconomical if there is parallelism in the algorithms, which could otherwise have been exploited. Finally, [11] discusses implementation of 2D FFT on media processors. This means that the hardware is not optimized in terms of resources and efficiency. Clearly, there is a need for optimized hardware which exploits certain features of the PRSI transform algorithms. Such dedicated hardware can then be deployed as co-processors to do the PRSI jobs in a quick way.

In this paper, a design for a co-processor capable of performing these PRSI transforms is presented. The processor consists of two 2D FFT units and a LPT unit. These units are reused to perform the other transforms. FMT is accomplished by two 2D FFTs and a LPT. POC is accomplished by three 2D FFT operations. The user has the freedom to choose the sequence of steps in which the transforms need to be performed (e.g. LPT followed by a FMT or an analytical FMT) which is usually the case while building such image-based embedded systems. Performing Fourier Mellin Transform (FMT), which is one of these PRSI transforms, using FFT is a rather ingenious way to accomplish the transformation fast. Performing correlation using FFTs has been there for a long time.

The proposed processor unlike [19] has a separate matrix transposing unit which eliminates the need for inter-process communication and its related issues like synchronization and simultaneous memory access. Different from [13] and [12], the processor design is targeted for FPGAs with internal block RAMs, and requires a different way of optimizing things. With 16 butterflies per 1D FFT unit, [18] is markedly different in its target applications compared to

978-0-7695-3277-6/08 $25.00 © 2008 IEEE
DOI 10.1109/DSD.2008.80

the proposed design. The presented design has one butterfly per 1D FFT unit. Lastly, different from [13], [12] and [18] the design has a separate controller unit which accomplishes matrix transpose in O(N) time using the parallelism of the algorithm and several block RAMs.

The paper is organized as follows. Section 2 briefly reviews the PRSI class of algorithms along with their implementation aspects. Section 3 deals with the 1D FFT design, the 2D FFT Design, details of Log Polar Transformation and radial warp unit. It ends with the details of the PRSI processor and how FMT, analytical FMT and POC can be performed. Section 4 discusses some of the results obtained from our FPGA implementation and their application areas. Finally, section 5 presents the concluding remarks.

2. PRSI Algorithms

PRSI transforms are based on 2D FFT, logarithm and exponential digital realizations and will be discussed here.

2.1 One Dimensional Fast Fourier Transform

FFTs computation exhibits a high degree of regularity in structure, comprising recurring basic kernels. The theories behind efficient hardware implementations have been well dealt with [10].

$$DFT : X(k) = \sum_{n=0}^{N-1} x(n) * e^{-\frac{j2\pi nk}{N}}; k = 0, 1, \ldots, N-1 \quad (1)$$

A scaling factor and a negative exponential are the only two differences which permit both FFT and IFFT operations on the same hardware. Radix-2 Decimation in Time (DIT) method has been chosen while designing the FFT module because of its regularity and low complexity of the controller involved.

2.2 Two Dimensional Fast Fourier Transform

2D DFT is given by:

$$X(k_1, k_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2) * e^{-\frac{j2\pi(k_1 n_1 + k_2 n_2)}{N}} \quad (2)$$

$$k_1 = 0, 1, \ldots, N-1; k_2 = 0, 1, \ldots, N-1$$

Performing 1D FFT on the rows of the matrix, followed by 1D FFT on the columns accomplishes this with a complexity proportional to $O(N^2 log(N))$. Using several 1D FFT units exploits the parallelism in the operation as explained next. Let $N = qp$ where N is the order of the squared input image, p is the number of 1D FFT processing elements and q is an integer. Each processing element is allocated a set of q rows. The algorithm consists of the following four steps:

Step 1. 1D FFT on rows: Processor P_i computes 1D FFT on q rows of input image assigned to it, where $i = 0, 1, \ldots, p-1$. Because each processor executes in parallel with the other P-1 processors, computation complexity reduces by a factor q/N.

Step 2. Transpose the matrix: Transposition of the output matrix is required before execution of another set of FFTs to reuse the control logic and address generation units which are configured to work on rows. Matrix transposition has $N * (N-1)/2$ exchange steps. By taking advantage of the distributed memory banks in the form of block RAMs of FPGA, the operation is completed in $O(N)$ time.

Step 3. 1-D FFT on new rows (old columns): This is same as *Step 1*, but works on the new rows which were columns before the matrix transposition step.

Step 4. Transpose the matrix again: A final matrix transposition is required to obtain the result image. This step can be skipped if both the host and the co-processor know the way data is being transferred. For example, row wise input of data to the co-processor and column wise output of the processed image will not require this step.

2.3 Log Polar Transform

The log-polar transformation is a nonlinear and non-uniform sampling of the spatial domain. Nonlinearity is introduced by polar mapping, while non-uniform sampling is the result of logarithmic scaling. Consider the log-polar $(log_{base}(\rho), \theta)$ coordinate system, where ρ denotes radial distance from the center (x_c, y_c) and θ denotes angle. Then,

$$\rho = \sqrt{(x - x_c)^2 + (y - y_c)^2} \quad (3)$$

$$\theta = tan^{-1}(\frac{y - y_c}{x - x_c}) \quad (4)$$

The processor employs a straightforward method for this transform which again takes advantage of the presence of several block RAMs. The logarithm and exponential calculations needed to perform this operation are done using the series expansion method. The polar conversion uses the nearest-neighbor approximation. The LPT can be a standalone transform as well as an intermediate step while performing FMT and analytical FMT. LPT by itself is not related to invariance. When applied to an image, it can turn scaling and rotation into translations which can be operated on by the translation invariant 2D FFT.

2.4 Fourier Mellin Transform and analytical Fourier Mellin Transform

The optical research community first introduced the FMT for pattern recognition [1]. The Mellin Transform in the polar coordinate system is:

$$M_f(k,v) = \frac{1}{2\pi}\int_0^\infty f(\rho,\theta)\rho^{-iv}e^{-ik\theta}d\theta\frac{d\rho}{\rho} \quad (5)$$

$$\forall(k,v) \in Z * R$$

It can be seen that this expression is a 2D Fourier Transform of a logarithmically scaled input $f(e^\tau,\theta)$ with $\tau = log(\rho)$. FMT is defined as 2D Fourier transform followed by a polar conversion and finally a Mellin Transform. This means that the FMT can be accomplished by performing a 2D FFT followed by a polar conversion and logarithmic scaling. Finally another 2D FFT is done.

How FMT is PRSI is illustrated through the following example. If an image $I_o(x,y)$ is rotated, scaled and translated to $I_1(x,y)$ by magnitudes α , σ and (x_o,y_o) respectively, then the Fourier transforms F_o and F_1 are related as

$$F_1(u,v) = e^{-j\pi\phi(u,v,x_o,y_o)}\sigma^{-2}$$

$$*|F_o(\frac{u*cos\alpha + v*sin\alpha}{\sigma}, \frac{-u*sin\alpha + v*cos\alpha}{\sigma})| \quad (6)$$

Considering only the magnitudes, it is seen that the translation effect has vanished. If the polar system for the magnitudes of the Fourier transforms of the two images $(F'_o(\rho,\theta), F'_1(\rho,\theta))$ is used, then

$$F'_1(\rho,\theta) = \sigma^{-2}F'_o(\frac{\rho}{\sigma}, \theta-\alpha) \quad (7)$$

Again considering the magnitudes, α appears only in the phase term. Scaling is further reduced to translation by doing a logarithmic sampling of ρ i.e. if $\tau = log(\rho)$ and $\beta = log(\sigma)$ Then,

$$F''_1(\tau,\theta) = \sigma^{-2}F''_o(\tau-\beta, \theta-\alpha) \quad (8)$$

This implies that a LPT after the initial Fourier Transform makes the scale and rotational terms appear as translations. These are eliminated by doing a subsequent Fourier Transform. Hence, these steps constituting FMT make it a PRSI transform.

By using the 2D FFT and LPT blocks, FMT is accomplished on the proposed chip. It has been observed that for some inputs, the FMT does not converge (see [5]). To counter this, a modified transform has been proposed by them called the analytical FMT (AFMT) where the FMT of a modified function or image given by $f_\mu(\rho,\theta) = \rho^\mu f(\rho,\theta)$ is performed. Though this helps in convergence, it also makes the AFMT appropriate only for extracting features that are translation and rotation invariant. Scale invariance does not hold true now and needs to be addressed by additional steps. On hardware, this multiplication with a ρ^μ is carried out after the polar conversion and before the logarithmic sampling in the LPT unit.

2.5 Phase Only Correlation

POC is accomplished by performing three 2D FFT operations. Two images are taken as inputs and a 2D FFT is performed on each of them. Then their phase spectra is separated and subtracted from each other. Finally an inverse 2D FFT is performed. A POC processor is described in [13].

3. Implementation

3.1 1D FFT

A 64 point complex 1D FFT is used as a processing element inside the 2D FFT. The core has one butterfly unit with one $16bit * 16bit$ Wallace tree based multiplier and four carry look ahead adders. The speed of the Wallace multiplier more than outweighs the issues with its irregularity. Speed of multiplication is of prime importance in determining the maximum clock frequency at which the processor can run. The 1D FFT is connected to a single dual port block RAM which stores the ROM twiddle factors as well as the input-output data of each stage (in-place substitution).

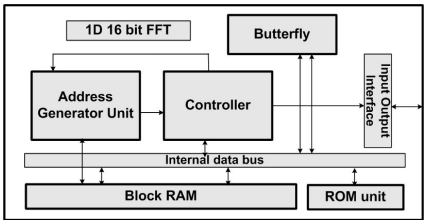

Figure 1. Block diagram of the 1D FFT unit

Each butterfly requires 8 clock cycles to generate $A \pm W * B$ from the inputs A, B and W where W represents the twiddle factor $(e^{-j\frac{2\pi kn}{N}})$ and (A,B) represent the input complex pair. The internal word length is 16 bits for the real and 16 bits for the imaginary part. Since, 8 cycles are required for each butterfly operation, and $\frac{64}{2}$ butterfly operations need to be performed for each of the $log_2(64)$ stages, the FFT conversion for 64 points amounts to $8 * 32 * 6 = 1536$ cycles. No LUT for address generation has been employed speeding up the operation in the process. Because twiddle factors (32 in number for a 64 point FFT) are stored in the ROM and ROM access times are slower than that

of RAM during operation (if we fabricate this design as an ASIC), the 1D FFT unit copies these factors from ROM to RAM on power up. Subsequent access of these twiddle factors is then, through the dual port RAM itself.

A scaling of N is done to prevent overflow during operation. The result of IFFT however is the same as that of the IDFT expression (1). A $2'$s-complementing unit is used to get the negative of the imaginary part of the twiddle factors required in the case of IDFT (i.e. obtain $-sin(\frac{2\pi kn}{N})$ from $sin(\frac{2\pi kn}{N})$).

Word length choice plays an important role in keeping the truncation errors within bounds. 16 bits have been chosen after the hardwares performance was evaluated against floating point FFT operation (figure 2).The SQNR (Signal to Quantization Noise Ratio) for the radix-2 FFT assuming white noise as given in Proakis et al. ([14]) is $\frac{\sigma_x^2}{\sigma_q^2} = \frac{1}{2N}\delta^{-2}$. In the proposed processor N=64 and number of bits ($b = -log_2\delta$) = 16, the SQNR comes to be 75.26 dB.

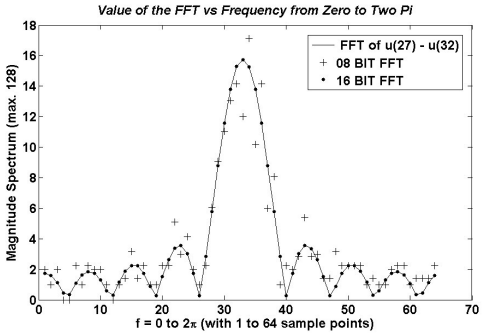

Figure 2. 8 and 16 bit 64 point 1D FFT are compared with a floating point 1D FFT (Matlab) at same scales. Input: $U(27) - U(32)$ where $U(n)$ is the unit step function.

3.2 2D FFT

Many of the 2D FFT designs do not address the corner turning (matrix transposition) problem present in the calculation. Some proposed solutions are impractical because the corner turning problem is solved by requiring row as well as column access, often through several ports, to a memory containing a complete image. This issue has been taken care of implicitly, by the use of 32 1D FFT processing elements. Each processing element has its own Address Generator Unit and a dual port ram connected to it. This implies that the matrix transposition needs to be carried out on 32 dual port RAM blocks. The transpose hence is having simultaneous read and write access to each RAM. Thus, while it is reading an element from a row (in RAM X), it also reads an element from a row (in RAM Y) and then swaps them in the next cycle. At the same time, two other rows (on a different set of RAMs) also undergo the same operation. Speed is obtained at the expense of fairly complex logic and bus architecture. This answers two issues. Firstly, it eliminates the need for N-port ($N > 2$) RAM, which would otherwise bring with it a new set of problems (like cache-misses and routing). Secondly, matrix transposition takes $\sim O(N^2)$ time if done sequentially on a single ram block. This time complexity is reduced to $\sim O(N)$ because of the presence of several dual port memories, to which the transposing unit is connected to through separate buses.

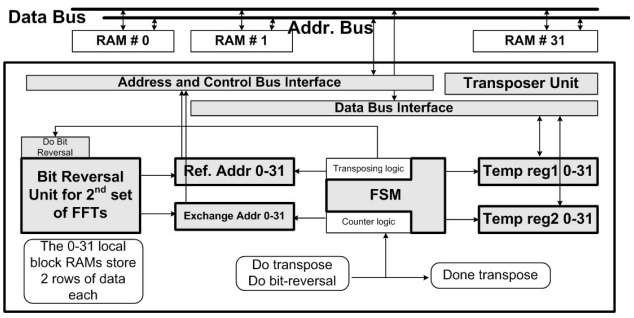

Figure 3. The transposing unit.

The 32 RAMs roughly sum up for the local storage of the input/output data. Data is fed to the hardware at the start signal from the outside. As soon as parts of data are read, 1D FFT units are scheduled to start working immediately. The 32 1D processing units are scheduled twice for the $64 * 64$ data. Each processing unit gets two rows. Similarly in a scaled $256 * 256$ point processor each 1D FFT unit gets 8 rows. The block level diagram of the 2D FFT is shown in Figure 4.

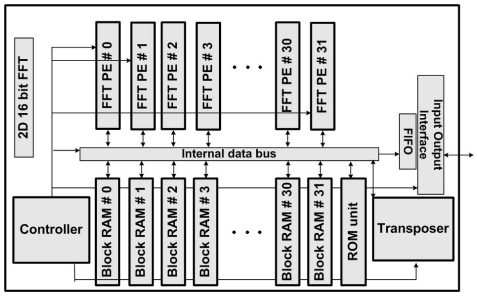

Figure 4. 2D FFT Block diagram

3.3 Log Polar and Radial Warp Unit

The logarithm sampling of the radial coordinate needs to be taken after Cartesian to polar conversion of the magni-

tude spectra takes place. This step happens after the first 2D FFT. The 16 bit logarithm unit uses a normalization (series expansion based) technique. It consists of a ROM (to store for example, the $log(1+2^i)$ values), a 20 bit carry look ahead adder and control logic. The exponential ($base^\tau$) also needs to be determined while doing the LPT. It is also designed using ROM and adders using the series expansion method. Extra circuitry is in place to keep a tab of the magnitude. For example, the logarithmic unit described above, needs to have an input in the range of 0.5 to 1, and hence preprocessing using peripheral circuitry becomes a necessity. Cartesian to polar conversion based on nearest neighbor approximation is done before the logarithmic sampling step. The LPT unit has access to two RAM clusters (each having a group of 32 RAMs) because of the absence of an in place algorithm. Each cluster has the size equal to the input data. The unit accesses the output values from the first cluster, does the polar conversion and logarithm sampling and then writes to the second RAM cluster. Again, by using two RAM clusters, parallelism has been again exploited to accomplish the operation in $O(N)$ time. A similar LPT not exploiting the parallelism, implemented on an Altera FPGA using a dual port external memory is shown in [6] which can process 25 images per second.

In analytical FMT, the matrix after the polar conversion needs to be multiplied with a factor (ρ^μ) before the Mellin Transform. This is achieved by using LUTs and a set of 32 Wallace tree based multipliers. This step is carried out between polar conversion and logarithmic sampling.

3.4 PRSI Processor

The processor block diagram is shown in Figure 5. The 2D FFT housing 32 1D FFT processing elements is instantiated twice and the LPT unit is instantiated once. The LPT along with log and polar conversion has control logic and multipliers to separate the phase and magnitude spectra which are required by the POC and the FMT/AFMT respectively. The global controller takes in the user operation mode to accomplish any of the 2D FFT, LPT, FMT, analytical FMT and POC operations.

3.5. Working memory

As shown in Figure 5, the processor contains two RAM clusters and a synchronous FIFO. The FIFO has a size equal to $4K * 16bit$. Each of the 32 RAMs in one cluster have a size to house 2 rows of complex data and space for the twiddle factors. For $64 * 64$ bit data, this amounts to $(256 + 128)*16$ bit. Hence, the total RAM cluster size is $32*384 \approx 12K * 16$ bit. The processor also has some local registers for transposing and accomplishing LPT which amount to $1K * 16bit$. Thus, the total memory requirement for the

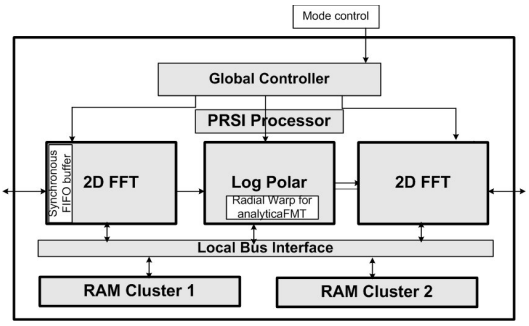

Figure 5. 2D PRSI Processor block diagram

processor is $25k * 16$ bit. Due to preassigned fixed sizes of the dual port block RAMs in FPGAs (such as Xilinx Virtex 2), total memory usage is in general more than this (see Tables 1 and 2).

4. Discussion and Applications

4.1 Simulation, results and discussions

The 1D FFT units were designed and tested for Spartan 2 and Virtex 2 platforms. The table below lists the variation in the resource usage for 8 and 16 bit implementations of 64, 128, 256, 512 and 1024 point FFT hardware for the Virtex 2 FPGA (XC2VP30). From the table it can be inferred that, the percentage increase in resources is negligible for transform lengths adjacent to each other. This means that, one can perform coarse FFT with a smaller number of points in a reduced number of cycles and then decide on doing a finer FFT with a higher number of points. These two step transforms can thus help in conserving resources and effort in many image processing based embedded systems. The existing design is fully parameterized at RTL level and can easily be tweaked to incorporate this scalability. A sampling rate conversion with interpolators and decimators before reading the data is usually needed for this kind of a task.

There are better architectures for 1D FFT in the literature. [20] is an energy efficient scalable 1D FFT on ASIC. It employs a BW multiplier in lieu of Wallace tree based multiplier. [2] proposes a 256 point 1D FFT in 0.18 CMOS based on the DIT radix-2 algorithm. Similar to both [20] and [2], the design proposed in this paper can be made into an ASIC if the memory units can be properly implemented. In fact the process of copying twiddles from ROM to RAM which the processor does on power up is beneficial only on an ASIC. [9] and [17] discuss similar 1D FFT designs on FPGAs with various optimizations using several multipliers. The purpose of the present design however, is to keep the 1D FFT design minimal. Since 32 of them will form

the 2D FFT block and each has one multiplier in its butterfly unit, 32 multipliers will be working simultaneously during the 2D FFT operation implying an adequate number of computations per cycle.

8 bit Design of complex 1-Dimensional FFT						
Transform Points		64	128	256	512	1024
Area	Normalized gate Count	1	1.007	1.016	1.032	1.943
	Number of slice flip-flops	248	259	273	286	297
	Number of 4 input LUTs	845	893	958	1111	1268
Clock Frequency (Mhz)		100	100	100	100	100
Memory (Kb)		18	18	18	18	18
$SQNR = \frac{1}{2N} 2^{2*bits}$ dB		27.0	24.0	21.0	18.1	15.1

Table 1. Resource usage and speed values of different implementations of 8 bit 1D FFTs on Virtex 2 (XC2VP30)

16 bit Design of complex 1-Dimensional FFT						
Transform Points		64	128	256	512	1024
Area	Normalized gate Count	1	1.010	1.026	1.871	3.552
	Number of slice flip-flops	333	345	359	371	387
	Number of 4 input LUTs	1786	1872	1994	2279	2787
Clock Frequency (Mhz)		57	56.8	57.9	57.8	57.5
Memory (Kb)		18	18	18	36	72
$SQNR = \frac{1}{2N} 2^{2*bits}$ dB		75.2	72.2	69.2	66.2	63.2

Table 2. Resource usage and speed values of different implementations of 16 bit 1D FFTs on Virtex 2 (XC2VP30)

Base equivalent gate counts for Tables 1 and 2 are 73337 and 80276 respectively.

The processor was tested for 2D FFT operation and was implemented on a Virtex 2 FPGA. A 2D rectangular input was given and the output of the $64 * 64$ 16 bit 2D FFT block was plotted and compared with a floating point computation using the *fft2* function in Matlab. In Figure 6 the output has been plotted. The magnitude of the error was found to be less than 1 percent.

The details of the 2D FFT implementation are given in Table 3 . FMT and POC were performed on a portion of the fingerprint shown in Figure 7. Figure 9 shows the output of the processor for a POC operation of the image on itself. A peak in the center is obtained as expected. Similarly, Figures 10 and 11 compare the processor with a floating point computation for the FMT operation on the same fingerprint image of Figure 7.

Lastly, it needs to be pointed out that a $64 * 64$ point architecture was chosen with 32 internal processing elements for ease of implementation. Only a few cases in the real world might require such low resolution $(64 * 64)$ image

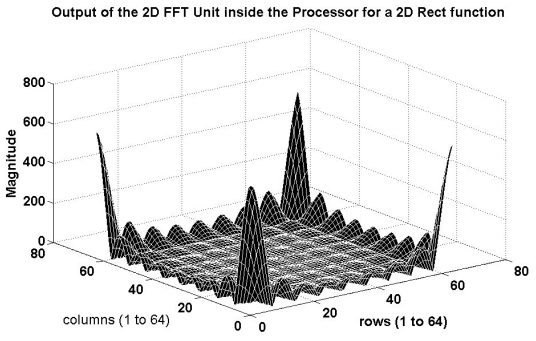

Figure 6. Output from the 16 bit processor doing the 2D FFT operation

Technology	Virtex II FPGA (XV2VP30)
Frequency of operation	45.5 Mhz
Internal word length	$16 + 16$ bits complex
Transform size	$64 * 64$ points
Area	2,591,464 equivalent gates
Memory	40 block RAMs of size 18Kb each

Table 3. Implementation details of 2D FFT on XC2VP30 (Xilinx Virtex II)

Figure 7. Sample Fingerprint image for FMT and POC analysis

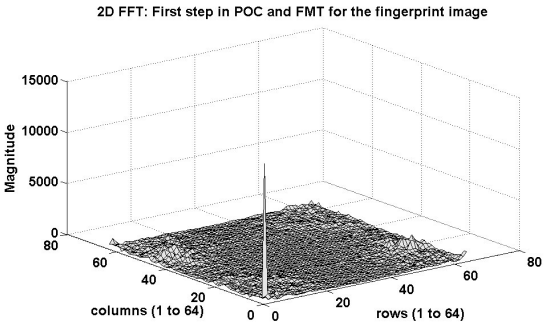

Figure 8. 2D FFT of the image performed by the Processor

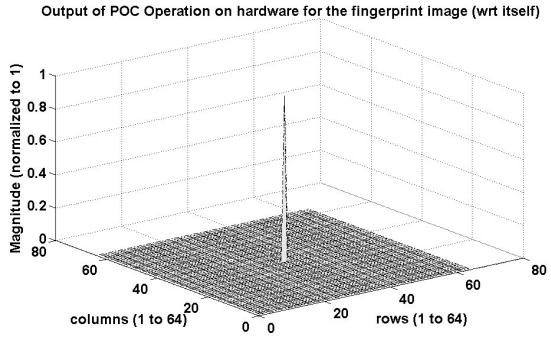

Figure 9. POC result from the Processor

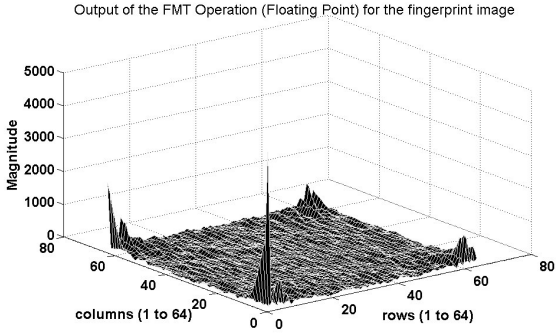

Figure 10. FMT output for the fingerprint image using floating point computation

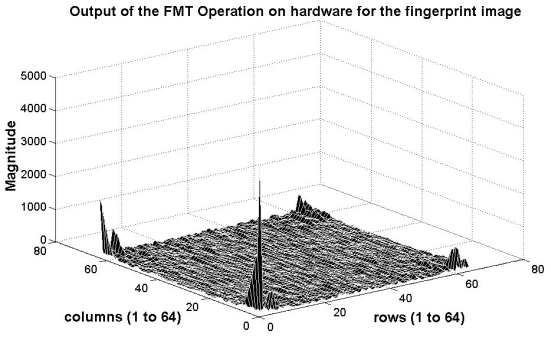

Figure 11. FMT output of the Processor

processing. Initially, the design had 256 point 1D processing elements. Nevertheless, the design still has provisions such that for example, by changing the 64 point 1D processing element to a 256 point unit, one can also achieve a $256 * 256$ point 2D FFT. Results for this kind of a processor will not be greatly different. Here, in expanding the number of points, lies the importance of the FIFO unit at the input port. More data can be accommodated in the buffer if the transform length is higher without getting affected by the number of cycles of operation, ensuring smooth transfer of data from the external world to the processor.

4.2 Application areas

The core of the processor:2D FFT, is an essential tool in image processing. LPT is also becoming increasingly popular. The PRSI processors applications can be grouped into those requiring analytical FMT and FMT on one hand, and LPT and POC on the other. [4] which deals with invariant Content Based Image Retrieval (CBIR) using Fourier Mellin descriptors uses the analytical FMT as its backbone. [22] uses POC as well as FMT for fingerprint matching. The processor is suitable in this case meeting the specifications of resolution as well as the speed of the application concerned. As we have seen earlier, POC and FMT can be achieved by 5 2D FFT and 1 LPT operations. The processor is capable of doing this with ease. [8] and [15] describe the use of FMT in image registration and recognition respectively, again proving the utility of the proposed processor. Some more applications involving FMT and analytical FMT are given in [3] and [7].

Image stabilization in [21], registration in [3] and [8] and other applications like [6] rely heavily on the Log Polar Transform. The processor can be used for these as well.

Goals of the architecture design were to include a set of transformations which an application engineer would most likely need but at the same time keep it general enough so that one can employ a custom designed algorithm with ease. For example, a face recognition processor using the engineers custom algorithm might invariably need some of the transformations that the presented architecture offers. This could then be followed by a PCA or any other analysis technique. A block level diagram of the steps involved in PRSI based on FFT is shown in [16]. This procedure is very well suited to be implemented on the proposed processor.

After PRSI transformation is done, either through FMT or through AFMT, a nominal correlation step is generally necessary to estimate the scaling factor σ. Further, other parameters (like translation and rotation) can also be computed (for example using phase correlation [16]). At the same time, plain 2D FFT, which is itself a versatile operation, can also be performed routinely if required by the host processor using the proposed hardware.

5. Concluding Remark

The design of a 2D PRSI Image Processor and its implementation in Xilinx Virtex 2 FPGA have been presented. The architecture is programmable to execute 2D Fast Fourier Transform, 2D Log Polar Transform, 2D Fourier Mellin Transform, 2D Analytical Fourier Mellin Transform and Phase Only Correlation. The results indicate that the processor can run at 45.5 MHz on the target platform. The architecture of the processor is easily scalable to $256 * 256$ point operation. The processor presented here uses Fourier methods and draws upon their fast implementations. Design choices for 1D FFT, 2D FFT and LPT units were based on the application areas listed above. Design reuse of 1D FFT and its scaling effects have also been explored thoroughly.

Acknowledgments

This work has been partially supported by the Indian Institute of Technology Kharagpur and also by the Portuguese Foundation for Science and for Technology (FCT) under the research project POSI/EEA-CPS/60765/2004.

References

[1] D. Casasent and D. Psaltis. New optical transforms for pattern recognition. *Proceedings of the IEEE*, 65(1):77–84, January 1977.

[2] E. Cetin, R. Morling, and I. Kale. An integrated 256-point complex fft processor for real-time spectrum analysis and measurement. *Instrumentation and Measurement Technology Conference, 1997. IMTC/97. Proceedings. 'Sensing, Processing, Networking'., IEEE*, 1:96–101, May 1997.

[3] Q. S. Chen, M. Defrise, and F. Deconinck. Symmetric phase-only matched filtering of fourier-mellin transforms for image registration and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(12):1156–1168, December 1994.

[4] S. Derrode, M. Daoudi, and F. Ghorbel. Invariant content-based image retrieval using a complete set of fourier-mellin descriptors. *IEEE International Conference on Multimedia Computing and Systems*, 2:877–891, July 1999.

[5] S. Derrode and F. Ghorbel. Robust and efficient fourier-mellin transform approximations for gray-level image reconstruction and complete invariant description. *Comput. Vis. Image Underst.*, 83(1):57–78, July 2001.

[6] A. Gasteratos, P. Gonidis, and I. Andreadis. A hardware structure for time-to-impact computation using log-polar images. *5th International Conference on Technology and Automation (ICTA'05)*, pages 348–353, October 2005.

[7] F. Ghorbel. A complete invariant description for gray-level images by the harmonic analysis approach. *Pattern Recognition Letters*, 15:1043–1051, 1994.

[8] X. Guo, Z. Xu, Y. Lu, and Y. Pang. An application of fourier-mellin transform in image registration. *Computer and Information Technology*, pages 619–623, September 2005.

[9] T. Lenart and V. Owall. Architectures for dynamic data scaling in 2/4/8k pipeline fft cores. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(11):1286–1290, November 2006.

[10] C. V. Loan. Computational framework of the fast fourier transform. *SIAM*, 1992.

[11] C. Mermer, D. Kim, and Y. Kim. Efficient 2d fft implementation on mediaprocessors. *Parallel Computing*, 29(6):691–709, 2003.

[12] M. Morikawa, A. Katsumata, and K. Kobayashi. An image processor implementing algorithms using characteristics of phase spectrum of two-dimensional fourier transformation. *Industrial Electronics, 1999. ISIE '99. Proceedings of the IEEE International Symposium on*, 3:1208–1213, 1999.

[13] M. Morikawa, A. Katsumata, and K. Kobayashi. Pixel-and-column pipeline architecture for fft-based image processor. *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 3:687–690, 2002.

[14] J. G. Proakis and D. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice Hall, 4 edition, 2007.

[15] S. Raman and U. Desai. 2-d object recognition using fourier mellin transform and a mlp network. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 4:2154–2156, November 1995.

[16] B. Reddy and B. Chatterji. An fft-based technique for translation, rotation, and scale-invariant image registration. *Image Processing, IEEE Transactions on*, 5(8):1266–1271, August 1996.

[17] S. Sukhsawas and K. Benkrid. A high-level implementation of a high performance pipeline fft on virtex-e fpgas. *VLSI, 2004. Proceedings. IEEE Computer society Annual Symposium on*, pages 229–232, February 2004.

[18] G. Taylor, R. Steinvorth, and J. McDonald. An architecture for a video rate two-dimensional fast fourier transform processor. *Computers, IEEE Transactions on*, 37(9):1145–1148, September 1988.

[19] I. Uzun, A. Amira, and F. Bensaali. A reconfigurable coprocessor for high-resolution image filtering in real time. *Electronics, Circuits and Systems, Proceedings of the 2003 10th IEEE International Conference on*, 1:192–195, December 2003.

[20] A. Wang and A. Chandrakasan. A 180-mv subthreshold fft processor using a minimum energy design methodology. *Solid-State Circuits, IEEE Journal of*, 40(1):310–319, January 2005.

[21] F. Yuan, H. Zhang, and R. Jia. Digital image stabilization based on log-polar transform. *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on*, pages 769–773, August 2007.

[22] J. Zhang, Z. Ou, and H. Wei. Fingerprint matching using phase-only correlation and fourier-mellin transforms. *Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on*, 2:379–383, October 2006.