FPGA Based Hardware Implementation of Image Filter With Dynamic Reconfiguration Architecture

***B. Rajan and **S.Ravi**
*Research Scholar, Dept. of ECE, Dr. M.G.R. University, Chennai-95.
**Professor and Head, Dept. of ECE, Dr. M.G.R. University, Chennai-95

ABSTRACT

Reconfigurable computing has been proposed for image and signal processing applications with various objectives, including high performance, flexibility, specialization, and most recently, adaptability. Reconfiguration is characterized by how fast the reconfiguration can occur and how many possible reconfigurations can be used and this feature is referred to as dynamic reconfiguration. For many image processing systems [5], it is possible to exploit variations in image signals to vary computation and memory requirements. In this paper, based on noise levels at a specific time instant, minimally sufficient hardware resources are dynamically allocated to meet the MDPP requirements of the application. These architectures can be characterized via a set of architectural parameters which can be determined experimentally. In this work, the analysis and hardware implementation of a dynamic reconfigurable unit based image filtering algorithm is described. This work is the first operational implementation of the reconfigurable architecture and its algorithm and is targeted to a Xilinx 600K Spartan-IIE FPGA to take advantage of computational specialization and parallelism. Our work has the capability to adapt the amount of computation performed and the amount of storage used at both a fine-timescale (ms) and coarse-timescale (s) level. Experimental results show that the overall run-time of the image filter implementation on a Spartan-IIE FPGA, including bus overhead, is up to 400 times faster than a software implementation on a 2.8GHz Pentium processor.

Keywords: Reconfigurable computing, FPGA implementation, Image processing.

1. INTRODUCTION

General-purpose image filters lacks the flexibility and adaptability for un-modeled noise types. The reconfiguration circuit based image filtering algorithm is considered as a means to enhance performance without compromising image visual quality. The hardware resource requirements can be altered in response to noise conditions for a fixed MDPP. A significant amount of noise demands a large number of repeated reconfigurations, to achieve a MDPP similar to that achieved for a less noisy image. In this work, in implementing the VRC based image filter, the dynamic reconfiguration architecture is proposed by considering both the reconfiguration options, *fine-timescale* and *coarse-timescale* reconfiguration. Coarse-timescale reconfiguration of the image filter is performed in accordance to variations in noise conditions over seconds. Reconfiguration at this time scale minimizes the performance impact of millisecond FPGA reconfiguration times. Coarse-timescale reconfiguration is motivated by changing noise characteristics from parameters such as weather, distance, or camera battery-power. These parameters result in a signal-to-noise ratio (*SNR*) that changes relatively slow. When more improved image quality is required, a lower clock-speed can be used. When less accuracy is required, a higher-performance circuit is swapped in. If dynamic reconfiguration was not allowed, the lower-performance circuit would always need to be resident. Current FPGA architectures [9] require reconfiguration times measuring in milliseconds.

2. IMAGE CHARACTERIZATION

In order to process an image it is essential to identify its basis components. The term image refers to a two dimensional light intensity function [5], denoted by g(x, y), where the value at spatial coordinates (x, y) gives the intensity of the image at that coordinate. As light

is a form of energy, g(x, y) must be non-zero and finite, i.e.

$$0 < g(x, y) < \infty$$

The basic nature of g(x, y) may be represented by two components namely: The amount of light incident on the object being viewed and the amount of light reflected by the object. These two components are called the illumination and reflectance components respectively, and are denoted by i(x, y) and r(x,y), so that,

$$g(x, y) = i(x, y) \times r(x, y) \quad ---(1).$$

In (1) the nature of i(x, y) is determined by light source and r(x, y) is determined by the characteristics of the object

3. FPGA BASED IMAGE FILTER

The proposed filter considers spatial domain approach and uses the overlapping window to remove the noise in the image. The approach chosen here is based on functional level evolution whose architecture contains many nonlinear functions and uses an evolutionary algorithm to evolve the best configuration [9]. The digital image filter contains dynamic reconfigurable circuit (DRC) together with genetic unit. The corrupted image is given as input to the reconfigurable circuit and the filtered image is obtained. The filtered image is compared with the original image and the fitness is evaluated. The DRC processes nine 8-bit inputs I0 – I8 and produces a single 8-bit output and consists of a total of 25 PEs. Every pixel value of the filtered image will be calculated using a corresponding pixel and its eight neighbors.

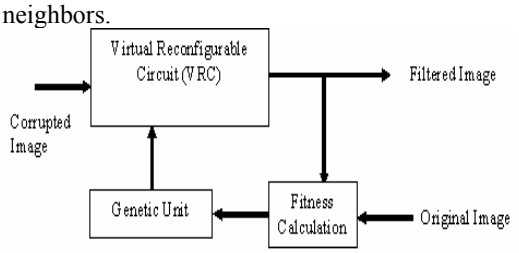

Figure 1 Block diagram of FPGA based image filter using VRC

4. IMPLEMENTATION OF THE GA PROCESSOR

The Genetic Algorithm (GA) is a powerful optimization algorithm inspired by natural evolution [8]. The optimization is generally performed by creating a population of solutions. In GA, the offspring are produced by standard genetic operators: reproduction, crossover, and mutation. The GA implementation configures the evolving design by placing individuals in Random Access Memory (RAM). In each generation, a selection scheme is used to select the survivors to the next generation according to their fitness values defined by users. With this artificial evolution, the solutions are gradually improved generation by generation. The GA process starts with a random population and iterates until the termination condition is met i.e. the optimal solution is found, or reaches the maximum number of generations. Over the past years, GA has been successfully applied to many hard optimization problems. On the limitations side, however, the GA process is time-consuming. For many real world applications, GA can run for days, even when it is executed on a high-performance workstation and, the algorithm's memory requirement and computation count pose a performance obstacle when configuration bit size and its fitness function evaluation are large.

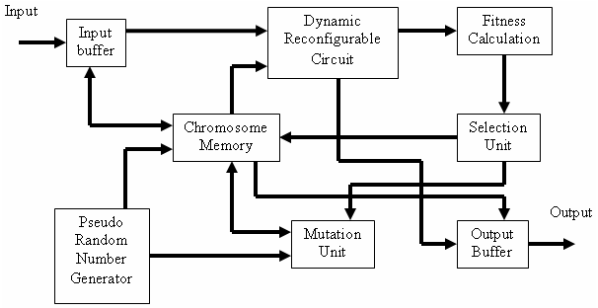

Figure 2 GA Processor (GAP) along with the dynamic reconfigurable block

The hardware implementation of GAP consists of the basic modules; pseudo random number generator, population memory, selection unit, mutation unit, fitness evaluator and output buffer as shown in figure 2. A Pseudo Random Number Generator (PRNG), is used in two of the major steps in GA. First, during initial population creation, and next to select individuals and mutate or copy. In this work, the PRNG for FPGA implementation is done using a Linear Feedback Shift Register (LFSR). A word size of 12 is chosen. It is important to choose a good polynomial to ensure that the RNG can generate a maximal sequence of 2^n-1 random numbers, while keeping the number of taps to a minimum for efficiency. For a 12 bit word the polynomial x^{12} (xnor) x6 (xnor) x^4 (xnor) x^1 is used. The

block diagram of the LFSR is shown in Figure 3. The RNG is designed such that a random number is generated in every clock. The 12th bit is taken as the random bit. To create a 10 bit random number, 10 single bit pseudo random number generators are combined in parallel.

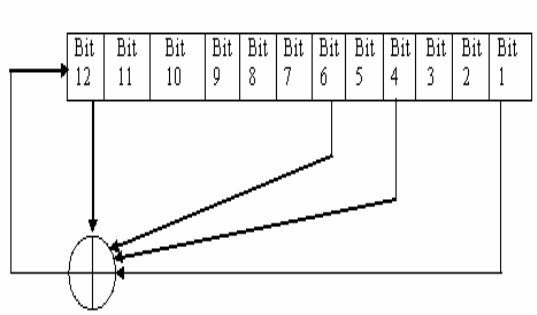

Figure 3 LFSR based 12-bit random number generator

4.1 VHDL IMPLEMENTATION OF GAP

The VHDL coding is used to describe the different modules of the GA processor. The 12-bit random number is obtained in parallel vector processors represented by signals s1 to s16. The implementation result captured using the Modelsim package is shown in figure 4.

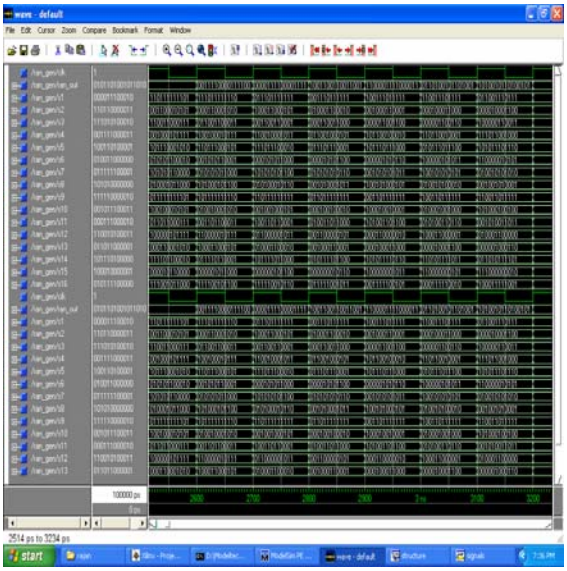

Figure 4 Synthesized test-bench waveforms of the GAP using Modelsim package

5. PERFORMANCE MEASURE AND FITNESS FUNCTION

Various performance measures and image features exist to measure image visual quality. The Peak Signal-to-Noise Ratio (PSNR) and the **M**ean **D**ifference **P**er **P**ixel (MDPP) are the commonly used approaches. The fitness function using the PSNR is given by

$$PSNR = 10\log_{10}\frac{255^2}{MSE} \quad dB$$

The fitness function using MDPP is given by

$$MDPP = \frac{1}{NxN}\sum_{i,j=1}^{N}|\,orig(i,j) - filt(i,j)\,|$$

where |orig(i,j) – filt(i,j)| is the absolute difference between the original and filtered images [5]. The MDPP fitness function is computationally easier for hardware implementation as compared to PSNR and hence, in this work MDPP function is taken for the fitness calculation.

6. ALGORITHM FOR IMAGE FILTERING

The algorithm for digital image filtering consists of the following seven steps:

1. Read the corrupted and original images and store in a buffer.

Input buffer consists of RAM. Original and distorted images are read from the external source and stored in the input buffer. During runtime pixels are given as input to the dynamic reconfigurable circuit (DRC) from the input buffer.

2. Generate initial population of size ‘n’ with a chromosome length of L each.

During **initial population** creation, a 250 bit chromosome is created using 10 bit random number generator in 25 clock cycles. Chromosomes are stored in the Block RAM of FPGA. The initial population size is taken as 16. Totally 16x25 clock cycles are needed for initial population generation.

3. For each chromosome in the population
   a) Take 3x3 overlapping window and input 9 pixel values to the VRC to replace the center pixel. Repeat for full image.
   b) Calculate the Mean Difference Per Pixel (MDPP) and Fitness.

c) Retain the chromosome that has maximum fitness.

Fitness function is used to select the best chromosome. The original and DRC output values are taken from the memory and the absolute difference between them is computed and the fitness is evaluated. The chromosome which has **highest fitness** is selected as the best chromosome and it is retained for subsequent generations.

4. Select parent chromosomes according to roulette wheel.
5. Apply crossover and mutation operations on the selected chromosomes to get Childs.
6. Replace the old population.

The chromosome which has highest fitness is selected for **mutation**. Bit by bit mutation is used for the creation of childs. Fifteen new childs are created in every generation and stored in the population memory.

7. Repeat from step 3 for N number of generations

After the specified number of generations the evolution is complete and the best chromosome is stored in the memory. The fitness value is obtained and stored in the **output buffer**.

7. FPGA BASED HARDWARE IMPLEMENTATION

A more effective approach is to perform image filtering using a FPGA based dynamic reconfigurable circuit (DRC) architecture as this does not require any apriori knowledge of noise characteristics. In the FPGA based implementation decoder computation and memory requirements are minimized to support faster performance. The algorithms discussed in section 6 of this paper are implemented in real-time on a high performance video processing system designed using FPGA architecture.

Figure 5 FPGA BASED Hardware Board

The developed hardware board is shown in figure 5. The architecture is compactly designed by exploiting the inherent parallelism built into FPGA based solutions. The designed module has the advantage that it is possible to execute video processing algorithms many times faster than that could be achieved if implemented using conventional DSP processors. The onboard FPGA acts as a video processing engine and multiple coefficient and data memories provides a general purpose platform to the user for implementing all the video processing needs

7.1 FEATURES OF THE HARDWARE MODULE

The hardware unit [7] consists of the following onboard modules:

- 600K Spartan-IIE FPGA chip
- Data Memory of size 4M x 16 each of which is arranged as 4 independent banks of 1M x 16 SRAMs
- Coefficient Memory of size 1M x 16 arranged as two independent banks of 512K x !6 Flash ROMs
- One analog input channel that can sample the input signal at the rate of 10 MSPS. The converted digital output is of size 14-bit.
- Onboard 10 bit video DAC to provide composite video signal RGB/YUV.
- Two channel full duplex serial port RS232 compatible interface
- 32-bit/33MHz PCI interface unit

The performance improvement achieved in this work can be attributed to the parallelism available within the FPGAs. Under reduced-noise conditions, a less complex hardware design can be swapped into the FPGA to achieve the same accuracy. Due to the reduced complexity of the algorithm, logic resource requirements are reduced by more than a factor of two compared to standard implementations. An empirical study shows that hardware requirements for our image filter grow at a rate substantially less than the exponential growth exhibited by standard algorithms.

8. EXPERIMENTAL RESULTS

8.1 IMAGE NOISE FILTERING

The original and distorted bitmap images are stored in input buffer initially. Simulations were performed using Gaussian noise distorted

bitmaps. Bitmap of IEEE test images were used as the target images at different distortion levels for testing the performance of the FPGA DRC architecture. All results were compared with the results obtained from a Gaussian filter. The results are shown in Mean Difference, MSE in dB and PSNR in dB for different images with different levels of variance in Tables 1 to 3.

Figure 6a

Figure 6b

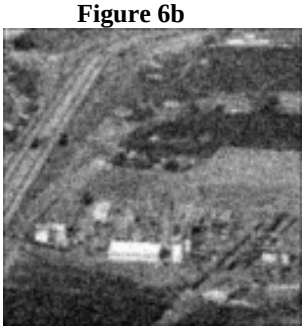

Figure 6c

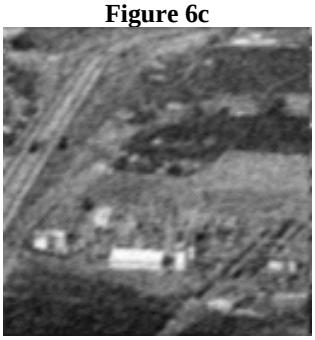

Figure 6d

Fig. 6a Original Chemical Plant image 128x128, Fig. 6b Image Distorted by Gaussian Noise of Mean 0 and Variance 0.006 Fig. 6c Image Filtered by Gaussian Filter, Fig. 6d Image Filtered by FPGA filter.

Figure 7a

Figure 7b

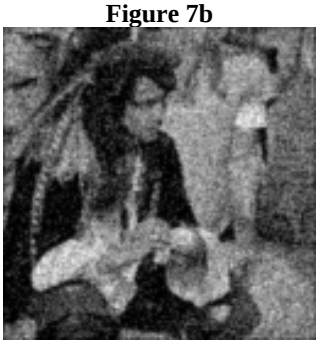

Figure 7c

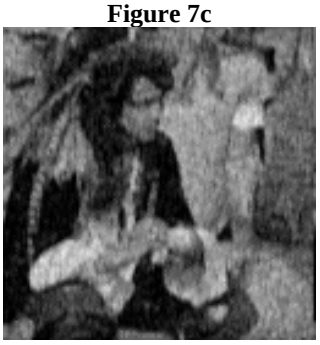

Figure 7d

Fig. 7a Original Man image 128x128, Fig. 7b Image Distorted by Gaussian Noise of Mean 0 and Variance 0.009, Fig. 7c Image Filtered by Gaussian Filter, Fig. 7d Image Filtered by FPGA filter.

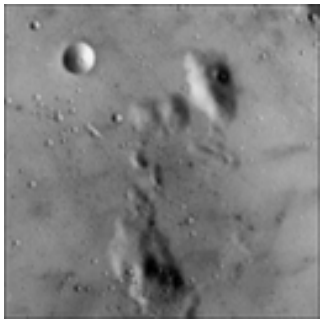

Figure 8a

Figure 8b

Figure 8c

Figure 8d

Fig. 8a Original Moon image 128x128, Fig. 8b Image Distorted by Gaussian Noise of Mean 0 and Variance 0.01,Fig. 8c Image Filtered by Gaussian Filter, Fig. 8d Image Filtered by FPGA filter.

Table 1 Comparison of Mean Difference Per Pixel for various standard test images

Image	Variance	Gaussian Filter	FPGA Filter
Chemical Plant	0.006	173449	143152
Man	0.009	203399	180606
Moon	0.01	217764	143384

Table 2 Comparison of Mean Square Error (dB) for various standard test images

Image	Variance	Gaussian Filter	FPGA Filter
Chemical Plant	0.006	22.44	21.08
Man	0.009	23.96	23.14
Moon	0.01	24.41	20.98

Table 3 Comparison of PSNR (dB) for various standard test images

Image	Variance	Gaussian Filter	FPGA Filter
Chemical Plant	0.006	25.69	27.05
Man	0.009	24.17	24.99
Moon	0.01	23.72	27.15

8.2 FPGA RESOURCE USAGE

The logic resources used by the dynamic reconfigurable algorithm described are measured in terms of configurable logic block (CLB) usage [6]. The time generation report for each of the individual operations of the FPGA based filter is shown in Table-4. Also, Table-5 summarizes the resource utilization of the algorithm on the Spartan-IIE FPGA. From Table 5, it can be observed that the implemented algorithm fits within the device. The values in Table-4 and 5 were obtained with the FPGA run at the maximum possible frequency. The measurement of the total time taken to execute the algorithm on the FPGA includes the 33 MHz PCI bus access time. The total time was then divided by the length of the sequence to obtain time per bit. The results indicate that the FPGA based implementation outperforms the processor based implementation.

Table 4 FPGA Operations

FPGA Operation	Time Taken (ms)
Image Storage	.409
Random Number generator (Population Generation)	.1
Individual output evaluation	.409
Single population	6.55
New population generation	.093

Raw FPGA decoding time per bit was obtained from HDL simulations.

8.3 COMPARISON TO A PROCESSOR IMPLEMENTATION

The performance of our Spartan-IIE FPGA based implementation of the algorithm was compared with a 2.8GHz Processor based implementation. A 'C' version of the algorithm was compiled. Subsequent simulation was performed and cycle counts were obtained [4]. Table-5 shows the performance comparison between the FPGA and Processor based implementations of the image filter. The comparison indicates that an FPGA implementation achieves a speed-up of up to 400x versus the processor implementation (without bus and API overheads). Table-5 shows that bus and API overheads slow down decoding by only a factor of 1.25 to 2.

Table 5 Performance comparison between FPGA and 2.8GHz processor implementation

CLBs	FF	Processing time (sec)			FPGA Clock (MHz)	Speed Up %
		CPU	FPGA (No Over head)	FPGA (PCI Over head)		
563	286	2250	2.996	5.376	40.5	418.6
1200	551	4762	6.09	8.496	20.1	560.5
1213	732	5263	6.161	8.598	19.9	612.1
1226	766	5620	6.218	8.756	19.7	641.85
1290	799	5814	6.963	9.040	17.6	643.14
1301	832	6944	7.087	9.276	17.3	748.6

9. CONCLUSION

In this paper, a novel dynamically reconfigurable circuit (DRC) based image filter was presented. The DRC was implemented on the Spartan-IIE FPGA board and important algorithm parameters were determined. The FPGA based implementation was applied to a PCI-based system. The approach has shown significant speed up versus software implementation on a 2.8GHz processor. Through hardware implementation it was shown that the present FPGA based image noise filter can be used to effectively improve overall performance by at least 25%. If noise corrupting the image increases, a more accurate but slower running DRC filter is swapped into the FPGA hardware and similarly any decrease in noise introduces the opposite effect.

10. REFERENCES

1. Murakawa, M., Higuchi, T., Iwata, M., Kajitani, I., Liu, W., and Salami, M. (1997). "Evolvable Hardware at Functional Level" IEEE.
2. Xilinx Corporation (2001) Xilinx Virtex Data Sheet, San Jose, CA. http://www.xilinx.com.
3. Iwata, M., Kajitani, I., Liu, Y., Kajihara, N., Higuchi, T. "Implementation of a Gate-Level Evolvable Hardware Chip" IEEE.
4. J.Anderson and S.Mohan, "Sequential coding algorithms: A survey and cost analysis", IEEE Trans . Commn., Vol. Com-32, Pp. 169-176, Feb 1984.
5. Anil k Jain, "Advanced Digital Image Processing", PHI India.
6. L.Shang, A.Kaviani and K.Bathala, "Dynamic in Proc. ACM/SIDGA Int. Symp. Field Programmable Gate Arrays, Monterey, CA, Feb 2002, Pp.157-164.
7. Xilinx Corporation, "ISE Manual", San Jose, CA,2001.
8. Goldberg, D. E. (1989). "Genetic Algorithms in Search, Optimization & Machine Learning". Pearson Education, Inc.
9. Hollingworth, G., Smith, S. and Tyrell, A. (2000). "Safe intrinsic evolution of Virtex devices". In Proc. of 2 nd NASA/DoD Workshop on Evolvable Hardware, IEEE.

11. BIOGRAPHIES

1) **Mr.B.Rajan** is presently a research scholar in ECE Dept. Dr.M.G.R. University, Chennai. His areas of interest include image processing, reconfigurable computing and VHDL programming.

2) **Dr.S.Ravi** is presently the professor and Head, Department of ECE, Dr.M.G.R. University, Chennai. He has published more than ten papers in international/national journals.