Digital Filter Design using Evolvable Hardware Chip for Image Enhancement

***A.Sumathi, and **Dr.R.S.D.Wahida Banu**

*Associate Professor, Adhiyamaan college of Engg., Hosur
**Professor, Government college of Engineering, Salem, India

Abstract: Images acquired through modern cameras may be contaminated by a variety of noise sources (e.g. photon or on chip electronic noise) and also by distortions such as shading or improper illumination. Therefore a preprocessing unit has to be incorporated before recognition to improve image quality. General-purpose image filters lacks the flexibility and adaptability for un-modeled noise types. The EHW architecture evolves filters without any apriori information. Adaptability is accomplished by calculating a score for the results obtained by the system on a set of data. This score is then used to manipulate the system in such a way that the system output will converge towards a desired result. Using evolutionary techniques on a digital filter-system is a possible method for obtaining system adaptability. 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. The proposed filter considers spatial domain approach and uses the overlapping window to remove the noise in the image.

Keywords: Image processing, Evolvable hardware, Genetic algorithm.

1. INTRODUCTION

Many of today's image and signal processing tasks are performed on real-time data. On systems that perform real-time processing of data, performance is often limited by the processing capability of the system. Providing a high processing speed is therefore often a crucial factor to be considered when implementing real-time systems. Also there is a need to have flexible systems that can be changed according to new specifications. Systems based on software are flexible, but often suffer from insufficient processing capability. Alternately, dedicated hardware can provide the highest processing performance, but is less flexible for changes. Reconfigurable hardware [1] devices offer both the flexibility of computer software, and the ability to construct custom high performance computing circuits. Thus, in many cases they make a good compromise between software and hardware solutions. The structure of a reconfigurable hardware device can be changed any number of times by downloading into the device a software bit string called configuration bits. Field Programmable Gate Arrays (FPGA) and Programmable Logic Devices (PLD) are typical examples of reconfigurable hardware devices.

Evolvable Hardware (EHW) is a new concept in the development of online adaptive machines. In contrast to conventional hardware where the structure is irreversibly fixed in the design process, EHW is designed to adapt to changes in task requirements or changes in the environment through its ability to reconfigure its own hardware structure online and autonomously (Higuchi 1999). The capacity for adaptation is achieved through evolutionary algorithms such as Genetic Algorithm (GA).

In this paper, an EHW chip is configured using evolutionary algorithms to remove the noise and improve the quality of the images. The basic idea of EHW is to regard the architecture bits of a reconfigurable device as a chromosome for GA, which searches for an optimal hardware structure. The GA chromosome (architecture bits) is downloaded onto the reconfigurable device during genetic learning [3]. Thus, EHW can be considered as an online adaptive hardware.

In the field of digital image processing, a broad and disparate range of applications using evolutionary computation can be found in the literature. The functional level EHW architecture is described in (Clark 1999). The hardware implementation of the Genetic Algorithm model is explained in (Lei 2002). The evolution of spatial masks to detect edges within gray scale images is described in (Hollingworth 1999). (Sekanina 2002) (Sekanina 2003) has achieved evolutionary design of image filters with virtual reconfigurable circuits in extrinsic EHW environment. Digital image processing operations, such as image smoothing, edge detection, and image compression, have been carried out in an extrinsic EHW environment, which exhibits the potential of EHW in digital image operator design. This paper presents complete evolvable

hardware architecture, dedicated for implementing high performance digital image filters on FPGA, so that the time for the evolution is greatly reduced. In Complete Hardware Evolution (CHE) the GA is implemented on the same chip as the evolving design. The GA implementation configures the evolving design by placing individuals in Random Access Memory (RAM). The fitness value is calculated by the GA from the feedback signals originating in the evolving design. Since the GA and the evolving design are implemented on the same chip, the evolution process may continuously observe the evolving design.

2. EVOLVABLE HARDWARE

Evolvable Hardware is the combination of Genetic Algorithms and the software reconfigurable devices. The structure of the reconfigurable device can be determined by downloading binary bit strings called the architecture bits (Gordon Hollingworth 2000). The architecture bits are treated as chromosomes in the population by the GA, and can be downloaded to the reconfigurable device resulting in changes to the hardware structure. The changed functionality of the device can then be evaluated and the fitness of the chromosome is calculated. The performance of the device is improved as the population is evolved by GA according to fitness. This process is repeated till the desired performance is achieved or particular number of generations. Figures 1 and 2 show the function level and gate level evolution model. The basic concept of evolvable hardware is shown in Figure 3.

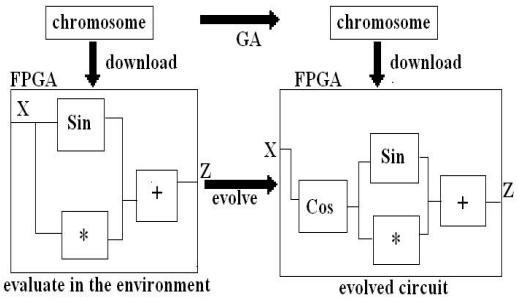

Figure 1 Functional level evolution

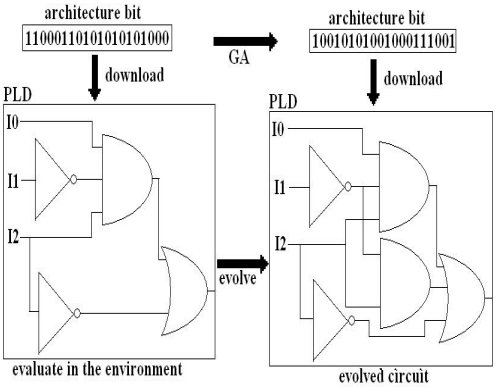

Figure 2 Gate level evolution

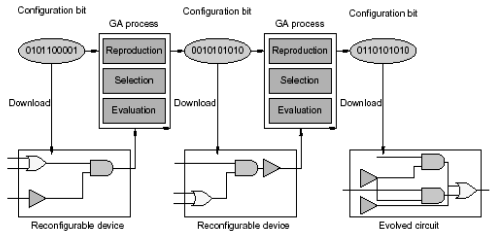

Figure 3 Basic concept of EHW

In gate-level evolution, the hardware evolution is based on primitive gates such as AND-gates and OR-gates as shown in Figure 1. The size of circuits generated by gate-level evolution is not very large because GA execution takes a long time to evolve large circuits. This makes it difficult to use gate-level EHW to produce hardware functions which are useful for practical applications.

In function level evolution, hardware synthesis involves higher-level hardware functions than the primitive gates of gate-level evolution. With function-level evolution, it is possible (1) to synthesize more useful hardware functions and (2) to design larger hardware circuits that are not possible with gate-level evolution.

EHW offers three advantages over traditional hardware and software systems. First, it can autonomously improve its performance by changing its hardware configuration according to the GA. Second, it processes information much faster than software systems. Third, the reconfigurable devices can change their functionality in an on-line fashion during execution. EHW can therefore be applied to new areas of application, where more inflexible traditional hardware systems are not efficient.

3. GENETIC ALGORITHM

Genetic Algorithm (Goldberg 1989) determines how the hardware structure should be reconfigured whenever a new hardware structure is needed for an improved performance. GA was proposed to model adaptation of natural and artificial systems through evolution, and is one of the most well known powerful search procedures. The canonical GA has a population of chromosomes; each of them is obtained by encoding a point in the search space. Usually, they are represented by the strings of binary characters.

The sequence of operations performed by the GA is shown in Figure 4. In the initial state, chromosomes in the population are generated at random, and processed by many operations, such as evaluation, selection, crossover and mutation. The latter three operations are called the genetic operations, and one cycle of the evaluation and the genetic operation is counted as a generation. The evaluation assigns the fitness values to the chromosomes, which indicates how well the chromosomes perform as solutions of the given problem. The crossover chooses some pairs of chromosomes, and exchanges their sub-strings at random. Finally, the mutation randomly chooses some positions in the chromosome and flips their values.

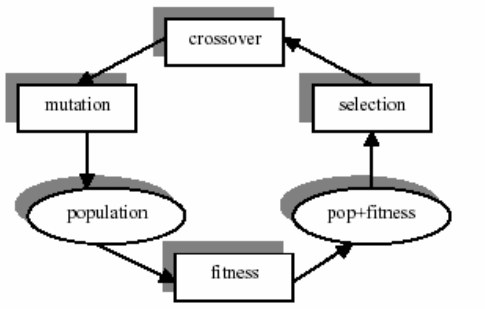

Figure 4 Flow of Genetic algorithm

The major advantages of GA are its robustness and superior search performance particularly in problems without aprior knowledge. If the evaluation can be executed very quickly by the specific hardware device, the most serious problem of GA i.e. time constraint for fitness evaluation can be solved, and one can use GA more effectively. Complete evolvable hardware is based on this concept, and utilizes the robust capability of GA by reducing its computational cost.

4. HARDWARE IMPLEMENTATION OF EHW

Although various evolvable systems have been implemented as Application Specific Integrated Circuits (ASIC), this solution is relatively expensive (Higuchi 1999). Hence a great effort is invested to designing evolvable systems at the level of FPGAs. These solutions can be divided into two groups:

1)FPGA is used for evaluation of circuits produced by evolutionary algorithm, which is executed in software.

2)The whole evolvable system is implemented in the FPGAs. This type of implementation integrates a hardware realization of evolutionary algorithm and a reconfigurable device.

The typical feature of these approaches is that the chromosomes are transformed to configuration bit stream and the configuration bit stream is uploaded into the FPGA. However, it is not easy to decode usually very complex configuration bit stream of FPGA vendors. Furthermore, most families of FPGAs can be configured only externally (i.e. from an external device connected to the configuration port). Internal reconfiguration means that a circuit placed inside an FPGA can configure programmable elements of the same FPGA. Although the Internal Configuration Access Port (ICAP) has been integrated into the newest Xilinx Virtex II family, it is still too slow (Sekanina 2004).

Virtual Reconfigurable Circuits (VRC) was introduced for digital evolvable hardware as a new kind of rapidly reconfigurable platform utilizing conventional FPGAs (Sekanina 2004). The approach utilizing VRC offers many benefits, such as

1) It is relatively inexpensive, because the whole evolvable system is realized using an ordinary FPGA.
2) The architecture of the reconfigurable device can be designed exactly according the needs of a given problem.
3) Since, the whole evolvable system is available at the level of Hardware Description Language (HDL) code it can easily be modified and synthesized for various target platforms (FPGA families).

5. IRTUAL RECONFIGURABLE CIRCUIT (VRC)

Figure 5 shows the VRC. It is in fact a new reconfigurable circuit consisting of 8 programmable elements realized on top of an ordinary FPGA. Slices have to implement a new array of programmable elements, new routing circuits and new configuration memory. The virtual circuit can be configured internally or from FPGA's I/O pins if new configuration memory is

connected to them. The main advantage of the VRC is that the array, the routing circuits and the configuration memory can be designed exactly according to the requirements of a given application. Furthermore, style of reconfiguration and granularity of new virtual reconfigurable circuit can exactly reflect the needs of a given application.

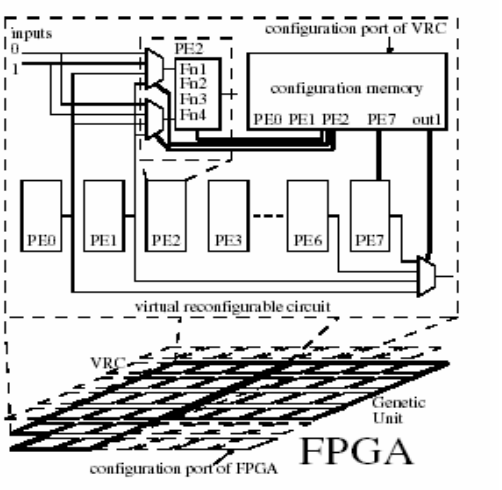

Figure 5 Example of VRC used in EHW

The elements in VRC are referred as Processing Elements (PEs). Configuration bits determine PEs function and the places where its inputs are connected to. The routing circuits are implemented using multiplexers. The configuration memory is composed of flip-flops. All bits of the configuration memory are connected to multiplexers that control routing and selection of functions in PEs. The number of PEs utilized in the virtual reconfigurable circuit depends on a given application. VRC can be described in HDL and synthesized with various constraints for different target platforms.

6. DIGITAL IMAGE FILTER DESIGN

The digital image filter contains virtual reconfigurable circuit together with genetic unit. The corrupted image is given as input to the virtual reconfigurable circuit and the filtered image is obtained. The filtered image is compared with the original image and the fitness is evaluated. According to the fitness value, genetic unit finds the most desirable architecture of the virtual reconfigurable circuit such that the difference between the filtered image and the original image will get reduced. This is shown in figure 6.

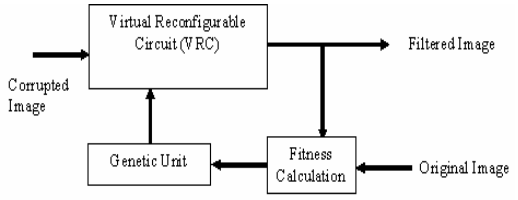

Figure 6 Block diagram of digital image filter using VRC

The VRC processes nine 8-bit inputs I0 – I8 and produces a single 8-bit output. Every pixel value of the filtered image will be calculated using a corresponding pixel and its eight neighbors. The VRC consists of 25 PE's as shown in Fig. 4.2. The PE's are indexed from the top, row-wise, and then column-wise. Four PE's are considered as a single stage of the pipeline.

Each PE has two 8-bit inputs and gives a single 8-bit output. The outputs of PE's are equipped with registers. The two inputs to every PE can be connected to one of the outputs from the previous *l* columns where *l* is the level back parameter. Every PE executes a certain function from Table 1, depending on the function code configuration, sel3 which is applied to its two inputs.

The architecture of VRC with the 25 PE's and that of the single PE are shown in Figures 7 and 8 respectively. The configuration bit stream consists of ten bits for each PE. First six bits determine the places where PEs inputs will be connected to. One of 16 functions is selected from Table-1 using the last four bits. The output of the PE is given by

$$\text{Output} = F\{\text{mux}(\text{sel1}), \text{mux}(\text{sel2}), \text{sel3}\}$$

Table-1

Function Code	Function	Function Code	Function
0000	X >> 1	1000	X & 0x0F
0001	X	1001	X & 0xF0
0010	~ X	1010	X | 0x0F
0011	X & Y	1011	X | 0x F0
0100	X | Y	1100	Min(X,Y)
0101	X ^ Y	1101	Max(X,Y)
0110	(X+Y)>>2	1110	Y<<1
0111	(X+Y) >>1	1111	X+Y

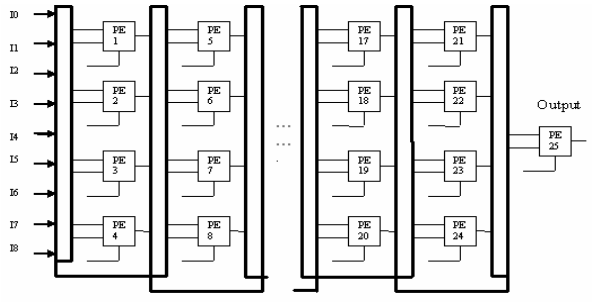

Figure 7 Architecture of VRC with 25 PE's

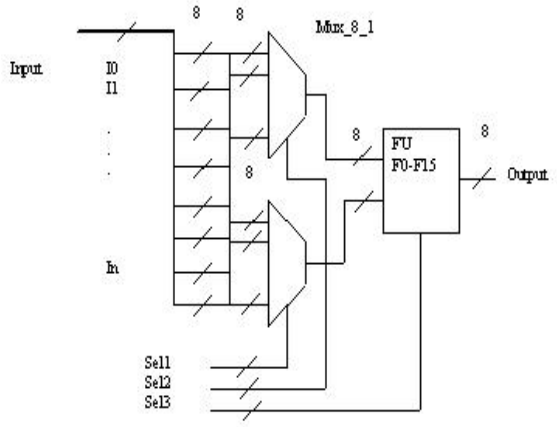

Figure 8 Architecture of a single PE

Both sel1 and sel2 should not exceed the number of the multiplexer inputs. The sel3 input is the binary representation of the number of functions in the Table-.1. The fewer the functions, the faster is the evolution. Further functions can be included but this is dependant on the resource requirements, as there is a trade-off between the functionality and the complexity of the hardware structure.

7. DETAILS OF EHW CHIP

The proposed hardware implementation of the complete evolvable system is composed of basic modules; input buffer, virtual reconfigurable circuit, pseudo random number generator, population memory, selection unit, mutation unit, fitness evaluator and output buffer as shown in Figure 9. Both the Genetic unit and the Virtual reconfiguration unit reside in the chip and hence the configuration is complete. The configuration word contains details about the interconnection between the processing elements (PE) of the VRC and the functional operations performed within each PE. In this work, the interconnection between the PE's is not restricted to its nearest neighbors.

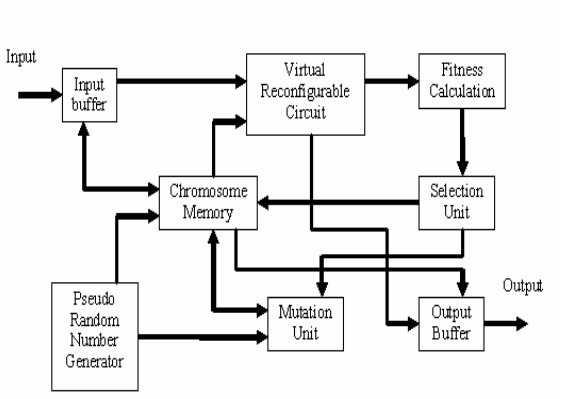

Figure 9 Block diagram of the complete EHW system

7.1 Pseudo Random Number Generator

The Pseudo Random Number Generator (PRNG) (Alfke 1996) is used in two of the major steps in GA. Firstly, during initial population creation, and secondly to select individuals for crossover and mutation. One of the most common PRNG for FPGA implementation is a Linear Feedback Shift Register (LFSR). In this work, a word size of 12 is chosen. It is important to choose a good polynomial to ensure that the PRNG can generate a maximal sequence of 2^n-1 random numbers, while keeping the number of taps to a minimum for efficiency. For the chosen 12 bit word the polynomial

$$x^{12} \text{ (xnor) } x^6 \text{ (xnor) } x^4 \text{ (xnor) } x^1$$

is used. The block diagram of the LFSR is shown in Figure 10.

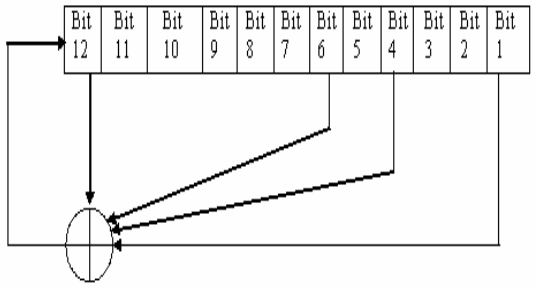

Figure 10 Pseudo random noise generator

The PRNG is designed so that a random number is generated in every clock. The 12^{th} bit is taken as the random bit. Initial seed value is loaded in to the PRNG. To create a 10 bit random number, 10 single bit PRNG are combined in parallel.

7.2 Input Buffer

Input buffer consists of RAM. Original and distorted images are read from the file and stored in the input buffer. During runtime pixels are given as input to the VRC from the input buffer.

7.3 Initial Population Generation

During initial population generation, 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.

7.4 Fitness Calculation

MDPP fitness function is used to evaluate the chromosome. The original and filtered images are taken from the memory and the absolute difference between the corresponding pixel values is added and the fitness is evaluated.

7.5 Selection Unit

Selection unit selects the chromosome which has highest fitness as the best chromosome and it is retained for subsequent generations.

7.6 Mutation Unit

The chromosome which has highest fitness is selected for mutation. Using PRNG, bit by bit mutation is done for the creation of Childs. Fifteen new Childs are created in every generation and stored in the population memory.

7.6 Output Buffer

After the specified number of generations the evolution is completed and the best chromosome is stored in the memory. The fitness value and filtered image signal are calculated and stored in the output buffer.

8. CHROMOSOME FORMAT

The logical configuration of the circuit is defined by a set of 250 bits, 10 bits for each one of the 25 PEs in the reconfigurable architecture as shown

111 000 0000 010 001 1100 110 010 0011 111 011 0010 110 111 0100 …… 101 000 1000 110 010 1010

The first six bits of each ten bits represent the source of inputs to the PE (sel1& sel2) as labeled in Figure 8. The other four bits of each ten bit (sel3) indexes the function from Table 4.1 to be applied by the PE.

9. FITNESS FUNCTION

Various approaches 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 MDPP fitness function is computationally easier for hardware implementation as compared to PSNR. In this work MDPP fitness function is taken for the fitness calculation.
The fitness function using the PSNR is given by

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

The fitness function using MDPP is given by

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

where

$$|\mathrm{orig}(i,j) - \mathrm{filt}(i,j)|$$

is the absolute difference between the original and filtered images (Jain 2003).

10. DISCUSSION OF RESULTS

The original and distorted bitmap images are stored in input buffer initially. It takes 128x128 clock cycles to store each image. At the same time using random number generator generates 16 initial chromosomes. 16x25x10 clock cycles are needed to generate the initial population. 128x128 clock cycles are used to evaluate the output for each chromosome. Totally 16x128x128 clock cycles are needed to evaluate the output for all chromosomes. To select the best chromosome 16 clock cycles are needed. Bit by bit mutation is used and to generate the new population 15x250 clock cycles are needed.

Simulations were performed using Gaussian noise distorted bitmaps. Gaussian noise is chosen independent of the image. Filtering operation can be done either with a frequency filter or with a spatial filter. Generally, a spatial filter is preferable as it is computationally cheaper than a frequency filter. Bitmap of IEEE test image Lena is used as the target image at different distortion levels for the generality of the EHW architecture. All results were compared with the filtered results from the Gaussian filter. The bitmap images contaminated by Gaussian noise with mean 0 and variance 0.01, 0.02, 0.05, 0.08 and 0.1 were used for the initial evolution. Figure 11 is the original Lena bitmap of 128x128. Figure 12 is the Lena bitmap distorted by Gaussian noise with mean 0 and variance 0.03. Figure 13 is the result from Gaussian filter. Figure 14 is the resulting image of the EHW filter.

The MDPP is 135602 for Gaussian filtered image and 127985 for evolvable hardware filtered image.

Figure 15 is the original Baboon bitmap image of size 128x128. Figure 16 is the image distorted by Gaussian noise with mean 0 and variance 0.008. Figure 17 and Figure 18 are the image filtered by Gaussian filter and the EHW filter respectively. The Mean Difference is 225353 and 215612 for Figure 17 and Figure 18 respectively.

Another image, Saturn, distorted by Gaussian noise of variance 0.007 is given as input to the evolved circuit and the output is verified. Figure 19 is the original Saturn bitmap image of size 128x128. Figure 20 is the image distorted by Gaussian noise. Figures 21 and 22 are the image filtered by Gaussian filter and the EHW filter respectively. The Mean Difference is 157912 for Gaussian filtered image and 100849 for evolvable hardware filtered image. The results are shown in Mean Difference, MSE in dB and PSNR in dB for different images with different levels of variance in Tables 2 to 4.

Table 2 Comparison of Mean Difference for Various Standard Test Images

Image	Variance	Gaussian Filter	EHW Filter
Lena	0.003	135602	127985
Test pattern	0.005	160621	139955
Chemical Plant	0.006	173449	143152
Saturn	0.007	157912	100849
Baboon	0.008	225353	215612
Chart	0.008	198807	100531
Man	0.009	203399	180606
Moon	0.01	217764	143384
Peppers	0.02	295040	237896

Table 3 Comparison of Mean Square Error (dB) for Various Standard Test Images

Image	Variance	Gaussian Filter	EHW Filter
Lena	0.003	20.46	20.36
Test pattern	0.005	21.90	21.21
Chemical Plant	0.006	22.44	21.08
Saturn	0.007	22.10	19.42
Baboon	0.008	24.79	24.64
Chart	0.008	24.18	23.55
Man	0.009	23.96	23.14
Moon	0.01	24.41	20.98
Peppers	0.02	27.07	26.08

Table 4 Comparison of PSNR (dB) for Various Standard Test Images

Image	Variance	Gaussian Filter	EHW Filter
Lena	0.003	27.67	27.77
Test pattern	0.005	26.23	26.92
Chemical Plant	0.006	25.69	27.05
Saturn	0.007	26.03	28.71
Baboon	0.008	23.34	23.49
Chart	0.008	23.96	24.58
Man	0.009	24.17	24.99
Moon	0.01	23.72	27.15
Peppers	0.02	21.06	22.05

11. CHIP IMPLEMENTATION RESULTS

The evolved filter is the result of the evolution of an array of 4x6 PEs with one PE at the output. The level back parameter is set as 2. Number of generations is 3000. The coding was done in VHDL and simulations were performed in ModelSim 6. The hardware evolution took 2 minutes on Xilinx Virtex FPGA xcv800 running at 49 MHz. This compares favorably with software simulations run previously which it took approximately 6 hours (Pentium III/800 MHz system) to achieve the best chromosome, the speed has been increased by 180 times and the evolution time has been greatly reduced. Number of generations was taken as 3000. Hardware evolution took 12 minutes in Xilinx VirtexE FPGA xcv2000e. This compares favorably with software simulations run previously which took approximately 6 hours to give the best chromosome.

The VRC takes 1754 slices of the Xilinx Virtex FPGA xcv800 (9408 slices) and the whole evolvable system including the GA takes 3204 slices. Since small amount of the resources are only used i.e only 34% of the resources, chromosomes can be operated in parallel and the processing time can be further reduced. The synthesis report obtained is given in Table-2 below.

Table 2 Synthesis Report - Device Utilization Summary
(Population Size = 16, Chromosome Length = 250)

Target information: Vendor: Xilinx Family: Virtex Device: v800fg680 Speed: -6 Optimization Goal: Speed		
Number of Slices	3204 out of 9408	34%
Number of Slice Flip Flops	1087 out of 18816	5%
Number of 4 input LUTs	6200 out of 18816	32%
Number of bonded IOBs	79 out of 516	15%
Number of BRAMs	8 out of 28	28%
Number of GCLKs	1 out of 4	25%
Minimum period	20.160ns	
Maximum Frequency	49.603MHz	
Minimum input arrival time before clock	27.706ns	
Maximum output required time after clock	6.887ns	

12. CONCLUSION

The work has presented a novel approach to digital image filter design based on the technique of evolvable hardware. FPGA model for the function level evolvable hardware is analyzed and associated with the evolutionary algorithms employed. The evolution time has been greatly reduced by implementing the evolutionary algorithm in hardware. The EHW architecture evolves filters without a priori information and out-performs conventional filter in terms of computational effort, filtered output signal and implementation cost.

13. REFERENCES

1. Vandenberg et al (1992), "Digital image processing techniques, fractal dimensionality and scale-space applied to surface roughness", Wear, 159, 17-30
2. Kiran et al (1998), "Evaluation of surface roughness by vision system", International J. Mach. Tools Manufact. Vol. 38, Nos 5-6, pp. 685-690.
3. Suresh et al (2002), "A Genetic algoritm approach for optimization of Surface roughness prediction model", The International Jnl. Of Machine Tools & Manufacture, Vol. 42, pp.675-680
4. Samhouri et al (2005), "Surface Roughness in Grinding: Off-line Identification with an Adaptive Neuro Fuzzy Inference system", Paper submitted to NAMRAC 33-2005 conference, May 24-27, Columbia.

14. BIOGRAPHIES

1) **M.Sumathi** is working as an associate Professor, ECE dept. In Adiamman Engg. College, Hosur, affiliated to the Anna university. Her areas of interest includes VLSI Signal processing, Evolvable Computing, Image processing and neural networks.

2**) R.S.D.Wahida Banu** completed her doctoral degree from Anna university in 1996. She is presently working as a professor, ECE dept. In Government college of Engg. Salem. Her areas of interest includes Computer networks, image processing, Networks and evolvable hardware.

Figure 11 Figure 12 Figure 13 Figure 14

Figure 11 represents Original Lena Image of size 128x128, Figure 12 Image Distorted by Gaussian Noise of Mean 0 and Variance 0.003, Figure 13 Image Filtered by Gaussian Filter, Figure 14 Image Filtered by EHW filter.

Figure 15 Figure 16 Figure 17 Figure 18

Figure 15 represents Original Baboon Image of size 128x128, Figure 16 Image Distorted by Gaussian Noise of Mean 0 and Variance 0.008, Figure 17 Image Filtered by Gaussian Filter, Figure 18 Image Filtered by EHW filter.

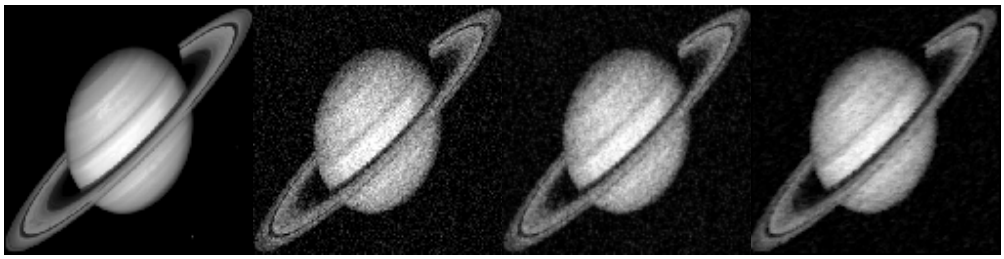

Figure 19 Figure 20 Figure 21 Figure 22

Figure 19 represents Original Saturn Image of size 128x128, Figure 20 Image Distorted by Gaussian Noise of Mean 0 and Variance 0.007, Figure 21 Image Filtered by Gaussian Filter, Figure 22 Image Filtered by EHW filter.