CHAPTER 2

EVLNVABLE HARDWARE AND GENETIC ALGORITHM

2.1 BASIC CONCEPT OF 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 2003). 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 elapsed. The basic concept of evolvable hardware is shown in Figure 2.1.

EHW has three advantages over traditional hardware and software systems. First, EHW can autonomously improve its performance by changing its hardware configuration according to the GA. Secondly, it processes information much faster than software systems. Thirdly, 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.
Based on how the hardware circuit is evolved, the evolution can be classified as (Higuchi 1997)

i) Gate level evolution and
ii) Function level evolution.

2.2 GATE-LEVEL EVOLUTION

In gate-level evolution, the hardware evolution is based on primitive gates such as AND-gates and OR-gates as shown in Figure 2.2. 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.
2.3 FUNCTION - LEVEL EVOLUTION

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

i) to synthesize more useful hardware functions and

ii) to design larger hardware circuits than those possible with gate-level evolution.

The function-level evolution is shown in Figure 2.3.

Figure 2.2 Gate-Level Evolution

Figure 2.3 Function-Level Evolution
2.4 EXTRINSIC, INTRINSIC AND COMPLETE HARDWARE EVOLUTION

The evolution of hardware devices can be either extrinsic or intrinsic or complete (Layzell 1998). Extrinsic EHW is the evolution of electronic circuits through simulation. It uses a software simulation of the underlying hardware to evaluate the fitness value of each individual; at the end of each generation the best individual is downloaded to the electronic device.

The second method, Intrinsic EHW, is when each genotype is assessed on the device by downloading the new configuration and testing the device directly. Intrinsic EHW, therefore, requires the hardware being reconfigured for each individual in every generation. This means that the hardware requires the ability to be reconfigured very quickly.

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.

2.5 GENETIC ALGORITHM

Genetic Algorithm (Goldberg 1989) determines how the hardware structure should be reconfigured whenever a new hardware structure is needed for a better performance. GA was proposed to model adaptation of natural and artificial systems through evolution, and is well known as one of
the most 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 strings of binary characters.

The sequence of operations performed by the GA is shown in Figure 2.4. At 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. According to the fitness values, the selection determines which chromosomes can survive into the next generation. The crossover chooses some pairs of chromosomes, and exchanges their sub-strings at random. Finally, the mutation randomly picks some positions in the chromosome and flips their values.

![Genetic Algorithm Flowchart](image)

**Figure 2.4 Genetic Algorithm Flowchart**

The major advantages of GA are its robustness and superior search performance in many type of problems without a priori knowledge. Indeed, an
image filter can be represented as a sequence of elementary operations which can easily be encoded to be handled by a genetic algorithm. However, the implementation of GA in software poses a time constraint for fitness evaluation. Hence, if the evaluation can be executed very quickly by the specific hardware device, the most serious problem of GA can be solved, and we can use GA more effectively. Thus, in this work, the implementation of GA in a FPGA chip by writing a VHDL program solves this time constraint since it consists of parallel hardware units. Such a migration helps in improving the convergence of the genetic algorithm towards a valid solution quickly.

2.6 IMPLEMENTATION OF GA PROCESSOR

The implementation of simple GA is composed of basic modules; pseudo random number generator, population memory, selection unit, mutation unit, fitness evaluator and output buffer.

2.6.1 Pseudo Random Number Generator (PRNG)

One of the most common PRNG for FPGA implementation is a Linear Feedback Shift Register (LFSR) and this technique is used in this work with a word size of twelve. 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 the twelve bit word the polynomial $x^{12} \text{ (XOR)} x^6 \text{ (XOR)} x^4 \text{ (XOR)} x^1$ was used. The block diagram of the LFSR used in this work is shown in Figure 2.5.
2.6.2 Input Buffer

Input buffer consists of RAM. Reference and noise corrupted images are read from the file and stored in the input buffer. During runtime pixels are given as input to the virtual reconfigurable circuit from this input buffer.

2.6.3 Initial population creation

A chromosome of required length is created using a 12 bit random number generator in multiple clock cycles and used as the initial chromosome. In hardware implementation, chromosomes are stored in the Block RAM of FPGA. The initial population size is chosen as 16.
2.6.4 Fitness calculation

Mean Difference per Pixel (MDPP) is used as the fitness function. 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.

2.6.5 Selection unit

The chromosome which has highest fitness is selected as the best chromosome and is retained for subsequent generations.

2.6.6 Mutation Unit

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

2.6.7 Output Buffer

After the specified number of generations the evolution is complete and the chromosome with the best fitness value is stored in the output buffer.

2.7 FPGA IMPLEMENTATION OF EHW

Recently, Field Programmable Gate Array (FPGA) technology has become a viable target for the implementation of algorithms suited to image processing applications. In this work, the EHW architecture is realized by including all the functional modules in a FPGA chip and this is the novelty of the proposed design. FPGAs represent reconfigurable computing technology
which is in some ways ideally suited for image processing. Reconfigurable computing elements are processors which can be programmed with a design, and then reprogrammed (or reconfigured) with virtually limitless designs as the designer’s needs change. FPGAs provide cost effective solution to quickly prototype a design.

The major advantages of FPGA based design are

i) Validating the design on the FPGA hardware minimizes the risk when entering into fabrication. FPGA hardware emulation enables development of software much ahead of implementation.

ii) Array of logic cells called Configurable Logic Blocks (CLB) connected via routing channels can be programmed. All of the logic in an FPGA can be rewired, or reconfigured, with a different design as often as the designer likes.

iii) Today, FPGAs can be developed to implement parallel design methodology. The gate densities from 40K to 10M Gates make this ideal for high-density designs.

iv) FPGAs contain multiplexers, Carry Logic, Arithmetic Logic Gates and two Storage Elements (Flip-Flop/Latch). Plenty of Fast 18 x 18 Multipliers (in some FPGAs) are available for image and signal processing applications.

viii) FPGAs with large amounts of 18k-Bit Select RAM all over the chip (Synchronous operation, configurable as Single or Dual ported, configurable in different sizes like 2kx9, 16kx1, 4kx4, etc) are available.

2.8 DETAILS OF HIGH SPEED IMAGE PROCESSING CARD

It was required that the card be able to connect to a high-
speed bus, such as the Peripheral Component Interconnect (PCI) bus. This is due to the fact that data must be sent from main PC memory to the FPGA for coprocessing, and back again once the coprocessing is complete. This transfer must be at high speed to ensure that the overall system performance is better than the software only implementation. It is desired that the FPGA card contain external RAM, to provide the coprocessing system local storage of image data. No minimum amount of RAM was set as a selection criterion, but obviously cards with greater amounts of on-board RAM are viewed more favourably. These criteria of cost, high-speed bus access and on-board RAM are the primary factors considered in the card selection process. An additional factor is that of software driver support. Preferences given to those cards that were sold with dedicated drivers included.

The image processing card is designed compatible for interfacing in the PCI bus and is a cost-effective platform for developing image and multimedia based applications. The card has an on board high speed ADC and DAC. The board supports real time processing of image signals. The on board SRAM and flash memories are used for storing data i.e. the configuration bits of the Virtual Reconfigurable Circuit(VRC). Each SRAM bits controls various multiplexers and switch blocks, providing quick and infinite reprogrammability. The aggregate of all the SRAM bits required to program an entire FPGA is called the configuration memory. The Genetic processor code is stored in the on board power PC. The advantage of the developed card is that it can be used as a standalone image processing board. The card has five fully independent banks of IM × 16 SRAM and three fully independent banks of 512 k × 16 flash PROM. The sampling rate is 30 MSPS. The prototype of the high speed card is shown in Figure 2.6.
The specifications of the hardware are as follows:

- Densities as high as 2,700 logic cells with up to 100,000 system gates.
- Unlimited re-programmability.
- User selectable RAM hierarchical memory
- 16 bits/LUT distributed RAM
- Configurable 40K bit block RAM
- Fast interfaces to external RAM
- Fully PCI compliant
- Low-power segmented routing architecture.
- Full read back ability for verification/operability.
• Dedicated carry logic (for high-speed arithmetic).
• Dedicated multiplier support.
• Cascade chain for wide-input functions.
• Abundant registers/latches with enable, set and reset.
• Four dedicated DLLs for advanced clock control.
• Four primary low-skew global clock distribution nets.
• IEEE 1149.1 compatible boundary scans logic.
• Versatile I/O and packaging.
• Compatibility to common packages.
• 16 high-performance interface standards.
• Hot swap compact PCI friendly.
• Zero hold time (simplifies system timing).

A complete description of the FPGA based hardware implementation of EHW is presented in chapter 3 of this thesis.

2.9 EXPERIMENTAL SET UP

The basic experimental set-up consists of a vision system (CCD Camera), PC with image processing hardware (discussed in section 2.8) and an appropriate lighting arrangement and is shown in Figure 2.7.
Illumination of the specimen was accomplished by a diffuse, white light source which is situated at an angle of approximately 45° incidence with respect to the specimen surface. The distance between the work-piece and camera is approximately 10 cm and is maintained constant throughout the experimental study. The specification of vision system is given in Table 2.1.
### Table 2.1 Specification of Vision system

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Features</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CCD Camera</td>
<td>TK-C1380U</td>
</tr>
<tr>
<td>2</td>
<td>Resolution</td>
<td>768(H) x 568(V)</td>
</tr>
<tr>
<td>3</td>
<td>Modes of Electronic shutter</td>
<td>9 Modes</td>
</tr>
<tr>
<td>4</td>
<td>AES range linear</td>
<td>1/60 through 1/10000 second</td>
</tr>
<tr>
<td>5</td>
<td>Magnification</td>
<td>4x to 100x</td>
</tr>
</tbody>
</table>