CHAPTER -5

RESULTS AND CONCLUSION

5.1 Real Time Execution Timing Analysis
5.2 FPGA Based Seeker Timing Analysis
5.3 Conclusion
5.4 Future Scope
5.1 Real Time Execution Results

This chapter deals with results obtained by implementation of algorithm on System on Chip (SoC) for object tracking.

The results include the following functions:

- To acquire the vehicles image displacement data.
- To estimate the movement of the vehicle.
- Development of VHDL code for achieving the above objectives.
- Synthesis on a selected family FPGAs.
- Implementation on selected family FPGA.

5.1.1 Overview of Design

Development and implementation of Projection curve analyser used for object tracking calculation of high speed (approx 100 Fr/Sec) Infrared Images. This SoC code is developed in VHDL. These 2-D Projective curve are useful for determination position information of desired objects in the images obtained from FPA/CCD Cameras. The processing frames of 512 x 512 are 256 gray level images. Processing SOC is done over 64 x 64 pixels segment of 512 x 512 pixel frames. The tools employed in the project for result verification are MATLAB 7 system generator and VHDL Xilinx ISE9.1 Platform. Actel tools are also Used for Image Enhancement part of research. The object tracking algorithm is defined for high speed IR objects and the solution conceptual methodology has been developed in MATLAB. This modeling in matlab enables us to define the problem clearly, efficiently and quick implementation in VHDL. Further, the time optimization of algorithms and effective programming techniques developed can be verified for large
The MATLAB model is then transformed to VHDL easily with recent tools. The realization is made on FPGA chip shown in Figure 5.1.1.

![Block diagram of object tracking control system.](image)

*Figure 5.1.1:* Block diagram of object tracking control system.

The potential application of velocity estimation is to find velocity high speed moving IR and Visual objects, which are in the field of view (FOV) range of 100 meters to 3000 meters away from the object tracking system. The technique is especially useful in tracking of military tanks in Anti Tank Guided Missile (ATGM) applications. The dynamic motion of the missile could be traced and controlled through the digital High speed closed loop feedback to maintain missile path on to the target. The complete real time system is shown in *Figure 5.1.2.*

### 5.1.2 Velocity Estimation using Image Processing

The principle of 2-D image histogram can be used to estimate the velocity of a moving object (military tanks). Velocity estimation plays an important role in automatic tracking of military tanks in defense applications. The identification of the target is based on SAD algorithm.
Figure 5.1.2 Real Time Range Image Capturing Setup

Figure 5.1.3 Before Firing Lock-On and After Firing Missile Tracking
One of the approaches used for velocity estimation and pattern recognition is by the analysis of the images of the moving targets by using image-processing techniques. Image processing involves processing and analyzing the images by various image-processing techniques depending upon the application. These images are generated by a CCD (Charge Coupled Device) camera.

Two consecutive image data frames are stored in the RAM processing. The processing involves pixel by pixel operation. The subtracted values are stored in the third frame. Then X, Y histograms are calculated for the resulting image. Then the moving average filtering (smoothing) is done over histograms. Finally to obtain the maxima "Find maxima index" algorithm has been developed. The difference between two maxims of successive frames x-histogram gives the $\Delta x$ and the difference between two maxims of successive frames y-histogram gives the $\Delta y$, which can be used to calculate the velocity of moving objects.

The difference in recording times of the two frames gives the time interval of the motion of object as

$$\Delta t = T_2 - T_1$$

Velocity of the objects in X-direction ($V_x$) = $dx/(t_2 - t_1)$

Velocity of the objects in Y-direction ($V_y$) = $dy/(t_2-t_1)$

Where $t_2-t_1 = \Delta t = 1/(\text{frame frequency})$.

where T is the time taken by the object to move from first position to second position, that can be calculated from the frame frequency (30 frames/sec), i.e.,

$$T = 1/\text{Frame frequency}$$

$\Delta x$ indicates the displacement moved horizontally with respect to the previous frame.
in $\Delta t$ time. $\Delta y$ indicates the displacement moved vertically with respect to the previous frame in $\Delta t$ time.

### 5.1.3 SOC Design Approaches

**IO Signals:**

- Digital data of two frames - Input
- Hx, Hy Indices - Output
- Velocity (16 bit) - Output

![Diagram](image)

**Figure 5.1.4** The Image Projection Curve Analysis System on Chip

**Figure 5.1.4** shows the basic inputs and outputs of the Image Projection Curve Analyzer System on chip that has been developed in this research. The frames are obtained from the CCD camera into an image grabber memory. There is proper synchronization between the camera and System. For each clock pulse, two consecutive frames are read into the chip for processing. The entire operation is carried on when the chip is enabled. After processing the position of the desired object, position values ($x, y$) are send to the output. These correspond to the position of the vertical and horizontal coordinates of IR object. The block diagram showing the basic operations is shown in **Figure 5.1.5** and the flow chart is shown in...
SoC ARCHITECTURE

The block diagram of the entire processing.

Figure 5.1.5  Functional Blocks diagram of Object Tracker SoC on FPGA
Figure 5.1.6 Functions of internal the data flow Blocks
Figure 5.1.6 above. The 2-D Image parallel image processing implementation contains mainly three important steps of processing to be made on the image:

- Obtaining horizontal and vertical histograms
- 2-point moving averaging of the two histograms
- Finding the maxima and their indices.

5.1.4 ARCHITECTURE OVERVIEW

This architecture is optimized and targeted Xilinx FPGAs to reduce the Object tracking processing time and SIMD parallel processing hardware SAD algorithm and H264 moving vector analysis. The design based on pipelined and SIMD parallel processing architecture. The logic targeted at XILINX FPGA families and hence more Vertex and Spartan family has been used instead of increasing system reliability by single chip solution. In this architecture are processing 8 pixels parallelism at a time i.e., difference calculation and finding X, Y Coordinates. 8 pixels of two consecutive frames are parallel processed. With this method, 16x128 read cycles are required to complete the operation on a 128x128 pixel frame.

Figure 5.1.7 Memory controller Block diagram
Every two consecutive frames of data is stored in the SRAM core. SRAM controller takes data from memory core and gives it to image_top module. Data path of SRAM controller is 128(8 pixels data of first frame) + 128(8 pixels data of second frame) bits. Total processing is divided into three **pipelined** stages.

### 5.1.5 Design & Implementation:

The different states involved in the processing is shown in figure 5.1.8. Reset signal initializes the contents of all the registers to zeros. First signal beside the arrow shows the transition criterion and the remaining shows the outputs of the state.

These stages explained more in the design and implementation.

1) Bringing data from SRAM core
2) Calculating hx, hy indices on 128x128 pixels of data.
3) Calculating velocity.

### 5.1.6 SRAM Controller:

This block reads the contents of SRAM as 16 bit data at a time and accumulates in a buffer up to 64 bits per frame. The data read from buffer is sent on 8 lines, each line containing one pixel data (8 bits). Two such reads are performed to send 8 pixel data of frame1 and 8 pixel data of frame2. The read data (2*64 bits) is given to the difference calculator block.

### 5.1.7 Projections Calculation

This block gets the 8 pixel data of two frames from SRAM controller and carries out parallel subtraction on these pixels by using absolute difference method and stored in
the Difference matrix. This data is used for parallel calculation of Hx, Hy. Hx is the array of elements determined by the sum of row elements. Hy is the array of elements.

Figure 5.1.8 Flow Chart of State machine
determined from the sum of column elements. Two point moving averaging is applied on the Hx and Hy arrays to determine Hx_avg and Hy_avg arrays.

Finding Hx, Hy, Hx_avg and Hy_avg are processed in parallel fashion to decrease the processing time for calculating the indices. For every 16 clock cycles one Hx element is calculated and stored in one of the two registers Hx_reg0 or Hx_reg1. Selection of Hx_reg0 or Hx_reg1 depends on whether the row processed is even or odd. Even_odd signal is used to indicate whether the row is even or odd. For every Hx element, Hx_avg value is calculated and compared with the maximum value of the Hx_avg array to process the maxima finding in parallel.

By the time processing of all the rows completes, maximum value of the Hx_avg array will be obtained in the Hx_max register. While processing every row of the differenced matrix, Hy array is updated by adding the Hy array elements with the corresponding indexed elements of the present processing row of difference matrix. While processing the last row of the difference matrix averaging and maxima finding of the Hy array will be processed such that Hy_max value is obtained by the time processing of frame is completed.

The index values corresponding to these Hx_max and Hy_max are used as the indices to estimate the direction and velocity of the object.

Implementation of averaging filter and calculation of maxima are processed concurrently to minimize the latency. Also the registers used in one cycle are used in the next clock cycle to minimize the hardware.
5.1.8 Velocity Controller:

Velocity controller receives the Hx, Hy indices from the Hx_Hy processing block and finds out the Velocity by using the Hx, Hy indices of previous frame and the time interval between these frames.

SoC design synthesized on recent Xilinx FPGAs. Behavioral and RTL architecture used for best fit of the code after synthesis on Xilinx’s Spartan-3, Vertex-4 and Vertex-5 Family FPGAs. The Total Algorithm implemented on the Chip. Percentage of resource utilization on selected FPGA results achieved on Xilinx’s ISE 9.1. Results are best matched with expected values. Highest possible number of frames processing per second are achieved.
## FPGA Based Seeker Timing Analysis Results

### Table 5.1: Design and Implementation Results Obtained with SoC Architecture

<table>
<thead>
<tr>
<th>Architecture Type</th>
<th>FPGA Family</th>
<th>(LUTs) Used</th>
<th>Latency (Cr. Path)</th>
<th>% Resource Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavioral</td>
<td>Spartan 3</td>
<td>Didn’t Fit</td>
<td>Didn’t Fit</td>
<td>Didn’t Fit</td>
</tr>
<tr>
<td></td>
<td>Vertex4</td>
<td>Didn’t fit</td>
<td>Didn’t fit</td>
<td>Didn’t fit</td>
</tr>
<tr>
<td></td>
<td>Vertex5 (xc5vlx50t-3-ffl136)</td>
<td>24833</td>
<td>580μs</td>
<td>86%</td>
</tr>
<tr>
<td>RTL</td>
<td>Spartan 3E  DSP (xc3sd1800a-4 cs484)</td>
<td>6050</td>
<td>60 μs</td>
<td>18%</td>
</tr>
<tr>
<td></td>
<td>Spartan 3E</td>
<td>6179</td>
<td>51.2μs</td>
<td>66%</td>
</tr>
<tr>
<td></td>
<td>Vertex-4    (xc4vlx15-12-sf363)</td>
<td>6064</td>
<td>29 μs</td>
<td>49%</td>
</tr>
<tr>
<td></td>
<td>Vertex5     (xc5vlx30-3-ff324)</td>
<td>4686</td>
<td>24 μs</td>
<td>24%</td>
</tr>
</tbody>
</table>

The goal of benchmarking is to compare the results for one FPGA family versus another over a variety of metrics. Since the FPGA industry does not conform to a standard benchmarking methodology. Benchmark results alone are not enough to answer the question of whether to use an FPGA in a new system design or which FPGA to choose. Designers need to understand how their choice of processing engine will affect development flow, implementation effort, and system design.
The complexity of today's designs together with the wealth of FPGA and computer-aided design (CAD) tool features available make benchmarking a difficult and expensive task. To obtain meaningful benchmarking results, a detailed understanding of the designs used in comparisons as well as an intimate knowledge of FPGA device features and CAD tools is required.
5.2.1 Simulation Results

Figure 5.2.1 Simulation Intermediate Results Input Null Vector data

Figure 5.2.2 Simulation Intermediate Results Input Vector data
Figure 5.2.3 Simulation Intermediate Results Enable data in Messages

Figure 5.2.4 Simulation Intermediate Results for Projection Hx Out
This section deals with Field Programmable Gate Array (FPGA) based hardware Implementation of Infrared Image (IRI) enhancement of thermo graphic images. The image enhancement capabilities and properties of the transform are analyzed. The transform is capable to perform both a nonlinear and a shape preserving stretch of the image histogram. FPGA Implemented results compared with Matlab Experiments and comparisons to histogram equalization are conducted.

Enhancing digital image to extract true image is a desired goal in several applications. Such transformation is known as image enhancement. Performing the task automatically without human intervention is particularly hard in image processing. Different approaches and techniques have been suggested to solve this problem. One well established method is the histogram equalization. Histogram equalization automatically flattens and stretches the dynamic range of the histogram of the image. Hence, an enhancement of the contrast in the image is achieved.

The Successive Mean Quantization Transform (SMQT) has properties that reveal the underlying structure in data. The transform performs an automatic structural breakdown of information. This can be interpreted as a progressive focus on details in an image. These characteristics make the transform interesting for automatic enhancement of any image. This paper deals with H/w implementation SMQT is applied for automatic image enhancement. An adjustment parameter is introduced to further control the enhancement. The image enhancement results are compared to histogram equalization.
5.2.3 DESCRIPTION OF THE SMQT

Let x be a pixel and the intensity of a pixel will be denoted \( V(x) \). The SMQT has only one parameter input, the level \( L \) (indirectly it will also have the number of pixels \( D \) as an important input. The output pixel set from the transform is denoted \( M(x) \). The transform of level \( L \) from \( D(x) \) to \( M(x) \) is denoted

\[
\text{SMQT}_L : D(x) \rightarrow M(x) \quad (5.2.1)
\]

The SMQT\(_L\) function can be described by a binary tree where the vertices are Mean Quantization Units (MQUs). A MQU consists of three steps, a mean calculation, a quantization and a split of the input set. The first step of the MQU finds the mean value of the pixels, denoted \( V(x) \), second, mean quantization of pixel set. The third step splits the input set into two subsets

\[
D_0(x) = \{ x \mid V(x) \leq V(x), \forall x \in D \} \quad (5.2.2)
\]

\[
D_1(x) = \{ x \mid V(x) > V(x), \forall x \in D \} \quad (5.2.3)
\]

where \( D_0(x) \) propagates left and \( D_1(x) \) right in the binary tree, see Figure 5.2.5. \( U(x) \) can be interpreted as the structure of \( D(x) \).
The first level transform, $\text{SMQT}_1$, is based on the output from a single MQU, where $U$ is the output set at the root node. The outputs in the binary tree need extended notation. Let the output set from one MQU in the tree be denoted $U(l,n)$ where $l = 1, 2, ..., L$ is the current level and $n = 1, 2, ..., 2^{(L-1)}$ is the output number for the MQU at level $l$. Weighting of the values of the pixels in the $U(l,n)$ sets are performed and the final $\text{SMQT}_L$ is found by adding the results. The weighting is performed by $2^{L-1}$ at each level $L$.

Today digital imaging devices typically use the range 0 ... 255, that is 8 bits is used. For automatic image enhancement of 8 bits images $L$ is chosen to 8. Nevertheless, it could be convenient to control the amount of enhancement applied. Given the original pixel set $D(x)$ and the $\text{SMQT}_8$ enhanced pixel set $M(x)$. 

\textit{Figure 5.2.5}. 2 out of 8 level operation of one Mean Quantization Unit (MQU).
5.2.4 INFRARED IMAGE ENHANCEMENT

A straightforward way to enhance an image is to use the SMQT directly. The only parameter to adjust is the level.

![Fig. 5.2.6. (Left) Original image. (Right) SMQT$_g$ enhanced image.](image)

In the original image histogram it is possible to see that this image does not take advantage of the full dynamic range.

![Fig. 5.2.7. (UP) Original image histogram. (DOWN) SMQT$_g$ enhanced image histogram. Intensity vs Percentage of Gray Values](image)
A) **Design Assumptions**

The design is based on the following assumptions

1. Image enhancement using successive mean quantization transform (SMQT) core implementation on FPGA.

2. The Image data assumed that image available in SRAM memory (FPGA – Block RAM) and image size is 64 x 64.

3. The FPGA- Block RAM image data loaded by external processor and it generates start command to SMQT core.

4. The enhanced image data should be available in memory and accessed by external Co-processor.

5. External clock frequency 50 MHz

6. The enhancement core implementation in two ways
   - Using Division (restoring) algorithm
   - Using multiplier logic.
5.2.6 SMQT CORE WITH CO-PROCESSOR

Fig: 5.2.9 Interface of SMQT core with Co-processor

5.2.7 REALIZATION OF HARD CORE SMQT

The image data come from external world to FPGA and resides in Block memory of FPGA. The memory data is taken and calculated mean for each iteration. This SMQT algorithm has 8 levels for gray scale image. In each level it has 2 power (L-1) iterations and for each iteration mean is calculated and quantized. The Fig 1.2 shows realization of SMQT core inside FPGA. The input image memory of FPGA (4096x8) image data loaded by external microprocessor. Here, image data considered as size of 64 x 64. Once image data loaded into memory then FPGA waits for start enhancement command from microprocessor. In FPGA, temporary memory contents are also filled with zeros during initialization process. Whenever microprocessor generates start command to FPGA then mean calculation module reads temp memory contents as
well as input image memory contents and compares both pixels and populates two sets for each iteration. In each iteration mean is calculated and registered in FPGA. Once mean is calculated then quantization for corresponding set pixels are updated in temp memory. Here no. of iterations depends on processing level. In the level1 only one iteration and similarly for level8 total 128 iterations takes place. The temporary memory contents are updated for each iteration based on mean value. In each level iterations completes then temporary memory contents are copied into quantization memory.

5.2.8 Mean calculation module:

The mean is calculated from input image memory and temporary memory contents. The mean calculation realization in hardware.
5.2.9 Synthesis and Performance Estimation

The SMQT core implemented using VHDL and synthesized for ACTEL FPGA - APA 600. The core includes BRAM's for storing as well as quantization of Image data.

**Core Cells**: 2123 of 21504 (10%)

**IO Cells**: 34 of 454 (7%)

**Block Rams**: 48 of 56 (86%)

Mean calculation module

Mean calculation with multiplication for further processing

Start processing of SMQT core

Image write into FPGA by co-processor
FPGA implementation of the SMQT has been applied and analyzed for automatic enhancement of infrared images. Properties of the SMQT on images by means of histogram change have been investigated. The SMQT was found to retain the basic shape of the histogram and performs a nonlinear stretch. Hence, the SMQT is found to perform a balanced and natural enhancement of images. A comparison with histogram equalization has been performed, which showed the advantage of the SMQT based enhancement.
Figure 5.2.13 Mean calculation module
Figure 5.2.14 Mean calculation with multiplication for further processing
Figure 5.2.15 Start processing of SMQT core
Figure 5.2.16 Simulation Results (Cont.)
Figure 5.2.17 Image write into FPGA by co-processor
5.2.10 OBJECT TRACKING HARDWARE TEST SETUP

In this object tracking system the test input to the FPGA is provided through DVD player which produce 60 fields/sec (i.e NTSC). The video is processed for Infrared object acquisition, processed and extraction and displayed on the screen at 50 frames/sec on VGA monitor. The current unprocessed video is displayed on the screen with svideo out from the DVD player.

Finally tracked coordinate information is displayed on screen.

Processing features

1. Recognition of IR Object
2. Eliminate the spurious objects
3. Calculating the Centroid
4. Real Time Tracking Data Plot
5. Display on the screen or store.

Figure 5.2.18 Missile tracking Test setup system

Real Time system is designed based on the performance requirements. Components of system i.e camera, FGPA and SDR/DDR are chosen accordingly.
5.2.11 Processing parameters:

1. Camera frame rate (up to 150 fps)
2. Video processing on Xilinx Vertex II Mil-Version.
3. Display refresh rate.

Figure 5.2.19 Real time target tracking embedded system (Test Setup)

5.2.12 High performance at low price

With the addition of the new Spartan-DSP series, the XtremeDSP Portfolio delivers 20 GMACS for under $30. The Portfolio fills the performance gap created by the growth in algorithmic complexity and limitations of traditional sequential processors in signal/image processing applications.

Algorithm Complexity: As demand for processing power is rapidly increasing, sequential processing cannot support the algorithmic complexities within the required response times; to overcome these architectural limitations, parallel processing offered by XtremeDSP devices is essential.
5.3 Conclusions

Infrared Object Tracking System on Chip (SoC) algorithm is developed and implemented on recent high end FPGAs. Code written in Behavioral and RTL architecture used for best performance and to fit on Xilinx’s Spartan-3, Vertex-4 and Vertex-5 family FPGAs. The complete algorithm implemented on single FPGA and proper percentage of resource utilization. Results are achieved with Xilinx’s ISE 9.1 tools. Results are compared with MatLab program Output. Highest possible number of frames processing per second in much better than the expedited before.

Infrared object tracking and image enhancement algorithms are independently tested. Maximum no of frames processing for Object Tracking is 4000 Frames/ sec is achieved with 25 percent of resource utilization Vertex-5 xc5vlx30-3-ft324. Maximum no of frames processing for Object Tracking is 1500 Frames/ sec is achieved with 18 percent of resource utilization Vertex-5 xc5vlx30-3-ft324. Remaining 88% may be used for improvement of algorithms.
Computer vision algorithms are getting more and more important. The realization of such computer vision algorithms needs enormous computing power of the hardware. Cutting edge DSPs provide enough performance for computer vision algorithms if optimized algorithms are used. Highly optimized code with the use of intrinsics saves runtime and helps them to meet real time constraints.

Porting the algorithm on an embedded system brings up questions like which hardware fits the needs of the application best, how fast can the algorithm be processed and how many resources are required. Selecting an embedded hardware has big influence on the performance. FPGA benchmarks that have been completed to provide a frame of reference when evaluating the benefits of new FPGA technology.
Virtex-5 FPGAs are optimized to execute faster and more efficiently using the single-cycle timed loop structure in FPGA. The fundamental building blocks for implementing digital logic inside FPGA chips are called slices, and each slice is composed of flip-flops and look-up tables (LUTs).

Previous-generation Virtex-II FPGAs use 4-input LUTs for up to 16 combinations of digital logic values. The new Virtex-5 FPGAs use 6-input LUTs for up to 64 combinations, increasing the amount of logic that can implement per slice. The single-cycle timed loop structure in FPGA takes advantage of six-input LUTs for substantially improved resource utilization. This means optimize more FPGA code to fit within Virtex-5 FPGAs and perform more operations per clock cycle. In addition, the slices themselves are placed in closer proximity to each other to reduce the propagation delay of electrons and increase overall execution rates.
References


algorithm for nonuniformity correction in focal-plane arrays," Applied Optics-

Descour, "Adaptive nonuniformity correction for in focal plano arrays using

Descour, "Adaptive retina-like preprocessing for imaging detector
arrays," Proceedings of the IEEE International Conference on Neural


[22] M. Cohen and G. Cauwenberghs, "Floating-gate adaptation for focal-plane on­
line nonuniformity correction," IEEE Transactions on Circuits and Systems

"Nonuniformity correction for improved registraron and high-resolution image
reconstruction in ir imagery," Proceedings of SPIE, vol. 3808, pp. 150-161,
1999.

nonuniformity correction with video sequences and registraron," Applied


