# Configurable Computing for Real-time Data: Performance Analysis

**By: Ali Zaidi**

**Supervisor: Dr. Oskar Mencer**

## Department of Computing
## Imperial College

# Abstract

Computation of real-time data can be accomplished by generic computational hardware such as microprocessors, micro controllers or DSPs. This solution, however, has certain limitations with regard to speed of data stream and quality of processing owing to abstract nature of employed hardware platforms. The alternative solution for performing high quality computation on a high speed data stream involves the introduction of reconfigurable devices such as FPGAs into the computing system either as stand alone components or as accelerators for more generic computing devices. This ISO will execute a comparative performance analysis on a computationally intensive algorithm for real-time processing of high speed data streams implemented using generic and custom hardware platforms. The results obtained for each computation would then be analysed in detail. Necessary caution will be exercised in the selection of the test algorithm so that the results obtained are generic and thus applicable for similar scenarios irrespective of the implementation and application details.

# Acknowledgements

This work would not have been possible without the unrelenting support and encouragement by Dr. Oskar Mencer. I would also like to acknowledge the patient support and sound advise provided by Carlos Tavarez.

# Table of Contents

# Report Organization

## Chapter 1- Introduction

This chapter lays the conceptual and theoretical foundation of the rest of the work. It starts by discussing the various computational platforms for the implementation of algorithms, their strengths and weaknesses. This discussion is followed by an overview of the parameters which effect the performance of 'Configurable Devices'.

## Chapter 2- Drowsy Driver problem

This chapter starts by giving a statistical overview of the 'Drowsy Driver' problem. It then goes on to discuss the various methods which can be used to detect driver drowsiness. Finally, it attempts to classify and provide a brief account of the research which has been conducted to detect driver drowsiness by using computer vision techniques.

## Chapter 3 - Design and Development

This chapter starts by listing the design goals which should be satisfied by a driver drowsiness detection system of acceptable quality. It then goes on to discuss the overall system architecture of the driver drowsiness system presented in this report. Finally, it discusses at length the various algorithmic steps and their respective implementation issues for software and hardware.

## Chapter 4 - Analysis of Results

This chapter starts by presenting the stepwise results obtained for the software and hardware implementation of the algorithm in order to verify the correctness of the algorithm as well as the implementations. The result comprise images captured during the various stages of the algorithm execution. These images have been subjected to post processing, in order to illustrate the working of the algorithm. Following the results verification phase, a comparative performance analysis is performed to highlight the computational cost of respective algorithmic steps as well as the performance of the various platforms.

## Chapter 5- Conclusion

This chapter starts by recommending the directions deemed feasible for future work. It then lists the various limitations of the driver drowsiness detection system presented in this report. Finally, it provides the conclusion, summarizing the work presented in this report and the results obtained.

# Chapter 1

# Introduction

The primary methods employed in contemporary computing applications for the implementation and/or acceleration of algorithms can be classified into three categories:

1. Implementations using 'Hard-Wired' devices
2. Implementations using 'Programmable' microprocessors
3. Implementations using 'Reconfigurable' devices

## 1.1 Implementations using 'Hard-Wired' devices

The first method employs 'Hard-Wired' devices for the implementation of algorithms. These devices include Application Specific Integrated Circuits (ASICs), configured to perform a certain computation, or several off-the-shelf, Large scale integration (LSI) or Medium Scale Integration (MSI) devices, connected on a Printed Circuit Board (PCB) to perform the required operations. Of these two, ASICs delivers better performance. This is largely due to the fact that the signal propagation delays are significantly smaller for on-chip connections as compared to PCB connections. However, the ASIC is prone to incur higher Non-Recurring Engineering (NRE) costs. Both of these solutions have a slight disadvantage when it comes to flexibility. In case of any design changes or upgrades, the cost for replacing the ASIC or redesigning the PCB is a significant factor which weighs against these solutions.

## 1.2 Implementations using programmable devices

The second method for the execution of algorithms is to use 'Programmable' microprocessors. The immediate advantage is that of flexibility. Microprocessors execute a set of instructions to perform the requisite computations. Any change in functionality of this computation can be easily incorporated by changing the set of instructions by reprogramming the device. However, there is a certain price to pay for this flexibility in terms of device performance and throughput. This is due to the fact that the very nature of the microprocessor allows for a sequential execution of tasks. The microprocessor must read each instruction from the memory, decode it, and only then executes it. The execution phase contributes towards the overall computational goal, while the fetch and decode cycles for every instruction can be termed as execution overheads. Additionally, the instruction set of a microprocessor is determined during the architectural design phase. Therefore, any attempt towards implementing an algorithm is limited by the instruction set for the particular microprocessor, although the control for the flow of instructions to be executed rests with the programmer.

# 1.3 Implementations using Reconfigurable devices

The diverse advantages and disadvantages of the implementation of computational algorithms using hard-wired and programmable solutions have resulted in a technology gap between the two domains. The last decade have witnessed the emergence of 'Reconfigurable' devices, which promise to bridge the gap between the traditional solutions by matching the performance of ASICs and being as flexible as microprocessors. Although several devices are included in the family of reconfigurable devices, however, Field Programmable Gate Arrays (FPGAs) are claimed by many to be the flag bearers of the potential of reconfigurable logic devices and generally assert the technological trends which shape the industry.

Reconfigurable devices, comprises of generic arrays of computational elements termed as 'Logic Blocks' whose functionality is determined by a configuration bit stream. The logic blocks are connected by sets of routing resources. These routing resources are implemented by on-chip 'Crossbar Switch Matrices' which, in turn are also configurable. Computational algorithms can be realized by dividing them into logic components which can be mapped directly on to the logic blocks. Subsequently, the logic blocks contributing towards the computational algorithm are connected by the configurable crossbar switch matrices to form the necessary circuit.

It can be stated that the implementation of computational algorithms in configurable devices is accomplished in 'Space Domain'. On the other hand, the implementation of computational algorithms in microprocessors are executed in 'Time Domain'. Unlike the microprocessors which broadcast instructions to the functional units on every clock cycle, instructions in reconfigurable devices are locally configured, allowing the reconfigurable device to compress instruction stream distribution and thus deliver more instructions into active silicon on each cycle. Reconfigurable devices provide a large number of separately programmable, relatively small computational units allowing for the execution of a greater range of computations per unit time. Resources such as memories, crossbar switch matrices and logic blocks are distributed rather than being centralized in large pools. Independent, local access allows for the utilization of on-chip resources in parallel, thus avoiding performance bottlenecks resulting from a large, central resource pool.

FPGAs and reconfigurable computing have been shown to accelerate a variety of applications. The paper at reference [1] lists the speed-ups achieved for implementing various algorithms in FPGA. By careful analysis of these works, we can categorize the circumstances in which the traditional microprocessors or the contemporary reconfigurable devices are preferable. For example, when the function and data granularity to be computed are well-understood and fixed, and when the function can be economically implemented in space, dedicated hardware provides the most computational capacity per unit area to the application. On the other hand, if we are limited spatially and the function to be computed has a high operation and data diversity, we are forced into reusing limited active space and accepting limited instruction and data bandwidth. In this case, conventional microprocessors are most effective since they dedicate considerable space to on-chip instruction storage in order to minimize off-chip instruction traffic while executing descriptively complex tasks.

# 1.4 Performance Factors in Reconfigurable Devices

The widespread popularity of microprocessors have resulted in large scale production, decreasing the cost per chip to record levels. Reconfigurable devices on the other hand are relatively expensive. However, the massive computational demands posed by contemporary applications such as Digital Signal Processing (DSP) and High Performance Computing (HPC) have resulted in a renewed interest in reconfigurable devices. This interest has resulted in a redoubling of efforts towards asserting the performance benefits offered by reconfigurable devices.

The performance of reconfigurable devices is a complex phenomenon and depends upon several parameters. Careful study has shown that the factors which effect the performance of a configurable devices include the following:

1. I/O Bandwidth
2. Logic Fabric Performance
3. Block RAM Performance
4. DSP Blocks Performance
5. Embedded Cores Performance

## I/O Bandwidth

In many cases, I/O bandwidth represents the most critical system bottleneck in FPGA-based systems so much so that it is substantially more difficult to achieve high data transfer rates in and out of the FPGA than it is to sustain internal processing performance. I/O bandwidth as implemented with parallel Low Voltage Differential Signalling (LVDS) chip-to-chip interfaces, external single-ended memory interfaces, or even serial multi-gigabit interfaces for backplanes, often becomes the limiting factor in the quest for performance.

## Logic Fabric Performance

The logic fabric for most FPGAs has been based on the fundamental 4-input Look-Up Table (LUT) architecture. Today's high performance 65 nm FPGAs, such as the Xilinx Virtex-5 family, offers a 6-input LUT based fabric. Moving to a 6-input LUT architecture provides the most effective trade-off between critical path delay, the determining factor for logic fabric performance and utilized die area.

## Block RAM Performance

The performance of on-chip memory (LUT-based memory, block RAM, or FIFO) is also critical to achieving higher system performance because it is used extensively to store data between algorithmic processes. Choosing the right memory hierarchy and fully utilizing the on-chip memory can greatly improve system performance. For example, distributed LUT RAM is best suited for smaller sizes (less than 4 Kb) and fast clock-to-data outputs, while block RAM can accommodate larger buffers at frequencies of up to 550 MHz.

## DSP Blocks Performance

Many image, signal, and data processing applications require dedicated logic with increased parallelism capable of implementing arithmetic algorithms at high throughput. Some of the latest FPGA offerings enable the designer to configure the DSP slices to implement multipliers, counters, multiply-accumulators, adders and many more functions, all without consuming logic fabric resources. To keep pace with the seemingly insatiable demand for more DSP performance, the DSP block clock rate has increased to 550 MHz and the precision have improved to 25 × 18 bits.

## Embedded Cores Performance

Reconfigurable devices are better suited to perform numerically intensive computations, while traditional microprocessors are better at executing control flow computations. Contemporary real-time systems requires high level of performance in both domains. Some contemporary high-end FPGAs provide up to two hard IP processor cores, each delivering more than 700 DMIPS performance with minimal power consumption. Using these hard IP core, it is now possible to design hybrid systems, with the computational power of reconfigurable devices and the control flow flexibility of a microprocessor within the confines of a chip.

Determining the performance parameters of reconfigurable devices is a complex task which is made increasingly involved by the varied nature of the parameters quoted by the manufacturers of these, devices. These parameters are often manufacturer, device family or device dependant and thus are not very helpful in performing comparative analysis on two or more devices. For example, the parameter termed as Gate Count represents a simplistic approach toward determining the logic resources available on a chip. However, Altera have stopped using this parameter opting to employ a different measure for the same parameter which is termed as “Logic Density”. On the other hand, Xilinx have opted to use a term “System Gates” essentially for conveying the same information. The extent of disagreement between major chip manufactures even extends to the definition of a gate. According to Altera,a gate is a 2 input AND, while Xilinx prefers the 2 input NAND. This results in significant differences in implementing any given design. For example a 4-input XOR gate, which is unanimously accepted as a complex gate is equivalent to 13 gates according to Altera and only 9 gates according to Xilinx.

In order to perform a credible analysis of device capabilities, one cannot rely upon these parameters. As a result, increasing number of researchers have chosen to follow an experimental path rather than an empirical one. The current trend in performance profiling of reconfigurable devices, therefore involves the implementation of a selected algorithm, followed by the accumulation and analysis of results to determine the performance parameters. The algorithm selected for implementation, understandably, should be generic enough so that these results can be applied to predict device performance for similar applications.

After careful consideration, the 'Driver Drowsiness Detection' problem was selected as the test algorithm. It is a perfect embodiment of a modern 'real-world', yet unsolved problem in the domain of computer vision which presents the usual challenges associated with high quality, real-time processing of high speed data stream, which in this case is the video feed from the on-board cameras. Because, it is an area of active research, it presents an intellectually inspiring and challenging situation to deal with.

# Chapter 2

# Drowsy Driver Problem

## 2.1 Statistical Overview

The increasing number of traffic accidents due to a driver's reduced vigilance level has become a problem of considerable scale. According to the paper at reference [2]:

- In Europe, statistics show that between 10% to 20% of all traffic accidents are due to drivers with a reduced vigilance level caused by fatigue or drowsiness
- In Heavy Vehicle Industry, about 60% of fatal truck accidents are related to driver fatigue
- According to the U.S. National Highway Traffic Safety Administration (NHTSA), falling asleep while driving is responsible for at least 100 000 automobile crashes annually
- An annual average of roughly 40 000 non-fatal injuries and 1550 fatalities result from these crashes

These figures only cover crashes happening between midnight and 6 AM, involving a single vehicle and a sober driver travelling alone, including the car departing from the roadway without any attempt to avoid the crash therefore, it can be stated that these figures are optimistic. According to the paper at reference [2], these statistics show some interesting trends. For example:

- GES statistics from 1992 indicate that drowsy driver crashes peak between midnight and dawn, with a second smaller peak in the afternoon
- Most of the crashes occur in non-urban areas, generally on roadways with 55-65 mph speed limits
- Eighty-four percent are single-vehicle roadway departure crashes or collisions with parked vehicles
- In 83 % of known cases, the crash occurs on a straight section of roadway with the pre-crash manoeuvre of "going straight" accounting for 85 % of reported incidents
- In 78% of crashes the driver is the only occupant of the vehicle, and typically the driver makes no corrective action (i.e., braking or steering) to avoid the collision

# 2.2 Classification of Techniques

In the past few years, significant research endeavours have been carried out to develop a safety system addressing the drowsy driver problem. These efforts can be coarsely classified into three categories:

1. Detection by Vehicle Controls
2. Detection by Physiological Measures
3. Detection by Visual Cues

## Detection by Vehicle Controls

A driver's state of vigilance can be characterized by indirect means such as monitoring the behaviour of vehicle like lateral position, steering and wheel movements, and time-to-line crossing. Although these techniques are not intrusive, they are subject to several limitations such as vehicle type, driver experience, geometric characteristics, condition of the road, etc. In addition, these methods have a poor response time and therefore do not work with the so called micro-sleeps i.e. when a drowsy driver falls asleep for a few seconds on a very straight road section without changing the lateral position of the vehicle. Following is list of systems which employ this technique to detect driver drowsiness [2]:

1. Toyota uses steering wheel sensors (steering wheel variability) and a pulse sensor to record the heart rate [3].

2. Mitsubishi has reported the use of steering wheel sensors and measures of vehicle behaviour (such as the lateral position of the car) to detect the driver's drowsiness in their ASV system [3].

## Detection by Physiological Measures

The techniques based upon monitoring the physiological parameter of the driver are generally accepted to be most accurate. These physiological measures include brain waves, heart rate, pulse rate and respiration which are potentially indicative of human vigilance levels. However, these techniques are termed as 'intrusive' since they require electrodes to be attached to the drivers, causing discomfort and in severe cases distraction. Notable efforts in this domain include:

1. MIT Smart Car, employs several sensors (electrocardiogram, electromyogram, respiratory, and skin conductance) embedded in the car while visual information is used for confirmation and validation of the results from the sensors [4].

2. Advanced safety vehicle (ASV) project conducted by Toyota uses a wristband for measuring heart rate of the driver. Others techniques monitor eyes and gaze movements using a helmet or special contact lenses [3].

Owing to the intrusive nature of these techniques, they have not been able to gain widespread acceptance.

## Detection by Visual Cues:

People experiencing fatigue and drowsiness show observable and thus detectable visual behaviours. Typical visual characteristics observable from the images of a person with a reduced alertness level include a longer blink duration, slow eyelid movement, smaller degree of eye opening (or even closed), frequent nodding, yawning, gaze (narrowness in the line of sight), sluggish facial expression, and drooping posture [11]. Computer vision can be a natural and non-intrusive technique for extracting visual characteristics that typically characterize a driver's vigilance from the images taken by a camera placed in front of the user.

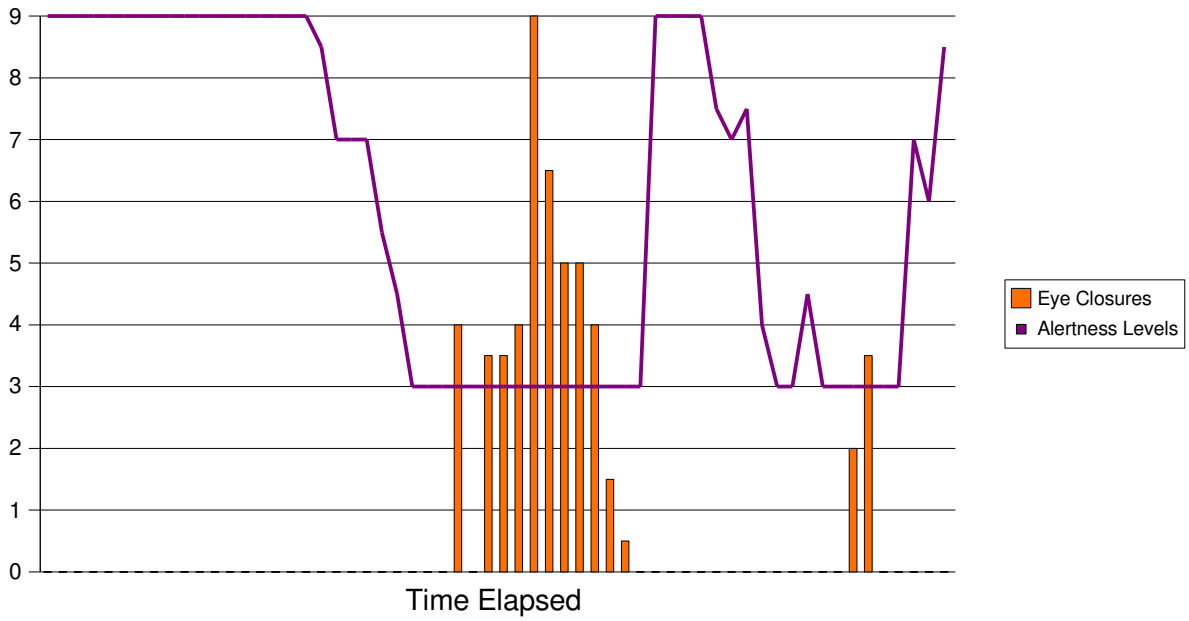


**Figure-1: The relation between the Alertness level and Eye Closures [11].**

The research undertaken in this particular direction can be categorized by the number of cameras employed for capturing the requisite visual information, and includes:

### Two Camera Systems:

Some notable efforts in this domain are following:

1. The method presented in [5] estimates the head pose and gaze direction. It relies on a two-dimensional (2-D) template searching and then a three-dimensional (3-D) stereo matching of the facial features. A 3-D model is then fit and minimized using virtual springs, instead of the least- squares fit approach for determining the head pose.

2. In [6], a method is presented based on a stereo template matching system to determine some specific facial features. A least-squares optimization is done to determine the exact pose of the head. Two eye trackers calculate the eye-gaze vector for each eye; these vectors are combined with the head pose to determine the gaze direction.

A vast majority of systems employing two cameras and some form of 3D template matching, including the systems discussed above, rely on a manual initialization of the feature points. The systems appear to be robust, but manual initialization is a limitation which hampers the popularity of such systems.

**Single Camera Systems**

The systems employing a single camera generally utilize domain knowledge to detect the symptoms of drowsiness in drivers. Some notable works in this domain are following:

1. Another successful head/eye monitoring and tracking system that can detect the drowsiness of drivers using one camera and based on colour predicates is presented in [7]. However, this system is based on passive vision techniques and its functionality can be problematical in poor or very bright lighting conditions. Moreover, it does not work at night, when the monitoring is more important.

2. In [8], a system with active infra-red LED illumination and a camera is implemented. Because of the LED illumination, the method can easily find the eyes and based on them, the system locates the rest of the facial features. The authors in [8] propose to analytically estimate the local gaze direction based on the pupil location. They calculate the eyelid movement and face orientation to estimate driver fatigue.

Most of the systems presented so far have only been tested in labs. Owing to the safety critical nature of the driver drowsiness detection system, it is necessary for such systems to be subjected to rigorous testing in realistic scenarios. A moving vehicle presents several new challenges like variable lighting, changing background, and vibrations which cannot be simulated in the lab.

The ambitious European project, named system for effective assessment of driver vigilance and warning according to traffic risk estimation (AWAKE) [9] was developed recently by a consortium including two major car manufacturers (Fiat, Daimler Chrysler), four automotive system developers [Siemens, Association de Coordination Technique pour L'Industrie Agro-Alimentaire (ACTIA), Navigation Technologies (NAVTECH), and Autoliv] and many research institutes and universities.
This system follows a multi-sensor approach, and requires initialization of parameters with respect to the driver, the vehicle, and the environment. This system accumulates data from a multitude of sources including on-board driver monitoring sensors (such as an eyelid camera and a steering grip sensor) as well as driver behaviour data (i.e., from the lane tracking sensor, gas/brake, and steering wheel positioning) and integrates this information via an artificial intelligent algorithm. This system unlike the ones mentioned earlier, has undergone exhaustive pilot testing to determine its functional performance and the user acceptance. However, the results concluded that this system cannot be used outside the well-structured highway scenario. It was recommended that in further research is necessary to declare this system commercially viable.

Our efforts stem from the work conducted by the paper at reference [10] employing a single camera, and domain knowledge to detect driver drowsiness. In an attempt to optimize the implementation of this algorithm for software and hardware, several digressions were successfully attempted, resulting in some differences in the software and hardware implementations. However, the crux of the algorithm remains intact and all the adulterations without exception have been attempted to achieve the optimal implementation for entirely different platforms and design procedures.

# Chapter 3

# Design and Development

## 3.1 System Requirements

Driver drowsiness system can be termed as a 'Safety Critical System', i.e. its success or failure determines the safety and well being of one or more human beings. Therefore, the design of such a system must assume a certain level of responsibility and care. In order to determine the specifications of the systems at design time, it is necessary to identify the requirements posed at such a system.

### Non Intrusive

Driver drowsiness detection system should ideally be non-intrusive in nature. A non-intrusive system can be defined as not having any instrumental probes or attachments to the body of the driver which can potentially cause irritation and distraction and therefore cannot be employed for long time periods.

### Real-time

Driver Drowsiness detection system should be able to respond in real-time in order to ensure accuracy in detecting drowsiness. Real-time behaviour of a driver drowsiness detection system is not an objective issue with a definite answer. It varies from driver to driver and also depends upon the state of vigilance, traffic density and speed limits, environmental conditions and many other factors which cannot be determined at the time of design or controlled at the time of execution. However, since the system is safety critical, every effort should be spared to induce every millisecond of performance out of this system, which in turn can result in saving human lives.

### Robust

In order to be practically viable, a Driver Drowsiness detection system should be robust. A robust system works in most acceptable environmental conditions and is required to provide an indication to the user about the level of performance which can be relied upon under the given environmental conditions. Generally, a driver drowsiness detection system should work in both daytime and night time conditions. It should be able to adapt itself to sharp changes in the light intensity owing to vehicles coming form the opposite direction, which is a common observation on vast stretches of rural highways as well as street lights which are common in more urban environments.

# 3.2 System Architecture

The system architecture for this comparative study is illustrated in Figure-2. Hybrid implementation and DSP implementation are not included in the scope of this study, however, they are illustrated in the figure to show the possibilities for future work.

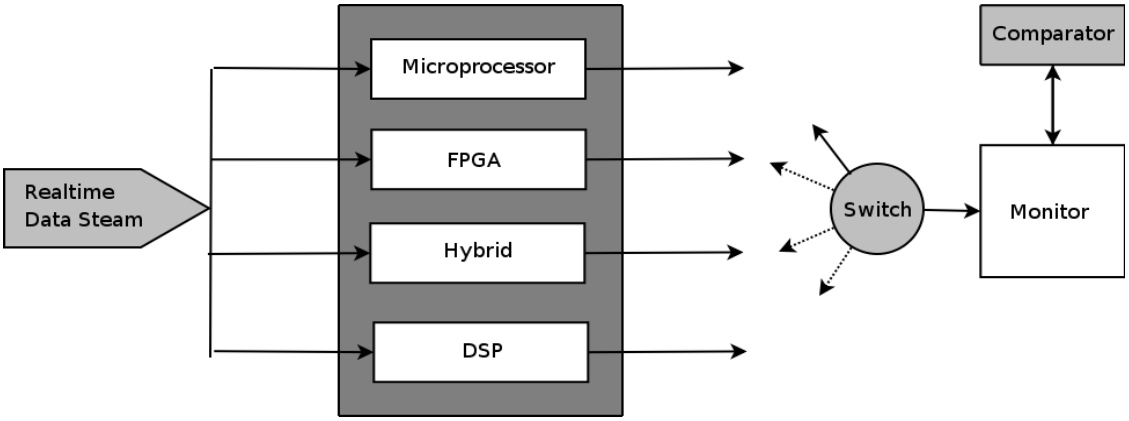


**Figure-2: System for comparative performance analysis of various platforms**

Following is a brief description of the various components of the system.

## Real-time data stream

The real-time data stream in the figure is suitably represented by a digital camera, which feeds visual data to the computing platform at a rate configurable within the confines of the device and interfacing technology. The camera employed for the lab testing of driver drowsiness system is Logitech QuickCam Pro 4000, which is capable of operating at 20 frames per second.

## Computing platforms

The computing platforms perform the steps required for the successful execution of the selected algorithm. The implementation of these steps on platforms which are architecturally diverse, warrants certain intuitive digressions from the original algorithm in order to better accommodate the strengths of the platform.

**Microprocessors**

Two different microprocessor based platforms were employed in our analysis. The first was an Intel Pentium M Processor with 1.8 Ghz Clock Frequency, 512Kb L2 Cache and 1GB RAM. The second platform comprised two Dual Core AMD Opterons, 2.2 Ghz Clock Frequency, 1MB per core L2 Cache, 2GB RAM. Open Source Computer Vision (Open CV) library was employed for the image processing operations. OpenCV is a library of programming functions mainly aimed at real time computer vision. Example applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, Motion Understanding; Structure From Motion (SFM); and Mobile Robotics.

**Reconfigurable Device**

The FPGA based computing platform was the Sepia Card with PCI and DVI interfacing options and two Xilinx Virtex II 2000 FPGAs. A Stream Compiler (ASC) was used for the implementation of algorithm on the FPGAs. ASC provides an interface for programming (or generating) stream architectures for flexible hardware such as Field Programmable Gate Arrays (FPGAs). The ASC interface uses the familiar C++ syntax with ASC specific semantics to simplify the task of generating flexible hardware accelerators running in parallel to a microprocessor to speed up the software running on the microprocessor. The core of ASC revolves around "programming language (semantics) based architecture generation," or more specifically the generation of stream architectures from a C++ specification of an algorithm [12].

# Switch

The switch symbol in this illustration is used as a logical symbol and represents multiplexing operation in space and time domain.

# Monitor

The results of the driver drowsiness detection algorithm are fed to the monitor for display. Needless to say, commercial implementations need not employ the monitor, or any other device for that matter to display the stepwise results from the algorithm.

# Comparator

The comparator is an observer which can be a special software or a human operator and performs the verification and comparison of the results.

# 3.3 Software Implementation

Software implementation of the driver drowsiness detection algorithm is characterized by a clear focus on harnessing higher clock speeds, significantly larger memory and lack of architectural support for highly parallel processing of independent tasks. Following are the steps necessary for achieving our goal:

## Face region searching:

In order to accurately detect the closure of eyes at a given instance, it is beneficial for the accuracy and robustness of the algorithm, although by no means necessary to detect the location of the driver's face region.

**Vertical Projection Function:**

For a 2D grey scale image of resolution n×m, vertical projection can be described mathematically by the equation:

$$PV(x) = \sum_{y=1}^{m} I(x,y)$$

where,

*PV(x)* is the Vertical Projection of the image
*I(x,y)* is the grey scale value of the pixel
*x* and *y* are the horizontal and vertical coordinates respectively

In order to successfully detect the face region by this step, it is necessary to assume a consistent background and proper placement of the camera. Given these conditions, the vertical projection will be represented by a wave crest or trough depending on whether the face of the driver is darker or lighter than the background.

**Smoothing**

As noise filtering, after this operation, a smoothing operation is recommended. However it was observed during the implementation phase that for the particular application, the required resolution and thus the image size was quite small and therefore, the smoothing operation can be neglected without having any considerable effect on the accuracy of the results.

$$SV(x) = \frac{1}{(k+1)} \sum_{x-k/2}^{x+k/2} PV(x)$$

where,

*SV(x)* is the vertical projection array after smoothing operation
*k* is a configurable value depending upon the image size

**Local Average Thresholding:**

The final stage of face region location is comparing the values to mark the start and end of face region. This can be accomplished by calculating the gradient at every point in the vertical projection array by a simple difference operation, followed by comparing the gradient values to get the maximum and minimum ones position. However, it was observed that in this particular case the transition between the pixel intensity values of the vertical projections tend to be very smooth and thus the gradients do not present a suitable solution for comparing these values. As an alternative to this approach, Local Average Thresholding (LAT) were attempted and found to be more successful. LAT involves the calculation of of the average value of vertical projection, followed by thresholding with respect to local average. Following equation provides the mathematical representation of this computation:

$$PVA = \frac{1}{n} \sum_{x=1}^{n} SV(x)$$

where,

$PVA$ is the average of $PV$

Following is a mathematical representation of the thresholding function:

$$SV(x) = \begin{matrix} 255, \forall\ x > PVA \\ 0, otherwise \end{matrix}$$

This technique proves to be significantly more robust when changing environmental conditions such as lights and shades are taken into consideration. The horizontal boundaries of the face detected by these means are represented by $FL$ and $FR$ respectively.

# Eyes region searching:

The basic theory and operation for the eyes region searching are the same as the face region search except the orientation of the resulting process.

**Horizontal Projection Function:**

The projection value is calculated in horizontal direction, utilizing the information extracted in the previous step to limit the search space to the width of the face defined by $FL$ and $FR$. The mathematical equation for this operation is:

$$PH(y) = \sum_{x=FL}^{FR} I(x,y)$$

where,

$PH(y)$ is the Horizontal Projection of the image
$I(x,y)$ is the grey scale value of the pixel
$x$ and $y$ are the horizontal and vertical coordinates respectively

**Smoothing operation:**

This is followed by the smoothing operation which again, is strictly optional owing to the low resolution of the image. Following equation describes this operation mathematically:

$$SH(y) = \frac{1}{(k+1)} \sum_{y-k/2}^{y+k/2} PH(y)$$

where,

*SH(y)* is the horizontal projection array after smoothing operation
*k* is a configurable value depending upon the image size

**Local Average Thresholding**

The projection values are compared by employing Local Average Thresholding in order to detect the facial boundaries represented by *FT* and *FB*. Following equation provides the necessary mathematical representation of this operation:

$$PHA = \frac{1}{m} \sum_{y=1}^{m} SH(y)$$

where,

*PHA* is the average of *PH*

Following is a mathematical representation of the thresholding function:

$$SH(y) = \begin{matrix} 255, \forall\ x > PHA \\ 0, otherwise \end{matrix}$$

Next, the Local Average Thresholding technique is again applied, this time restricted to the facial boundaries. Although the facial features are evident in the first iteration of *LAT,* however, it was observed that the execution of a second iteration significantly increases the robustness of the algorithm in unpredictable lighting conditions. Mathematically this operation can be represented by the following equation:

$$PHAlt = \frac{1}{(FB - FT)} \sum_{y=FT}^{FB} SH(y)$$

where,

*PHAlt* is the average of *PH* within the facial boundaries

Following is a mathematical representation of the thresholding function:

$$SH(y) = \begin{matrix} 255, \forall\ x > PHAlt \\ 0, otherwise \end{matrix}$$

Generally, in human face, it can be safely assumed that in the presence of appropriate lighting, the colour of skin is brighter owing to the natural glow of skin pigments, as compared to the darker textures evident in other facial features such as hair and eyes. Therefore, the first local minima along the y-axis should be the hair, followed by the eyebrows, eyes, nostrils and finally the mouth. Needless to say, it is assumed that the driver in focus is not wearing hat, cap, mask or any other accessory which can distort or even restrict the observation of visual features.

The detection of vertical boundaries of facial features is followed by the detection of the 'Eye window', which quite simply is the second low intensity island in the vertical direction, sandwiched between similar islands produced by eyebrows and nostrils. The horizontal boundaries of the Eye Window will hereafter be called *EU* and *EL*.

**Edge detection:**

This process detects the outlines of both eyes and boundaries between eyes and face. Since it is not required to extract the features of the driver's eyes, therefore the comparatively simple Prewitt operator is selected for this application. The Prewitt operator consists of two components: the vertical edge component is calculated with the kernel $K_x$ and the horizontal edge component calculated with the kernel $K_y$, as shown in Figure-3. $|K_x|+|K_y|$, gives an indication of the intensity of the gradient in the current pixel.

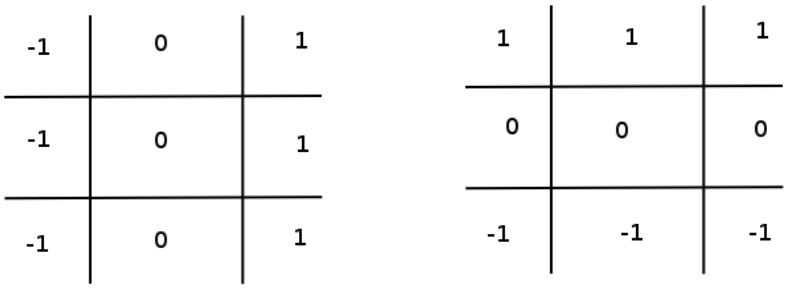


**Figure-3: Horizontal Prewitt kernel ($K_x$) and Vertical Prewitt kernel ($K_y$) respectively**

**Complexity function:**

When the driver's eyes are closed, the only feature captured in the eye area image is a thick curve representing the upper and lower eyelids, accentuated by the eyebrows. However, when the eyes are open, the features detected are considerably more complex, consisting of the upper and lower eye lids and eyebrows around the pupils and the iris. These differences in the visual characteristics of the two images can be highlighted by subjecting the eye area image to a complexity function defined as:

$$compl = \sum_{i=FL}^{FR} \sum_{j=EU}^{EL} \left( b(i,j) - b(i,j-1) \right) \times k(i)$$

where,

*compl* is the complexity value of the 'Eye window'
*b(i,j)* is the binary image resulted by the edge detection
*k(i)* is a weight constant as shown in Figure-4.

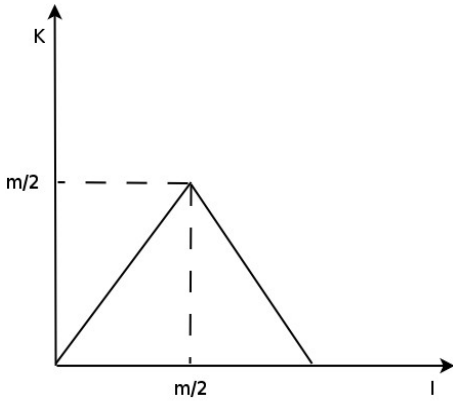


**Figure-4: k(i) function**

Comparison of results demonstrate that the complexity function has a value which varies by a factor of *2X*, depending upon whether the eyes are open or closed.

It is pertinent to mention here that the initial efforts in the algorithm to detect facial boundaries and the 'Eye window' are computationally much less complex as compared to the latter steps of edge detection and complexity function calculation. Therefore, with the advantage of hindsight, we can justify the extra computational effort to mark out the 'Eye window' as it results in significant decrease in the overall computational load.

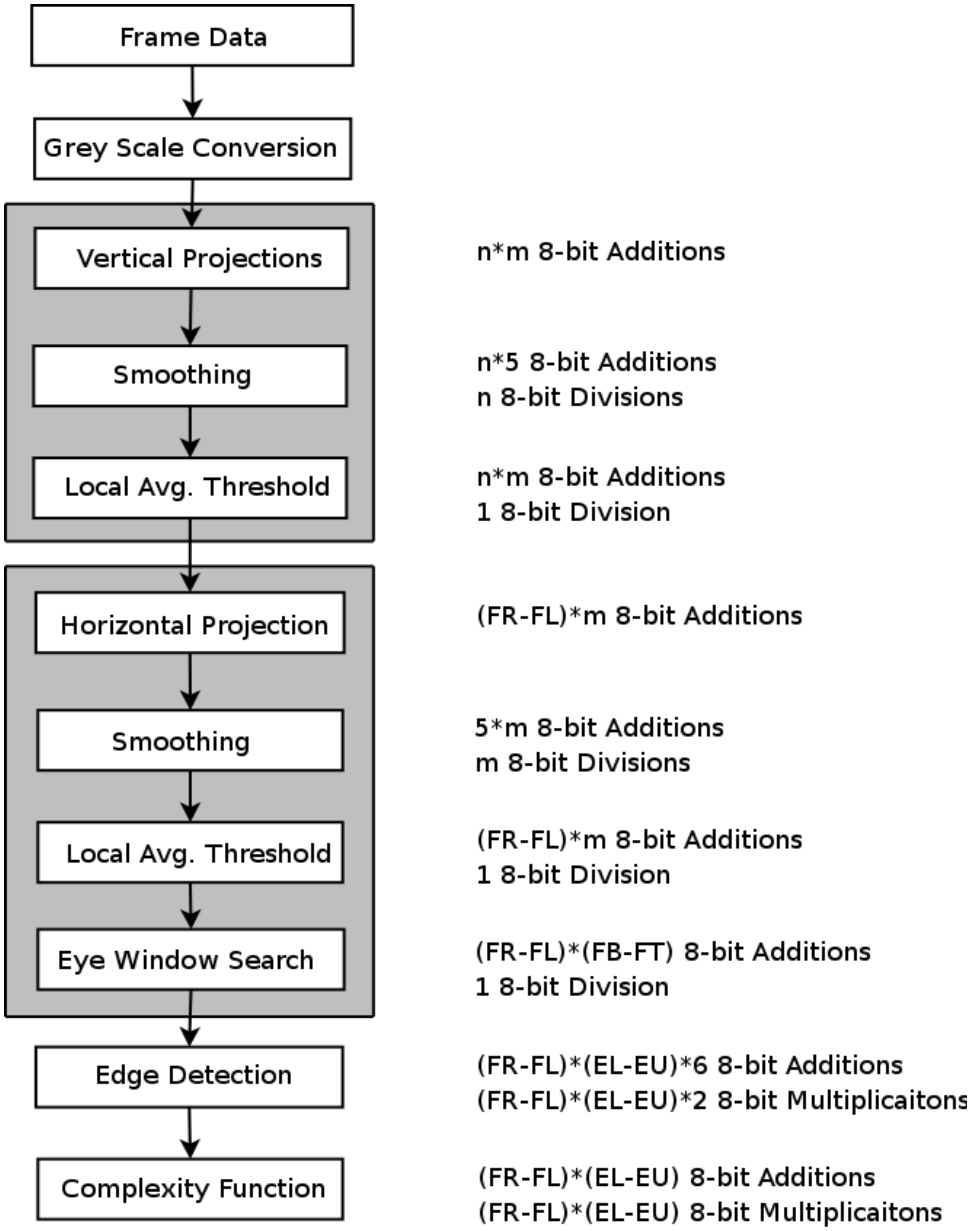


**Figure-5: Software Implementation of Driver Drowsiness System**

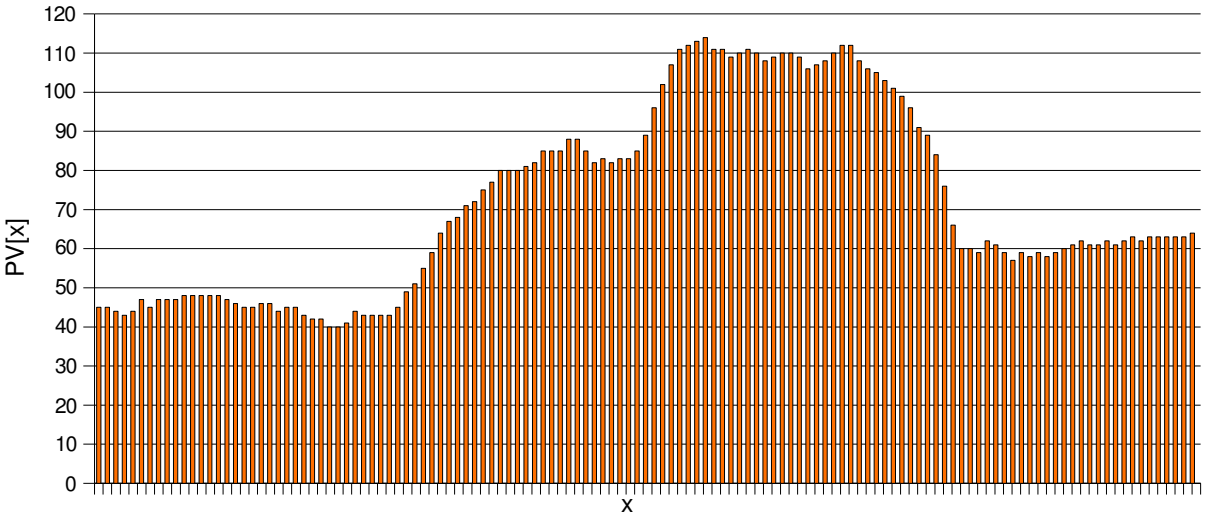


**Figure-6: Vertical Projections**

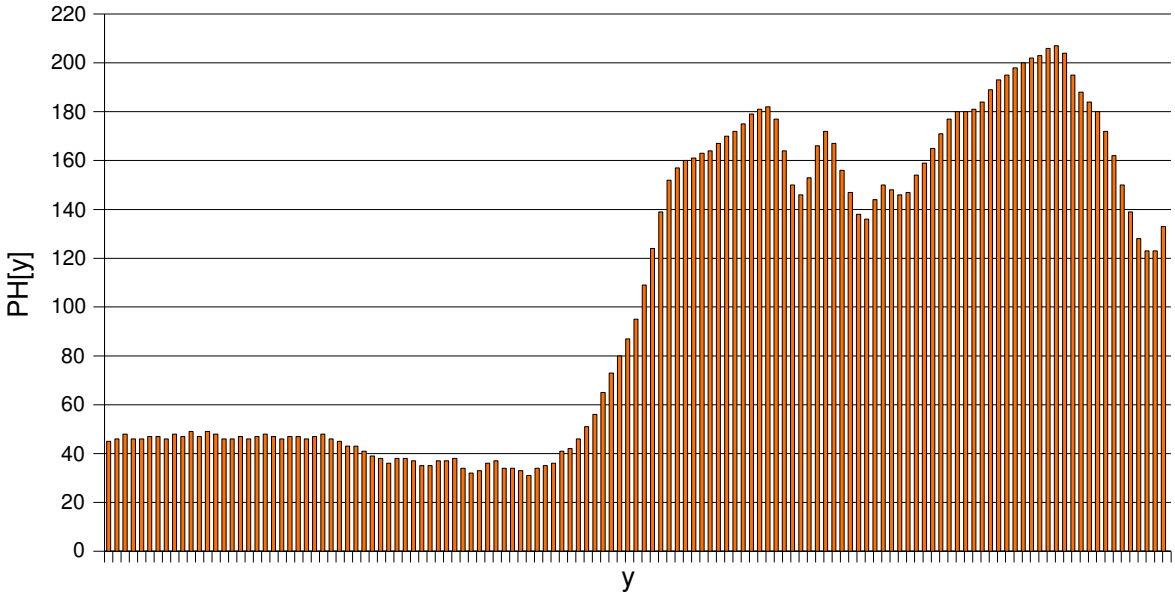


**Figure-7: Horizontal Projections**

# 3.4 Hardware Implementation

Hardware implementation of the driver drowsiness detection algorithm attempts to utilize the advantages of the contemporary FPGA architectures while minimizing the effect of potential pitfalls. It employs the massive potential for parallel execution of independent routines, even in effect introducing redundancy in the system for harnessing the power of stream architecture. It avoids the use of on-chip dual port RAM in order to attain a certain amount of independence form the camera parameters such as resolution and image size. Off-chip RAM available on the Sepia Card is avoided for the obvious benefits in terms of throughput. Following are the variations in the steps necessary for achieving an optimal hardware implementation of the algorithm:

## Face region searching:

The hardware implementation utilizes what is termed a 'Stream Architecture'. The reason is that FPGAs offer different trade-off than custom processors, namely: large area, slow clock frequency (loss of about 5x over microprocessors), fully flexible instruction set (ALUs), a fully flexible interconnection network, and of course the option to reconfigure the entire chip within tenths of a second. As a consequence ASC generates stream architectures which are designed to take full advantage of the FPGAs strengths while hiding the FPGAs weaknesses as much as possible. In more practical terms this means that, given sufficient area, ASC generates an architecture that can achieve data rates equal (or at least close) to the clock frequency of the circuit. On the other hand the FPGA area cost of accessing what is termed as 'Stream History' grows with the length of the time delay after which a particular stream value is recalled, therefore it is necessary in order to achieve maximum possible throughput, to avoid excessive use of this feature. As an alternative, it is recommended to employ external or internal memories to achieve more efficient implementations. However, as previously discussed, any attempt for buffering the image captured from the camera will result in reduced design abstraction and throughput. This leads us nicely to a potential bottleneck in the implementation of out algorithm.

As described in the system architecture, the DVI input form the VGA card is fed directly to the Sepia Card, providing the steady stream of pixels containing the necessary visual information required for the detection of drier drowsiness. This stream follows the standard spatial pattern of CRT devices as shown in Figure-8

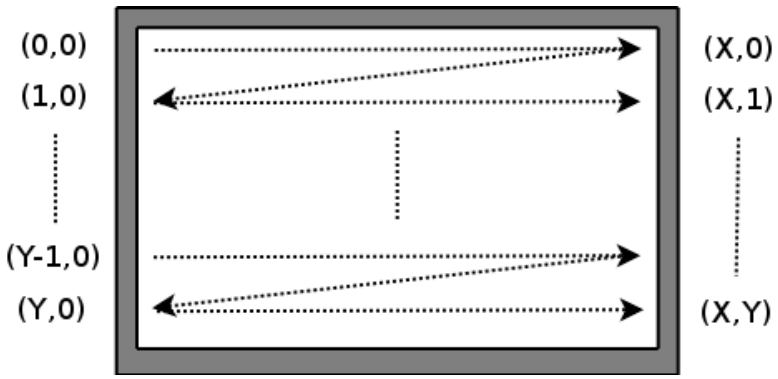


**Figure-8: Scanning operation in a graphic device**

Evidently, the vertical projections of the image, can only be calculated after the last row of pixels is fed through to the Sepia Card. Therefore, the computation of the latter steps of the algorithm can be achieved in two ways:

1. Buffering the image in On-chip or Off-chip RAM. This solution instantly remedies the situation however, as discussed earlier, using the On-chip or Off-chip RAM results in disadvantages in design abstraction and throughput respectively.

2. Benefiting form the fact that the usual operating frame rates (20-30 fps), are high enough not to allow any significant movement by the driver in consecutive frames, thus allowing for the latter steps of the algorithm to be executed on the data from latter frames. This approach can be useful however the particular application can be termed as 'Safety Critical', and it can be stated that a delay spanning a tenth of a second can result in loss of human life. Therefore it is not feasible to utilize 2 or possibly even 3 frames for the necessary results.

This predicament was solved by completely dropping this step from the sequence of computations. The testes conducted after this, demonstrated that this innovation resulted in a worst time delay which was lesser than that incurred by employing the second technique discussed above while in the normal circumstances, the results obtained were better than the ones produced by the first technique presented above.

This decision however, had many implications on the latter steps of the algorithm.

# Eyes region searching:

### Horizontal Projection Function

The basic operation of the Eyes region search remains the same as in the software implementation. However, in this case, we do not have the facial boundaries to limit the search space. Therefore the horizontal projections obtained at the end of this step are comparatively less accurate and the facial features reflected in the horizontal projections are less pronounced due to the inclusion of background noise and light/shade variations in the computations. This horizontal Projection function can be represented by the following equation:

$$PH(y) = \sum_{x=0}^{n} I(x,y)$$

where,

*PH(y)* is the Horizontal Projection of the image
*I(x,y)* is the grey scale value of the pixel
*x* and *y* are the horizontal and vertical coordinates respectively

**Contrast Enhancement**

In order to extract the requisite features out of the image, it is necessary in this case to pronounce these features to a certain extent. Contrast enhancement can be used to good effect in this regard. Mathematically, Contrast enhancement can be represented by the equation:

$$CH(y) = (PH[y] \times Gamma) + Beta$$

where,

*CH(y)* is the Horizontal Projection of the image with enhanced contrast
*Gamma* and *Beta* are constants and their values can be found experimentally

The phenomenon can be illustrated by Figure-9. Contrast enhancement is responsible for introducing a certain gap between valid values of the horizontal projection, thus helping in the thresholding process.

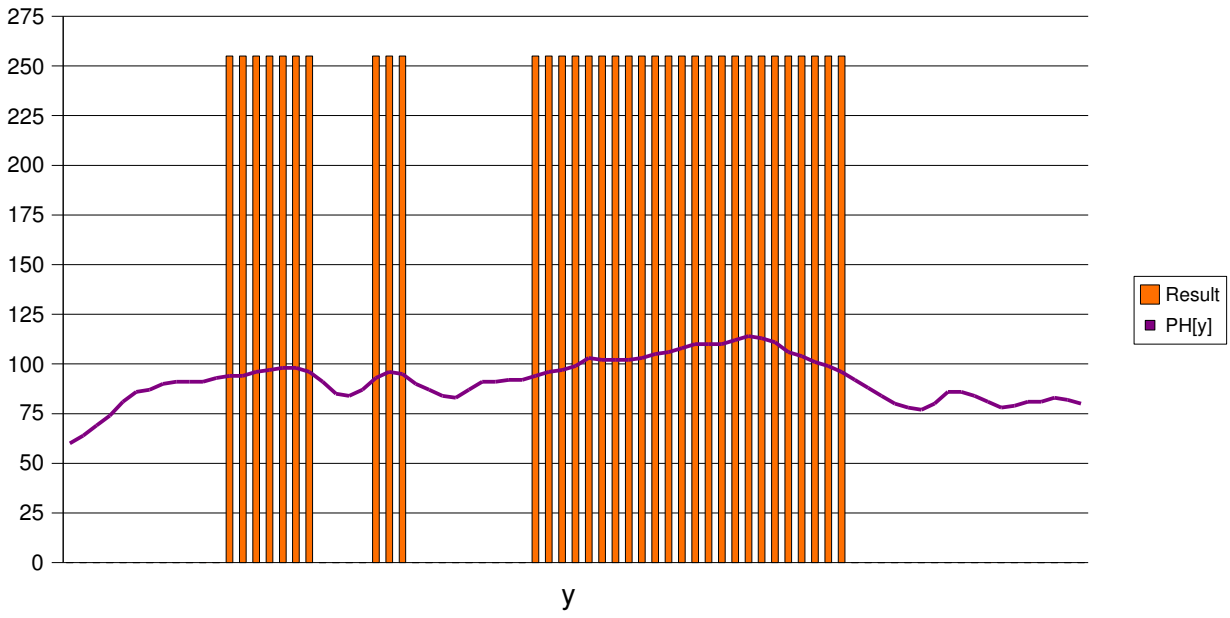


**Figure-9: Contrast Enhancement Function**

After contrast enhancement, the new values of horizontal projection are subjected to a thresholding operation, which can be defined by the following equations:

$$CH(y) = \begin{matrix} 255, \forall\ x > 255 \\ 0, otherwise \end{matrix}$$

This thresholding operation results in the marking of the relatively dark facial features such as hair, eyes, mouth etc. against the lighter tone of skin. The thresholding is followed by the comparison of the values and search for local minima representing the eye window.

# Edge detection

The process of edge detection is conducted as before, however, the Sobel operator is employed instead of the Prewitt operator. The Sobel operator is computationally more complex requiring 2 multiplication or bit shifting operations for every pixel. However, owing to the parallel execution of computations in hardware implementation, the added complexity does not have any notable adverse effect on the performance and throughput of the device. The Sobel operator, just like the Prewitt operator consists of two components: the vertical edge component is calculated with the kernel $K_x$ and the horizontal edge component calculated with the kernel $K_y$, as shown in Fig. 1. $|K_x|+|K_y|$, gives an indication of the intensity of the gradient in the current pixel.

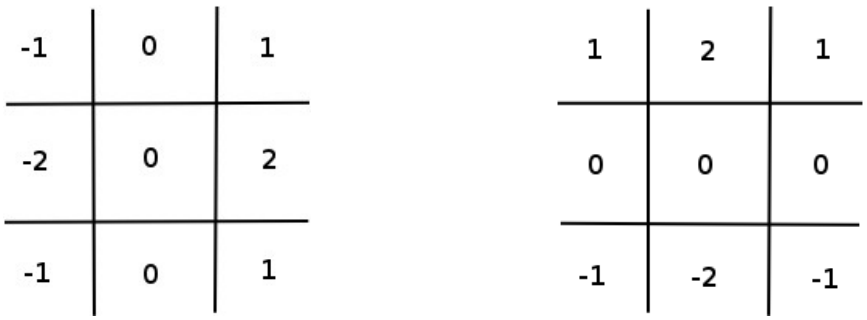


**Figure-10: Horizontal Sobel kernel ($K_x$) and Vertical Sobel kernel ($K_y$) respectively**

In addition to the added edge detection operator complexity, the hardware implementation has to do without the considerable reduction in search space resulting from executing the face region search in the software implementation resulting in a much larger and less refined Eye Window. However, as in the case of edge detection, the added complexity does not have any notable effect on the execution of the algorithm owing to the parallel implementation of the algorithmic steps.

**Complexity function:**

Complexity function is calculated manner similar to the software version. Consistent with the earlier implementation, the complexity function introduces a difference in the order of *2X* between the values computed with closed eyes and those computed with opened eyes, thus making it possible to detect the vigilance level of the driver.

It is important to note here that the more complex and computationally demanding steps in the algorithm, such as edge detection and complexity function computation are ideal for the massively parallel architectures offered by FPGA.

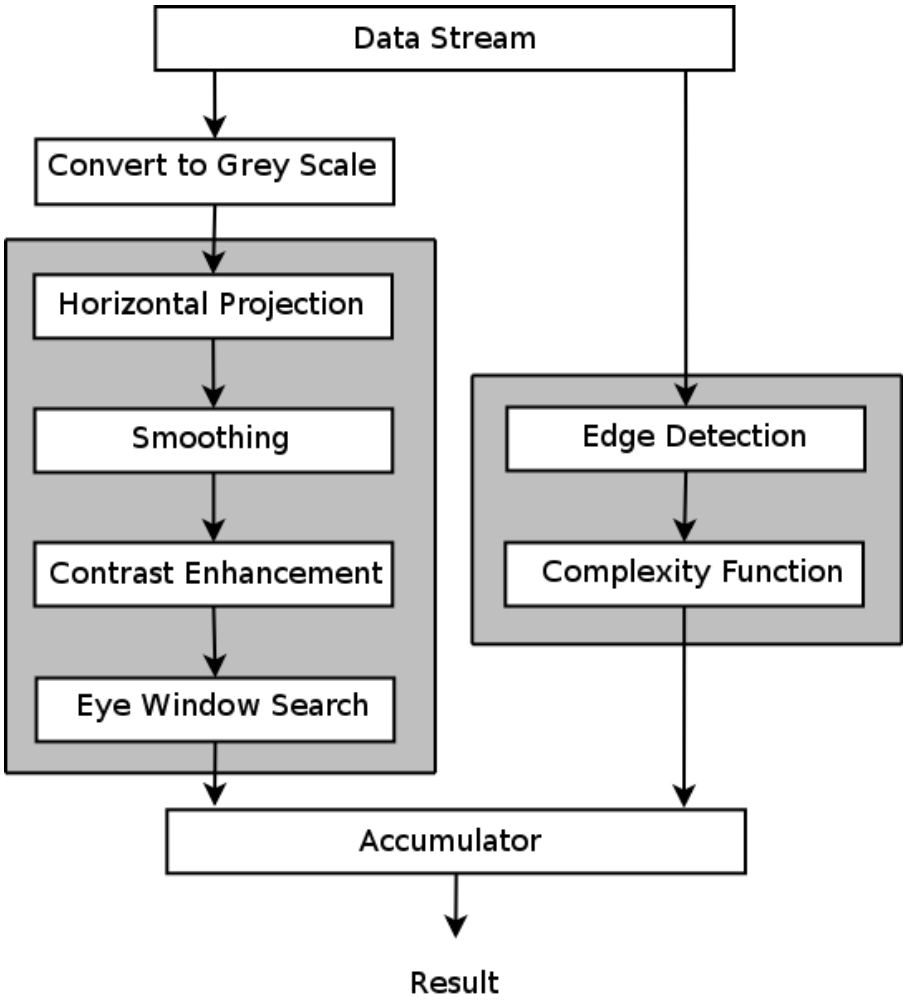


**Figure-11: Hardware Implementation of Driver Drowsiness Detection algorithm**

# Chapter 4

# Analysis of Results

## 4.1 Software implementation

Stepwise results of the software implementation of the driver drowsiness detection algorithm are presented below. These results were obtained by executing the software implementation of driver drowsiness detection algorithm on static images of resolution 150x200, 300x400 and 600x800. However, only the results of 300x40 resolution have been included in this report. These images were obtained from the Logitech QuickCam Pro 4000 digital camera. These images have been subjected to post-processing to illustrate the operation of the algorithmic step.

### Face Region Search

Figure-12 represent the results of the face region search performed with the driver's eyes opened and closed respectively. Notice the horizontal strip on the bottom of the pictures. These are the normalized values of the vertical projection function. Notice the silhouettes of the driver's face reflected in this strip. The facial boundaries detected as a result of this step are also shown.

### Eye Region Search

Figure-13 represent the results of the horizontal projection function performed with the driver's eyes opened and closed respectively. Notice the vertical strip on the right of the pictures. These are the normalized values of the horizontal projection function. Notice the silhouettes of facial features reflected in this strip. The facial boundaries detected as a result of this step are also shown.

Figure-14 represent the results of the final step in the eye region search. The LAT is performed for the second time, this time limited by the facial boundaries on all the four sides. Eyes are represented by the second cluster of dark pixels in the horizontal projection function as shown by the white lines.

Figure-15 is the result of edge detection and complexity function performed on the eye window marked in the previous steps. Notice for the picture when eyes are closed, there are significantly fewer edges, resulting in value of complexity function which is smaller by a magnitude of *2X*.

**Figure-12: Result of the Face Region Search**

**Figure-13: Result of the Eye Region Search**

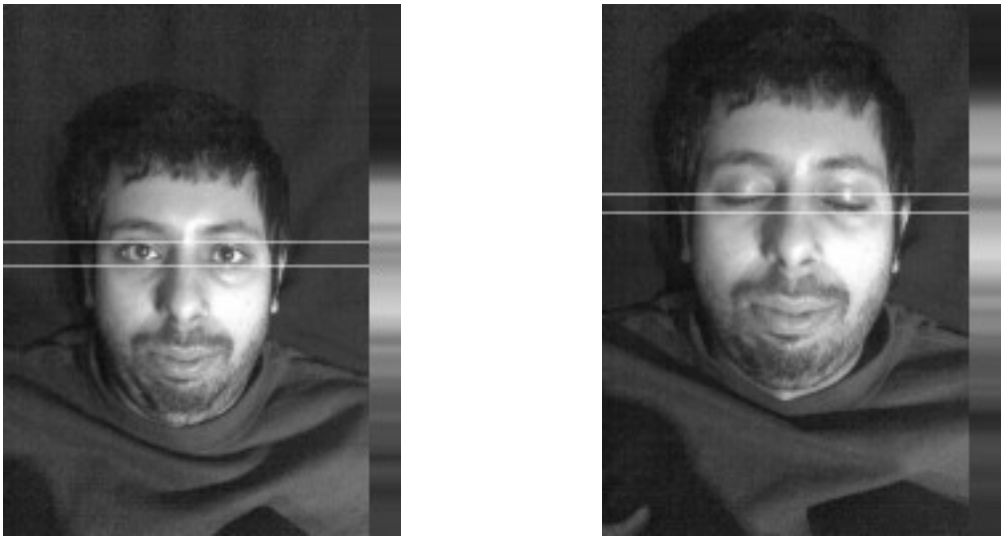

**Figure-14: Result of the Eye Region Search**

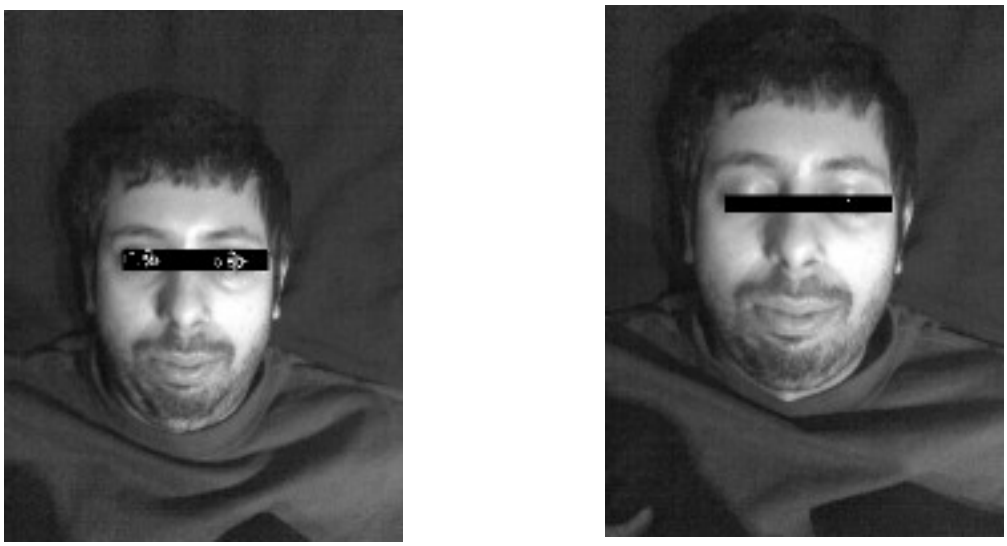

**Figure-15: Result of the Edge Detection**

# 4. Hardware Implementation

The hardware implementation was tested in simulation using images of resolution 150x200, 300x400 and 600x800. Thee images were captured from Logitech QuickCam Pro 4000 digital camera. The images shown below were obtained by accumulating the results from the various algorithmic steps and employing a software to convert these results into images for the purpose of illustration.

Figure-15 is the result of horizontal projection function performed when the driver's eyes are opened and closed respectively. Notice the darkening of the silhouettes in the strip representing the normalized values for the horizontal projection function. This is due to the inclusion of background pixels in the horizontal projection limit.

Figure-16 represent the results of edge detection and complexity function when the driver's eyes are opened and closed respectively. Notice the effect of Contrast enhancement function performed in the hardware implementation on the vertical strip representing normalized horizontal projection values.

**Figure-16: Result of the Eye Region Search**

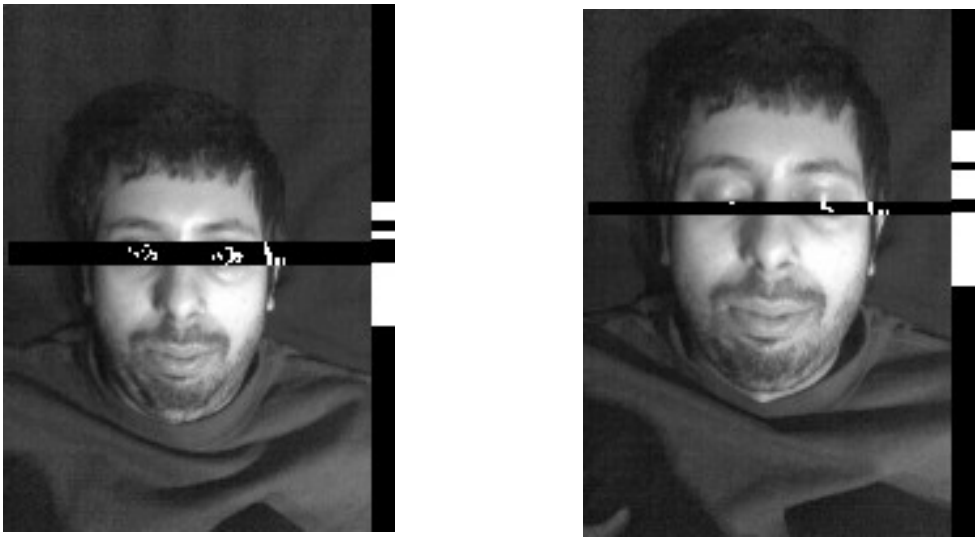

**Figure-17: Result of the Edge Detection**

# 4.3 Performance Comparison

The algorithm implemented in software was subjected to profiling operations. The results obtained give a fair estimation of the computational cost and execution time required by each step of the algorithm. Although the edge detection and complexity function incurred the most computational cost per pixel, calculation of vertical projections required the most execution time averaging at 24.1 milliseconds for the Intel Pentium M processor and 13.8 milliseconds for the Dual core AMD Opteron for an image resolution of 300x400. This was followed by the calculation of horizontal projection which averaged at 6.7 milliseconds for the Intel Pentium M processor and 4 milliseconds for the Dual core AMD Opteron for the image resolution of 300x400. This was largely due to the number of pixels on which the respective operations are performed. However, these operations result in an estimated decrease of 80 percent in the number of pixels for which the more computationally complex operations including edge detection and complexity function are performed. The results of profiling various algorithmic steps are illustrated in Figure-18 to Figure-20.

Given a data stream of significantly high speed, the system throughput is restricted by the latency of the hardware implementation of driver drowsiness detection system actually realized in the FPGA. The latency of our design was estimated to be 0.25 microseconds with the maximum operating clock frequency recommended by the synthesis report to be 20 MHz. This figure can be improved significantly by further optimization efforts. The time required to process a frame can be estimated by multiplying the latency with the frame resolution.

Figure-21 presents the performance comparison of the software platforms in executing the driver drowsiness detection algorithm as compared to the hardware platform. The comparison is illustrated in terms of execution speed. It was found that the hardware implementation provided an average speed-up of 12.18 for the Intel Pentium M processor and an average speed-up of 6.63 for the Dual core AMD Opteron. The maximum estimated speed-up achieved for Intel Pentium M processor was 15.67 for the resolution of 150x200 and for Dual core AMD Opteron, it was 7.07 for the resolution of 150x200.

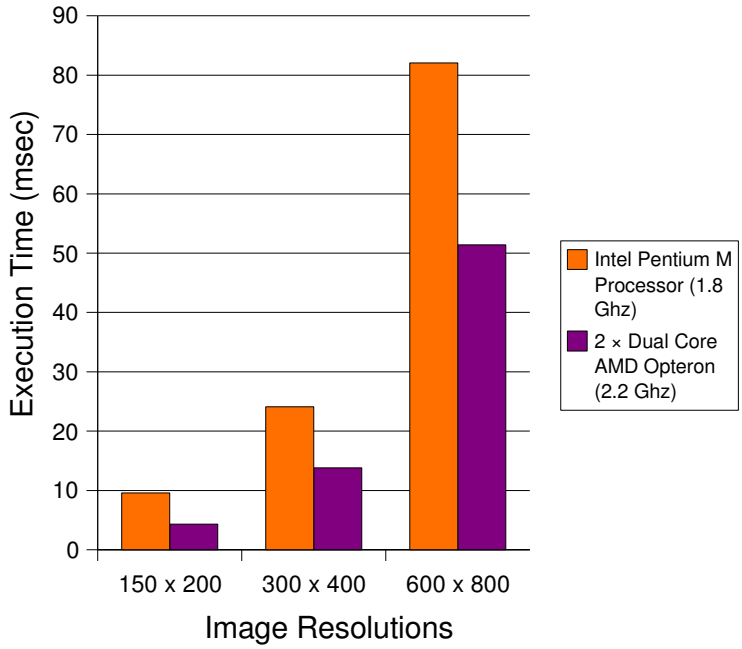


**Figure-18: Performance analysis for Vertical Projection Function**

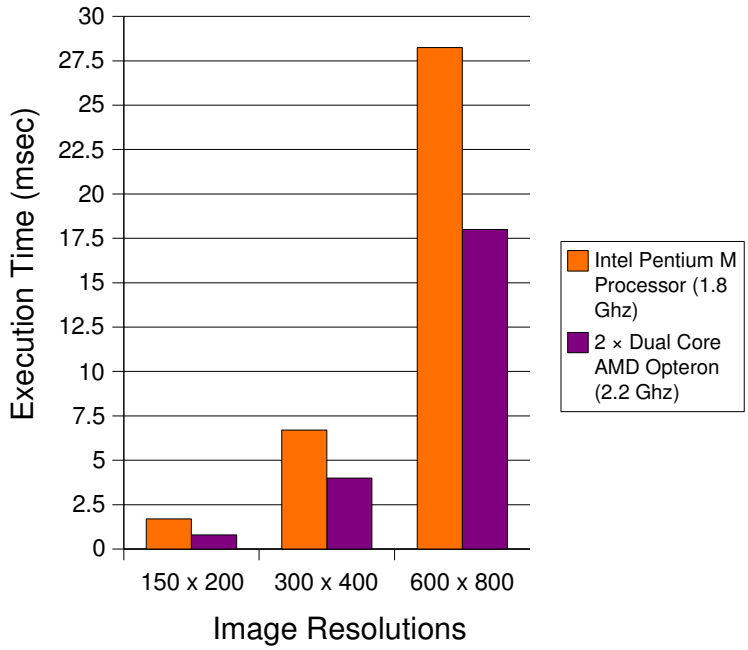


**Figure-19: Performance analysis for Horizontal Projection Function**

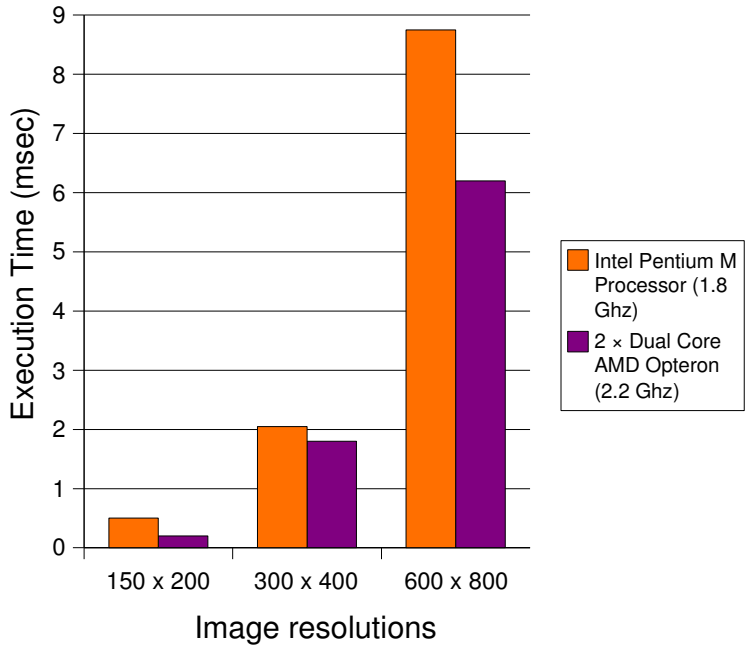


**Figure-20: Performance analysis for Edge Detection and Complexity Function**

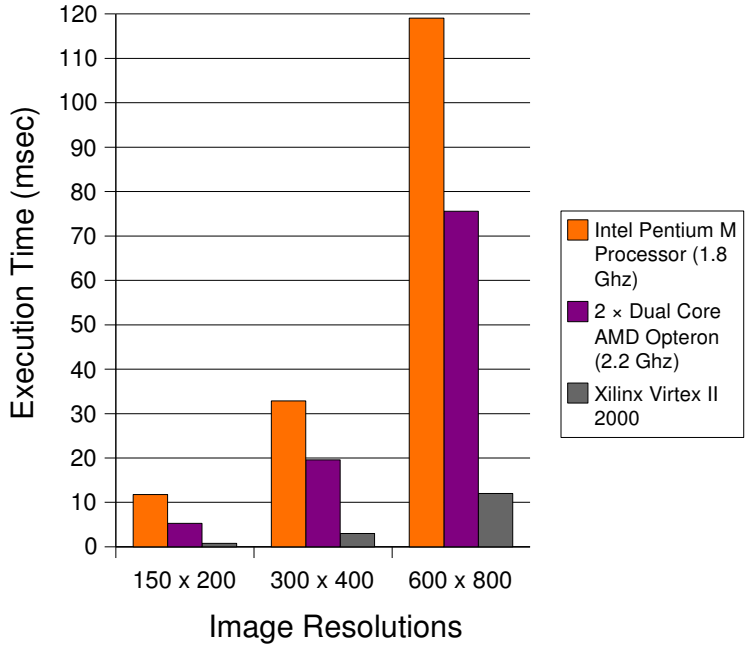


**Figure-21: Comparative Performance of the three platforms**

# 4.4 Limitations

During laboratory testing of the driver drowsiness detection system, following limitations were observed:

**Suitable Illumination**

A correct illumination scheme is a necessary for ensuring that the image has the sufficient contrast. For conditions when ambient light is poor (night time), a light source must be present to compensate. The placement of the light source must ensure maximum light being reflected back from the face. Since the algorithm is highly dependant on light, careful consideration should be given to the fact that different parts of objects are lit differently due to the variations in the angle of incidence of light In addition, brightness values vary due to the degree of reflectiveness of the different objects.

**Camera placement**

During lab testing of this system, the camera was positioned in front of the driver, approximately 30 cm away from the face. However, theoretically the distance between the face of the driver and the camera should not pose a serious issue regarding the performance of the system as long as the resolution of the image is such that it represents the facial features of the driver with a passing amount of accuracy. The angle of the camera position should not pose any problems as well because the algorithm is designed to process an image with a capture angle of about 30 degrees with respect to the normal to facial plane.

Resolution of the camera is closely related to the placement. Generally, the smaller the image the better the performance of the drowsiness detection system, however smaller image requires smaller resolution, which in turn requires a smaller distance between the camera and the driver's face. During lab testing of the system, several different resolutions were employed and it was found that the optimum resolution for a distance of 30 cm was between 150x200 and 300x400. Smaller images failed to portray the facial features with the required accuracy, while larger images required significantly more processing time while demanding a smoothing operation to be performed after every step in the algorithm.

**Background and Ambient Light**

Because the driver drowsiness detection based on the facial intensity variations, it is crucial that the background does not contain any object with strong intensity changes. Highly reflective object behind the driver, can be picked up by the camera, and be consequently mistaken as the face. Since this design is a prototype, a controlled lighting area was set up for testing. Low surrounding light (ambient light) is also important, since the only significant light illuminating the face should come from the drowsy driver system. If there is a lot of ambient light, the effect of the light source diminishes. The testing area included a black background, and low ambient light. This set up is somewhat realistic since inside a vehicle, there is no direct light, and the background is fairly uniform.

**Appearance of Driver:**

In order for the driver drowsiness detection system to work properly, the driver must refrain from using any type of head gear. The implementation of the algorithm relies on accurate representation of facial features in the captured frames. Head gears can hamper the acquisition of the requisite information from the image. Simulated testing of the system has indicated successful drowsiness detection if the colour of the head gear is darker than the skin colour of the driver. However, wearing head gear does result in a certain decrease in the robustness of the system with respect to the angle between the camera and normal to the facial plane.

The algorithm assumes that the skin colour of the driver would be lighter in relation to the background. This supposition may not hold true if the driver's face is darker than the background.

# 4.5 Future Work

Hybrid implementation of the driver drowsiness detection is not included in the scope of this study, however, owing to the understanding and insight developed by the previous implementations, we are in a position to advocate the pros and indicate the cons of such implementation.

Hybrid implementation has the potential to provide the best of both worlds. The control portions of the algorithm e.g. the detection of eye window as well as the serial portions e.g. computation of vertical and horizontal projection call for microprocessor implementation while the computationally expensive portions of the algorithm e.g. edge detection and complexity function computation should benefit from the custom hardware options provided by FPGA. However, the implementation of the algorithm on hybrid platform must take into account the time to transfer the data to be processed to the reconfigurable device as well as the time to accumulate the results. These timings can have a significant effect on determining the feasibility of the hybrid implementation.

An extremely beneficial direction for future work can be the automated detection of the requirement of hardware acceleration in an algorithm, followed by the automated mapping of the critical areas of the algorithm on to reconfigurable hardware for algorithm acceleration. This development will result in generic accelerator cards available for immediate application independent deployment for the execution of performance critical areas of program code.

Several improvements can be made to the driver drowsiness detection system presented in this report. Especially the robustness of this system in real-life situation can be questioned due to lack of testing in highway conditions. Testing of this system in real-life situations may bring some interesting insights to the driver drowsiness detection problem, which can be a good direction for future work. I addition, the system presented in this report has been designed for consistent background. The algorithm can be improved to detect the facial boundaries of the driver in any background, which can also be a good direction for future work.

# 4.6 Conclusion

Driver Drowsiness Detection System was selected as the test algorithm for the comparison of measure of performance of software and hardware implementations. The software platforms participating in this comparative study were an Intel Pentium M processor and a 2 x Dual Core AMD Opteron. The hardware platform was a Xilinx Virtex II 2000 FPGA on the Sepia Card. The selected algorithm was implemented on the various platforms. The implementations were tailored for optimal performance on the respective platforms. The obtained by each implementation were verified for correctness of the algorithm as well as the implementation. After verification, these results were subjected to comparative performance analysis. It was found that for the particular algorithm, the hardware implementation was faster than the software implementation by an factor of 12.18 for the Pentium M processor and a factor of 6.63 foe the 2 x Dual core AMD Opteron. This speed-up accounts for the detection of driver drowsiness by hardware implementation, as much as a tenth of a second earlier as compared to the software implementation. A car travelling at a speed of 100 km/h, covers a distance of about 3m for every tenth of a second. Owing to the safety critical nature of the application, it can be stated that the result obtained from this study justify the use of reconfigurable devices for the implementation of this particular algorithm. In addition, the results obtained for this algorithm are generic enough to be applied to different algorithms of compatible domains and computational cost.

# References

**[1]** Katherine Compton and Scott Hauck. 2002. Reconfigurable Computing: A Survey of Systems and Software. ACM Computing Surveys, Vol. 34, No. 2, pp. 171-210

**[2]** Luis M. Bergasa, Jesús Nuevo, Miguel A. Sotelo, Rafael Barea, and María Elena Lopez. 2006. Real-Time System for Monitoring Driver Vigilance. IEEE Transactions on Intelligent Transportation Systems, Vol. 7, No. 1.

**[3]** A. Kircher, M. Uddman and J. Sandin. 2002. Vehicle Control and Drowsiness. Swedish National Road and Transport Research Technical Report. VIT-922A.

**[4]** J. Healey and R. Picard. 2000. SmartCar: Detecting Driver Stress. Proc. 15th Int. Conf. Pattern Recognition.

**[5]** Y. Matasumoto and A. Zelinsky. 2000. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurements. Proc. IEEE 4th Int. Conf. Face and Gesture Recongnition.

**[6]** T. Victor, O. Blomberg and A. Zelinsky. 2001. Automating the measurement of driver visual behaviours using passuive stereo vision". Proc. Int. Conf. Series Vision Vehicles (VIV9).

**[7]** P. Smith, M. Shah and N. D. V. Lobo. 2003. "Determining driver visual attention with one camera. IEEE Trans. Intell. Transp. Syst. Vol 4, No. 4, pp 205-218

**[8]** Q. Ji and X. Yang. 2002. Realtime Eye, gaze and face pose tracking for monitoring driver vigilance. Realtime Imaging, vol. 8, no. 5, pp 357-377.

**[9]** AWAKE Consortium (IST 2000-28062). 2001-2004. System for assessment of driver vigilance and warning according to traffic risk estimation. http://www.awake-eu.og.

**[10]** Fei Wang and Huabiao Qin. 2005. A FPGA based Driver Drowsiness Detecting System. International Conference on Vehicular Electronics and Safety.

**[11]** Hiroshi Ueno Masayuki Kaneda Masataka Tsukino. 1994. Development of Drowsiness Detection System. IEEE Vehicle Navigation & Information Systems Conference Proceedings.

**[12]** ASC User Manual