DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEVELOPMENT OF A COST EFFECTIVE LVDS INTERFACE TEST SYSTEM WITH ETHERNET COMMUNICATION

by
Mustafa ÇALLI

August, 2010
İZMİR

DEVELOPMENT OF A COST EFFECTIVE LVDS INTERFACE TEST SYSTEM WITH ETHERNET COMMUNICATION

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical and Electronics Engineering, Electrical and Electronics Engineering Program

by
Mustafa ÇALLI

August, 2010
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DEVELOPMENT OF A COST EFFECTIVE LVDS INTERFACE TEST SYSTEM WITH ETHERNET COMMUNICATION"** completed by **MUSTAFA ÇALLI** under supervision of **ASST. PROF. DR. SALİH ZAFER DİCLE** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Asst. Prof. Dr. Salih Zafer DİCLE

Supervisor

.....	.....
Asst. Prof. Dr. Yavuz ŞENOL	Prof.Dr. Yalçın ÇEBİ
(Jury Member)	(Jury Member)

_____

Prof. Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I express my deepest gratitude to my advisor Asst. Prof. Dr. Salih Zafer DİCLE for his valuable guidance and support in every stage of my research. The experience I have gained by his supervisory is a valuable asset for me.

I would like to thank to my manager, chief and colleagues in Vestel Electronics R&D Test Development Engineering Department for their technical contributions during my thesis research.

This thesis work is supported by Republic of Turkey, Ministry of Industry and Trade under “00397.STZ.2009-1” numbered San-Tez Project. I also would like to thank to the Ministry for their support.

Finally, I am grateful to my family for their never ending support, patience and encouragement throughout my life.

Mustafa ÇALLI

DEVELOPMENT OF A COST EFFECTIVE LVDS INTERFACE TEST SYSTEM WITH ETHERNET COMMUNICATION

ABSTRACT

Today LVDS (Low Voltage Differential Signaling) is used extensively in diverse industrial areas like communication networks, laptop computers, office imaging systems, industrial vision systems, test and measurement systems, medical solutions, and automotive. In addition to these, high data throughput of LVDS standard makes it suitable for transmission of high speed digital video signals over inexpensive copper wires and FPD (Flat Panel Display) Link became the first use of LVDS standard for LCD (Liquid Crystal Display) TV (Television) sets.

In this thesis work it is aimed to design a low cost system to capture, decode and analyze LVDS image data from a signal generator (in this case it is the mainboard of an LCD TV).

The research comprises of image data capture card hardware design, embedded microcontroller software design and PC (Personal Computer) application software design parts. Communication between image data capture card and PC is implemented using Ethernet protocol.

By realizing proper signal translation in hardware design and necessary data decoding methods in software design the same system architecture can be used to acquire and analyze any kind of differential signal.

Keywords: LVDS (Low Voltage Differential Signaling), high speed embedded hardware design, embedded software design, image comparison, image processing, digital video, and Ethernet communication.

DÜŞÜK MALİYETLİ, ETHERNET HABERLEŞMELİ BİR LVDS ARAYÜZ TEST SİSTEMİNİN TASARLANMASI

ÖZ

LVDS (Düşük Gerilimli Diferansiyel Sinyalizasyon) bugün haberleşme ağları, dizüstü bilgisayarlar, ofis görüntüleme sistemleri, endüstriyel görüntüleme sistemleri, test ve ölçüm sistemleri, tıp çözümleri ve otomotiv gibi farklı endüstriyel alanlarda yaygın bir şekilde kullanılmaktadır. Bunlara ilaveten LVDS standardının yüksek veri taşıma kapasitesi bu standardı yüksek hızlı sayısal video sinyallerinin ucuz bakır kablolar üzerinden taşınması için uygun kılmıştır ve FPD (Düz Panel Ekran) Link LVDS standardının LCD (Likit Kristal Ekran) TV'ler (Televizyonlar) için ilk kullanım alanı olmuştur.

Bu tez çalışmasında bir sinyal üretecinden (bu durumda bir LCD TV anakartı) gelen LVDS resim bilgisinin yakalanıp, çözüleceği ve analiz edileceği düşük maliyetli bir sistemin tasarlanması hedeflenmiştir.

Araştırma veri yakalama kartı donanım tasarımı, gömülü mikrodenetleyici yazılımı tasarımı ve PC (Kişisel Bilgisayar) uygulama yazılımı tasarımı kısımlarından oluşmaktadır. Veri yakalama kartı ile PC arasındaki haberleşme Ethernet protokolü kullanılarak gerçekleştirilmiştir.

Donanım tasarımında uygun sinyal çevrimlerinin ve yazılım tasarımında gerekli veri çözme yöntemlerinin gerçeklenmesi ile aynı sistem mimarisi her tür diferansiyel sinyalin yakalanması ve analizinde kullanılabilinir.

Anahtar sözcükler: LVDS (Düşük Gerilimli Diferansiyel Sinyalizasyon), yüksek hızlı gömülü sistem donanım tasarımı, gömülü yazılım tasarımı, görüntü karşılaştırma, görüntü işleme, sayısal video ve Ethernet haberleşmesi.

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 Data Acquisition and Processing

Data acquisition is simply the gathering of information about a system or process. It is a core tool to the understanding, control and management of such systems or processes. Parameter information such as temperature, pressure or flow is gathered by sensors that convert the information into electrical signals. The signals from the sensors are transferred by wire, optical fiber or wireless link to an instrument which conditions, amplifies, measures, scales, processes, displays and stores the sensor signals. This is the data acquisition instrument.

Today, powerful microprocessors and computers perform data acquisition faster, more accurately, more flexibly, with more sensors, more complex data processing, and elaborate presentation of the final information. As a result, most scientists and engineers use PCs (Personal Computers) with ISA (Industry Standard Architecture), EISA (Extended Industry Standard Architecture), PCI (Peripheral Component Interconnect) or PCMCIA (Personal Computer Memory Card International Association) bus for data acquisition in laboratory, research, test and measurement, and industrial automation (Rongen, n.d.). Many applications use plug-in boards to acquire data and transfer it directly to computer memory. Others use DAQ (Data Acquisition) hardware remote from the PC that is coupled via parallel port, serial port, GPIB (General Purpose Interface Bus) or other network.

Data acquisition technology continues to evolve, with high speed data interfaces and networking forcing major change to previous practices. Sensitive low level signals can now be left in the field, with just the desired data being returned to a remote computer for analysis. This is the function of a data taker, data logger or DAQ box, providing the functionality and speed of a DAQ board, adding the standalone capability to process, consolidate and log data for later downloading. A

series of data loggers interconnected by a network allows data gathering closer to sensors, for improved signal quality and reduced installation cost.

1.2 Introduction to the Work

Most modern LCD (Liquid Crystal Display) TV (television) sets and notebook computers use LVDS (Low Voltage Differential Signaling) interface to transfer video signals from their mainboards to flat panel displays. Demand of high quality and high resolution video output on the screen is resulted in using such a high frequency and high throughput data transfer interface in TV systems.

In this thesis work it is aimed to design a complete, flexible and low cost system to capture, decode, and analyze LVDS image data from a signal generator (in this case it is the mainboard of an LCD TV). In this point of view the system can be viewed as a distributed data acquisition and central data processing solution like a server-client model.

Wide bandwidth (around 500 MHz for TV display panel video signals) and serial nature of LVDS signals make it impossible to observe and interpret those signals using ordinary oscilloscopes. In addition to these LVDS signals are not suitable to be carried over a long distance which eliminates use of mono-block multi-input data acquisition equipments. The only chance is to use a local display panel or utilize an expensive local LVDS frame grabber system.

A flat panel can be used to interpret the image data originates from the mainboard of a TV. In such a case the tester is the human eye, but the fact that today the cheapest LCD TVs have 24-bit color depth which is roughly equal to 16 million colors for each pixel, so human eye can only notice some serious color deviations in the picture. As a result flat panel and human eye combination is not an adequate solution to test picture quality.

The second alternative is to use an LVDS frame grabber which is available in the market, but such test equipments are usually so expensive to use in production lines. These equipments also require fast PCs to operate adequately and most of the time they offer much more than tester's needs. In addition to that interfacing these equipments with TV mainboards is a challenging task for mass production purposes. Also these devices are not suitable for distributed data gathering systems. In this thesis, the designed system overcomes the problems mentioned in here.

The system design can be divided into three major parts:

- *Hardware design:* In this part a low cost microcontroller based electronic circuit card is designed. LVDS output of TV mainboard is fed to this card. After extraction and processing of image data, it is transferred to a PC using an Ethernet link.
- *Embedded software design:* The microcontroller software is developed in this part. It performs LVDS data acquisition, image data extraction and communication with PC.
- *PC application software design:* PC application software is designed in this part. The software is responsible to retrieve image data from the capture card. Image processing is also carried out by this software, and pass or fail result is presented to user together with detailed test results.

Republic of Turkey, Ministry of Industry and Trade has a program called as San-Tez (Industrial Thesis). The main aim of the Ministry is to create university and industry co-operation on projects which are needed by industry. Hence, scientific research opportunities of university and product creation abilities of industry is brought together to create new projects.

This thesis work is found eligible by Republic of Turkey, Ministry of Industry and Trade to be supported by "00397.STZ.2009-1" numbered San-Tez Project. And it is realized with co-operation of Dokuz Eylül University, Electrical & Electronics Engineering Department and Vestel Electronics Inc., Research & Design Test Development Engineering Department.

1.3 Thesis Outline

The thesis has seven chapters. Each chapter (excluding Chapter 1, Chapter 6, and Chapter 7) is organized in a form that first few sections are comprised of theoretical information and the last section(s) includes detailed information about the design of the related part of the overall system architecture.

Chapter 1 presents data acquisition basics together with introduction to the work.

In Chapter 2, differential signaling theory and its applications are discussed. Some important information about TFT (Thin Film Transistor) LCD panels is revealed. In application part, extraction of RGB (Red Green Blue) data from LVDS data is implemented in hardware.

Chapter 3 presents one of the most important parts of the work. In this chapter DMA (Direct Memory Access) and data buffers are discussed. Temporal image storage is explained. In application part, high speed data acquisition hardware is implemented using a low speed microcontroller together with the help of DMA and data buffers.

Ethernet communication is the main topic of Chapter 4. Firstly, some theoretical background related to implementation of Ethernet protocol is build up in this chapter. The chapter has two application parts: the first one is the design of a fast communication system on embedded hardware/software side and the second one is the implementation of Ethernet communication on PC application side.

Various image processing methods are examined in Chapter 5. Nature of color and color spaces are researched. In application part, theoretical work is applied on real life examples to process images on PC application software.

In Chapter 6 overall system is tested on real TV systems, and some results are presented.

The thesis ends with Chapter 7. This chapter presents a conclusion and reveals some motto for future work on same research area.

The thesis has a wide appendix part that includes seven sections.

Appendix-A1 contains flowchart of embedded software with core functions. The software consists of nearly 3000 lines of C source code, so it is not included in thesis, but can be supplied separately on a digital storage medium.

Appendix-A2 contains flowchart of PC application software with core functions. PC application software is developed by using National Instrument's LabVIEW software development environment. Application software includes nearly 40 modules, so it is not included in thesis, but can be supplied separately on a digital storage medium.

Appendix-A3 contains printed circuit board design of the hardware.

Appendix-A4 and Appendix-A5 contain top side of empty printed circuit board and top side of printed circuit board after auto insertion, respectively.

Appendix-A6 contains detailed hardware design schematics of the capture card.

Appendix-A7 contains embedded system block diagram which is a clear map of the system architecture.

CHAPTER TWO
DIFFERENTIAL SIGNALING

2.1 LVDS Standard

LVDS (Low Voltage Differential Signaling) is a high-speed digital interface that has become the solution for many applications that demand low power consumption and high noise immunity for high data rates. Since its standardization under ANSI (American National Standards Institute)/TIA (Telecommunications Industry Association)/EIA (Electronic Industries Alliance)-644, LVDS has been implemented in a diverse set of applications and industries.

The LVDS standard provides guidelines that define the electrical characteristics for the driver output and receiver input of an LVDS interface, but stop short of defining a specific communication protocol, required process technology, media, or voltage supply (National Instruments, 2009). The general, non-application-specific nature of the standard has been conducive to the adoption of LVDS across a wide variety of commercial and military applications.

Moreover, growing demands for bandwidth have resulted in the emergence of high-performance technologies such as PCI (Peripheral Component Interconnect) Express and Hyper Transport, which are based on high-speed LVDS connections. The low power and high noise immunity aspects of LVDS, along with the abundance of commercial off-the-shelf LVDS components has led many military and aerospace applications to select LVDS as a robust, long-term solution for high-speed data transmission.

The LVDS standard defines the electrical characteristics of the transmitter and receiver of an LVDS interface. LVDS uses differential signals with low voltage swings to transmit data at high rates. Differential signals contrast to traditional single-ended signals in that two complementary lines are used to transmit a signal instead of one line. That is, two signals are generated of opposite polarity, and then

the data transmission references the two signals to one another. This transmission scheme provides the kind of large common-mode rejection and noise immunity to a data transmission system that a single-ended system referenced only to ground cannot provide.

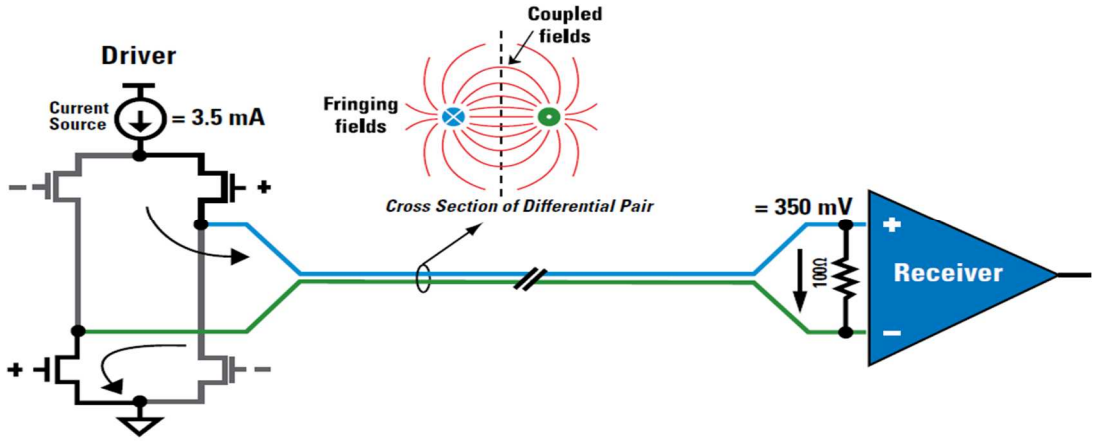


Figure 2.1 Illustration of a typical LVDS transmitter (National Semiconductor, 2008)

Figure 2.1 illustrates a typical LVDS transmitter. This transmitter consists of a current-mode driver, which provides around 3.5 mA of current through the transmission lines of the differential pair. At the receiver, a 100 Ω termination resistor is used to match the impedance of the transmission line that connects the receiver to the driver. Closely matching the impedance of this termination resistor with the impedance of the transmission lines reduces harmful signal reflections that decrease signal quality. The termination resistor also provides a path between the complementary signal paths of the system. The high input impedance of the receiver causes the 3.5 mA current coming from the driver to flow through the 100 Ω termination resistor, resulting in a voltage difference of 350 mV between the receiver inputs. As the path for the current within the driver changes from one path to another, the direction of the current flowing through the termination resistor at the receiver changes as well. The direction of the current through the resistor determines whether a positive or negative differential voltage is read.

As shown in figure 2.2, a positive differential voltage represents logic-high level, and a negative differential voltage represents logic-low level.

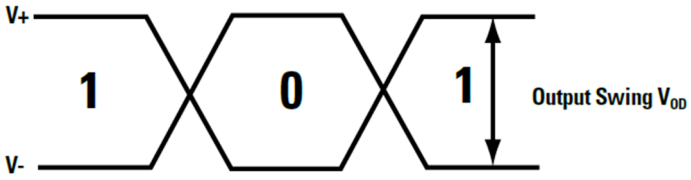


Figure 2.2 Differential signal representing logical levels

As mentioned previously, the ANSI/TIA/EIA-644 standard provides a set of specifications to which all LVDS devices must adhere. Figure 2.3 shows a differential signal labeled with some of the key parameters defined by the standard.

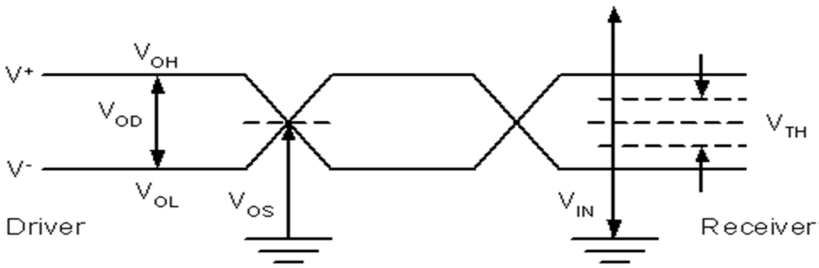


Figure 2.3 Key parameters of a differential signal

The first parameter is the differential output voltage (V_{OD}). This voltage is the absolute value of the difference in voltage measured between the two output lines of the driver and is specified to be between 247 and 454 mV, with 350 mV being typical. V_{OH} and V_{OL} are voltage output high and voltage output low, respectively. These parameters are not specified for LVDS devices, but they can be determined by combining the output offset voltage range (V_{OS}) with the differential output voltage (V_{OD}). V_{OH} and V_{OL} are the output voltages of the driver with respect to ground and should always be within the input range of the receiver.

The standard defines the input voltage range of the receiver, V_{IN}, to be 0 to 2.4 V. This input voltage range is significantly larger than the range of expected voltages from the driver. This difference provides the ability to absorb and reject common-mode noise, noise that is present on both lines of the differential pair, and allow for offsets between the driver and the receiver.

The offset voltage is the common-mode voltage of the differential signal and is essentially the average voltage of the two lines of the differential pair with respect to ground. The minimum and maximum values for V_{OS} according to the standard are 1.125 and 1.375 V. A typical value for V_{OS} is 1.2 V. This value places the differential signal in the center of the voltage input range, V_{IN}, for the receiver. With a voltage swing of 350 mV centered at 1.2 V, a margin of 1.025 V is available on each side of the signal. With this margin, the receiver effectively rejects common-mode noise and ground shifts within this margin.

Another important parameter is threshold voltage (V_{TH}) of the receiver. The threshold voltage is the minimum difference in voltage between the lines of the differential signal that can be registered as a valid logic state. This voltage is specified as |100 mV|; therefore, the positive line of the differential pair must be at least 100 mV greater than the complementary line for the receiver to register logic high level, and the positive line must be at least 100 mV less than the complementary line for the receiver to register a logic low level. Compared to other differential technologies, LVDS and its derivatives have some of the lowest voltage swings. This low voltage swing is one reason why LVDS can achieve very high data rates while consuming lower power than other available data transmission technologies (National Instruments, 2009). Smaller swing requires less power and results in faster transition times between logic states, and this is a key factor in the overall data bandwidth of a transmission path. ANSI/TIA/EIA-644 specifies that the maximum data throughput of a system is dependent on the transmission times of the signal. This relationship is expressed in a maximum output rise and fall time specification of 30% of the unit interval. For example, in order for a system to be classified as 1 Gbps (Gigabits per second) (unit interval of 1 ns), the signals must have rise and fall times smaller than 300 ps (30% of 1 ns) (National Instruments, 2009).

One of the very important features defined by the LVDS standard is the LVDS fail-safe feature. In an LVDS interface, the fail-safe specification forces the receiver to provide logic-high level under certain input conditions. The receiver outputs logic high level when one of the following conditions is true:

- The driver is disconnected from the receiver or powered off while the receiver is still powered on.
- The two lines of the differential pair become shorted.
- The inputs of the receiver are left open.

This fail-safe mode prevents the receiver from providing invalid data because of unexpected voltages on inputs.

The differential nature of LVDS has many inherent advantages. The most fundamental of these advantages is the ability to reject common-mode noise. When the two lines of a differential pair run adjacent and in close proximity to one another, environmental noise, such as EMI (Electromagnetic Interference), is induced upon each line in approximately equal amounts. Because the signal is read as the difference between two voltages, any noise common to both lines of the differential pair is subtracted out at the receiver. The ability to reject common-mode noise in this manner makes LVDS less sensitive to environmental noise and reduces the risk of noise related problems, such as crosstalk from neighboring lines. As a result, LVDS can use a much lower voltage swing compared with traditional single-ended schemes that rely on higher voltage swings to maintain an adequate threshold for noise tolerance. Figure 2.4 represents an illustration of this common-mode noise rejection.

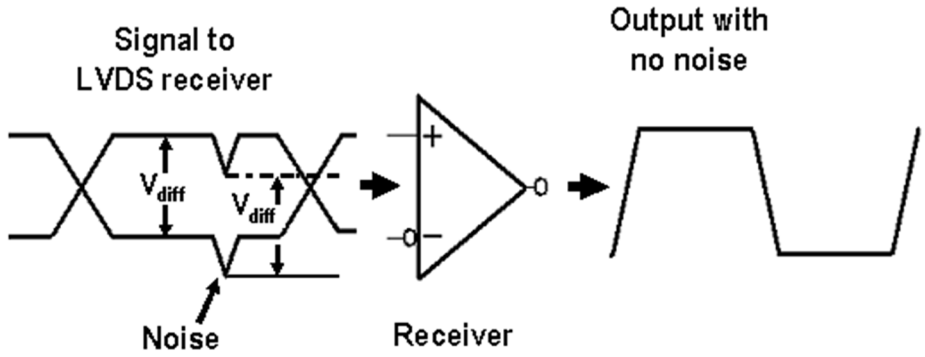


Figure 2.4 Representation of common-mode noise rejection on LVDS

The differential nature of LVDS not only reduces the effects of common-mode noise, it also results in a reduced amount of noise emission. When the two adjacent lines of a differential pair transmit data, current flows in equal and opposite directions, creating equal and opposite electromagnetic fields that cancel one another

as depicted in figure 2.5. The strength of these fields is proportional to the flow of current through the lines. Thus the lower current flow in an LVDS transmission line produces a weaker electromagnetic field than other technologies.

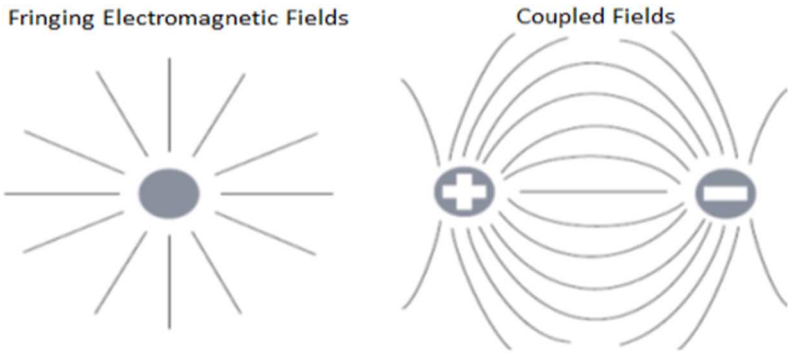


Figure 2.5 Representation of fringing and coupled electromagnetic fields

At first glance it may seem that one of the drawbacks of using LVDS in an application rather than a traditional single-ended data transmission method is that it requires twice as many wires to transmit the same number of channels. In reality, an LVDS application can easily reduce wires between the transmitter and receiver. With the higher data rates available in LVDS, the same amount of data can be transmitted serially across a single channel, avoiding the necessity of transmitting multiple bits in parallel at slower data rates to achieve the same throughput. Multiple channels of slower parallel data can be serialized onto a single high-speed LVDS channel and transmitted from one point to another. At the receiver the data can then be de-serialized and separated into the slower parallel channels. The combination of a serializer and deserializer (SerDes) is a common architecture found in many applications today including Camera LINK and PCI Express.

Another major benefit of LVDS is the low power consumption of it. The current-mode driver of LVDS provides a constant 3.5 mA of current through the differential pair. The power consumption at the load can be calculated using equation 2.1:

$$P = I^2R \qquad 2.1$$

Given the 3.5 mA of current through the 100 Ω termination resistor, equation 2.1 can be written as equation 2.2:

$$(3.5\ mA)^2\ x\ 100\ \Omega = 1.2\ mW \qquad 2.2$$

In comparison, another differential data transmission technology, RS422, dissipates 90 mW of power at the load. Other differential signaling technologies, such as RS485, ECL (Emitter Coupled Logic), and PECL (Positive Emitter Coupled Logic), also dissipate significantly more power than LVDS.

2.2 Various Differential Signals

There are plenty of various high-speed differential signaling technologies. Differential technologies generally share certain characteristics but vary widely in performance, power consumption, and target applications. Table 2.1 lists various attributes of the most common differential signaling technologies.

Table 2.1 Classification of most common differential signaling technologies (National Semiconductor, 2008)

Signaling Standard	Industry Standard	Maximum Data Rate	Output Swing (VOD)	Power Consumption
LVDS	TIA/EIA-644	3.125 Gbps	± 350 mV	Low
LVPECL	N/A	10+ Gbps	± 800 mV	Medium to High
CML	N/A	10+ Gbps	± 800 mV	Medium
M-LVDS	TIA/EIA-899	250 Mbps	± 550 mV	Low
B-LVDS	N/A	800 Mbps	± 550 mV	Low

Industry standards bodies define LVDS and M-LVDS (Multipoint LVDS) technologies in specifications ANSI/TIA/EIA-644 and ANSI/TIA/EIA-899, respectively. Some vendor datasheets claim LVDS I/Os (or pseudo-LVDS) but in fact they may not meet the required common mode or some other important parameters. Therefore, compliance to the LVDS specification ANSI/TIA/EIA-644 is an important consideration (National Semiconductor, 2008).

Current Mode Logic (CML) and Low Voltage Positive Emitter Coupled Logic (LVPECL) are widely used terms throughout the industry, although neither

technology conforms to any standard controlled by an official standards organization. Implementations and device specifications therefore often varies between vendors. AC (Alternating Current) coupling is used extensively which helps resolve threshold differences that might otherwise cause compatibility issues.

For higher data rates, technologies such as CML or LVPECL are required. These technologies can support very high data rates in excess of 10 Gbps. Achieving these very high data rates requires extremely fast, sharp-edge rates and typically a signal swing of approximately 800 mV (National Semiconductor, 2008). For these reasons, CML and LVPECL generally require more power than LVDS.

Sharp, fast edge rates include a significant amount of very-high-frequency content and since transmission loss in cables and FR4 PCB (Printed Circuit Board) traces increases with frequency; these technologies often require signal conditioning when driving long cables or traces (National Semiconductor, 2008).

2.2.1 CML

CML (Current Mode Logic) is a high-speed point-to-point interface that can support data rates in excess of 10 Gbps. As shown in figure 2.6, a common feature of CML is that termination networks are integrated typically into both drivers and receivers. CML uses passive pull-ups to the positive rail, which are typically 50 Ω. Most implementations of CML are AC coupled, and therefore require DC-balanced data. DC-balanced data contains, on average, an equal number of ones and zeros.

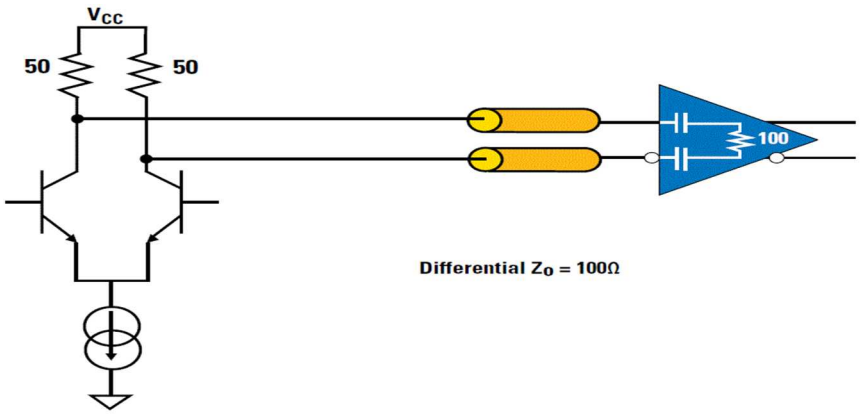


Figure 2.6 Representation of CML transmitter (National Semiconductor, 2008)

2.2.2 LVPECL

LVPECL (Low Voltage Positive Emitter Coupled Logic) and PECL are both offshoots of the venerable ECL technology first introduced in the 1960s. ECL is powered commonly between ground and -5.2 V. Because of the negative rail requirements and ECL's incompatibility with other logic families, a positive rail technology was introduced known as PECL. ECL, PECL, and LVPECL all require a 50 Ω termination into a termination rail that is about 2 V less than the most positive rail. ECL drivers are low-impedance open-emitter outputs that generate typically 700 mV to 800 mV. The output stage remains in the active region, preventing saturation, and results in very fast and balanced edge rates (National Semiconductor, 2008).

Positive features of LVPECL are the sharp and balanced edges and high drive capability. Drawbacks of LVPECL are relatively high power consumption and sometimes the need for a separate termination rail. A typical implementation of LVPECL is shown in figure 2.7.

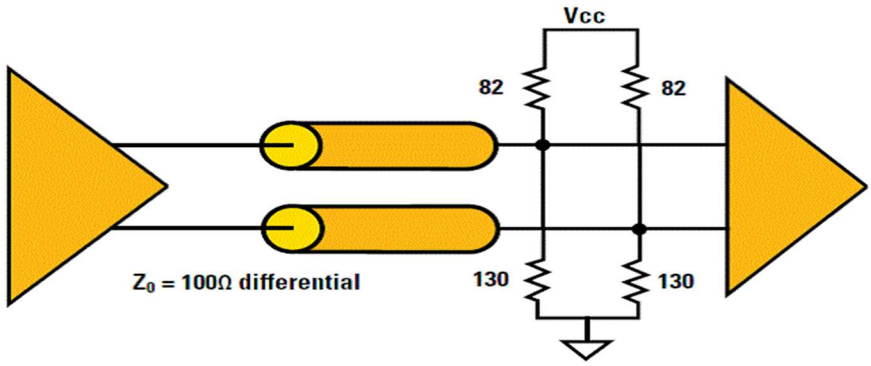


Figure 2.7 A typical implementation of LVPECL (National Semiconductor, 2008)

2.2.3 Comparison of Common Data Transmission Technologies

Data transmission, as the name suggests, is a means of moving data from one location to another. Choosing the best transmission standard to accomplish this task requires evaluation of many system parameters. The first two considerations encountered are how fast, and how far (Texas Instruments, 2002).

How fast refers to the signaling rate or number of bits transmitted per second. How far is concerned with the physical distance between the transmitter and receiver of the data. Consideration of these two primary system parameters usually results in a significant narrowing of the possible solutions. Figure 2.8 shows the speed and distance coverage of some familiar data transmission choices.

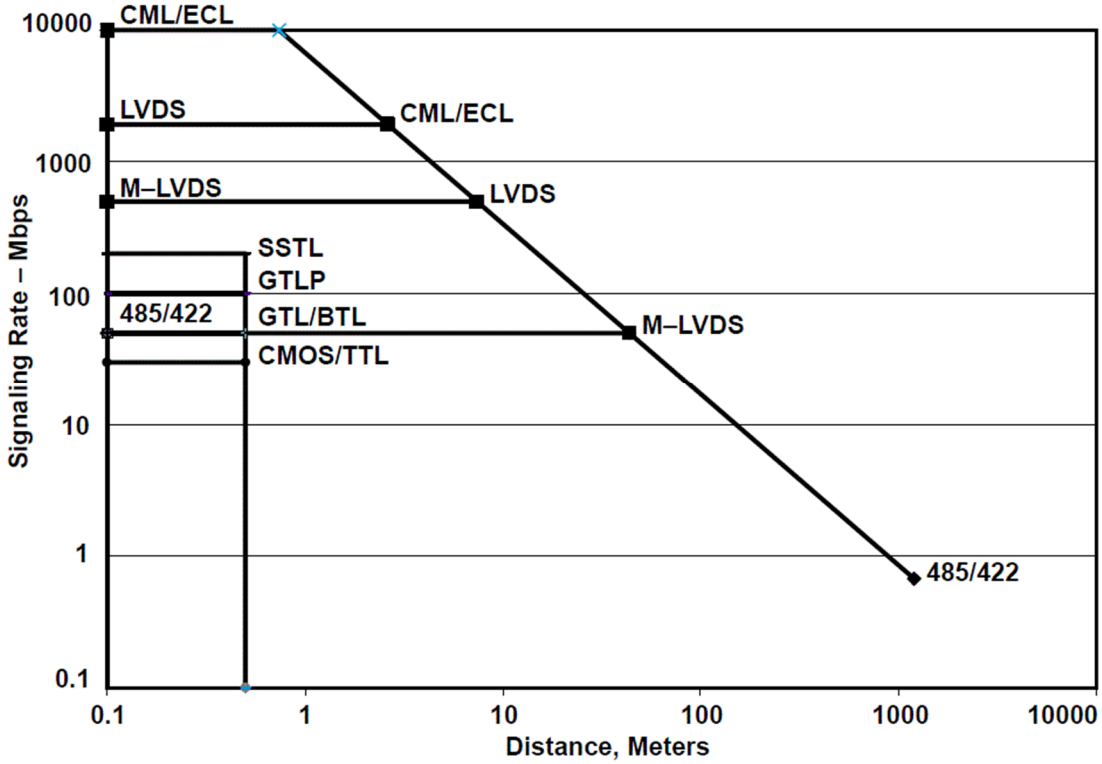


Figure 2.8 Speed and distance coverage of some current signaling technologies (Texas Instruments, 2002)

Figure 2.8 shows that signaling rate eventually decreases as transmission distance increases. While steady state losses may become a factor at the longest transmission distances, the major factors limiting signaling rate, as the distance is increased, are time varying. Cable bandwidth limitations, which degrade the signal transition time and introduce inter-symbol interference (ISI), are primary factors reducing the achievable signaling rate when transmission distance is increased.

Figure 2.8 also shows that general-purpose, single-ended logic, including BTL (Backplane Transceiver Logic), GTL (Gunning Transceiver Logic), and GTLP (Gunning Transceiver Logic Plus) provide satisfactory interface solutions when the

transmission distance is short (< 0.5 m) and the signaling rate is moderate. When transmission distance is increased, standards with higher voltage swings or differential signaling often move the data.

If data transmission over about 30 m and less than 50 Mbps (Megabits per second) is required, differential signaling standards TIA/EIA-422 and TIA/EIA-485 should be considered. High differential outputs, sensitive receivers, and the capability to operate with up to 7 V of ground noise make these interfaces ideal for long direct connections between equipments. TIA/EIA-422 and TIA/EIA-485 use similar voltage levels but differ in the bus topologies they can support. TIA/EIA-422 is used for multidrop (one driver and many receivers) operation, while TIA/EIA-485 allows for multipoint signaling (many drivers and receivers).

For signaling rate greater than 50 Mbps or in low-power applications, LVDS or M-LVDS provides an attractive solution. Introduced in 1996, LVDS offers high signaling rates and low power consumption for point-to-point or multidrop buses. M-LVDS, specified in TIA/EIA-899, was introduced in 2002 and offers similar benefits for the multipoint application (Texas Instruments, 2002).

When the signaling rate requirement exceeds the capabilities of LVDS, CML circuits are used. Signaling at 10 Gbps is possible with ECL/PECL devices. The high speed is achieved at the cost of high power consumption. Figure 2.9 shows voltage swing levels of some differential signaling technologies.

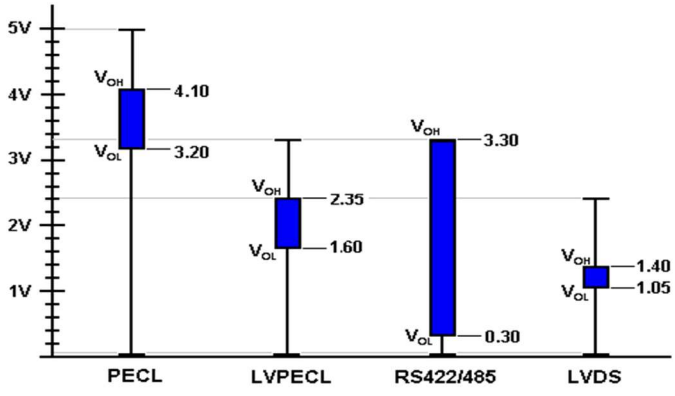


Figure 2.9 Voltage swing levels of some differential signaling Technologies (Fairchild Semiconductor, 2005)

2.3 Working with Differential Signals

With the existence of various differential technologies, a need for some guidance in selecting an optimal signaling technology for an application is obvious. The following are the factors under consideration when selecting an optimal technology for a given application (National Semiconductor, 2008):

- Required bandwidth
- Ability to drive cables, backplanes, or long traces
- Power budget
- Network topology (point-to-point, multidrop, multipoint)
- Serialized or parallel data transport
- Clock or data distribution
- Compliance to industry standards
- Need or availability of signal conditioning

LVDS is the most common differential signaling interface. The low power consumption, minimal EMI, and excellent noise immunity are the features that have made LVDS an interface of choice for many applications. In addition, the LVDS wide-input common mode makes LVDS devices easy to interoperate with other differential signaling technologies. The latest generation of LVDS operates from DC to as high as 3.125 Gbps, allowing many applications to benefit from LVDS. These multi-gigabit LVDS devices feature pre-emphasis and equalization that enables signal transmission over lossy cables and PCB traces.

Applications requiring data rates greater than 3.125 Gbps likely require CML signaling. In addition, certain communication standards such as PCI Express, SATA (Serial Advanced Technology Attachment), and HDMI (High Definition Media Interface) mandate the use of specific signaling technologies or describe a set of conditions such as signal amplitude and reference to V_{CC}, consistent with CML.

For applications with data rates between 2 Gbps and 3.125 Gbps, the optimum choice depends on the desired functionality, performance, and power requirements.

For relatively short distance transmission where signal conditioning is not required the device power and jitter dominates, with CML generally having the lowest jitter and LVDS the lowest power. For long-reach requirements, losses in the media dominate and the best choice is generally the device with the best signal conditioning solution for the data rate and media. Both LVDS and CML use techniques such as equalization and pre-emphasis or de-emphasis.

Understanding the loss characteristics of the transmission media and the best signal-conditioning solution enables the user to select the appropriate device. In the light of above explanations figure 2.10 places popular differential signaling techniques on data rate vs. power consumption graph.

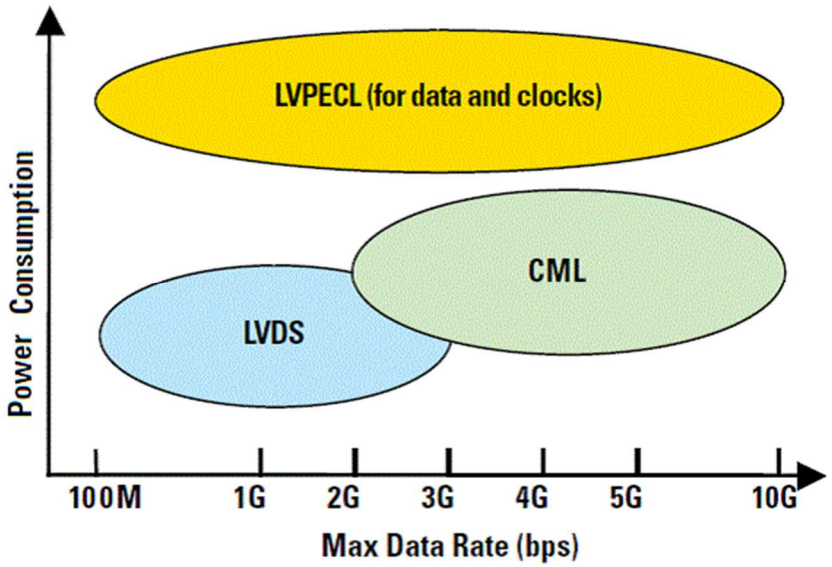


Figure 2.10 Data rate versus power consumption graph for popular differential signaling technologies (National Semiconductor, 2008)

2.4 Structure of Flat Panel TFT LCD

In this part, hardware sections of a flat panel TFT (Thin Film Transistor) LCD (Liquid Crystal Display) screen are detailed. Although a TFT LCD screen has a complex hardware structure and technical specifications, only necessary parts and their specifications are revealed.

TFT LCD screens vary greatly both in terms of panel size and panel resolution, but in general they all have same common properties. At this point general specifications of a sample 42” HD (High Definition) TFT LCD screen are examined, and necessary parameters are further explained.

Figure 2.11 shows the block diagram of a color active matrix liquid crystal display with an integral external electrode fluorescent lamp backlight system. The matrix employs a silicon thin film transistor as the active element. The sample is a transmissive display type which operates in the normally black mode. It has a 42.02 inches diagonally measured active display area with WUXGA (Widescreen Ultra Extended Graphics Array) resolution (1080 vertical by 1920 horizontal pixel array). Each pixel is divided into red, green and blue sub-pixels or dots which are arrayed in vertical stripes. Luminance of the sub-pixel color is determined with a 10-bit gray scale signal for each dot. Therefore, the panel can present a palette of more than 1.06 billion colors. The unit has been designed to be driven by 10-bit 2-port LVDS interfaces.

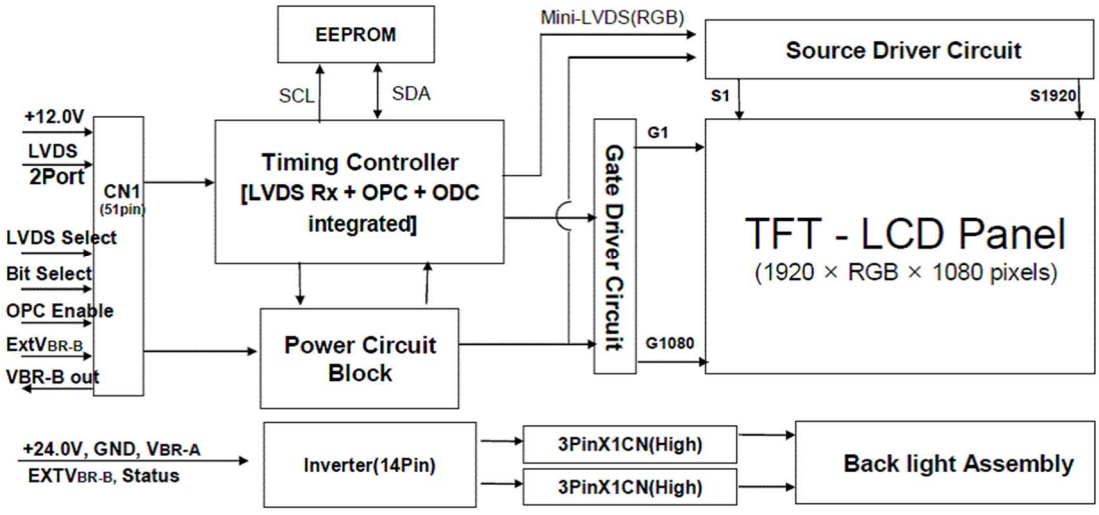


Figure 2.11 The block diagram of a color active matrix liquid crystal display with an integral external electrode fluorescent lamp backlight system (LG Display, 2008)

All necessary image data is transferred over LVDS ports, and other input signals to panel are used for some other functionalities, such as backlight control. So these extra input signals are not needed to acquire the image data.

Three specifications of a TFT LCD screen determine the characteristics of the LVDS communication configuration:

- Panel resolution
- Panel color depth
- Panel refresh frequency

All of these three factors determine the total data capacity of the link between TV mainboard and TFT LCD screen. In terms of these variables various LVDS port combinations can be derived and applied. Table 2.2 does not cover all possible panel types and resolutions, but it represents most commonly used panel resolutions and LVDS configuration schemes among TV producers. It is also worth to mention that table 2.2 does not act as an industry standard.

Table 2.2 Configuration of LVDS port according to different aspects of the display

Native Panel Resolution			**Preferred Number of LVDS Ports**	**Channel Distribution for Each Port**	**Panel Refresh Frequency (Hz)**	**Panel Color Depth**	**Overall Minimum Data Transfer Rate (Gbps)**
Number of Horizontal Lines	**Number of Vertical Lines**	**Display Standard**					
1024	768	XGA	1	4 data pair 1 clock pair	60	3 x 8	1.05
				5 data pair 1 clock pair		3 x 10	1.32
1280	720	(W)XGA	1	4 data pair 1 clock pair	60	3 x 8	1.24
				5 data pair 1 clock pair		3 x 10	1.54
1366	768	WXGA	1	4 data pair 1 clock pair	60	3 x 8	1.41
				5 data pair 1 clock pair		3 x 10	1.76
			2	4 data pair 1 clock pair	100	3 x 8	2.34
				5 data pair 1 clock pair		3 x 10	2.93
1920	1080	WXGA	2	4 data pair 1 clock pair	60	3 x 8	2.78
				5 data pair 1 clock pair		3 x 10	3.48
			4	4 data pair 1 clock pair	100	3 x 8	4.63
				5 data pair 1 clock pair		3 x 10	5.79

Native panel resolution represents the number of horizontal and vertical stripes which constitute the overall image. Some common resolutions are listed in table 2.2.

Number of preferred LVDS ports is determined by panel resolution together with panel refresh frequency. These are named as single channel, double channel and quad channel in industry. Panel color depth usually does not have any effect on this parameter.

In table 2.2 common panel refresh frequencies are also displayed. Usually panel specifications allow the designer to drive the panel in a frequency band, for example between 50 Hz and 70 Hz. Although there is no input source over 100 Hz (except for some PC monitor resolutions), higher frequency panels (for example 100 Hz, 200 Hz) are used together with some motion sensing algorithms to develop a better viewing experience.

Panel color depth determines number of channels in each LVDS port. For an 8-bit panel, each color element (red, green and blue) is coded with 8-bit, total of 24-bit. 8-bit panels use 4 data pairs and 10-bit panels use 5 data pairs together with one clock pair for each port.

The last column of table 2.2 shows overall minimum data transfer rate. In every second, this much of image data is transferred from TV mainboard to panel. These values are derived by using the formula in equation 2.3:

$$BW = HL\ x\ VL\ x\ F\ x\ CD\ x\ 3\ x\ 1024^{-3} \qquad 2.3$$

In equation 2.3, BW denotes overall minimum data transfer rate in Gbps, HL denotes number of horizontal lines, VL denotes number of vertical lines, F denotes vertical panel refresh frequency and CD denotes color depth for each color channel.

Result of equation 2.3 is viewed as a minimum data transfer rate. Due to transmission of other data signals (horizontal and vertical synchronization, data

enable, padding, etc.) the actual data rate is slightly larger than this calculated value in practice.

When the last column of table 2.2 is examined it is clearly observed that the practical bandwidth of each channel is over 300 Mbps and this is much lower than the achievable maximum bandwidth (over 3 Gbps as stated in table 2.1) of LVDS standard. The primary reason of that, routing such high speed differential signals on PCBs result in a lot of signal degradation, inter symbol interference and cross-talk. Adding connector and transmission cable losses into this figure puts a great design challenge. In order to keep the product cost at an acceptable level, stress on LVDS communication is decreased by using lower bandwidth channels otherwise some additional hardware components are needed for signal conditioning.

Figure 2.12 shows an example for PCB losses. The left-hand signal is a 3.125 Gbps LVDS signal which is measured at generator side directly by oscilloscope; the right-hand signal is measured after 71.2 cm of FR4 PCB trace. The attenuation of lossy media is clearly seen that the eye diagram tends to close.

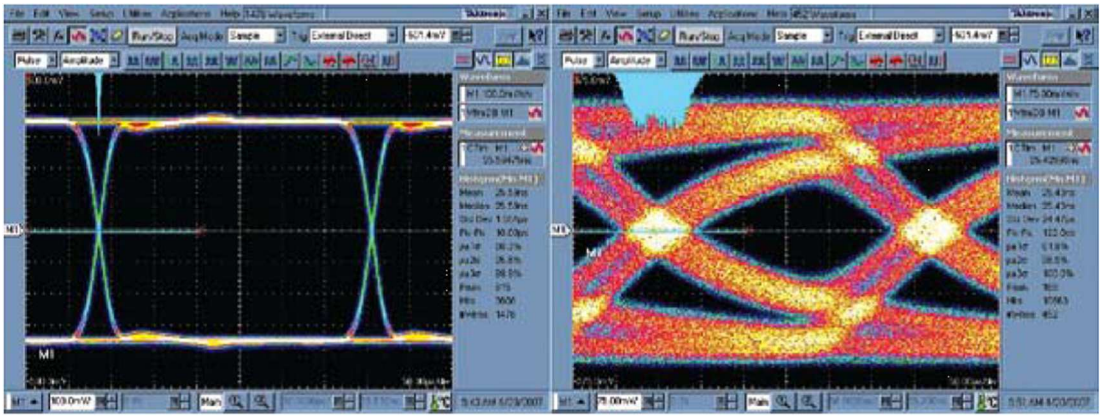


Figure 2.12 An example showing PCB losses during transmission of a differential signal (National Semiconductor, 2008)

2.5 Industry Standards of TFT LCD Signalization

Video data is processed in parallel inside of a TV mainboard, and then this data is serialized onto an LVDS data stream, this data stream is transferred to TFT (Thin

Film Transistor) panel's LVDS port, timing controller (so called TCON) integrated circuit of TFT panel de-serializes this data and drives the TFT panel properly. This data flow can be viewed in figure 2.13.

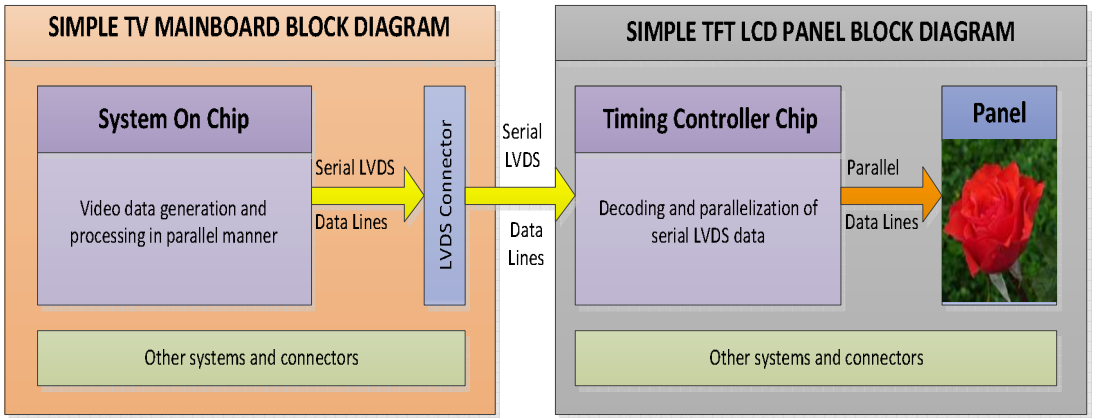


Figure 2.13 Transmission of LVDS data from TV mainboard to LCD Panel

As can be seen from figure 2.13 image data is transferred in a serial LVDS stream from TV mainboard to TFT LCD (Liquid Crystal Display) panel, therefore a standard or communication protocol is needed to serialize or de-serialize this data correctly. Video Electronics Standards Association (VESA) and Japan Electronics Industry Development Association (JEIDA) developed "TV Panel Standard" and "Digital Interface Standards for Monitor" standards respectively. These standards are widely accepted and used in TV industry.

Data mapping of parallel video stream onto serial LVDS lines is clearly explained on these standards together with timing and voltage specifications of LVDS signals. Here the most important part is the data mapping, because both standards use different mapping schemes, but they adopt timing and voltage specifications of waveform from ANSI/TIA/EIA-644.

Figure 2.14 shows data mapping for 10-bit JEIDA interface. This scheme consists of 1 clock pair and 5 data pairs, total of 12 cables. For each clock cycle 7-bit data is transferred on each pair, in other words data signals are 7 times faster than common clock signal, and each clock cycle contains color information for each pixel. Each pixel consists of three color elements (RGB), and each color element is represented

with 10-bit data. R10, G10, B10 denote least significant bit of (R10:R19) (G10:G19) (B10:B19) color data and R19, G19, B19 denote most significant bit. V_{SYNC} and H_{SYNC} denote vertical and horizontal synchronization information respectively. DE denotes data enable signal, and X denotes reserved data for future use.

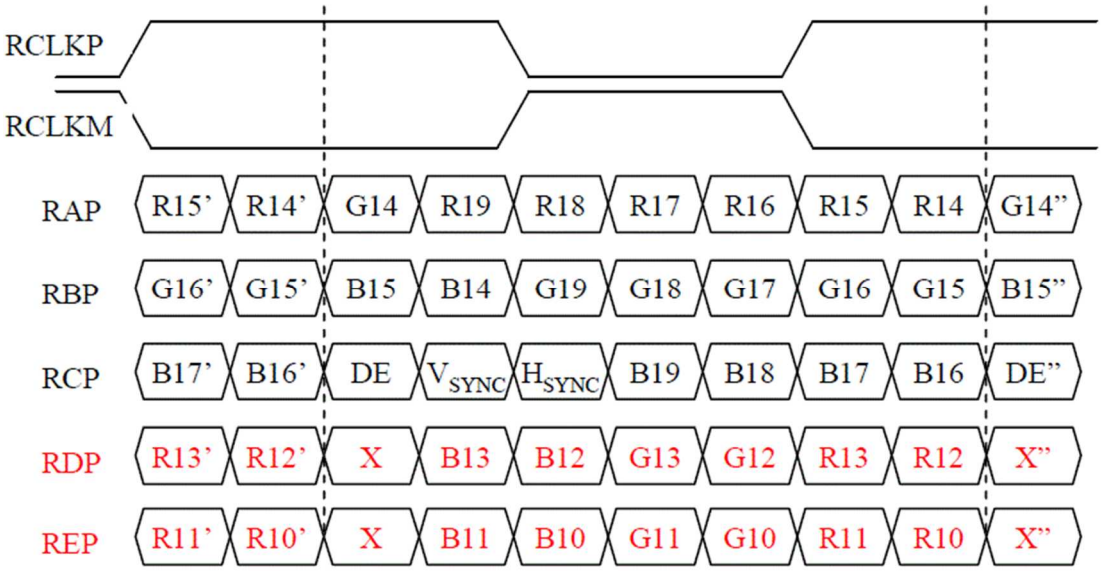


Figure 2.14 Data mapping for 10-bit JEIDA LVDS standard (JEIDA, 1999)

Figure 2.15 shows data mapping for 10-bit VESA interface. Apart from color mapping, this scheme has same properties with 10-bit JEIDA interface as in figure 2.14.

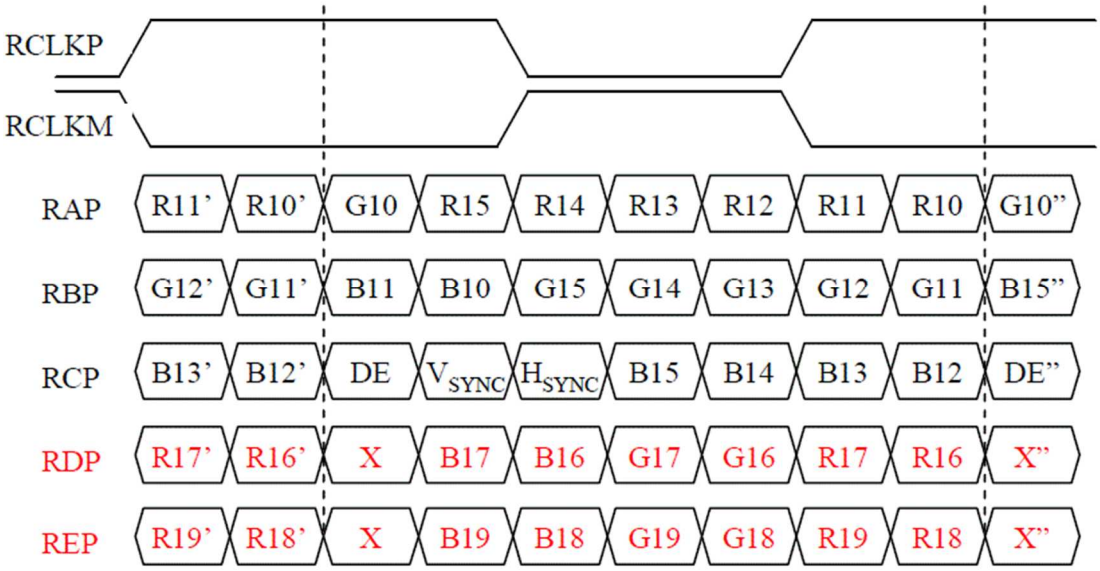


Figure 2.15 Data mapping for 10-bit VESA LVDS standard (VESA, 2008)

Figure 2.16 shows data mapping for 8-bit JEIDA interface. This scheme consists of 1 clock pair and 4 data pairs, total of 10 cables. For each clock cycle 7-bit data is transferred on each pair, in other words data signals are 7 times faster than common clock signal, and each clock cycle contains color information for each pixel. Each pixel consists of three color elements (RGB), and each color element is represented with 8-bit data. R10, G10, B10 denote least significant bit of (R10:R17) (G10:G17) (B10:B17) color data and R17, G17, B17 denote most significant bit. V_{SYNC} and H_{SYNC} denote vertical and horizontal synchronization information respectively. DE denotes data enable signal, and X denotes reserved data for future use.

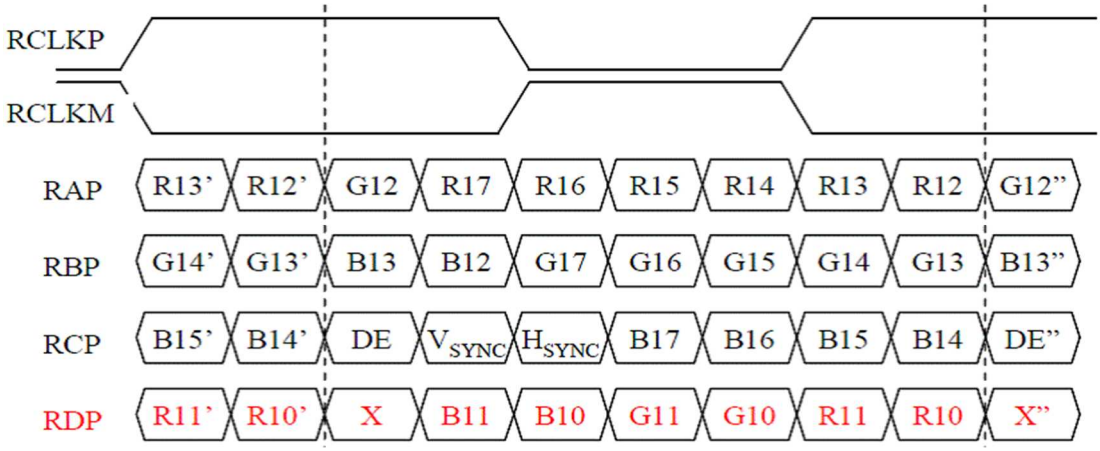


Figure 2.16 Data mapping for 8-bit JEIDA LVDS standard (JEIDA, 1999)

Figure 2.17 shows data mapping for 8-bit VESA interface. Apart from color mapping, this scheme has same properties with 8-bit JEIDA interface as in figure 2.16.

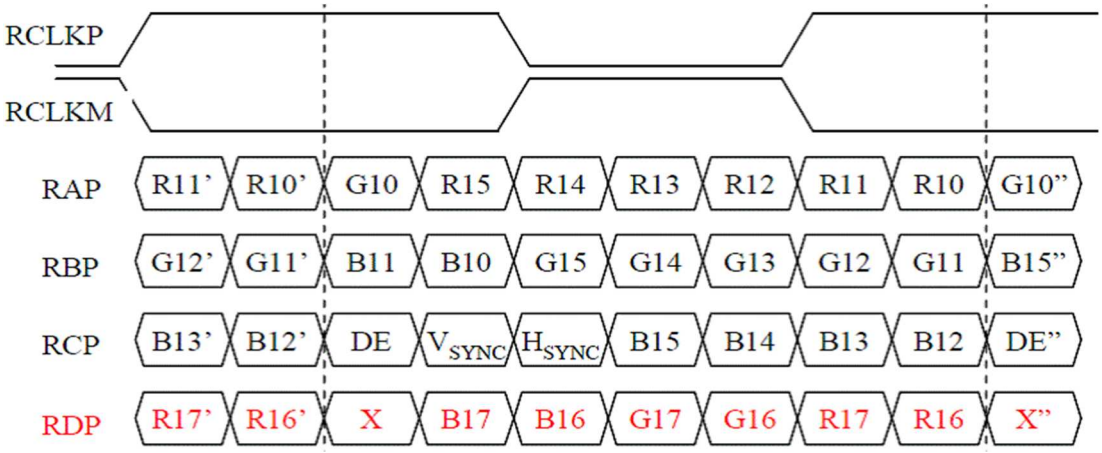


Figure 2.17 Data mapping for 8-bit VESA LVDS standard (VESA, 2008)

Various resolutions and display frequencies need different number of LVDS ports to transfer image data to TFT LCD panel as shown in table 2.2. When more than one channel is used for constitution of image, each LVDS port is responsible to carry a region of image data. Then the display panel combines each data stream from different ports and creates the image on the screen. Figure 2.18, figure 2.19 and figure 2.20 show the image formation for different number of LVDS ports utilization.

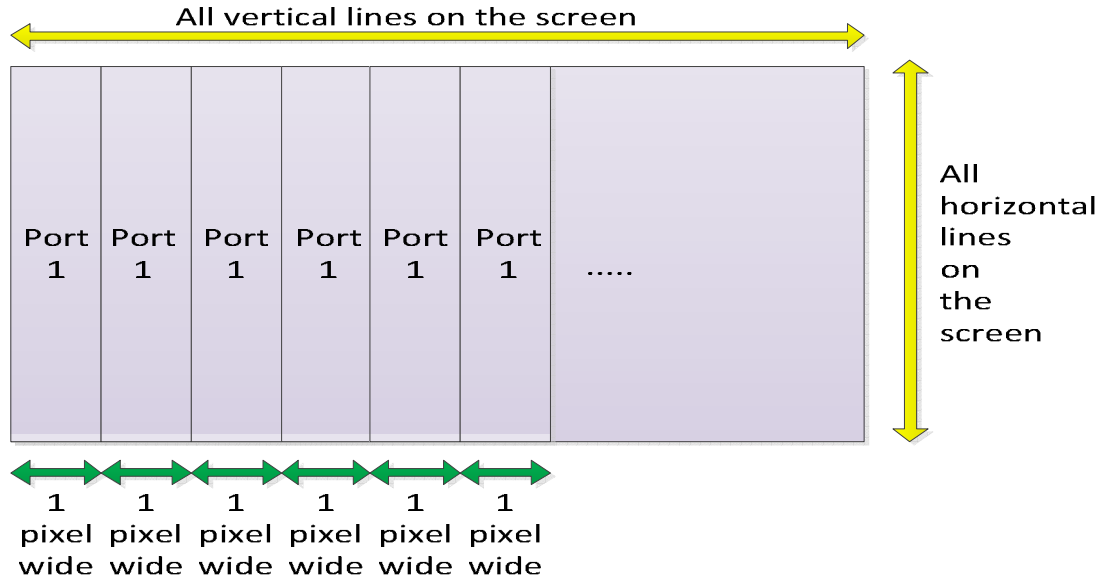


Figure 2.18 The image formation on the screen where the number of LVDS ports is equal to 1

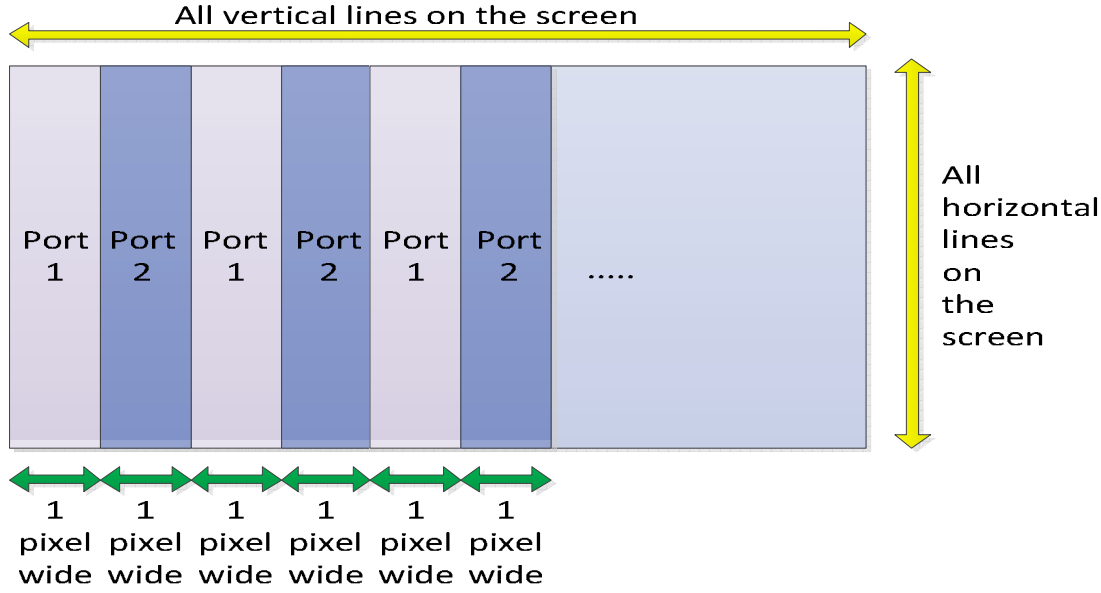


Figure 2.19 The image formation on the screen where the number of LVDS ports is equal to 2

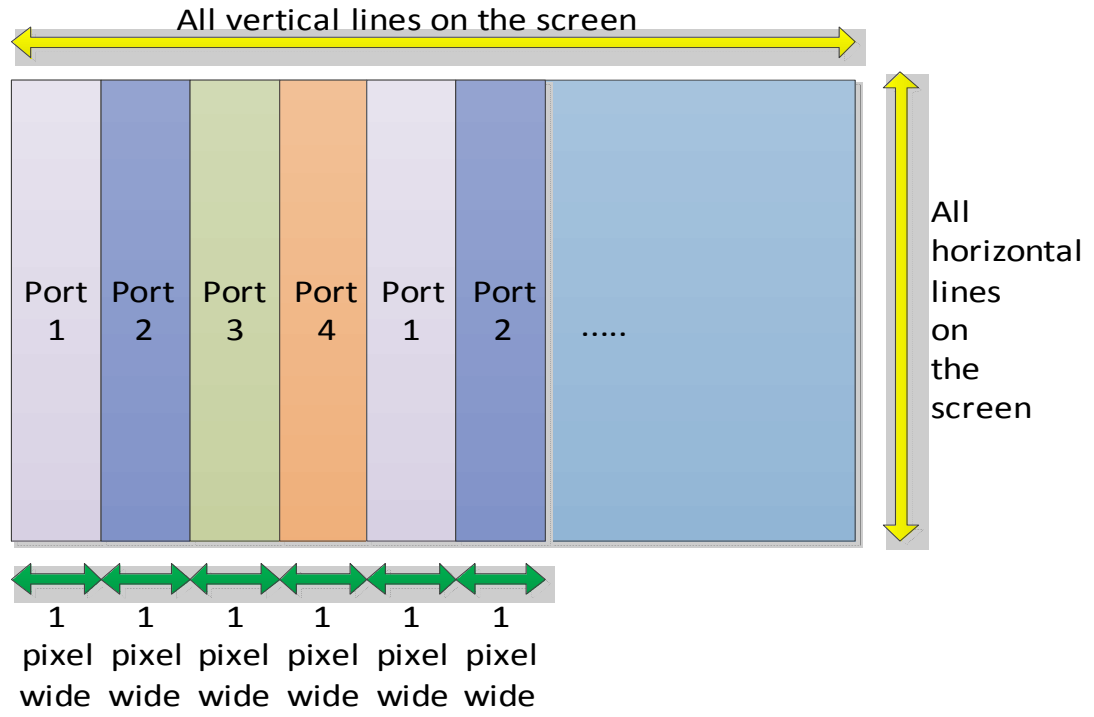


Figure 2.20 The image formation on the screen where the number of LVDS ports is equal to 4

2.6 Hardware Application: Extracting RGB Data from LVDS

Extraction of RGB (Red Green Blue) data from LVDS (Low Voltage Differential Signaling) signals constitutes the first part of the hardware design. The designed hardware processes image data in parallel like most image processing equipments.

There are two LVDS connectors on the card and each has two LVDS ports on them. In other words, the card has a 4-port LVDS input. These connectors are fed with LVDS data from a TV mainboard with a proper LVDS cable like the one in figure 2.21.

Figure 2.21 An LVDS cable connecting LVDS ouput of the TV mainboard to LVDS input of the picture grabbing card

Usually an AWG30 (American Wire Gauge) cable is used to transfer LVDS signals from TV mainboard to TFT LCD panel. Here the most important property of the cable is that each LVDS channel must be a twisted pair in order to maximize noise immunity of the cable. Otherwise blurry or scattered images can be grabbed by the acquisition card.

In the thesis, designed card has the ability of capturing 24-bit color depth images, so each LVDS port has 4 data pairs and 1 clock pair. Most PC operating systems store picture files in 24-bit RGB format, that's one of the reasons of working with 24-bit color depth, but the same card can also be used to grab 30-bit color depth images from TV mainboards, in this case the least significant 2-bit (for JEIDA standard) or the most significant 2-bit (for VESA standard) are lost for each color element.

Key electronic component of this part is the LVDS signal de-serializer. This component takes LVDS signals and transforms them into parallel data streams. For this purpose an integrated circuit from Thine Electronics, THC63LVDF84B is used. This is a 24-bit color depth LVDS receiver IC (Integrated Circuit). This IC converts the four LVDS data streams back into 28 bits of CMOS/TTL (Complementary Metal Oxide Semiconductor/Transistor Transistor Logic) data with falling edge clock. CMOS/TTL outputs contain 24-bit color data, control signals, and horizontal and vertical synchronization signals. Figure 2.22 shows the block diagram of the receiver IC. PLL (Phase Locked Loop) part of the IC locks on to incoming LVDS clock, and then successfully decodes LVDS data signals.

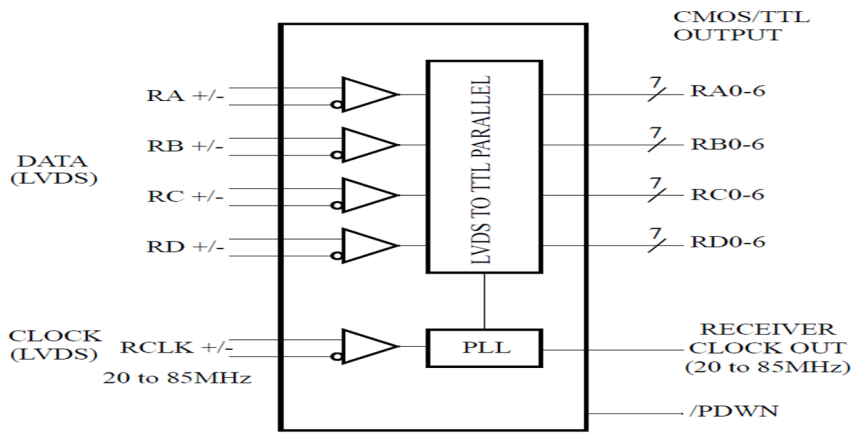


Figure 2.22 Block diagram of the LVDS receiver IC (Thine Electronics Inc., n.d.)

Figure 2.23 shows the related part of the hardware schematics drawing. Due to the high speed of LVDS, impedance matching is very important, even for very short runs. Any discontinuities in the differential LVDS traces causes signal reflections, thereby degrading the signal quality. These discontinuities also increase the common mode noise and are radiated as EMI. The LVDS outputs, being current mode outputs, need a termination resistor to close the loop and do not work without the resistor termination. The value of this termination resistor is chosen to match the differential impedance of the transmission line and as 100 Ω in this case. PWRDWN (Power Down) input of IC is used to disable receiver when not in use. D2_ [24:1] bus is used to transmit parallel RGB data from receiver to FIFO (First In First Out) memory. Although the receiver IC has horizontal and vertical synchronization outputs, these are not used in the design. Instead of these, PDE (Pixel Data Enable) and LCLK (LVDS clock) outputs are used. RGB data bus from receiver to memory is arranged according to VESA standard, by the help of shifting operations inside of microcontroller data can easily be converted to JEIDA standard.

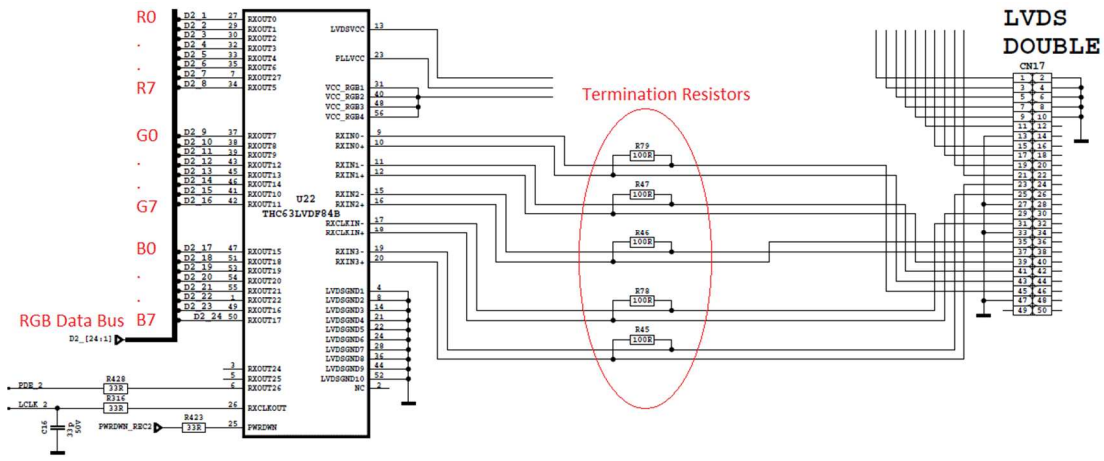


Figure 2.23 Hardware design schematic section showing LVDS receiver IC connection

TFT LCD panels are usually driven with fixed LVDS clock which is called as pixel clock. At each pixel clock, color data of one pixel is transferred. This clock value varies between 60 MHz and 85 MHz for most common panels.

As stated, panel is usually driven with a fixed frequency, but broadcasting signal standard may have a different frequency, for example PAL (Phase Alternating Line)

standard has a vertical refresh frequency of 50 Hz and NTSC standard (National Television System Committee) has 60 Hz. This vertical refresh frequency can be as low as 24 Hz for some HDMI inputs (EIA, 2002). In these cases, system on chip of TV changes number of vertical and horizontal blanking lines to match panel's pixel clock frequency. PDE pin output of the receiver IC is used to detect horizontal and vertical blanking periods. EIA/CEA-861-B standard (EIA, 2002) defines video timing requirements, discovery structures, and a data transfer structure (InfoPacket) that is used for building uncompressed, baseband, digital interfaces on digital TVs (DTV) or DTV monitors, and this standard clearly defines necessary timing parameters for different video sources and resolutions (also in VESA, 2003).

Figure 2.24 simply illustrates PDE and LCLK signals for a panel with a native resolution of 1366 (horizontal) by 768 (vertical) pixels and fed by single LVDS port (LG Display, 2008). The screen is scanned progressively.

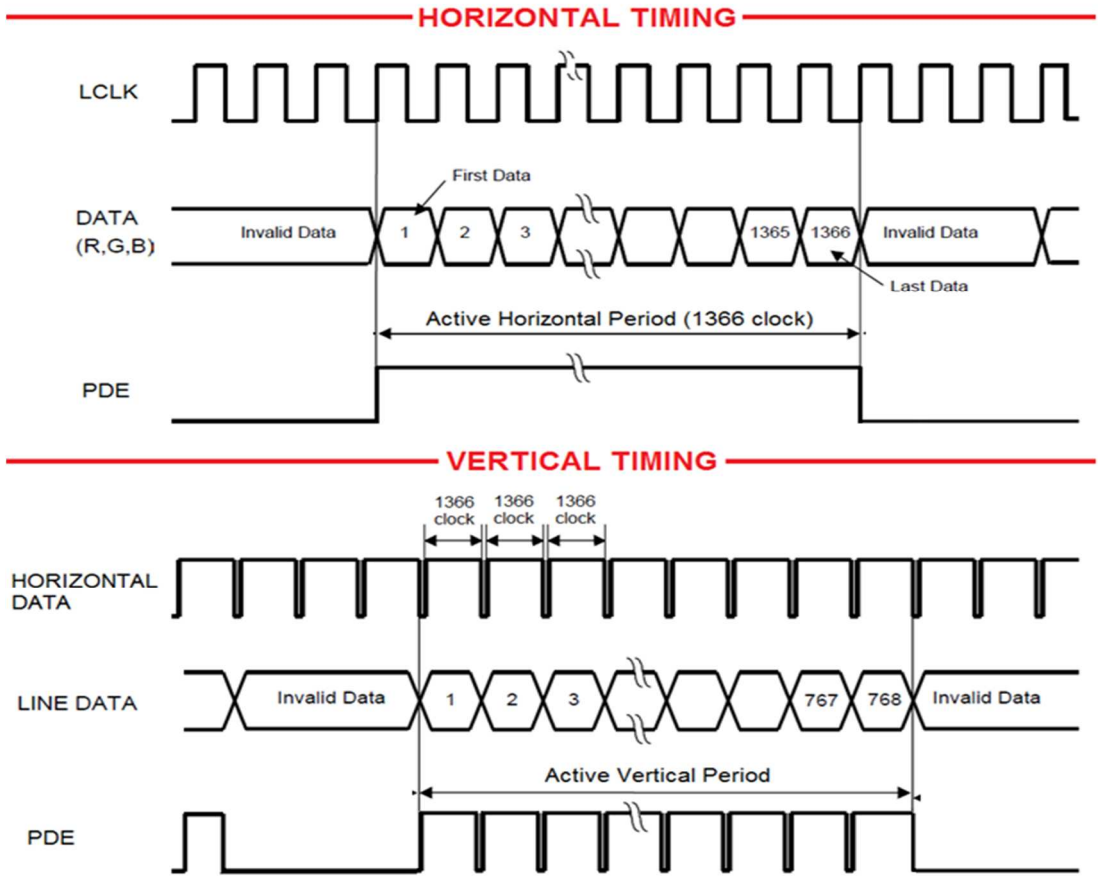


Figure 2.24 Sample timing diagram for a 1366 by 768 LCD panel showing horizontal and vertical timing details

Figure 2.25 shows the LVDS connector section together with receiver ICs on PCB design. While designing PCB of the hardware schematics, the following rules (Altera, 2000) are followed to overcome any performance issues related to LVDS lines:

- To ensure minimal reflections and maintain the receiver's common mode noise rejection, differential traces are run as closely as possible after they leave LVDS connector. Also, to avoid discontinuities in the differential impedance, the distance between the differential LVDS signals remain constant over the entire length of the traces.
- To minimize skew, the electrical lengths between the differential LVDS traces are same. Arrival of one of the signals before the other creates a phase difference between the signal pair, which impairs the system performance by reducing the available receiver skew margin.
- No vias are used on the signal path to minimize signal discontinuity.
- To avoid signal discontinuities, arcs or 45° traces are used instead of 90° turns.
- All high speed signal lines including LVDS lines are impedance controlled traces whose typical impedances are 100 Ω.
- For maximum performance LVDS termination resistors must be placed to receiver IC as close as possible, but for this design other components prevent them to be placed closely to receiver IC. Although this is not a preferable action, current signaling rate and relatively short distance of LVDS lines can permit them to be placed closely to connector side (far side) in this design.

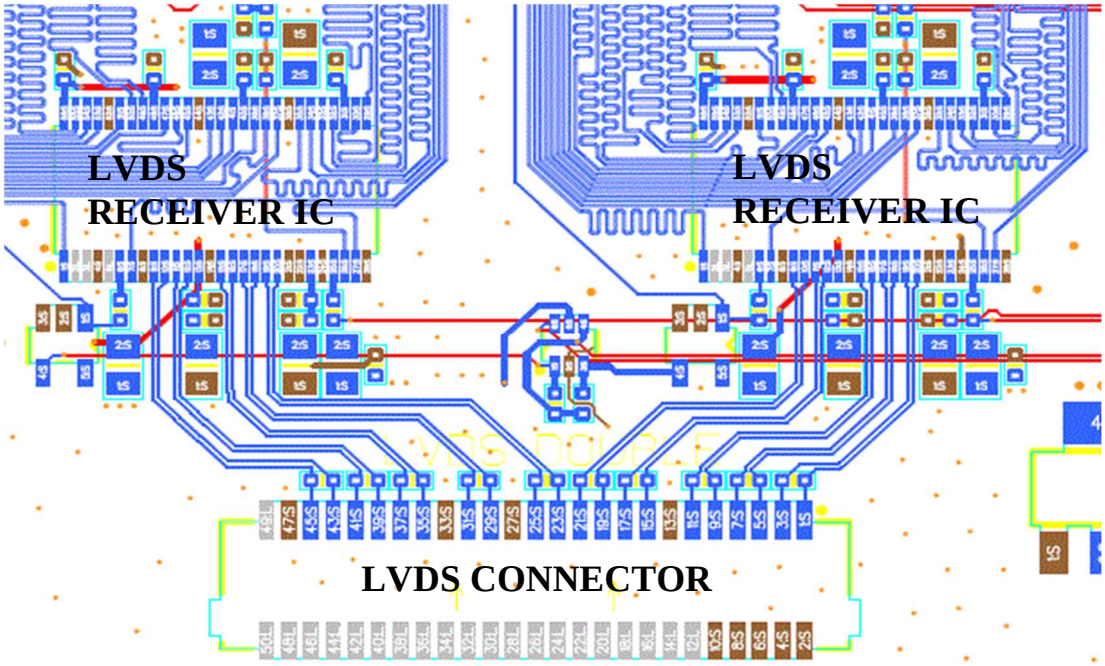


Figure 2.25 PCB design section showing LVDS receiver IC connection

Figure 2.26 shows the LVDS receiver part of the design on PCB after auto insertion of components.

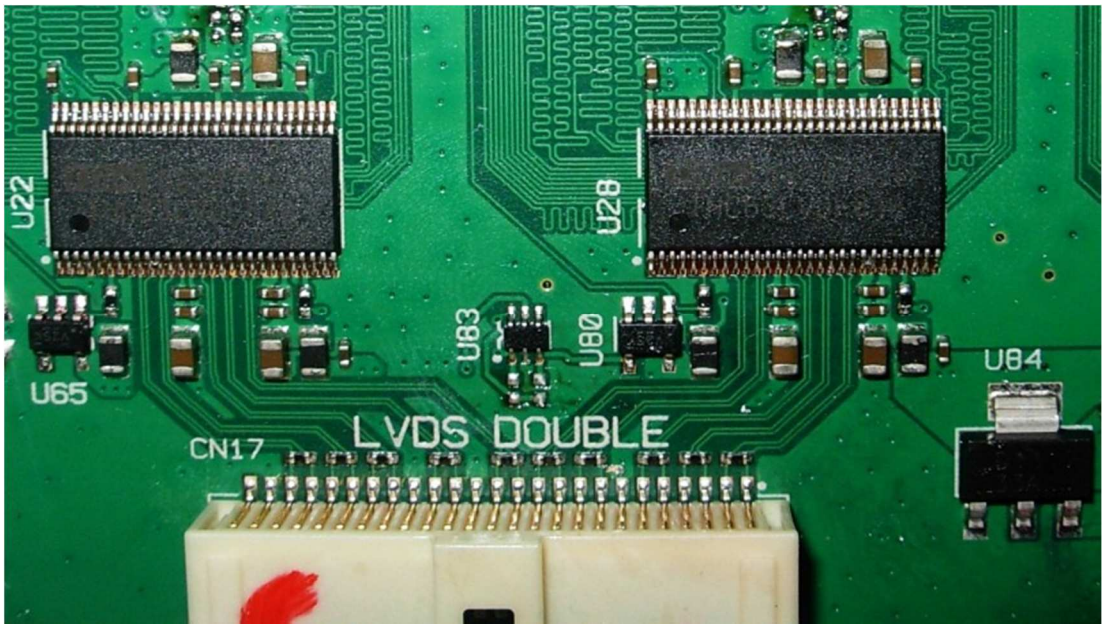


Figure 2.26 A hardware section from picture grabbing card showing LVDS receiver IC connection

CHAPTER THREE
DMA AND DATA BUFFERS

3.1 What is DMA?

In computer-based data acquisition applications, data incoming or outgoing through computer I/O (Input/Output) devices must often be managed at high speeds or in large quantities. The three primary data transfer mechanisms for computer-based data acquisition are polling, interrupts (also known as programmed I/O) and DMA (Direct Memory Access). Polling is a form of foreground data acquisition in which the processor is dedicated to acquiring the incoming data, often by waiting in a loop. The main program calls an acquisition subroutine that waits until the processor collects the required data. With interrupts, the processor is periodically interrupted from executing the main program to store incoming data in a buffer for later retrieval and processing. Interrupts are a form of background acquisition because the main program contains no code that reads data from the input device. Instead, the processor is invisibly stolen periodically from the main program to perform this function. With DMA, a dedicated data transfer device reads incoming data from a device and stores that data in a system memory buffer for later retrieval by the processor. This DMA process occurs transparently from the processor's point of view (Corbet & Rubini, 2001).

DMA has several advantages over polling and interrupts. DMA is fast because a dedicated piece of hardware transfers data from one location to another and only one or two bus read/write cycles are required per piece of data transferred. In addition, DMA is usually required to achieve maximum data transfer speed, and thus is useful for high speed data acquisition devices. DMA also minimizes latency in servicing a data acquisition device because the dedicated hardware responds more quickly than interrupts, and transfer time is short. Minimizing latency reduces the amount of temporary storage (memory) required on an I/O device. DMA also off-loads the processor, which means the processor does not have to execute any instructions to transfer data. Therefore, the processor is not used for handling the data transfer

activity and is available for other processing activity. Also, in systems where the processor primarily operates out of its cache, data transfer is actually occurring in parallel, thus increasing overall system utilization (Corbet & Rubini, 2001).

An embedded processor core as in the case of this thesis is capable of doing multiple operations in a single cycle, including calculations, data fetches, data stores and pointer increments/decrements. In addition, the core can orchestrate data transfer between internal and external memory spaces by moving data into and out of the register file. In reality, optimum performance can only be achieved in an application if data can move around without constantly bothering the core to perform the transfers. This is where a DMA controller comes into play. Processors need DMA capability to relieve the core from these transfers between internal/external memory and peripherals, or between memory spaces (Katz & Gentile, 2007).

There are two main types of DMA controllers. "Cycle-stealing" DMA uses spare (idle) core cycles to perform data transfers. This is not a workable solution for systems with heavy processing loads like multimedia flows. Instead, it is much more efficient to employ the second type: a DMA controller that operates independently from the core (Corbet & Rubini, 2001).

Imagine if a processor's video port has a FIFO that needs to be read every time a data sample is available. In this case, the core has to be interrupted tens of millions of times each second. As if that's not disruptive enough, the core has to perform an equal amount of writes to some destination in memory. For every core processing cycle spent on this task, a corresponding cycle would be lost in the processing loop. That situation clearly indicates the need for DMA.

3.2 Use of DMA for High Speed Data Transfer

A DMA (Direct Memory Access) controller is a unique peripheral devoted to moving data around a system. It can be thought as a controller that connects internal and external memories with each DMA-capable peripheral via a set of dedicated

buses. It is a peripheral in the sense that the processor programs it to perform transfers.

It is unique in that it interfaces to both memory and selected peripherals. Notably, only peripherals where data flow is significant (Megabytes per second or greater) need to be DMA-capable. Good examples of these are video, audio and network interfaces. Lower-bandwidth peripherals can also be equipped with DMA capability, but it's less of an imposition on the core to step in and assist with data transfer on these interfaces.

In general, DMA controllers include an address bus, a data bus, and control registers. An efficient DMA controller possesses the ability to request access to any resource it needs, without having the processor itself get involved (Harvey, 1991). It must have the capability to generate interrupts. Finally, it has to be able to calculate addresses within the controller.

Each DMA controller has a set of FIFOs that act as a buffer between the DMA subsystem and peripherals or memory. For Memory DMA, a FIFO exists on both the source and destination sides of the transfer. The FIFO improves performance by providing a place to hold data while busy resources are preventing a transfer from completing.

A DMA controller is typically configured during code initialization, the core should only need to respond to interrupts after data set transfers are complete. The DMA controller can be programmed to move data in parallel with the core, while the core is doing its basic processing tasks, the jobs on which it's supposed to be focused.

In an optimized application, the core would never have to move any data, but rather only access it in its cache. The core wouldn't need to wait for data to arrive, because the DMA engine would have already made it available by the time the core was ready to access it. Figure 3.1 shows a typical interaction between the processor and the DMA controller. The steps allocated to the processor involve setting up the

transfer, enabling interrupts, and running code when an interrupt is generated. The interrupt input back to the processor can be used to signal that data is ready for processing.

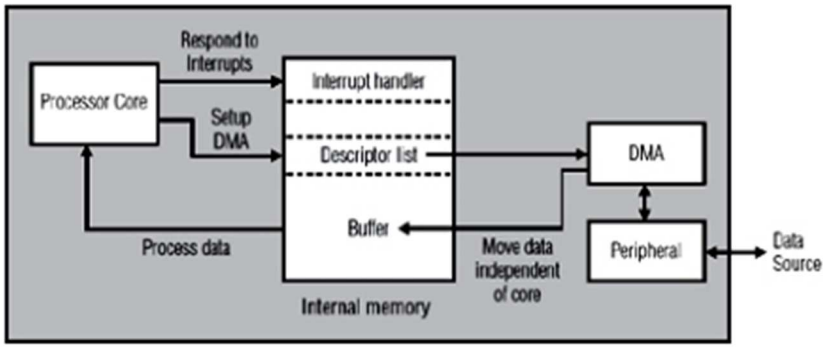


Figure 3.1 A typical interaction between the processor and the DMA controller (Katz & Gentile, 2007)

Figure 3.2 shows some typical DMA data flows. First one (a) illustrates data flow from memory to peripheral device, the second one (b) illustrates data flow from peripheral device to memory, and the third one (c) illustrates data flow from memory to memory, so called memory DMA.

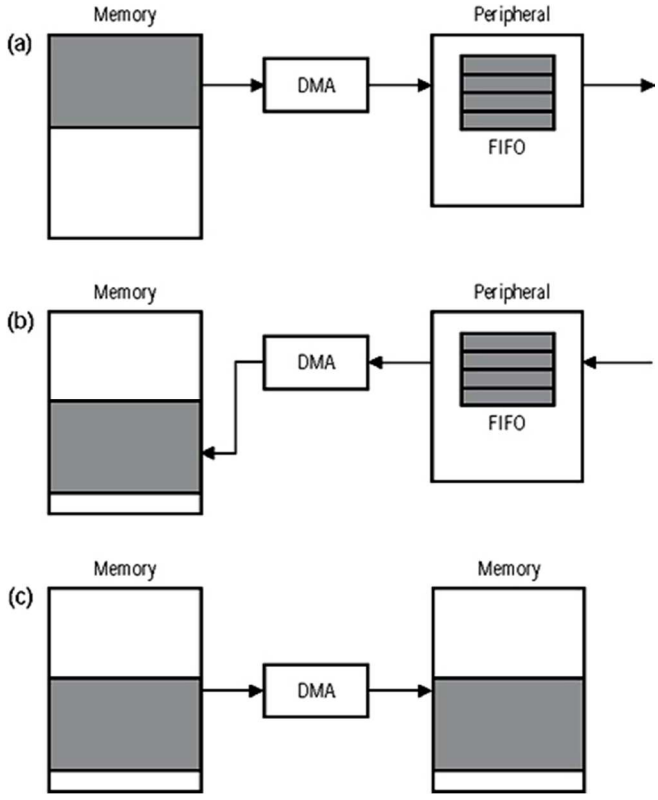


Figure 3.2 Representation of some typical DMA data flows (Katz & Gentile, 2007)

For any type of DMA transfer, it is always needed to specify a starting source and destination address for data. In the case of a peripheral DMA, the peripheral's FIFO serves as either the source or the destination. When the peripheral serves as the source, a memory location (internal or external) serves as the destination address. When the peripheral serves as the destination, a memory location (internal or external) serves as the source address (Katz & Gentile, 2007).

In the simplest memory DMA case, it is needed to tell the DMA controller the source address, the destination address and the number of words to transfer. With a peripheral DMA, either the source or the destination is specified, depending on the direction of the transfer. The word size of each transfer can be 8, 16 or 32 bits. This type of transaction represents a simple one-dimensional (1D) transfer with a unity "stride."

As part of this transfer, the DMA controller keeps track of the source and destination addresses as they increment. With a unity stride, the address increments by 1 byte for 8-bit transfers, 2 bytes for 16-bit transfers, and 4 bytes for 32-bit transfers. The above parameters configure a basic 1D DMA transfer. Figure 3.3 shows an example for this transfer (a) unity stride transfer and (b) non-unity stride transfer.

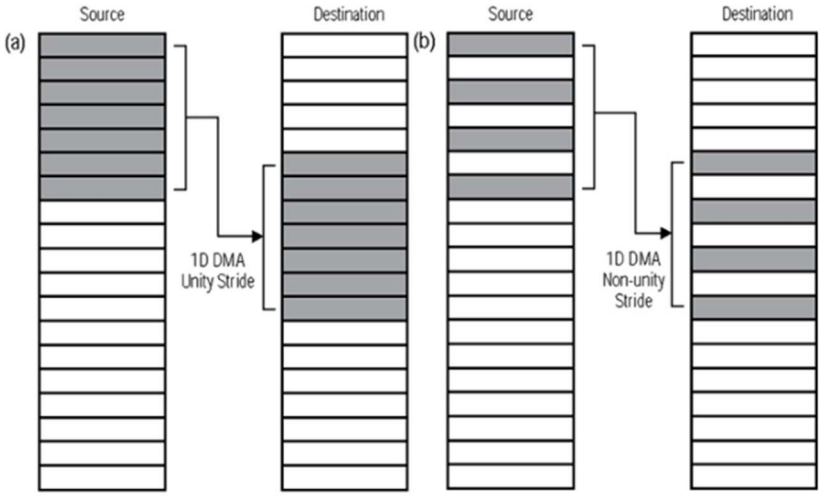


Figure 3.3 Representation of 1D DMA data transfer (Katz & Gentile, 2007)

More flexibility can be added to a one-dimensional DMA simply by changing the stride. For example, with non-unity strides, addresses in multiples of the transfer sizes can be skipped. That is, specifying a 32-bit transfer and striding by 4 samples results in an address increment of 16 bytes (four 32-bit words) after each transfer.

For a peripheral DMA, the "memory side" of the transfer can be either 1D or 2D. On the peripheral side, though, it is always a 1D transfer. The only constraint is that the total number of bytes transferred on each side (source and destination) of the DMA has to be the same. For example, if a peripheral was fed from three 10-byte buffers, the peripheral would have to be set to transfer 30 bytes using any possible combination of supported transfer width and transfer count values available.

Memory DMA offers a bit more flexibility. For example, one can set up (a) a 1D-to-1D transfer, (b) a 1D-to-2D transfer, (c) a 2D-to-1D transfer, and of course (d) a 2D-to-2D transfer, as shown in figure 3.4. The only constraint is that the total number of bytes being transferred on each end of the DMA transfer block has to be the same.

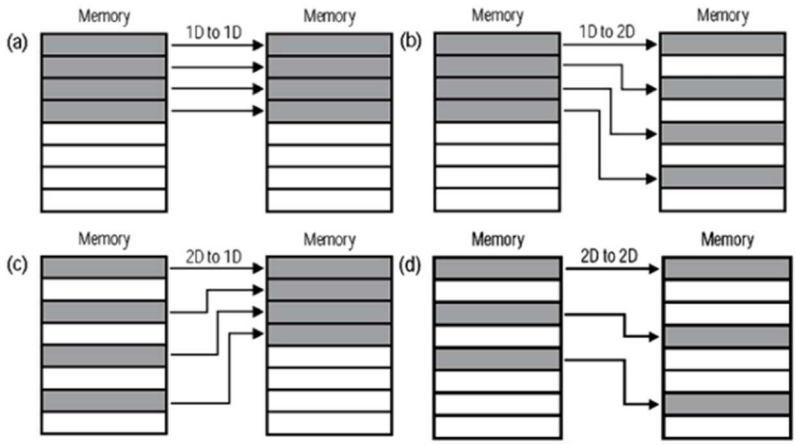


Figure 3.4 Representation of different types of memory to memory DMA data transfers (Katz & Gentile, 2007)

3.3 Line and Frame Buffering for Temporal Data Storage

Most audio/video processing equipments make use of DMA to maximize data transfer speed between various locations. The hardware designed in this thesis also

utilize DMA to enable a low speed microcontroller to interface a high speed data receive port.

Use of DMA requires an external fast memory apart from internal microcontroller memories, and every LVDS port has its own external memory on the system. In this configuration a peripheral to memory DMA is realized. LVDS output of TV mainboard is modeled as a peripheral; the data source and system memory on the designed hardware is the memory; the data sink.

These fast memory ICs are used for temporal image data storage, because incoming RGB data speed from receiver IC (180 Megabytes/sec for 1366 by 768 24-bit color depth panel) is far more beyond that microprocessor's data fetching speed. Even data is read by a high speed microcontroller it is impossible to transfer that data at that speed via Ethernet which has maximum 100 Mbps data bandwidth. In the thesis it is aimed to transmit a complete picture of the screen (1 frame), so these memory ICs must be able store one LVDS port data for later retrieval of that frame by microprocessor.

TFT LCD screen is scanned progressively, in other words, first incoming pixel data for each frame belongs to top-left corner of the screen and last incoming pixel data for each frame belongs to bottom-right corner of the screen. So, this incoming RGB data stream can be pushed into the memory, and then this memory is read by microcontroller from the beginning to constitute the image. Hence the resolution is known, microcontroller knows how much data it has to read. The memory ICs can be called as FIFOs due to their functions on the system.

Various panel resolutions and panel display frequencies need different number of FIFOs and FIFO organizations. Figure 3.5 and figure 3.6 show FIFO organizations for some common TV mainboard – LCD panel pairs.

Figure 3.5 shows FIFO organization for a 1366 by 768 panel whose color depth is 24-bit and has a vertical refresh frequency of 60 Hz. Here, LVDS data is transmitted over only one (first) port.

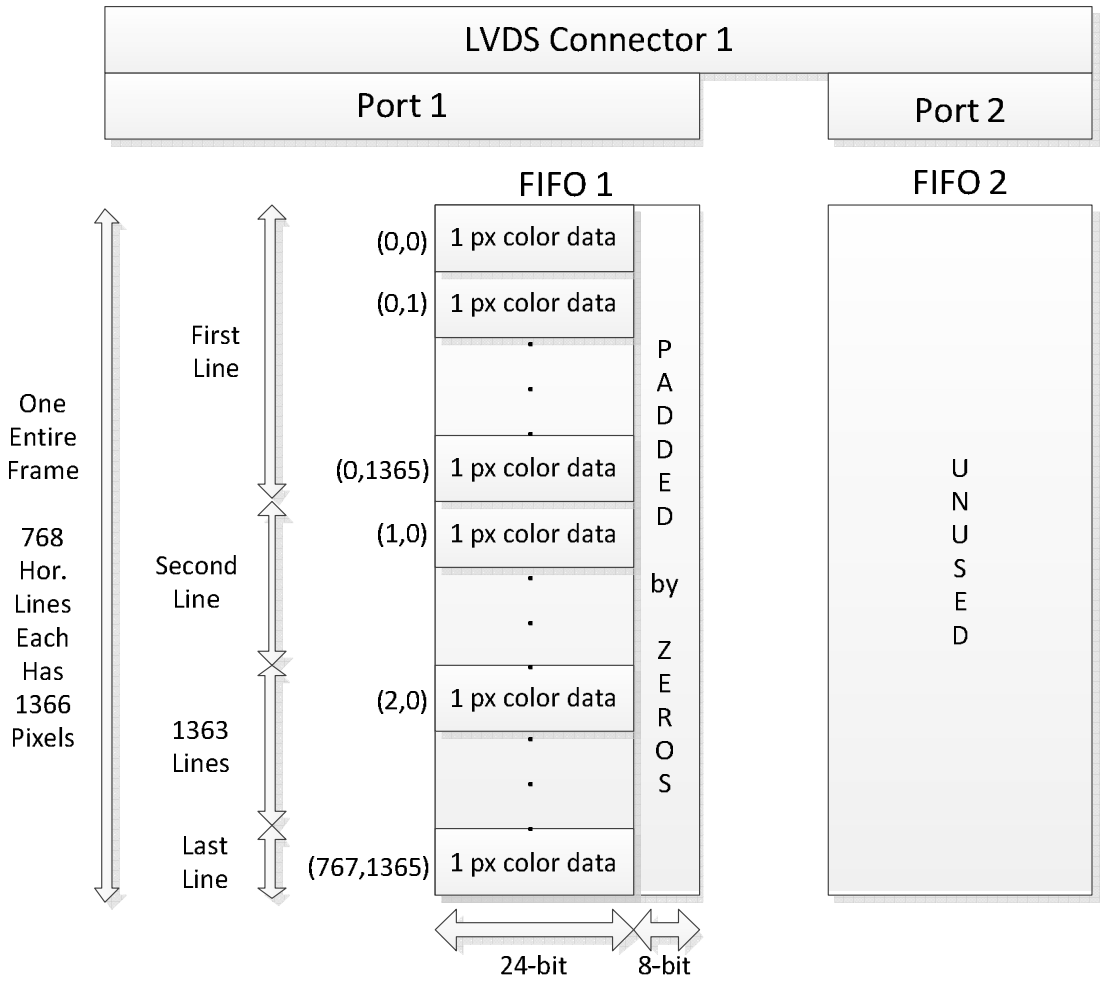


Figure 3.5 FIFO organization for a 1366 by 768 panel whose color depth is 24-bit

Figure 3.6 shows FIFO organization for a 1920 by 1080 panel whose color depth is 24-bit and has a vertical refresh frequency of 60 Hz. Here, LVDS data is transmitted over two ports, in other words first LVDS connector is fully utilized.

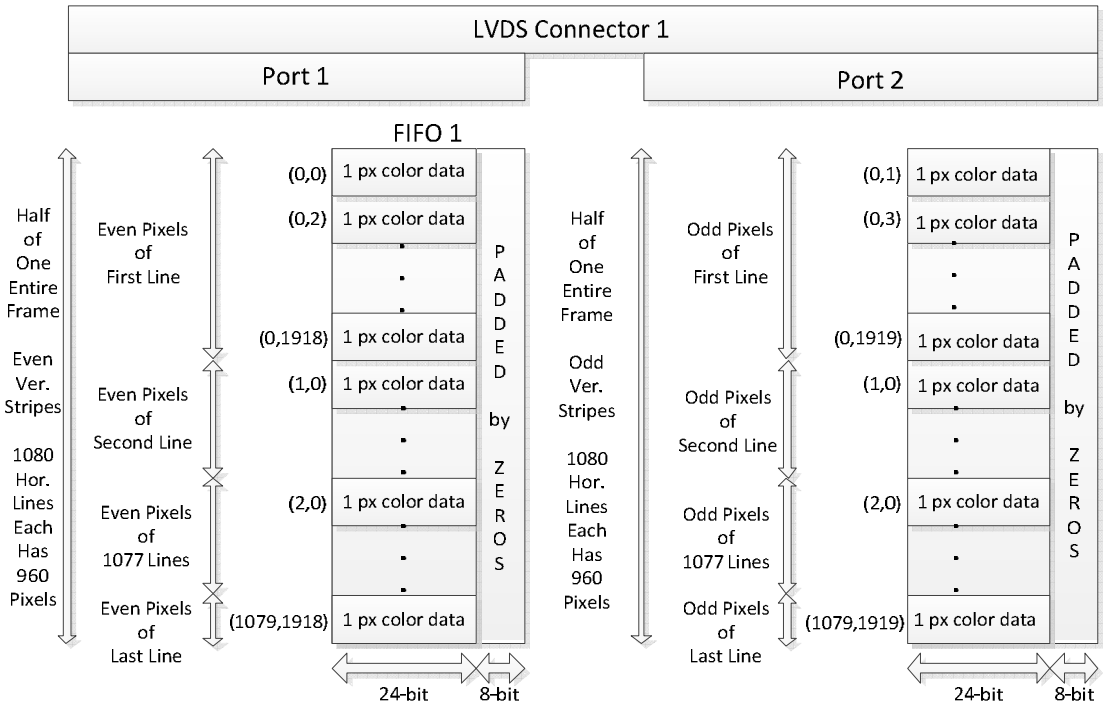


Figure 3.6 FIFO organization for a 1920 by 1080 panel whose color depth is 24-bit

Other FIFO configurations can be derived in same way for different TV mainboard – LCD panel configurations. For example FIFO organization for a 1920 by 1080 panel whose color depth is 24-bit and has a vertical refresh frequency of 100 Hz uses 4 LVDS ports. This configuration requires 4 FIFOs, and each FIFO is filled with one quarter of entire image data.

As conclusion, FIFOs serve as a frame buffer for incoming image data to store them temporarily.

3.4 Hardware Application: ARM Microcontroller and Custom DMA

The center of the hardware design in this thesis is the microcontroller. In point of system requirements embedded microcontroller must have the following properties:

- Ethernet communication for fast data transfer to a PC
- In system programming to program the microcontroller without removing it
- Large number of I/O pins to accommodate large data bus and a number of different control lines

- UART (Universal Asynchronous Receiver Transmitter) for easy debugging of the system without having a debugger
- JTAG (Joint Test Action Group) interface for line by line code debugging
- At least two timers to keep track of various properties of incoming image data (like sync signals)
- Easy code generation using C to rapidly develop a running software on the microcontroller
- Availability at market to easily reach the required number of microcontrollers via different distributors
- A fair processing speed to achieve a reasonable image capturing period
- And a reasonable price to keep the hardware cost at minimum level

Some different microcontroller families such as ARM (Advanced RISC Machines), Xscale, and Blackfin etc. can meet these requirements. In this point an ARM architecture based microcontroller, NXP LPC2468 microcontroller is selected. This microcontroller satisfies all requirements listed above.

Although LPC2468 is a highly capable microcontroller and a simpler microcontroller can be utilized for the current design, it is readily available in Vestel Electronics' warehouse and schematics library to speed up the board assembly process.

The ARM processor core is a key component of many successful 32-bit embedded systems. ARM cores are widely used in mobile phones, handheld organizers, and a multitude of other everyday portable consumer devices. In fact, the ARM core is not a single core, but a whole family of designs sharing similar design principles and a common instruction set. For example, one of ARM's most successful cores is the ARM7TDMI. It provides up to 120 Dhrystone MIPS (Mega Instruction Per Second) (Dhrystone MIPS is a small benchmarking program) and is known for its high code density and low power consumption, making it ideal for mobile embedded devices (Sloss, Symes, & Wright, 2004).

The ARM core uses RISC (Reduced Instruction Set Computing) architecture. RISC is a design philosophy aimed at delivering simple but powerful instructions that execute within a single cycle at a high clock speed. The RISC philosophy concentrates on reducing the complexity of instructions performed by the hardware because it is easier to provide greater flexibility and intelligence in software rather than hardware. As a result, a RISC design places greater demands on the compiler.

In contrast, the traditional complex instruction set computer (CISC) relies more on the hardware for instruction functionality, and consequently the CISC instructions are more complicated. Figure 3.7 illustrates these major differences.

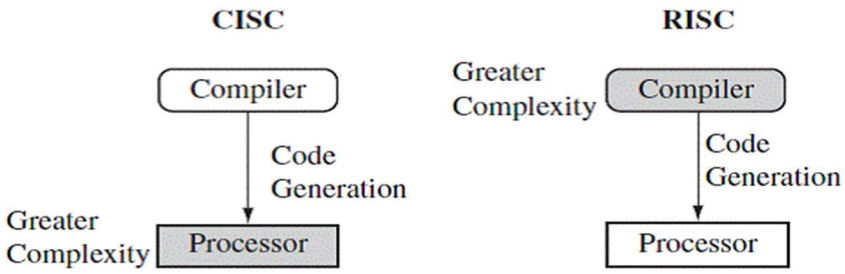


Figure 3.7 Comparison of CISC and RISC architectures (Sloss & others, 2004)

There are a number of physical features that have driven the ARM processor design. First, portable embedded systems require some form of battery power. The ARM processor has been specifically designed to be small to reduce power consumption and extend battery operation-essential for applications such as mobile phones and personal digital assistants (PDAs). Although this is not a critical requirement in thesis work, a low power microcontroller means less heat production during the operation and relaxes power block design during the development.

High code density is another major requirement since embedded systems have limited memory due to cost and/or physical size restrictions. High code density is useful for applications that have limited on-board memory, such as mobile phones and mass storage devices. This is the most important advantage of the LPC2468 in this thesis work, since the implementation of TCP/IP (Transmission Control Protocol/Internet Protocol) stack in a microcontroller requires a large number of lines

coding. Although a limited version of the stack is implemented in the thesis the microcontroller can host even a dynamic web page and act as a web server.

In addition, embedded systems are price sensitive and use slow and low-cost memory devices. For high-volume applications like digital cameras, every cent has to be accounted for in the design. The ability to use low-cost memory devices produces substantial savings (Martin, 2007).

Another important requirement is to reduce the area of the die taken up by the embedded processor. For a single-chip solution, the smaller the area used by the embedded processor, the more available space for specialized peripherals. This in turn reduces the cost of the design and manufacturing since fewer discrete chips are required for the end product (Sloss & others, 2004).

ARM has incorporated hardware debug technology within the processor so that software engineers can view what is happening while the processor is executing code. With greater visibility, software engineers can resolve issues faster, which have a direct effect on the time to develop and reduces overall development costs.

The ARM core is not a pure RISC architecture because of the constraints of its primary application-the embedded system. In some sense, the strength of the ARM core is that it does not take the RISC concept too far. In today's systems the key is not raw processor speed but total effective system performance and power consumption (Martin, 2007).

Figure 3.8 shows a typical embedded device based on an ARM core. Each box represents a feature or function. The lines connecting the boxes are the buses carrying data. The device can be separated into four main hardware components:

- The ARM processor controls the embedded device. Different versions of the ARM processor are available to suit the desired operating characteristics. An ARM processor comprises a core (the execution engine that processes instructions and manipulates data) plus the surrounding components that

interface it with a bus. These component scan include memory management and caches.

- Controllers coordinate important functional blocks of the system. Two commonly found controllers are interrupt and memory controllers.
- The peripherals provide all the input-output capability external to the chip and are responsible for the uniqueness of the embedded device.
- A bus is used to communicate between different parts of the device.

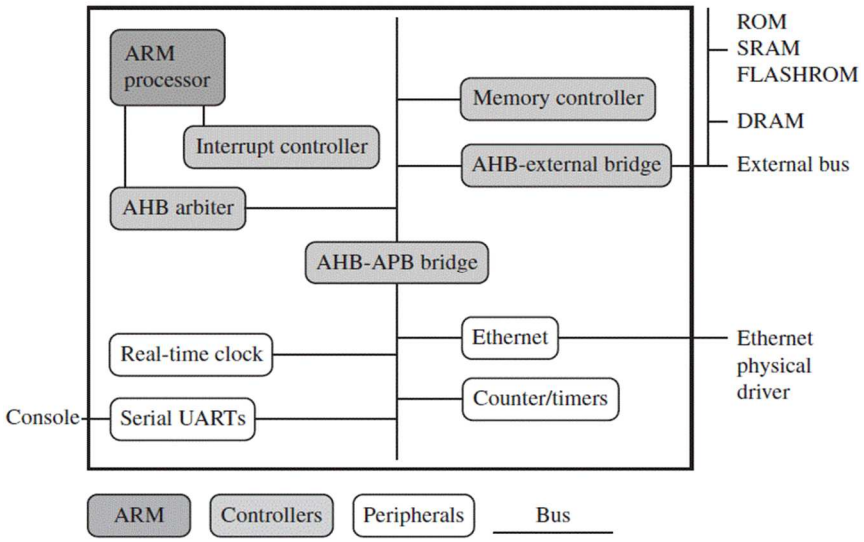


Figure 3.8 Block diagram of a typical embedded device based on an ARM core (Sloss & others, 2004)

The LPC2468 microcontroller consists of an ARM7TDMI-S CPU (Central Processing Unit) with emulation support, the ARM7 local bus for closely coupled, high speed access to the majority of on-chip memory, the AMBA (Advanced Microcontroller Bus Architecture) AHB (ARM High Performance Bus) interfacing to high speed on-chip peripherals and external memory, and the AMBA APB (ARM Peripheral Bus) for connection to other on-chip peripheral functions. The microcontroller permanently configures the ARM7TDMI-S processor for little-endian byte order.

The LPC2468 implements two AHB buses in order to allow the Ethernet block to operate without interference caused by other system activity. The primary AHB, referred to as AHB1, includes the VIC (Vector Interrupt Controller), GPDMA

(General Purpose Direct Memory Access) controller, and EMC (External Memory Controller).

AHB2 includes only the Ethernet block and an associated 16 kB (kiloByte) SRAM (Static Random Access Memory). In addition, a bus bridge is provided that allows AHB2 to be a bus master on AHB1, allowing expansion of Ethernet buffer space into off-chip memory or unused space in memory residing on AHB1.

In summary, bus masters with access to AHB1 are the ARM7 itself, the GPDMA function, and the Ethernet block (via the bus bridge from AHB2). Bus masters with access to AHB2 are the ARM7 and the Ethernet block.

AHB peripherals are allocated a 2 MB (Megabyte) range of addresses at the very top of the 4 GB (Gigabyte) ARM memory space. Each AHB peripheral is allocated a 16 kB address space within the AHB address space. Lower speed peripheral functions are connected to the APB bus. The AHB to APB bridge interfaces the APB bus to the AHB bus. APB peripherals are also allocated a 2 MB range of addresses, beginning at the 3.5 GB address point. Each APB peripheral is allocated a 16 kB address space within the APB address space.

Pipeline techniques are employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

The ARM7TDMI-S processor also employs a unique architectural strategy known as Thumb, which makes it ideally suited to high-volume applications with memory restrictions, or applications where code density is an issue.

The key idea behind Thumb is that of a super-reduced instruction set. Essentially, the ARM7TDMI-S processor has two instruction sets:

- The standard 32-bit ARM set
- A 16-bit Thumb set

The Thumb set's 16-bit instruction length allows it to approach higher density compared to standard ARM code while retaining most of the ARM's performance.

Although LPC2468 has many features, some key specifications of it regarding the hardware design phase of the thesis are listed below (NXP, 2008) together with table 3.1, and figure 3.9 shows the block diagram of LPC2468.

- ARM7TDMI-S processor, running at 72 MHz of core clock (in the thesis, it also runs at 72 MHz)
- 64 kB of SRAM on the ARM local bus for high performance CPU access
- 16 kB SRAM for Ethernet interface
- 512 kB on-chip Flash program memory with In-System Programming (ISP) and In-Application Programming (IAP) capabilities. Flash program memory is on the ARM local bus for high performance CPU access
- Dual AHB system allows memory access by multiple resources and simultaneous program execution with no contention
- VIC, supporting up to 32 vectored interrupts
- Ethernet MAC (Media Access Controller) with MII (Media Independent Interface) /RMII (Reduced Media Independent Interface)
- Four UARTs with fractional baud rate generation
- 160 general purpose I/O pins with configurable pull-up/down resistors
- Four general purpose timers/counters with 8 capture inputs and 10 compare outputs. Each timer block has an external count input
- Watchdog Timer (WDT)
- Standard ARM test/debug interface for compatibility with existing tools
- Emulation trace module supports real-time trace
- Single 3.3 V power supply (3.0 V to 3.6 V)
- Four external interrupt inputs configurable as edge/level sensitive. All pins on PORT0 and PORT2 can be used as edge sensitive interrupt sources
- On-chip power-on reset
- On-chip PLL allows CPU operation up to the maximum CPU rate without the need for a high frequency crystal

- Versatile pin function selections allow more possibilities for using on-chip peripheral functions

Table 3.1 Summary of the specifications for LPC2468 (NXP, 2008)

Type number	Flash (kB)	SRAM (kB)					External bus	Ethernet	USB OTG/ OHC/ DEV + 4 kB FIFO	CAN channels	SD/ MMC	GP DMA	ADC channels	DAC channels	Temp range
		Local bus	Ethernet buffer	GP/USB	RTC	Total									
LPC2468FBD208	512	64	16	16	2	98	Full 32-bit	MII/ RMII	yes	2	yes	yes	8	1	–40 °C to +85 °C

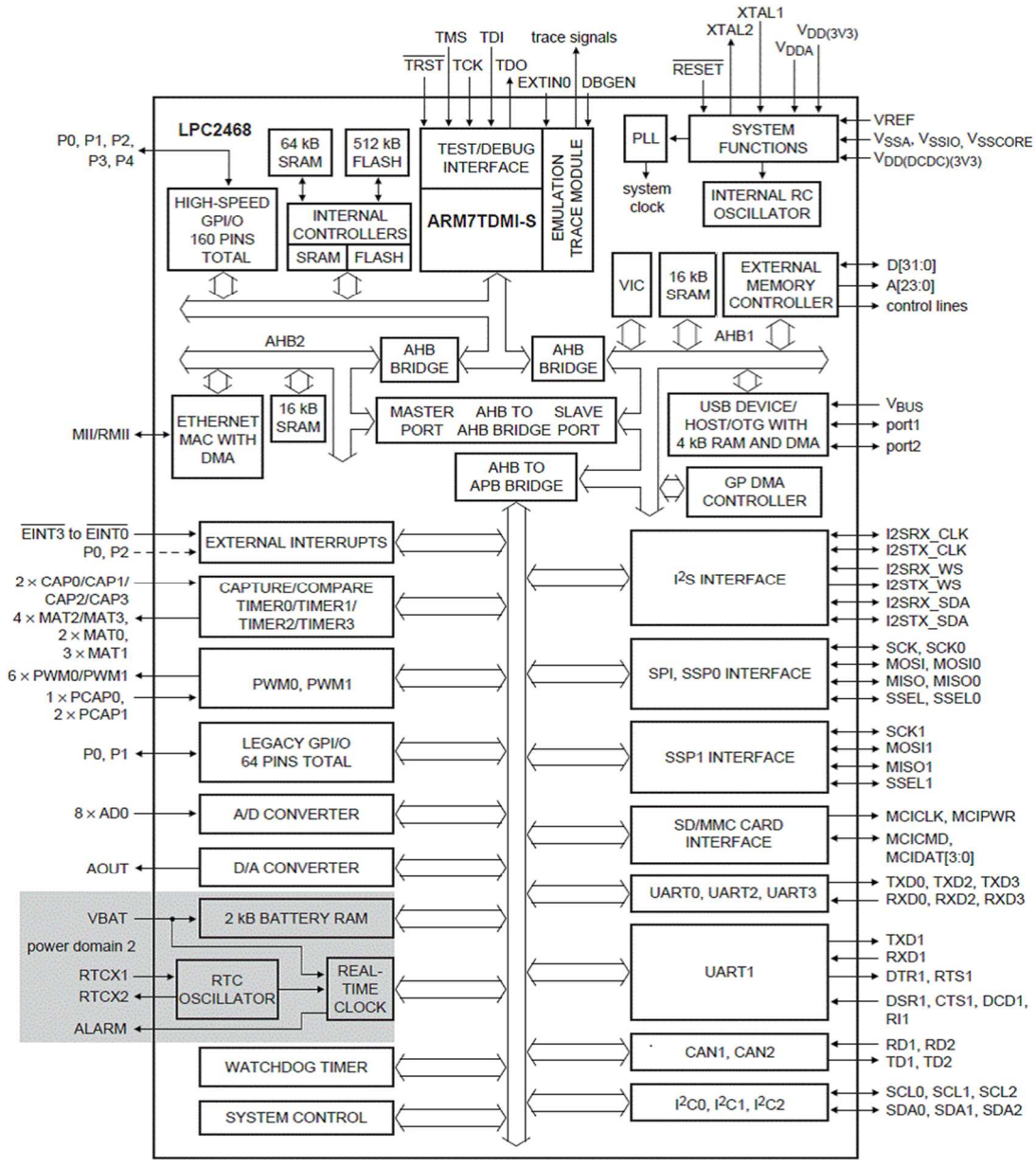


Figure 3.9 System block diagram of LPC2468 (NXP, 2008)

Custom DMA (Direct Memory Access) implementation constitutes the second part of the hardware design. In the previous hardware design section (Section 2.6) color data was taken from LVDS receiver in parallel, in this part it is stored in a temporary memory (FIFO) and is fetched by the microcontroller to be processed.

There are four FIFOs on the board, and their utilization is configured according to input LVDS signal, so one, two or four of them are controlled at a time.

Key electronic component of this part is the FIFO memory. The FIFO memory must be fast enough to catch up the speed of incoming RGB data bus, because FIFO memory is filled with data without microcontroller intervention. The core clock speed of the low cost microcontroller is 72 MHz, but pixel clock output of the receiver IC can be as high as 85 MHz, so it is impossible that the microcontroller can fetch the data inside its internal RAM or an external RAM. In this case the only solution is to use DMA.

The MS81V32322 from OKI Semiconductor Industry is specifically selected for DMA purposes. The OKI MS81V32322 is a high performance 32 megabit, 1088K x 32-bit, field memory (Oki Semiconductor, 2007). It is especially designed for high-speed serial access applications such as HD TVs, conventional NTSC TVs, VTRs (Video Tape Recorders), digital movies and multi-media systems. In the design, RGB data arrives with a 24-bit wide bus, hence it can be stored in a RAM like this in a serial way. Use of a serial RAM also eliminates the need for wide address busses. LCD screen is scanned progressively and RGB data stream is transmitted in scan order, then automatic manipulation of address bus without microcontroller intervention can be easily achieved. Memory's minimum cycle time is 6.6 ns (150 MHz) (Oki Semiconductor, 2007). When compared with maximum pixel clock frequency output of the LVDS receiver IC (85 MHz) the cycle time of the selected memory is far below that value.

Each of the 32-bit planes has separate serial write and read ports. These employ independent control clocks to support asynchronous read and write operations.

Different clock rates are also supported that allow alternate data rates between write and read data streams. Although the RAM is only used in synchronous mode in the design, support of different clock rates for read and write operations is one of the prime necessities, because microcontroller reads the RAM more slowly than it is filled by the incoming RGB data stream.

Selected memory IC provides high speed FIFO operation without external refreshing (Oki Semiconductor, 2007). The memory refreshes its DRAM (Dynamic Random Access Memory) storage cells automatically, so that it appears fully static to the users. Moreover, fully static type memory cells and decoders for serial access enable the refresh free serial access operation, so that serial read and/or write control clock can be halted high or low for any duration as long as the power is on. Internal conflicts of memory access and refreshing operations are prevented by special arbitration logic. As stated above the RAM does not used by asynchronously in the thesis design, but its first-in first-out property makes data storage and retrieval process extremely easy. And auto self-refresh mechanism of the memory (Oki Semiconductor, 2007) makes it even more suitable to be used in this application without having extra hardware design.

Additionally, the memory has write mask function or input enable function (IE), and read-data skipping function or output enable function (OE). The differences between write enable (WE) and input enable (IE), and between read enable (RE) and output enable (OE) are that WE and RE can stop serial write/read address increments, but IE and OE cannot stop the increment, when write/read clocking is continuously applied to memory. The input enable (IE) function allows the user to write into selected locations of the memory only, leaving the rest of the memory contents unchanged. The memory’s block diagram is shown in figure 3.10.

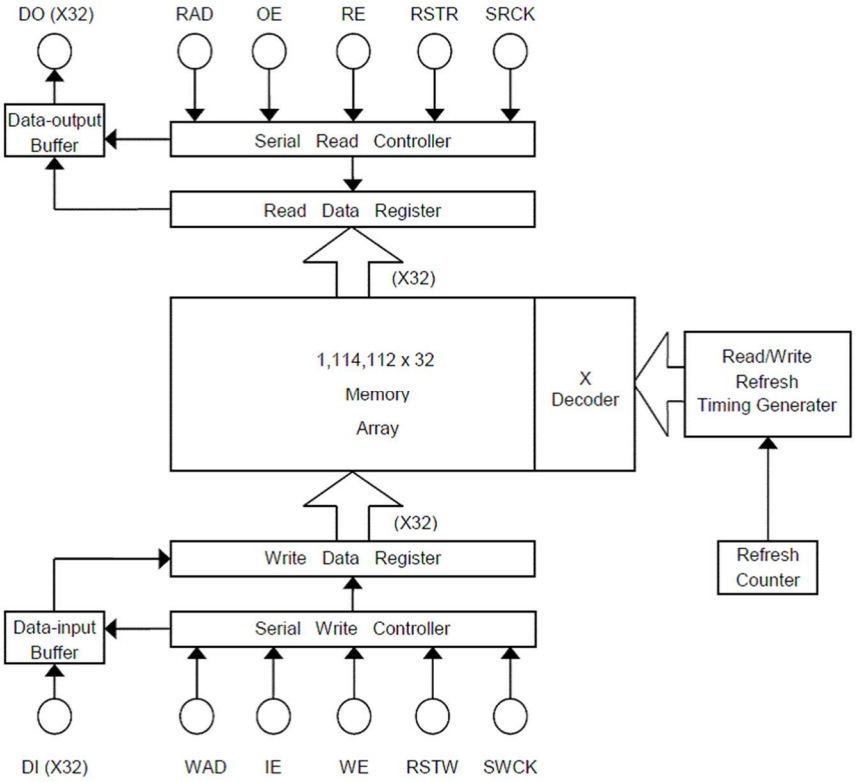


Figure 3.10 Block diagram of MS81V32322 showing its internal structure (Oki Semiconductor, 2007)

Figure 3.11 shows the related part of the hardware schematics drawing, due to large size of the components only necessary sections of them are included. All FIFOs have their own LVDS receivers, hence a 24-bit data bus (here D1_[24:1]) arrives from output of LVDS receiver, but all FIFOs share a common 24-bit data bus from FIFO output to microcontroller input. Although each FIFO output bus can be associated to a different input port of the microcontroller it would have no effect on the performance, because the microcontroller can access one input port (each has 32-bit wide) at a time. Such a design scheme also makes PCB drawing easier. As can be seen from figure 3.11 clearly, the microcontroller does not have any interruption on the incoming RGB data bus; it only receives data from whichever FIFO it wants.

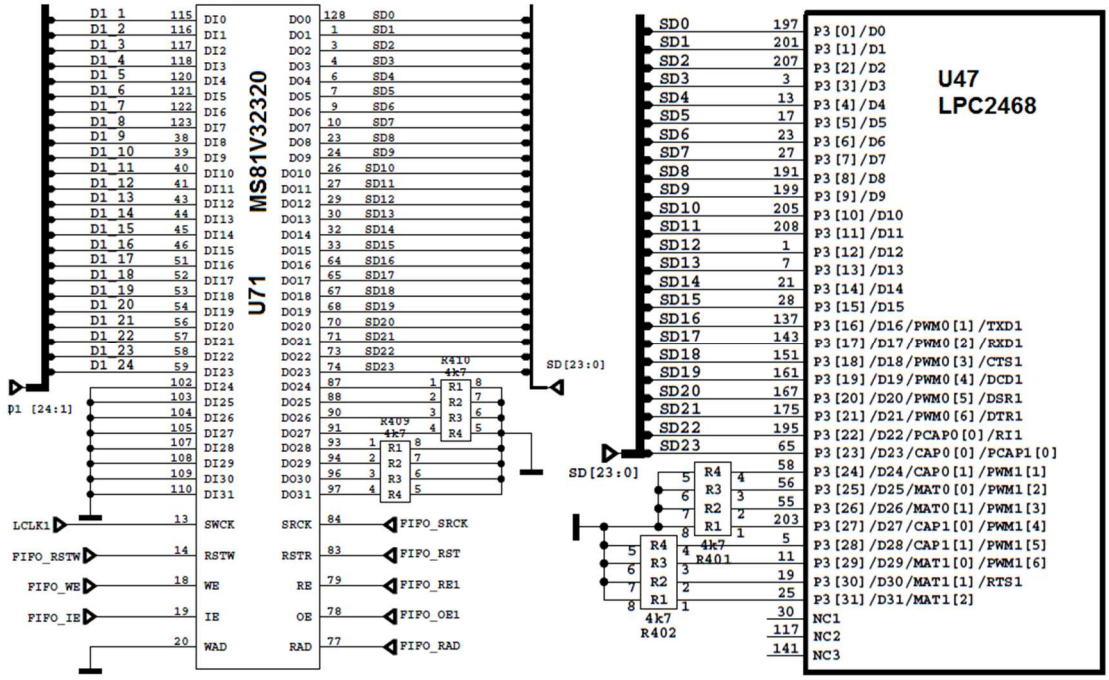


Figure 3.11 Hardware design schematic section showing FIFO interfacing

Here a question about the size of the memory can arise: Is memory large enough to store largest frame or frame part? To answer this question table 2.2 must be extended to see maximum frame or frame part size on single LVDS port as shown at table 3.2.

Table 3.2 Distribution of number of pixels on the LVDS ports according to different panel sizes

Native Panel Resolution		**Preferred Number of LVDS Ports**	**Number of All Pixels in the Frame**	**Number of Pixels in Each Port**
Number of Horizontal Lines	**Number of Vertical Lines**			
1024	768	1	786,432	786,432
1280	720	1	921,600	921,600
1366	768	1	1,049,088	**1,049,088**
		2		524,544
1920	1080	2	2,073,600	1,036,800
		4		518,400

As can be seen from table 3.2 maximum data that can be transferred to single FIFO is 1,049,088 x 3 bytes. Each pixel is composed of 24-bits, and occupies a single row in the memory. The MS81V32322 has 1,114,112 rows, so the memory size is enough for worst case condition.

The most critical part of the hardware design is DMA structure. The simple block diagram of the DMA structure is shown in figure 3.12. As can be deducted from figure 3.12, working mechanism of the DMA block is quite simple, although RGB data is always available to input port of the FIFO, DMA let this data to be written into FIFO in correct time by the help of microcontroller. In other words the microcontroller manages data write into FIFO over DMA.

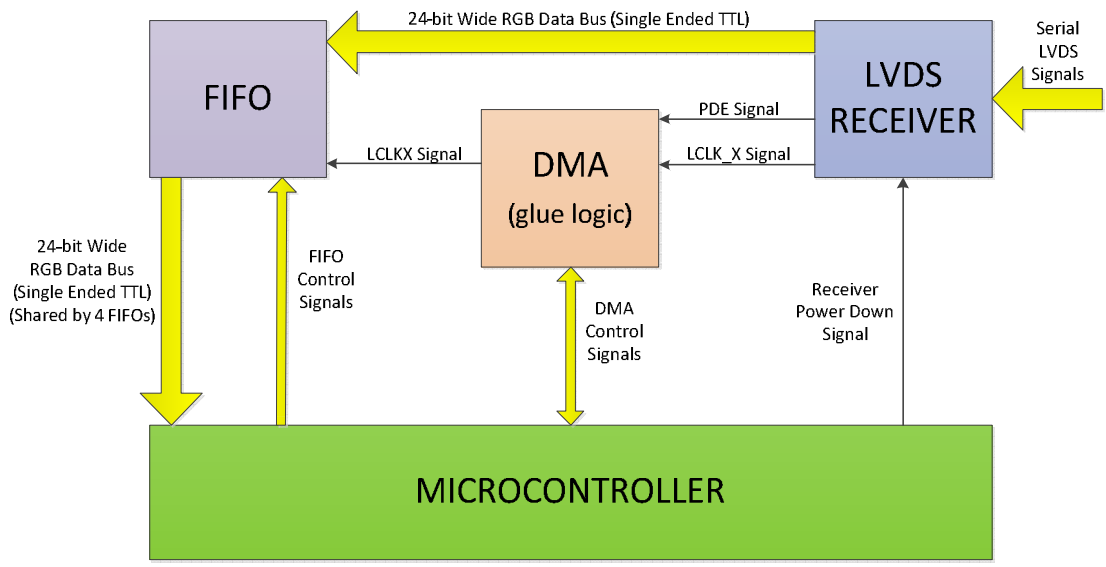


Figure 3.12 Block diagram of the DMA structure implemented on hardware design

A deeper understanding of DMA operation can be gained by examining figure 3.13. Figure 3.13 contains the DMA related parts of the hardware design. To keep things simpler only one LVDS receiver-FIFO pair is shown in here. In case of the utilization of more than one LVDS port all DMA parts work synchronously.

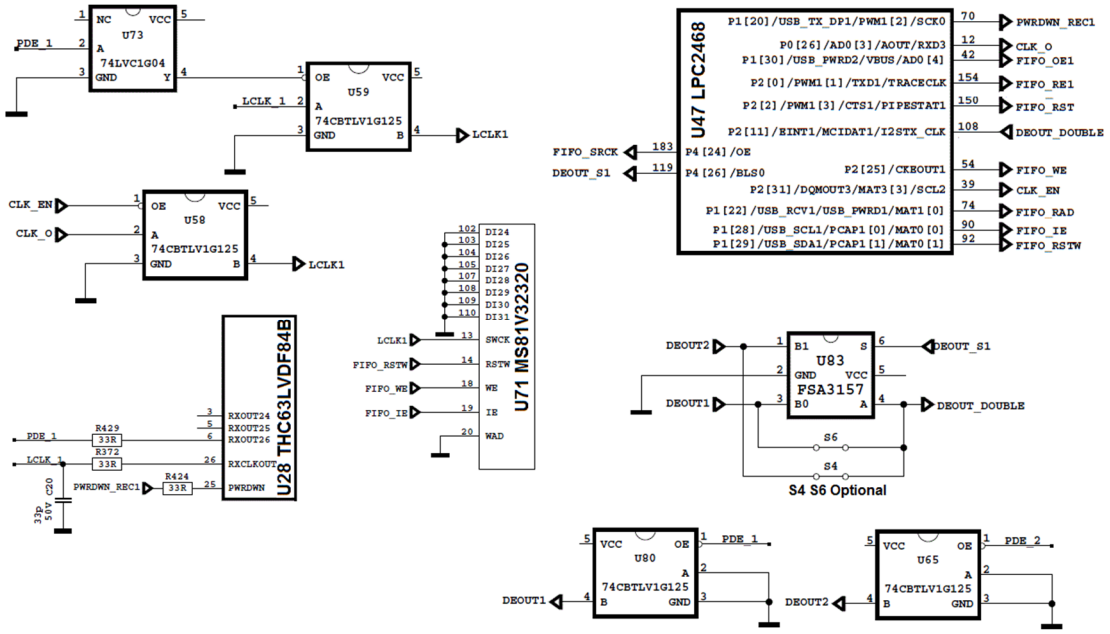


Figure 3.13 Hardware design schematic section showing DMA implementation

Here DMA operation is described for a single port 1366 by 768 pixels resolution panel. LVDS receiver (U28) produces RGB data signals, PDE_1 and LCLK_1 signals as long as LVDS data is present at its input port and PWRDWN input is at high. RGB data signals output of the LVDS receiver directly feed data input of FIFO, but LCLK_1 and PDE_1 signals are not directly fed to FIFO.

The MS81V3232 is an easy to manage FIFO. Its WAD (Write Address Input) input is at always zero which means when the FIFO is reset its write address input points to the beginning of the memory, hence the write operation always start from the top of the memory as intended. FIFO_IE input of the FIFO determines whether the incoming data is allowed to written into memory or not. FIFO_WE input of the memory determines whether the write address pointer is incremented in each SWCK (Serial Write Clock) cycle or not. FIFO_RSTW signal is used to set write address pointer of the memory together with WAD input of the memory. SWCK input of the memory increments write address pointer of the memory by one at each clock cycle. This is the most important feature of the memory. When the write address pointer of the memory is set to zero once, it is incremented by one automatically by LCLK1 signal. LCLK1 signal is pixel clock, so at each cycle of LCLK1 signal new RGB data is available, in each cycle of LCLK1 signal FIFO increments its write address pointer

by one and incoming RGB data is written into the next memory location. This is also shown in figure 3.14.

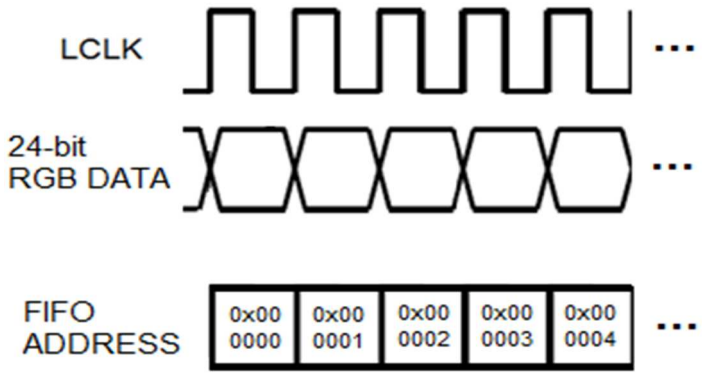


Figure 3.14 Relation between FIFO addressing and LCLK signal

The FIFO is filled by the help of LCLK1 (pixel data clock) signal and LCLK1 signal is coherent with RGB data signals. Such an arrangement sets up a perfect timing match between LCLK1 signal and RGB data signals, but another important point is to find the exact time to start filling the memory. “Finding the exact time to start filling the memory” operation is accomplished by microcontroller, and PDE_1 signal is used for this purpose. Here the microcontroller is responsible to find the beginning of each picture frame by probing PDE_1 signal.

Prior to explaining the function of PDE_1 signal, it is better to mention about the function of U83 (analog switch), U80 (line buffer) and U65 (line buffer) glue logic group. As explained before more than one port is used when larger amounts of data to be transmitted over LVDS. In such a case each LVDS receiver produces PDE_X output, and this glue logic group is used to select one of the receiver’s PDE_X output to be probed by the microcontroller. The action of this glue logic group can be perceived as a further hardware design expansion for future use.

As shown in figure 2.24 PDE signal stays at high whenever a valid pixel data is transmitted by the LVDS receiver. Figure 3.15 shows PDE signal in more detail to be discussed in here how it is used to keep the synchronization of the incoming RGB data. Timings in the figure 3.15 indicate some typical values for a 1366 by 768 pixels

resolution panel. Exact values of these parameters can vary between different panels, but changes in these values do not alter the operation of the designed hardware.

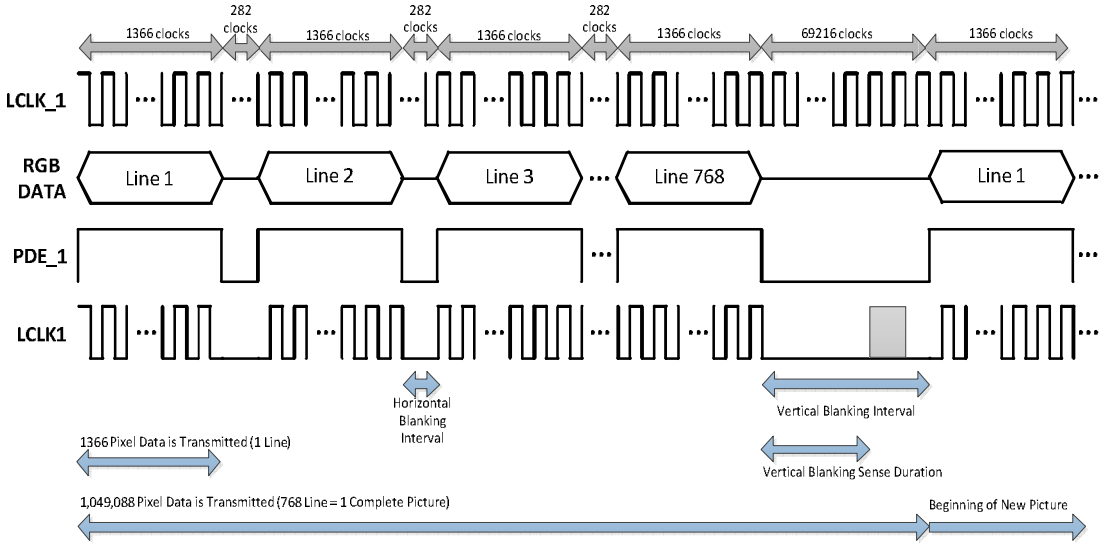


Figure 3.15 Detailed display timing diagram for a sample 1366 by 768 LCD panel

When figure 3.15 is examined it is clear that each new picture starts with a vertical blanking interval. For a typical panel operation it is 69,216 clocks duration, if panel operates at 80 MHz of pixel clock frequency, vertical blanking interval equals to 0.86 ms (Chunghwa, 2006). Such a long time interval can easily be detected by the microcontroller whose operating frequency is 72 MHz. Hence this signal (DEOUT_DOUBLE, after selection) is fed to one of the microcontroller's external interrupt pins, so whenever the microcontroller senses low value of DEOUT_DOUBLE for a long time (call it as vertical sense duration), it reset write pointer of the FIFO and the FIFO becomes ready to be filled by next picture. This vertical sense duration is set inside of microcontroller, and must be much longer than horizontal blanking interval to prevent microcontroller to sense horizontal blanking intervals as new frame beginning points. In addition, this vertical sense duration must be shorter than actual vertical blanking interval to have some time to initiate the FIFO for new picture. Optimum value of vertical sense duration can be discovered by examining various panel specifications, and can be easily changed inside of embedded software. This is also one of the strongest abilities of the hardware design

that it is insensitive to changes in pixel clock, vertical refresh and horizontal line rates.

LCLK1 signal is controlled by microcontroller all the time but not produced by it most of the time. As can be seen from figure 3.13, LCLK1 output of U59 (line buffer) is produced by LVDS receiver and LCLK1 output of U58 (line buffer) is produced by microcontroller. Hence LCLK1 is shaped out by using PDE, and now this signal can be fed to SWCK input of the FIFO to increment write address pointer by one whenever a valid pixel data is transmitted.

The microcontroller controls LCLK1 line by its CLK_EN (Clock Enable) and CLK_O (Clock Output) outputs; because when the FIFO must be reset, its SWCK input must be fed with a predetermined waveform as indicated in the FIFO producer's data sheet. As indicated in figure 3.15 when the microcontroller senses that the vertical blanking sense duration overflows, it asserts necessary signals (indicated by grey box) on LCLK1 line to make the FIFO ready for next incoming picture. The duration of the grey box is the only interval where the LCLK1 signal is produced by the microcontroller.

This process loops forever, as long as LVDS input signal is present. A typical TFT panel screen is refreshed 60 times in a second, so the FIFO is re-written 60 times in a second. When a "capture picture" command is received by microcontroller, it waits for next vertical blanking interval and transmits the most recent picture.

Figure 3.16 shows a part from the PCB design which includes an LVDS receiver-FIFO pair. As can be seen from figure 3.16 all RGB data lines between LVDS receiver and FIFO are approximately in same length, this is accomplished by during PCB drawing at route equalizing process. By this way approximately all data signals arrive to FIFO in same moment thus line to line signal transmission time differences are minimized.

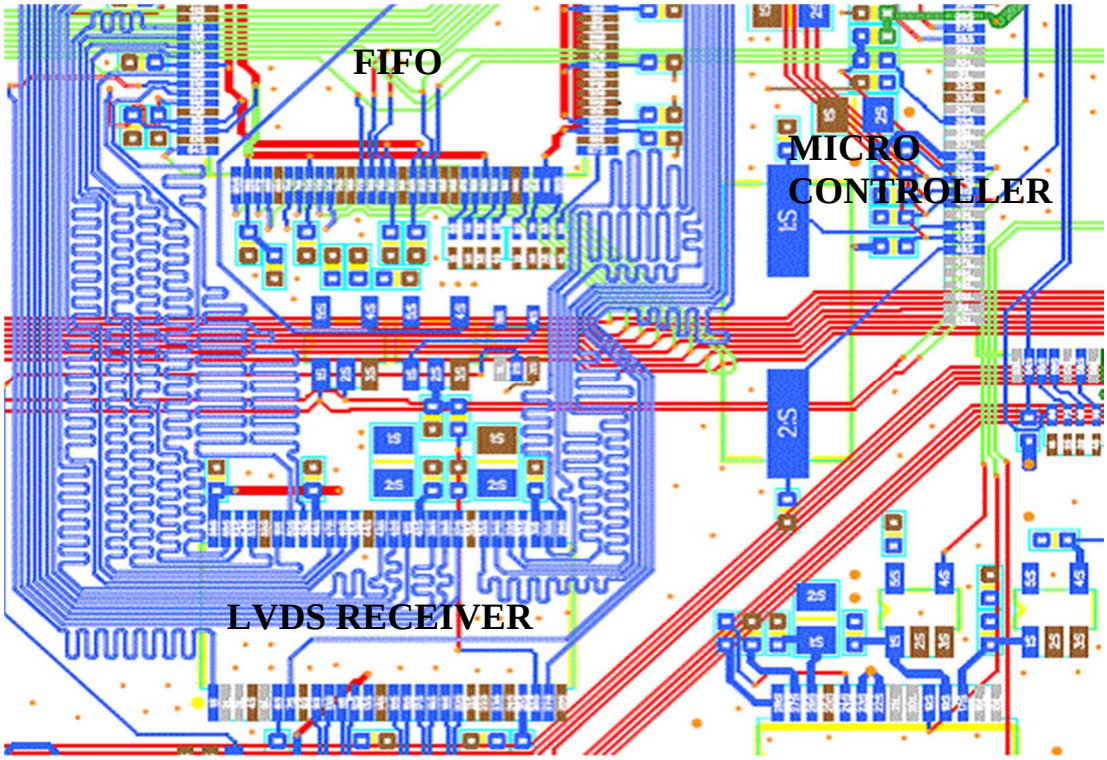


Figure 3.16 PCB design section showing FIFO interfacing

Figure 3.17 shows the FIFO, DMA and the microcontroller part of the design on PCB after auto insertion of components.

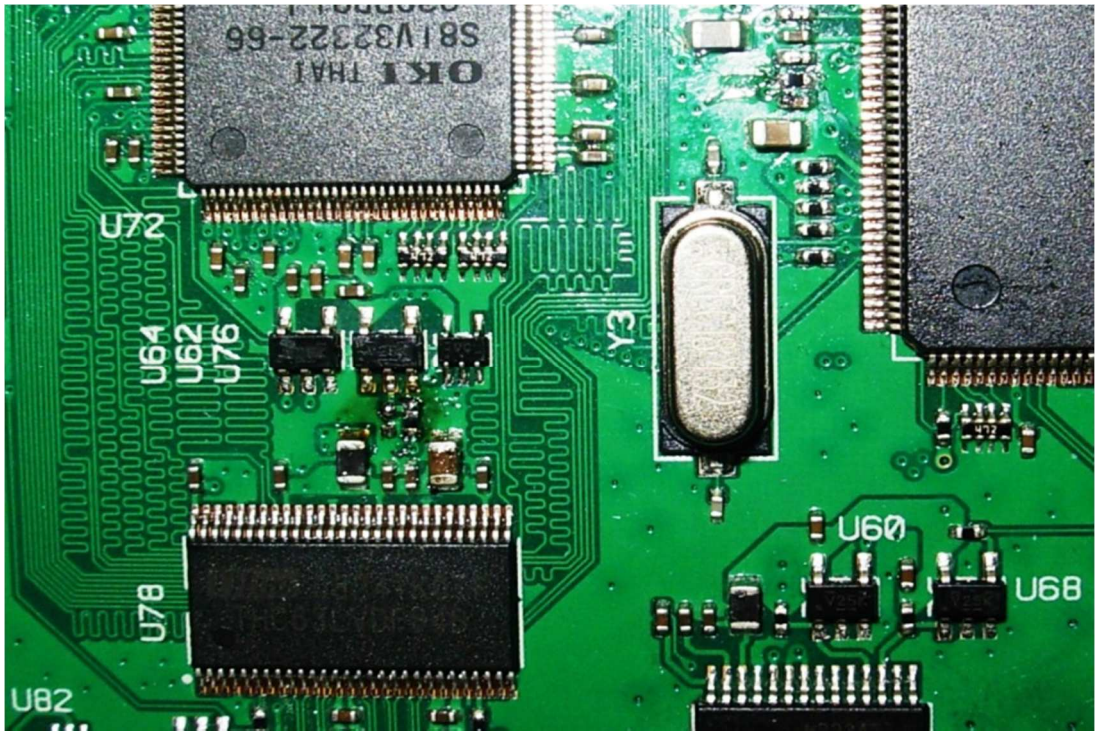


Figure 3.17 A hardware section from picture grabbing card showing FIFO interfacing

CHAPTER FOUR
ETHERNET COMMUNICATION

4.1 Data Acquisition with Ethernet Communication

Today much attention has been focused on the use of Ethernet technology directly at the device level for most industrial test equipments. The decreasing cost and increasing capabilities of network interfaces and microprocessors have accelerated the movement of communications network connections down to the instrument and device level. The Ethernet-Based I/O device is a measurement or I/O device with an Ethernet connection directly on the device itself as it is intended in this thesis. This approach provides a relatively inexpensive option for networked data acquisition, and provides greater versatility in terms of locating the measurement device in size-constrained areas or harsh environments (Potter, n.d.).

Compared to some other simpler communications links in use today, Ethernet does typically require a more powerful microprocessor and more memory to execute the communications stack (Potter, n.d.). However, the evolution of Ethernet technology indicates that the implementation cost will continue to decrease.

An Ethernet-based measurement device does have a unique set of requirements, relative to a standalone or PC-based measurement system. Reliable, autonomous operation of a distributed measurement device requires a more intelligent device with a local microprocessor that handles communications, system management, and diagnostics (Potter, n.d.). This is in addition to the needed components to collect and process the actual measurements.

The most important advantage of using Ethernet on this thesis is to be able to distribute data acquisition devices on the field freely. LVDS output of TV mainboards cannot drive long cables, and they are typically no longer than 50 cm for performance issues. In this point using USB (Universal Serial Bus) port can be an option to transfer picture data from designed hardware to PC. Although theoretical

bandwidth of USB2.0 (480 Mbps) is much higher than 100Base-TX (100 Mbps) Ethernet, connecting a few number of USB2.0 enabled picture grabbing cards to the same PC can be a problem, in addition to that USB2.0 has also cable length limitations, practically carrying USB2.0 data signals over 2 meters of distance requires use of expensive cables. In this point networkability of Ethernet provides great distinction together with its long cable driving capability (100 m for 100Base-TX). By this way a single server computer can control many picture grabbing cards through an Ethernet network switch.

The electronic circuit card designed in thesis captures a picture from the LVDS output of TV mainboard and transmit that picture via Ethernet. PC sends a picture grab command through the Ethernet to the hardware and hardware grabs a picture, and then transmits it.

There are many different communication methods over Ethernet protocol. Generally they all use the same hardware structure but differ in application. In this thesis, TCP/IP is used as communication stack. TCP/IP is preferred to OSI (Open Systems Interconnection), since it has fewer layers and easy to use. TCP/IP is composed of four layers as indicated in figure 4.1.

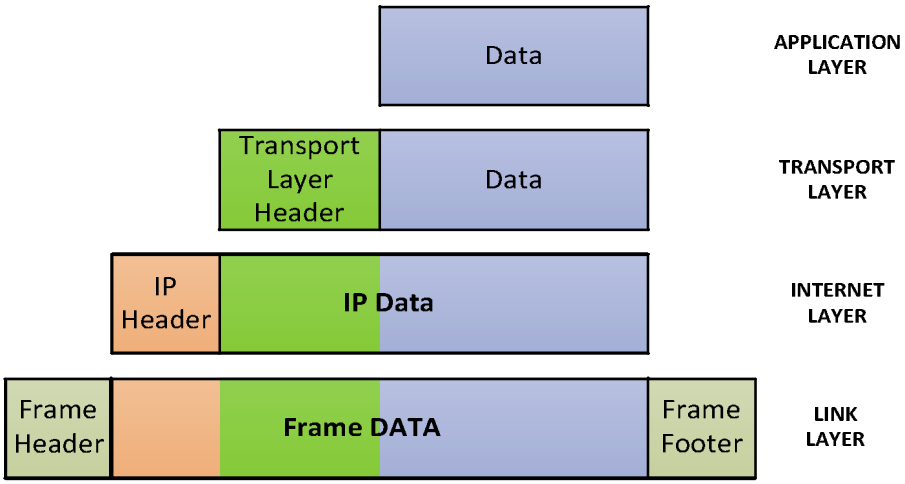


Figure 4.1 Layers of TCP/IP

Most popular transport layer protocols are TCP and UDP (User Datagram Protocol) in TCP/IP stack, and one of the most important parts of embedded software

design is to select one of the transport layer protocols. Rest of the layers does not have much importance on performance or code complexity.

TCP is the most commonly used protocol on transport layer. The reason for this is because TCP offers error correction. When the TCP protocol is used there is a "guaranteed delivery." This is due largely in part to a method called "flow control." Flow control determines when data needs to be re-sent, and stops the flow of data until previous packets are successfully transferred. This works because if a packet of data is sent, a collision may occur. When this happens, the client re-requests the packet from the server until the whole packet is complete and is identical to its original.

UDP is another commonly used protocol on the Internet. However, UDP is never used to send important data such as webpages, database information, etc.; UDP is commonly used for streaming audio and video. Streaming media use UDP because it offers speed. The reason UDP is faster than TCP is because there is no form of flow control or error correction. The data sent over the network is affected by collisions, and errors are present. UDP is only concerned with speed. This is the main reason why streaming media is not high quality.

As stated TCP and UDP have different properties, but for thesis application the performance is a prime issue. The picture data is not transmitted over the Internet, it travels through some network switches in a local network, so there is no need to establish a “guaranteed delivery” system. A main computer can manage more than one picture grabbing card, hence the computer ensures that number of currently data transmitting cards cannot exceed maximum specified value, maximum number of simultaneously transmitting cards can vary for different network structures and this number can be determined after some benchmarks. In addition to these TCP is a connection-oriented protocol whereas UDP is connectionless protocol. Setting up a connection and tracking that connection requires more coding in software and make the overall transmission process more complicated.

Figure 4.2 shows the frame structure for TCP and UDP side by side. As there is a limitation for frame size, TCP also consumes more space for non-payload data. In other words overhead of TCP is higher than UDP, it can be thought that difference is a few bytes, but a complete picture is transmitted by thousands of frame, so protocol overhead must be kept at minimum to achieve maximum throughput on the communication system. Data payload of a TCP frame starts from 192nd bit whereas data payload of UDP frame starts from 64th bit. It must also be noted that "options" area of a TCP frame can be 320-bit wide, but it is represented here as 32-bit.

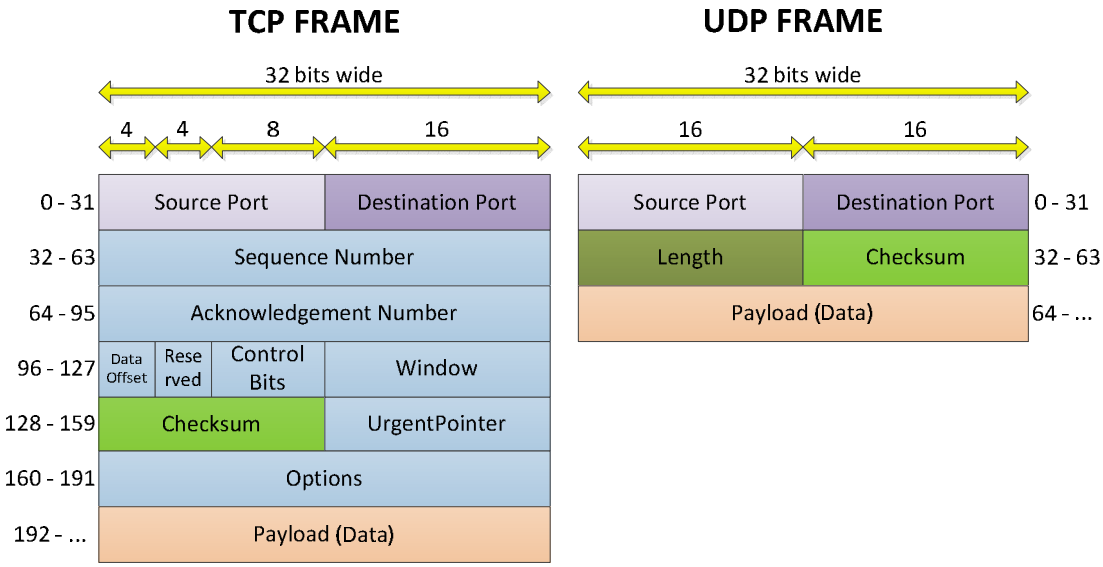


Figure 4.2 Comparison of TCP and UDP frames

4.2 Selecting a TCP/IP Stack for a 32-bit Microcontroller

The formal requirements for the protocols in the TCP/IP (Transmission Control Protocol/Internet Protocol) stack are specified in a number of RFC (Request For Comments) documents published by IETF (The Internet Engineering Task Force). Each of the protocols in the stack is defined in one more RFC documents and RFC1122 collects all requirements and updates the previous RFCs. So it is obvious that the implementation of TCP/IP stack from bottom to top is a rather challenging task and requires working on all related RFCs which is a serious software work and coding. Due to the complexity of TCP/IP stack a ready solution is searched in this stage of the thesis.

When compared with PCs embedded microcontrollers have many limitations, such as processing speed, memory amount, data bandwidth etc. Because of these reasons developing a TCP/IP stack for an embedded microcontroller is quite different for those of PCs. Some companies develop TCP/IP stack for embedded microcontrollers, but some voluntary people also work on the same subject.

Interniche Technologies Inc. developed two stacks for LPC24XX family microcontrollers. One of them is Nichelite and the other one is NicheStack IPV4. Architectural overviews of these two stacks are shown in figure 4.3. These two stacks require an RTOS (Real Time Operating System) working on the microcontroller, but in the thesis a single-process software design is aimed, because the designed hardware only serves as a capture card. In addition to that Niche stacks contain layer services such as FTP (File Transfer Protocol), DHCP (Distributed Host Control Protocol), DNS (Domain Name System), and NAT (Network Address Translation) etc. that are not necessary for the software design of the capture card. These services also add code complexity to the project.

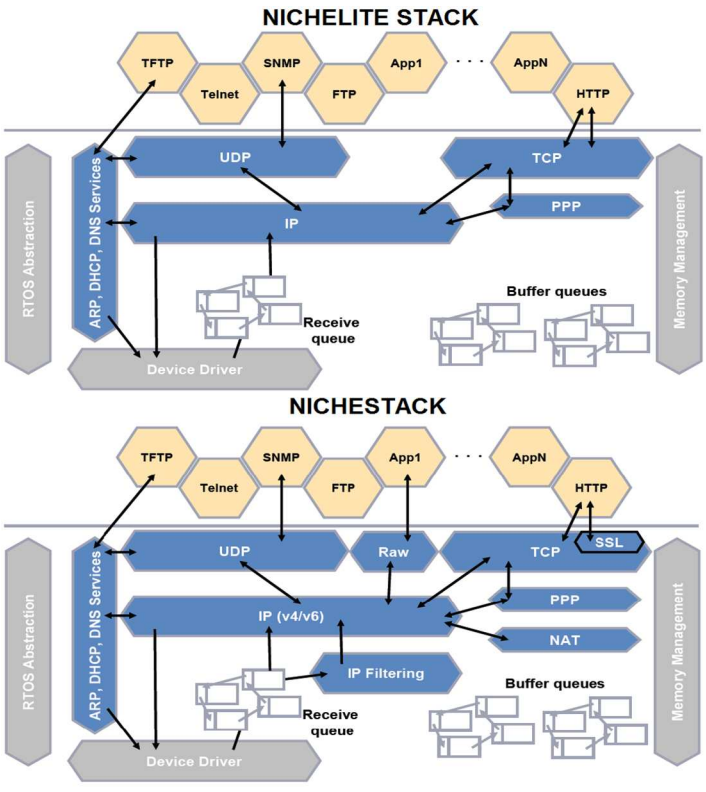


Figure 4.3 Nichelite and Niche Stacks for TCP/IP implementation (NXP, 2007)

RealView Real Time library developed by Keil Company also presents a TCP/IP stack for embedded microcontrollers, but that stack also requires a working RTOS at the base, and again contains many layer services that are unnecessary for the project.

Most commercial TCP/IP stacks have the shortage of being lightweight and target specific stacks. In addition to that customers cannot modify these stacks to suit them for their applications. At this point an open source TCP/IP stack is chosen to be used in the project. EasyWeb and uIP are two well-known open source TCP/IP stacks. uIP was successfully ported to RTOS based systems and EasyWeb was ported to single-process systems.

EasyWeb is a small TCP/IP stack and web server, it was published in the extra issue “Embedded Internet” of the magazine Design & Elektronik. The original stack is developed by Andreas Dannenberg from HTWK Leipzig, University of Applied Sciences, and this stack is ported to LPC2XXX family by Keil Company and can be freely distributed. Embedded software of the microcontroller is developed in C by using Keil uVision4 Integrated Development Environment. In addition to porting of stack to LPC2XXX family, NXP and Keil also supply some sample codes for this stack to speed up the development process.

EasyWeb TCP/IP stack handles the ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol), IP and TCP protocols. It is optimized for low resource consumption, not for performance. Because of the low resource consumption, the TCP/IP stack has some restrictions as stated at table 4.1, but none of these restrictions affects the operation of the design in the thesis as explained next to the restrictions.

Table 4.1 EasyWEB TCP/IP stack restrictions together with their effects

Stack Restriction	**Stack Restriction Effect on the Design**
Only one active TCP session at the same time	No effect, UDP is used in the software design instead of TCP and these options are related with TCP
No buffering of TCP segments received in wrong order	
Received TCP options are ignored	
No support for “type of service” and security options	
No reassembling of fragmented incoming IP frames	No effect, microcontroller will never subject to receive large frames
No checksum verification of received data	

4.3 Hardware Application: Fast RGB Data Transfer Using UDP

In Section 3.4 it is explained in detail how the FIFO is filled with the picture data. In this section, retrieval of picture data from the FIFO and transmission of it via Ethernet is explained.

Figure 4.4 shows the related section of the hardware design which contains the microcontroller, the FIFO and the Ethernet PHY (Physical Layer) device. Power, ground, some signal lines and passive components are not drawn in here to keep the figure simpler.

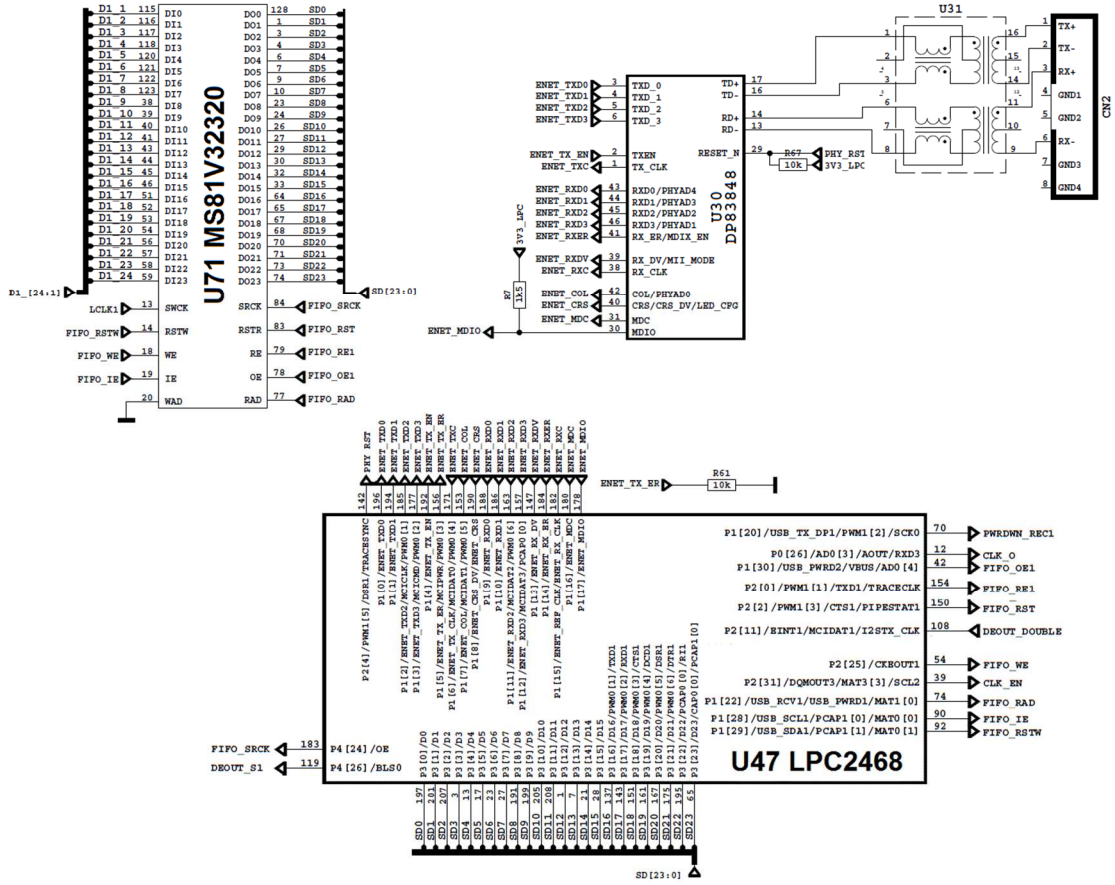


Figure 4.4 Hardware design schematic section showing Ethernet communication implementation

When the microcontroller receives “grab picture” command from the PC, it waits for the next vertical blanking interval in the video, and then this time it does not reset the FIFO. The microcontroller immediately disables the input of the FIFO by making

FIFO_WE and FIFO_IE inputs low, by this way incoming RGB data cannot alter the current content of the FIFO. Then microcontroller wraps the read pointer of the FIFO up to the first memory location where the RGB (Red Green Blue) data of (0, 0) coordinate on the screen is stored. By using FIFO_RAD signal the FIFO can be started to read from any location, but the aim is to retrieve the entire picture so the pointer is set to the beginning of the FIFO. The microcontroller uses FIFO_SRCK (FIFO Serial Read Clock) output to control address pointer of the FIFO. At each cycle of FIFO_SRCK the FIFO increments its read address pointer by one and data at the next memory location is present at the output data bus of the memory and presented to the microcontroller. The microcontroller repeats to alternate FIFO_SRCK line until it completely reads and transmits the entire picture in the FIFO, it knows how much data to read by the received command from the PC.

The Ethernet block of LPC2468 contains a full featured 10 Mbps or 100 Mbps Ethernet MAC designed to provide optimized performance through the use of DMA hardware acceleration. The Ethernet block interfaces between an off-chip Ethernet PHY using the MII or RMII protocol and the on-chip MIIM (Media Independent Interface Management) serial bus. Figure 4.5 shows the Ethernet subsystem of the LPC2468.

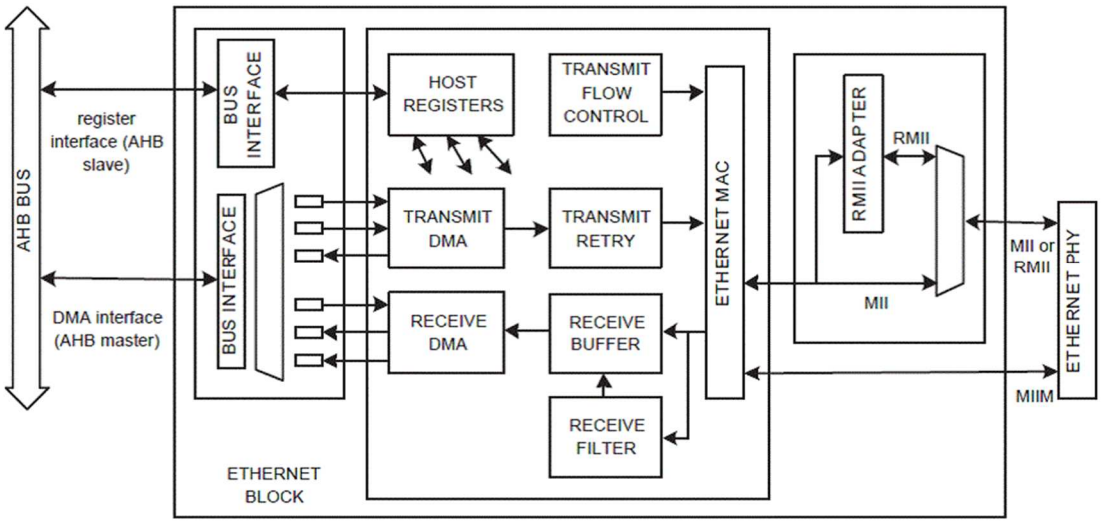


Figure 4.5 Ethernet subsystem of LPC2468 (NXP, 2008)

The block diagram for the Ethernet subsystem consists of (NXP, 2008):

- The host registers module containing the registers in the software view and handling AHB accesses to the Ethernet block. The host registers connect to transmit and receive datapath as well as the MAC.
- The DMA to AHB interface. This provides an AHB master connection that allows the Ethernet block to access the Ethernet SRAM for reading of descriptors, writing of status, and reading and writing data buffers.
- The Ethernet MAC and attached RMII adapter. The MAC interfaces to the off-chip PHY.
- The transmit datapath, including:
  - The transmit DMA manager which reads descriptors and data from memory and writes status to memory.
  - The transmit retry module handling Ethernet retry and abort situations.
  - The transmit flow control module which can insert Ethernet pause frames.
- The receive datapath, including:
  - The receive DMA manager which reads descriptors from memory and writes data and status to memory.
  - The Ethernet MAC which detects frame types by parsing part of the frame header.
  - The receive filter which can filter out certain Ethernet frames by applying different filtering schemes.
  - The receive buffer implementing a delay for receive frames to allow the filter to filter out certain frames before storing them to memory.

A complete Ethernet hardware interface normally consists of 4 major parts: The MAC, the PHY, the magnetics, and the connector. Connectors with integrated magnetics are also popular in nowadays. The block diagram of the Ethernet part of the design is shown in figure 4.6. The MAC of LPC2468 handles the high level portions of the Ethernet protocol (framing, error detection, when to transmit, etc.) and the PHY handles the low level logic such as 4B/5B encoding/decoding, SERDES (serialization/deserialization), NRZI (Non Return to Zero Inverted) encoding/decoding and analog portions.

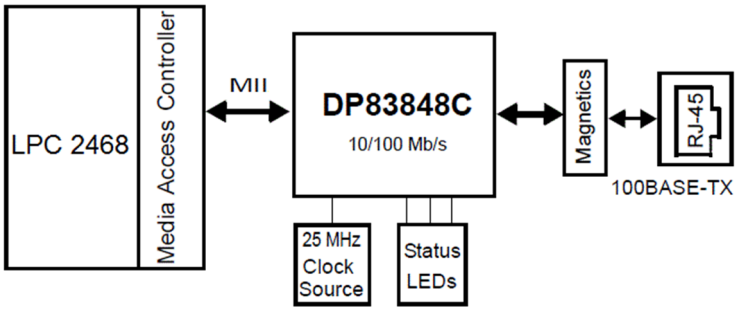


Figure 4.6 Interfacing of PHY device

The MII is a standard interface used to connect a Fast Ethernet (i.e. 100 Mbps) MAC-block to a PHY chip. The MII may be used to connect a MAC chip to a PHY chip on the same PCB. Being media independent means that any of several different types of PHY devices for connecting to different media (i.e. Ethernet, fiber optic) can be used without redesigning or replacing the MAC hardware.

RMII is one of the other possible interfaces between the MAC and PHY. By comparison, the MII interface requires two additional data lines in each direction, RX_DV (Receive Data Valid) and CRS (Carrier Sense) are separate rather than multiplexed, a separate TX_CLK (Transmit Clock) and RX_CLK (Receive Clock) are used instead of a shared reference clock, and a collision signal is added for a total of 7 additional lines (National Semiconductor, 2008). The added pin count is more of a burden on microcontrollers with built in MAC, FPGAs (Field Programmable Gate Arrays), multiport switches or repeaters, and PC motherboard chipsets than it is for a separate single port Ethernet MAC which partially explains why the older MII standard was more wasteful of pins.

In the low level Ethernet communication coding of the embedded software MII standard is used between the microcontroller and the PHY. This portion of the software is adopted from Keil's example source codes.

The DP83848C Phyter from National Semiconductors is chosen as PHY device in the design of this part. This IC supports both MII and RMII. Some of the key features of this component are listed below:

- Low power consumption, typically less than 270 mW

- 3.3 V MAC Interface
- Automatic medium dependent interface crossover for 10/100 Mbps
- IEEE (The Institute of Electrical and Electronics Engineers) 802.3u MII
- Error-free operation up to 137 meters
- Single register access for complete PHY status

The PHY itself is a rather complicated device which has over 20 configuration registers that controls the operation of the device. So a completely separate code file in embedded software deals with the control of these low level operations.

After the microcontroller receives the RGB data from the FIFO, it prepares this data for the transmission by using UDP (User Datagram Protocol) protocol. The first step is the decision of whether the data is transmitted without manipulation or some processing is necessary. As previously mentioned RGB outputs of the LVDS receiver is arranged for VESA standard. If incoming LVDS data has JEIDA standard, then the microcontroller performs necessary rotating operations on each color channel to correct the received data in accordance with JEIDA standard.

The LPC2468 has a limitation on maximum transfer unit size for Ethernet, and this value is 1500 bytes, so the size of a complete Ethernet frame cannot exceed 1500 bytes (NXP, 2007). This restriction limits the data payload area size of UDP protocol in the microcontroller to a maximum value of around 1470 bytes. In the second step microcontroller takes a part of the picture (less than 1470 bytes) and adds transport layer header (UDP header) into it. After that IP header is added to this IP data, and at last Ethernet frame header and footer are added to frame data. Now the data link layer frame is ready to be sent via Ethernet. The microcontroller passes this frame to the PHY and the PHY places this data on the transmission medium as bit stream. A cheap UTP (Unshielded Twisted Pair) cable is used to connect the picture grabbing card to the PC directly or through a network switch.

Figure 4.7 shows the PCB design part which includes the LPC2468, the PHY and the Ethernet line transformer.

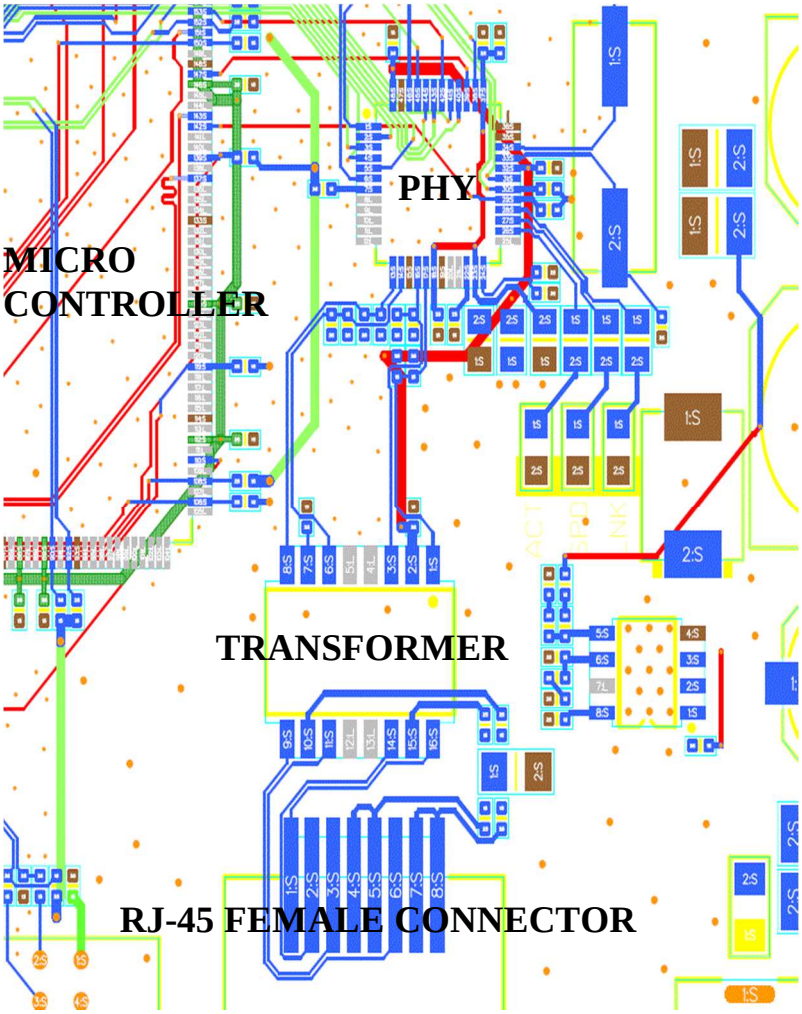


Figure 4.7 PCB design section showing Ethernet implementation

Data link layer signals of the Ethernet are differential signals. As can be seen from figure 4.7 these signals are routed in differential and equal length PCB traces when they leave PHY device and until they arrive RJ-45 connector. Careful signal routing on PCB design also plays an important role for the satisfactory operation of the Ethernet block of the design.

Although figure 4.7 does not contain FIFO to microcontroller data bus, length of each single line of this data bus must be equalized to each other as much as possible to eliminate different arrival times of data signals to the microcontroller. Figure 4.8 shows the Ethernet related part of the design on PCB after auto insertion of components.

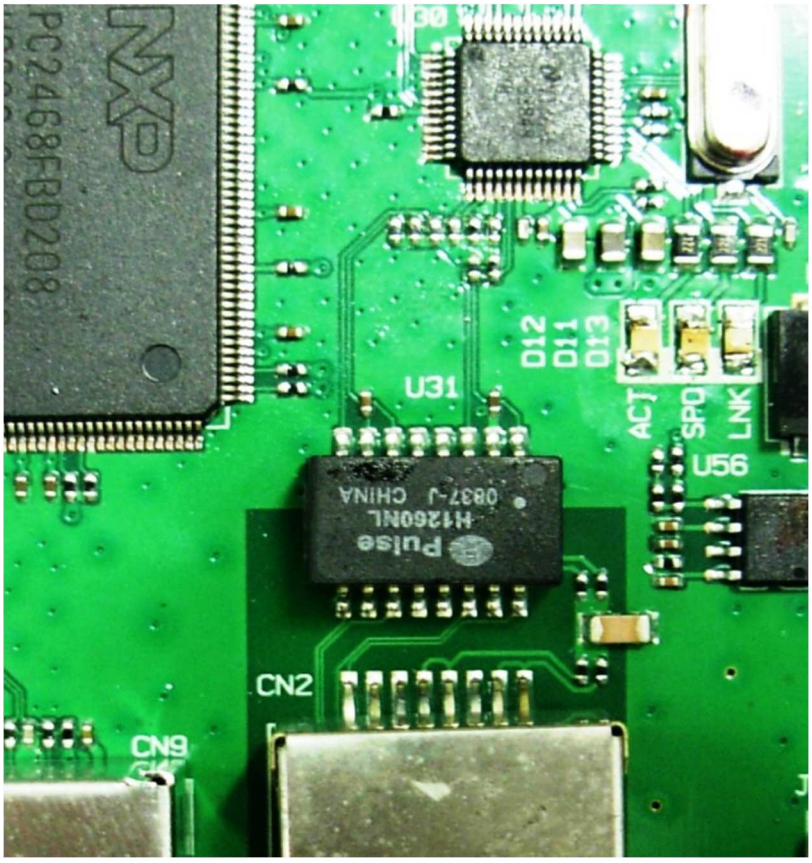


Figure 4.8 A hardware section from picture grabbing card showing Ethernet implementation

4.4 Software Application: Low Payload, High Speed UDP Packages

High level software development on the PC is performed on National Instrument's LabVIEW visual software development environment. When compared with other programming languages, LabVIEW presents a user friendly coding interface and it has a high level of abstraction. In this way the programmer spends more time to how to organize the overall structure of the program instead of writing time and memory efficient functions to perform every operation. LabVIEW software development environment is greatly reduced the time required to develop the complete PC application and a lot of previously written basic functions are used to keep the code more clean and understandable.

LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where

instructions determine the order of program execution, LabVIEW uses dataflow programming, where the flow of data through the nodes on the block diagram determines the execution order of the VIs (Virtual Instruments) and functions. VIs are LabVIEW programs that imitate physical instruments.

In LabVIEW, the user interface is built by using a set of tools and objects. The user interface is known as the front panel. After the front panel is built, code is added using graphical representations of functions to control the front panel objects. This graphical code, also known as G code or block diagram code is added to the block diagram. In some ways, the block diagram resembles a flowchart.

Figure 4.9 shows a sample front panel and its associated block diagram. The front panel is built by using controls (a and b in here) and indicators (a+b and a-b in here), which are the interactive input and output terminals of the VI, respectively. Controls are knobs, push buttons, dials, and other input mechanisms. Indicators are graphs and other output displays. Controls simulate instrument input mechanisms and supply data to the block diagram of the VI. Indicators simulate instrument output mechanisms and display data the block diagram acquires or generates. After the front panel is built, the code is added using graphical representations of functions to control the front panel objects. The block diagram contains this graphical source code. Front panel objects appear as terminals on the block diagram.

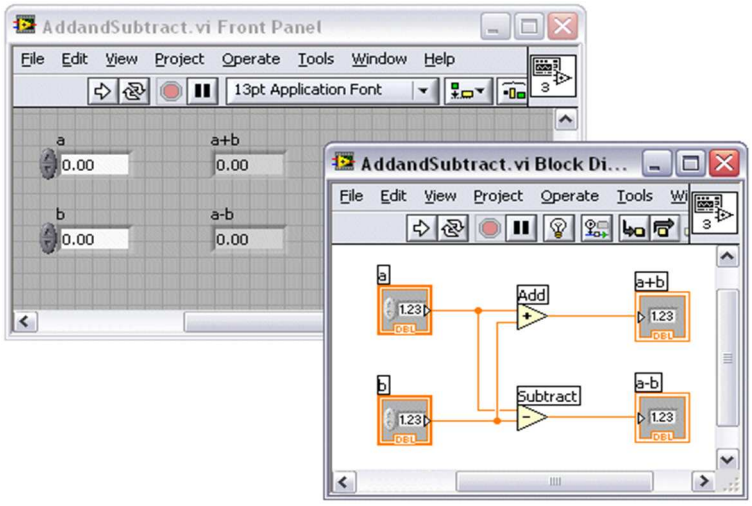


Figure 4.9 A sample front panel and its associated block diagram from LabVIEW software development environment

Data is transferred among block diagram objects through wires. In figure 4.9, wires connect the control and indicator terminals to the Add and Subtract functions. Each wire has a single data source, but it can be wired to many VIs and functions that read the data. Wires are different colors, styles, and thicknesses, depending on their data types.

In section 4.3 it is mentioned that the LPC2468 has a restriction on maximum Ethernet packet size which is equal to 1500 bytes, and this restriction limits data payload of each frame transmitted by the microcontroller to a value around 1470 bytes. Although this limitation can be interpreted as a transmission speed shortage and many modern PC operating systems can support much higher values for UDP (User Datagram Protocol) data payload, the microcontroller is able to transmit these packages in really fast speed due to the low complexity of the UDP stack. On the other side PC is not fast enough to capture and transfer this data to the application layer. To investigate that issue some benchmarking work is done under normal operating conditions of the Ethernet protocol on the PC. Results show that the PC cannot match the transfer speed of the microcontroller, so nearly half of the packages transmitted by the microcontroller cannot reach the application layer on the PC program and are lost at PC's Ethernet interface.

The problem is that the microcontroller is generating low payload, high speed UDP packages, but PC systems are designed to operate with large payload, low speed UDP packages. As there is no chance to modify the microcontroller's limits, this problem must be solved at PC side.

Most Ethernet NICs (Network Interface Cards) on the PCs include some on-board buffer memory for both data transmission and receiving, and this buffer is usually very limited in size for example at around 1024 bytes. The size of this buffer is so small for high speed operations, but Microsoft Windows lets users to configure some main system memory as a secondary buffer for Ethernet communication during socket initialization. The key design action is to use the dynamic link library wsock32.dll to increase receive buffer size of the Ethernet NIC by using some

portion of the main system memory. Now the operating system can receive entire picture data into its main memory and faster flushing of the Ethernet NIC buffer is achieved. The usage of this function is shown on VI in figure 4.10.

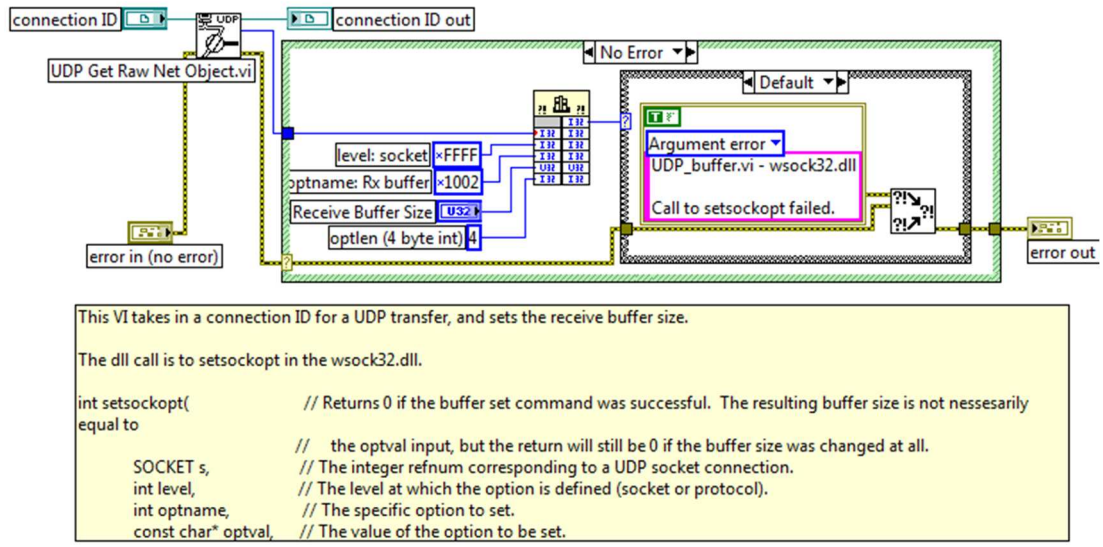


Figure 4.10 Setting UDP parameters in Microsoft Windows using wsock32.dll

Main application software that runs on the PC consists of two separate blocks: image capture and image processing. Image processing is discussed at Chapter 5.

Tabbed structure of the application software divides it into three parts: LVDS capture and display part, picture compare part (PSNR) and picture edge detection part. Last two tabs are explained in detail at the end of Chapter 5.

LVDS capture and display tab is composed of four parts:

- Connection settings
- Other settings
- Captured picture
- Capture progress bar and capture picture button

Figure 4.11 shows a screenshot from LVDS capture and display part of the application software.

Figure 4.11 LVDS capture and display tab of the PC application software

Connection settings part of LVDS capture and display tab is composed of seven settings:

- *Network address of system:* Every IP network operates on an IP pool. The size of this pool is determined by subnet mask of the system. An IPv4 network address is composed of 32 bits. Here "/16" suffix means that first 16 bits of "10.3.0.0" address is network address, and rest 16 bits are used to address IP clients on the network. The network address of the system is "10.3.0.0" and "10.3.255.255" is reserved for broadcasting purposes, so (2^{16}–2) different IP clients can work on this network. IP addresses in corporate networks are usually distributed by a DHCP server, the application software can operate on either a private or a DHCP-enabled network, but IP addresses must be chosen carefully to place both the PC and the capture card into the same IP pool.
- *IP address of PC:* This indicator finds and shows the PC's IP address. This value can be assigned to the PC either statically or by DHCP server, but it must also be specified in embedded software of the microcontroller.

- *IP address of capture card:* IP address of the capture card must be entered into this area. This IP address is specified in embedded software of the microcontroller, but can be changed to any value to meet network requirements.
- *Capture card request port:* This port value is also specified in embedded software of the microcontroller. This number indicates request listen port of the microcontroller. Capture request command generated by the PC uses this number as UDP destination port. So the microcontroller listens to this port continuously to check any "picture grab" command is received.
- *Data receive port on PC:* This port value is also specified in embedded software. When the microcontroller receives a "picture grab" command, it starts to transmit picture data by using IP address and data receive port of the PC. In network terminology combination of IP address and port number is called as socket, i.e. "10.3.89.1:50004" represents a socket.
- *Card detected? :* The application software periodically checks the capture card to verify the connection. The green color indicates that the connection is OK, the yellow color indicates that the software is checking the connection and the red color indicates that the capture card is not connected.
- *Mode of operation:* This control is designed to isolate PC application software from microcontroller's embedded software in debug mode; hence the PC application software can be developed in absence of the capture card to verify basic functionality of the PC software. Four debug modes are defined for differently configured LVDS ports, and one real mode lets the system to run with a real capture card.

Other settings part of LVDS capture and display tab is composed of three settings:

- *LVDS type and video standard:* This setting is used to select LVDS configuration of the currently tested TV mainboard. As shown previously at table 2.2, LVDS interface can be single, double or quad port; so the right configuration must be selected in here. And there are two common LVDS signaling standards, JEIDA and VESA, again the right standard must be chosen in here.

- *Set working directory:* This control lets the user to select where the captured files are stored. When the image data is received from the capture card, it is saved into this folder as bitmap picture by using current date and time as file name.
- *Picture preview size:* This control lets the user to zoom into the currently displayed captured picture.

Captured picture part shows a preview for captured image data. This picture is also saved into the working directory, so it can be viewed later, too.

Capture picture button sends the “grab picture” command to the microcontroller via Ethernet connection in accordance with the info supplied in settings section. For each different LVDS type and video standard setting, the application software sends a different command to the capture card, so the microcontroller knows which LVDS port(s) is used and how much data is gathered. Capture progress bar (the blue horizontal bar) indicates image data transfer progress after issuing a capture request to the card.

CHAPTER FIVE
IMAGE PROCESSING

5.1 Understanding Color and Color Spaces

A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components. When this model is associated with a precise description of how the components are to be interpreted (viewing conditions, etc.), the resulting set of colors is called a color space.

In the RGB color space, red, green, and blue, referred to as primary colors are combined in various proportions to produce other colors, which are referred to as secondary colors.

Figure 5.1 shows a cut-away color cube showing the mixing of red, green, blue to produce other colors.

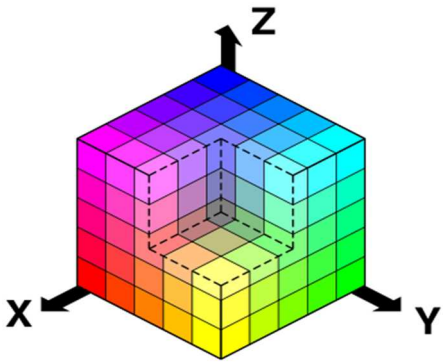


Figure 5.1 Mixing of red, green and blue
to produce all other colors (June, 2006)

The left most image shown in figure 5.2 is an RGB image, along with its separate R, G, and B components. As one can see from figure 5.2, the white snow consists of strong red, green and blue colors; the brown barn is made up of strong green with little red or blue, and the sky is composed of strong blue and moderately strong red and green. In a computer display, an image is naturally represented in the RGB color space. Color CRTs (Cathode Ray Tubes) and LCDs show an image by emitting red,

green and blue lights at each pixel with intensities proportional to the values of the components. When viewed from a distance, the separate red, green and blue lights merge to give the appearance of a 'true' color (June, 2006).

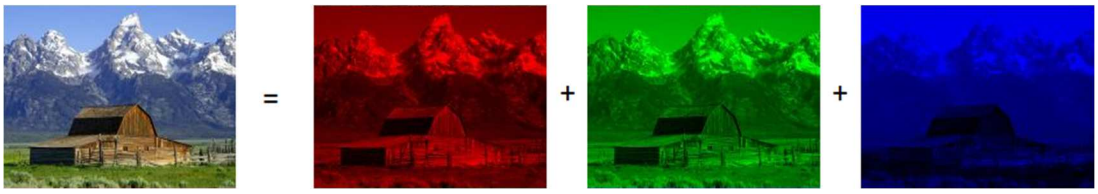


Figure 5.2 Separation of a color image into its red, green and blue components (June, 2006)

The RGB color space may be best in describing computer graphics or displaying images. However, it is not a good representation when it comes to process image information. This is because the human visual system (HVS) is less sensitive to color than to luminance (brightness). So if it is needed to discard some information carried by an image, firstly some color information is threw away rather than that of luminance. The YCbCr (and YUV as its analog counterpart) color space lets an image to be described by using luminance and colors. In this representation, an image is composed of a Y component, which is the luminance (luma), and three color difference (chrominance or chroma) components, blue chroma Cb, red chroma Cr, and green chroma Cg. The components can be calculated from R, G, B using formulas in equations 5.1, 5.2, 5.3 and 5.4:

$$Y = k_r R + k_g G + k_b B \quad (5.1)$$

$$Cb = B - Y \quad (5.2)$$

$$Cr = R - Y \quad (5.3)$$

$$Cg = G - Y \quad (5.4)$$

where k_x values are weighing factors with $k_r + k_g + k_b = 1$.

Cg can be obtained from Cb, Cr and Y. So in practice, only two of the three chroma components need to be saved.

Figure 5.3 presents the same RGB image discussed in figure 5.2 but along with the Y, Cb, and Cr components. It can be seen that that even if Cb, and Cr

components are totally threw away, retaining only the Y component, there are still some important features of the original image on luma.

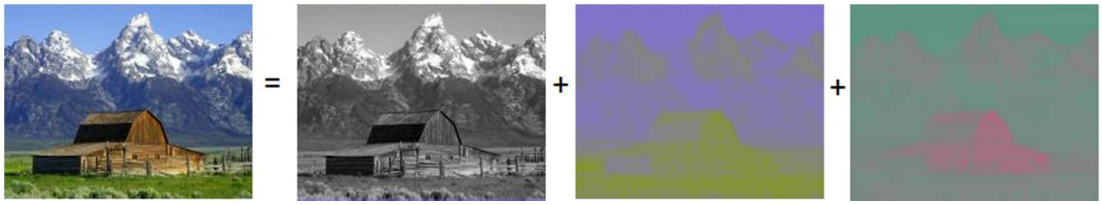


Figure 5.3 Separation of a color image into its luma and chroma components (June, 2006)

In a more generic form, the conversion from RGB to YCbCr can be done using formulas in equations 5.5, 5.6 and 5.7:

$$Y = k_r R + (1 - k_b - k_r) G + k_b B \quad 5.5$$

$$Cb = \frac{B-Y}{2(1-k_b)} \quad 5.6$$

$$Cr = \frac{R-Y}{2(1-k_r)} \quad 5.7$$

ITU (International Telecommunications Union) defines two standards for the values of k_x coefficients: BT.601 and BT.709.

BT.601 coefficients are used to decompose an SD (standard definition) video frame into its YCbCr components, in this case k_b = 0.114 and k_r = 0.299 and equations 5.8, 5.9 and 5.10 are used:

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \quad 5.8$$

$$Cb = 0.564\,(B - Y) \quad 5.9$$

$$Cr = 0.713\,(R - Y) \quad 5.10$$

BT.709 coefficients are used to decompose an HD (high definition) video frame into its YCbCr components, in this case k_b = 0.0722 and k_r = 0.2126 equations 5.11, 5.12 and 5.13 are used:

$$Y = 0.2126\,R + 0.7152\,G + 0.0722\,B \quad 5.11$$

$$Cb = 0.5389\,(B - Y) \quad 5.12$$

$$Cr = 0.6350\,(R - Y) \quad 5.13$$

Some common TV broadcasting resolutions are classified in table 5.1.

Table 5.1 Classification of broadcasting video quality

Format	Label	Resolution
SD (Standard Definition)	576 interlaced / progressive	720 x 576
	480 interlaced / progressive	720 x 480
	480 interlaced / progressive	640 x 480
HD (High Definition)	720 progressive	1280 x 720
	1080 interlaced / progressive	1920 x 1080

Most image processing methods work on gray scale images because human cognition is most sensitive to brightness information in the picture (Chan, n.d.), so first step in image processing is to extract luma component of a color picture, and then necessary analyses can be carried out. When extracting luma component either BT.601 or BT.709 coefficients are used. Video format of the TV signal determines which standard is used.

5.2 Edge Detection

5.2.1 Introduction to Edge Detection

In Section 4.4 a still picture is grabbed from the TV mainboard. And second stage in thesis work is to process that image to check whether grabbed image satisfies a predefined test limit or not.

In this section edge detection methods are examined to extract features of the captured image and reference image. In order to achieve these, two major groups of edge detection methods are studied, these are traditional edge detection methods and wavelet based edge detection methods.

The first major category is traditional edge detection methods. Laplacian, Prewitt and Sobel methods are some well-known methods in this category.

The second major category is wavelet transformation. Discrete Wavelet Transformation (DWT) is used for image edge detection by help of Biorthogonal and Haar wavelets.

Points in an image where brightness changes abruptly are called edges or edge points. There are different types of sharp changing points in an image. Edges can be created by shadows, texture, geometry, and so forth. Edges can also be defined as discontinuities in the image intensity due to changes in image structure. These discontinuities originate from different features in an image. Edge points are to be associated with the boundaries of objects and other kinds of changes. Edges within an image generally occur at various resolutions or scales and represent transitions of different degree, or gradient levels.

Edge detection refers to the process of identifying and locating sharp discontinuities in an image. Edge detection is a common operation issue in image analysis. Edges can be considered as transients in a signal or mathematically defined as local singularities.

Variables involved in the selection of an edge detection operator include (Rangarajan, n.d.):

- *Edge orientation:* The geometry of the operator determines a characteristic direction in which it is most sensitive to edges. Operators can be optimized to look for horizontal, vertical, or diagonal edges.
- *Noise environment:* Edge detection is difficult in noisy images, since both the noise and the edges contain high-frequency content. Attempts to reduce the noise result in blurred and distorted edges. Operators used on noisy images are typically larger in scope, so they can average enough data to discount localized noisy pixels. This results in less accurate localization of the detected edges.
- *Edge structure:* Not all edges involve a step change in intensity. Effects such as refraction or poor focus can result in objects with boundaries defined by a gradual change in intensity. The operator needs to be chosen to

be responsive to such a gradual change in those cases. Wavelet-based techniques actually characterize the nature of the transition for each edge in order to distinguish, for example, edges associated with hair from edges associated with a face.

Traditional edge detection methods may be grouped into two categories, namely, gradient based edge detection and Laplacian-based edge detection (Rangarajan, n.d.). In the gradient based edge detection, an estimate of the gradient magnitude is calculated using the smoothing filter and the calculated estimate is used to determine the position of the edges. In other words the gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. In the Laplacian method the second derivative of the signal is calculated where the derivative magnitude is maximum when second derivative is zero. In short, Laplacian method searches for zero crossings in the second derivative of the image to find edges.

Wavelet analysis is a local analysis and is suitable for time-frequency analysis. In wavelet edge detection technique, the transform used is DWT and the filter is one which searches for the local maxima in a wavelet domain. The wavelet transform offers a multi scale analysis, which can be applied to edge detection.

Edge detection is susceptible to noise. This is due to the fact that the edge detectors algorithms are designed to respond to sharp changes, which can be caused by noisy pixels. Noise may occur in digital images for a number of reasons. The most commonly studied noises are white noise, “salt & pepper” noise and speckle noise. To reduce the effects of noise, preprocessing of the image is performed. The preprocessing can be performed in two ways, filtering the image with a Gaussian function, or by using a smoothing function. The problem with the above approaches is that the optimal result may not be obtained by using a fixed operator.

Canny proposed the operators of different widths to obtain better signal to noise ratios in the detection of patterns that appear at different scales in the image. Marr

proposed a multi resolution scheme to detect primitives related to human visual system behavior. Bergholm used Canny's operator and was able to differentiate between shadow contours from perfect ones. All of them used multi resolution scheme but had difficulty in analyzing the information appearing at different scales. After the introduction of wavelet multi resolution analysis that was based on the Gaussian kernel, the results obtained for edge detection and classification of the edges appearing in the image were better. Mallat and Zhong utilized the local maxima of the absolute value of the wavelet transform to do edge detection. Peytavin studied a multi-oriented and multi-resolution edge detection technique using a wavelet, which is the first derivative of the smoothing function. Barlaud used a biorthogonal wavelet transform for edge detection.

5.2.2 Traditional Edge Detection Methods

The majority of traditional edge detection methods may be grouped into two categories:

- *Gradient based:* The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.
- *Laplacian based:* The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location.

Suppose the signal shown in figure 5.4, with an edge shown by the jump in intensity.

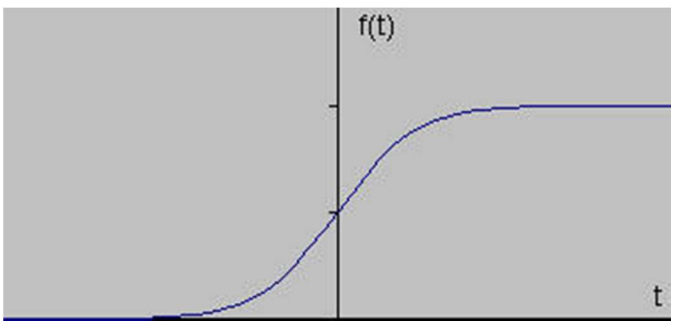


Figure 5.4 An f(t) signal with a sample intensity jump

If the gradient of signal in figure 5.4 (which, in one dimension, is just the first derivative with respect to t) is taken, the signal in figure 5.5 is get.

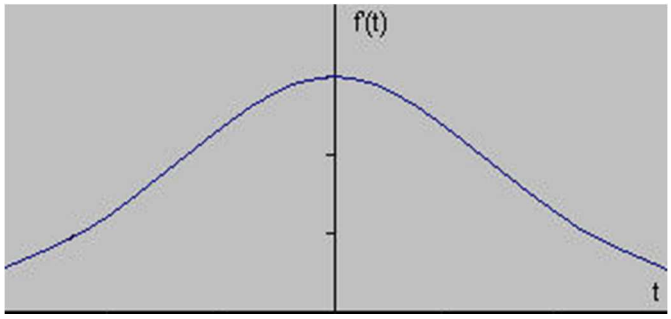


Figure 5.5 First derivative of f(t) signal, f'(t)

Clearly, the derivative shows a maximum located at the center of the edge in the original signal. This method of locating an edge is characteristic of the "gradient filter" family of edge detection filters and includes the Sobel method.

A pixel location is declared an edge location if the value of the gradient exceeds some threshold. As mentioned before, edges have higher pixel intensity values than those surrounding it. So once a threshold is set, the gradient value can be compared to the threshold value and whenever the threshold is exceeded an edge is detected. Furthermore, when the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. This method is known as the Laplacian and the second derivative of the signal in figure 5.4 is shown in figure 5.6:

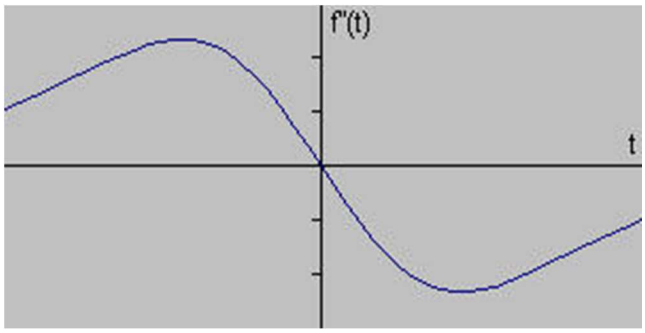


Figure 5.6 Second derivative of f(t) signal, f''(t)

Sobel operator consists of a pair of 3x3 convolution kernels as shown in figure 5.7. One kernel is simply the other rotated by 90°.

Gx =

-1	0	1
-2	0	2
-1	0	1

Gy =

1	2	1
0	0	0
-1	-2	-1

Figure 5. 7 Kernels of Sobel operator (Li, 2003)

Sobel kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these Gx and Gy). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient (Li, 2003). The gradient magnitude is given by equation 5.14:

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad 5.14$$

Typically, an approximate magnitude is computed using equation 5.15:

$$|G| = |G_x| + |G_y| \qquad 5.15$$

which is much faster to compute.

The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by equation 5.16:

$$\theta = \arctan(\frac{G_y}{G_x}) \qquad 5.16$$

The Roberts Cross operator performs a simple, quick to compute, 2D spatial gradient measurement on an image. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point (Li, 2003).

The operator consists of a pair of 2x2 convolution kernels as shown in figure 5.8. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator.

Gx =

1	0
0	-1

Gy =

0	1
-1	0

Figure 5.8 Kernels of Cross operator (Li, 2003)

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these Gx and Gy). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by equation 5.17:

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad 5.17$$

Although typically, an approximate magnitude is computed using equation 5.18:

$$|G| = |G_x| + |G_y| \qquad 5.18$$

which is much faster to compute.

The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by equation 5.19:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) - \frac{3\pi}{4} \qquad 5.19$$

Prewitt operator is very similar to the Sobel operator and is used for detecting vertical and horizontal edges in images by using the kernels in figure 5.9.

Gx =

1	1	1
0	0	0
-1	-1	-1

Gy =

-1	0	1
-1	0	1
-1	0	1

Figure 5.9 Kernels of Prewitt operator (Li, 2003)

The Laplacian is a 2D isotropic measure of the second spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian smoothing

filter in order to reduce its sensitivity to noise. The operator normally takes a single gray-level image as input and produces another gray-level image as output.

The Laplacian L(x,y) of an image with pixel intensity values I(x,y) is given by equation 5.20:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \qquad 5.20$$

Since the input image is represented as a set of discrete pixels, a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian has to be found. Two commonly used small kernels are shown in figure 5.10.

Type I =

0	1	0
1	-4	1
0	1	0

Type II =

1	1	1
1	-8	1
1	1	1

Figure 5.10 Two commonly used Laplacian kernels (Fisher, Perkins, Walker, & Wolfart, 2003)

Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian smoothed before applying the Laplacian filter. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

In fact, since the convolution operation is associative, the Gaussian smoothing filter can be convolved with the Laplacian filter first of all, and then this hybrid filter is convolved with the image to achieve the required result. Doing things this way has two advantages:

- Since both the Gaussian and the Laplacian kernels are usually much smaller than the image, this method usually requires far fewer arithmetic operations.
- The LoG (Laplacian of Gaussian) kernel can be pre-calculated in advance so only one convolution needs to be performed at run-time on the image.

The 2D LoG function centered on zero and with Gaussian standard deviation σ is represented as in equation 5.21:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad 5.21$$

and is shown in figure 5.11.

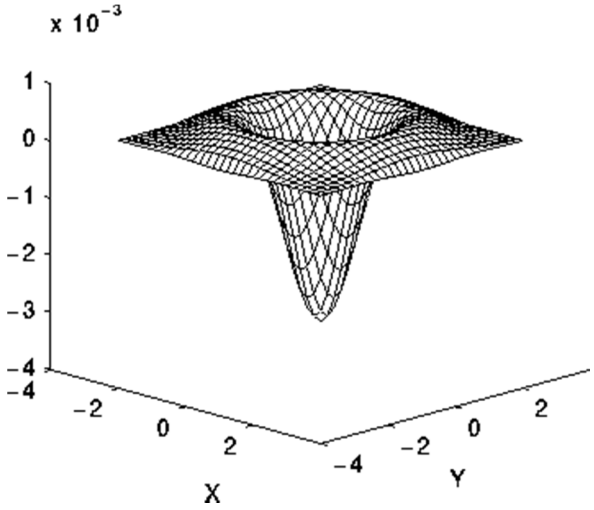


Figure 5.11 A 2D LoG function centered on zero and with Gaussian standard deviation σ (Fisher & others, 2003)

The kernel in figure 5.12 shows a discrete approximation to LoG function with Gaussian σ = 1.4

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figure 5.12 A discrete approximation to LoG function with Gaussian σ = 1.4

Note that as the Gaussian is made increasingly narrow, the LoG kernel becomes the same as the simple Laplacian kernels shown in figure 5.10. This is because smoothing with a very narrow Gaussian ($\sigma < 0.5$ pixels) on a discrete grid has no effect. Hence on a discrete grid, the simple Laplacian can be seen as a limiting case of the LoG for narrow Gaussians (Fisher & others, 2003).

Gradient-based algorithms such as the Prewitt filter have a major drawback of being very sensitive to noise. The size of the kernel filter and coefficients are fixed and cannot be adapted to a given image. An adaptive edge-detection algorithm is necessary to provide a robust solution that is adaptable to the varying noise levels of these images to help distinguish valid image contents from visual artifacts introduced by noise.

The performance of the Laplacian method depends heavily on the adjustable parameter, σ, which is the standard deviation for the Gaussian filter. The bigger the value for σ, the larger the size of the Gaussian filter becomes. This implies more blurring, necessary for noisy images, as well as detecting larger edges. As expected, however, the larger the scale of the Gaussian, the less accurate is the localization of the edge. Smaller values of σ imply a smaller Gaussian filter which limits the amount of blurring, maintaining finer edges in the image. The implementation can tailor the algorithm by adjusting this parameter to adapt to different environments.

Figure 5.13 and figure 5.14 show two original images together with some edge operators applied those original images. A simple LabVIEW application is written to perform these operations.

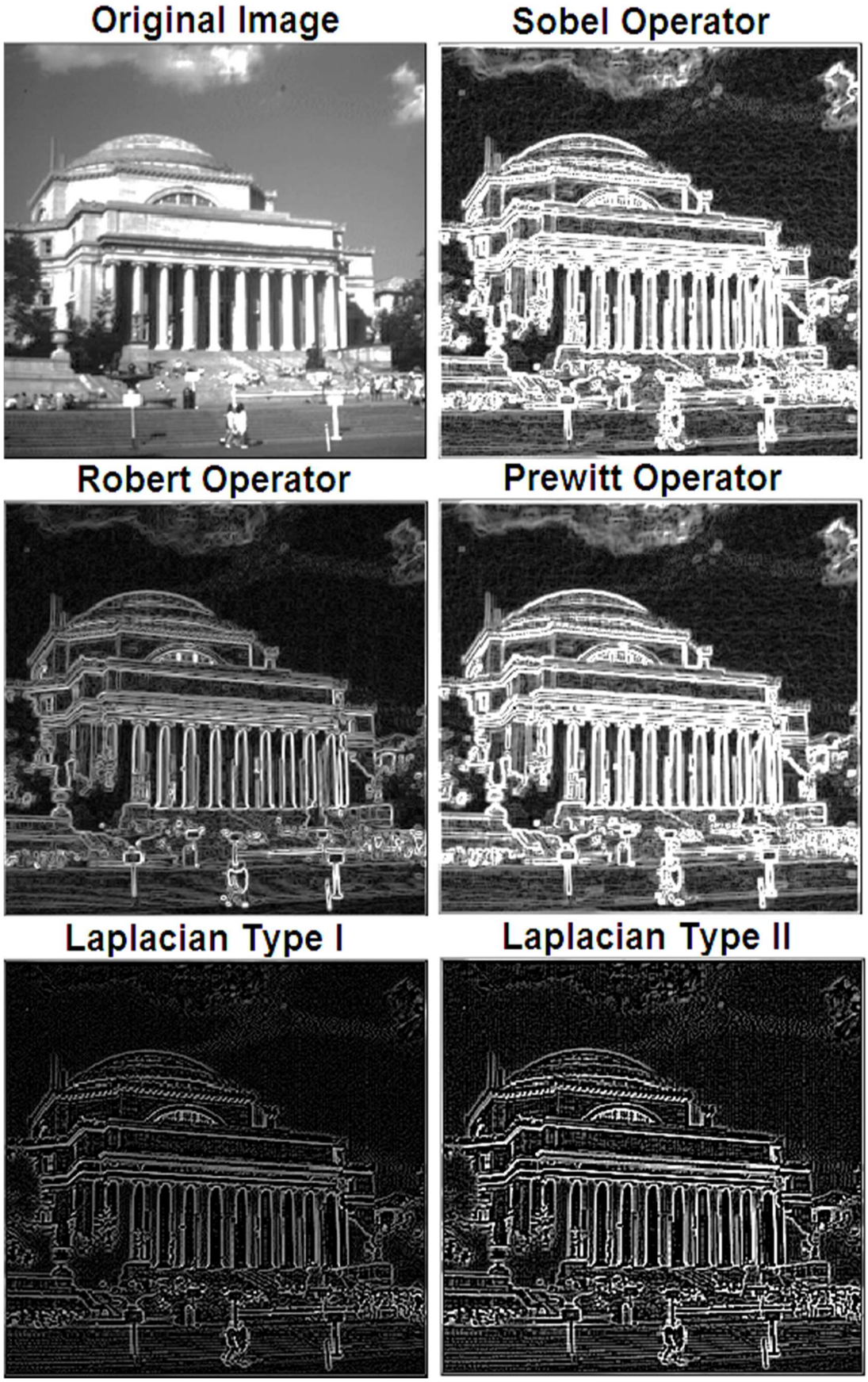


Figure 5.13 Image edge detection with traditional methods

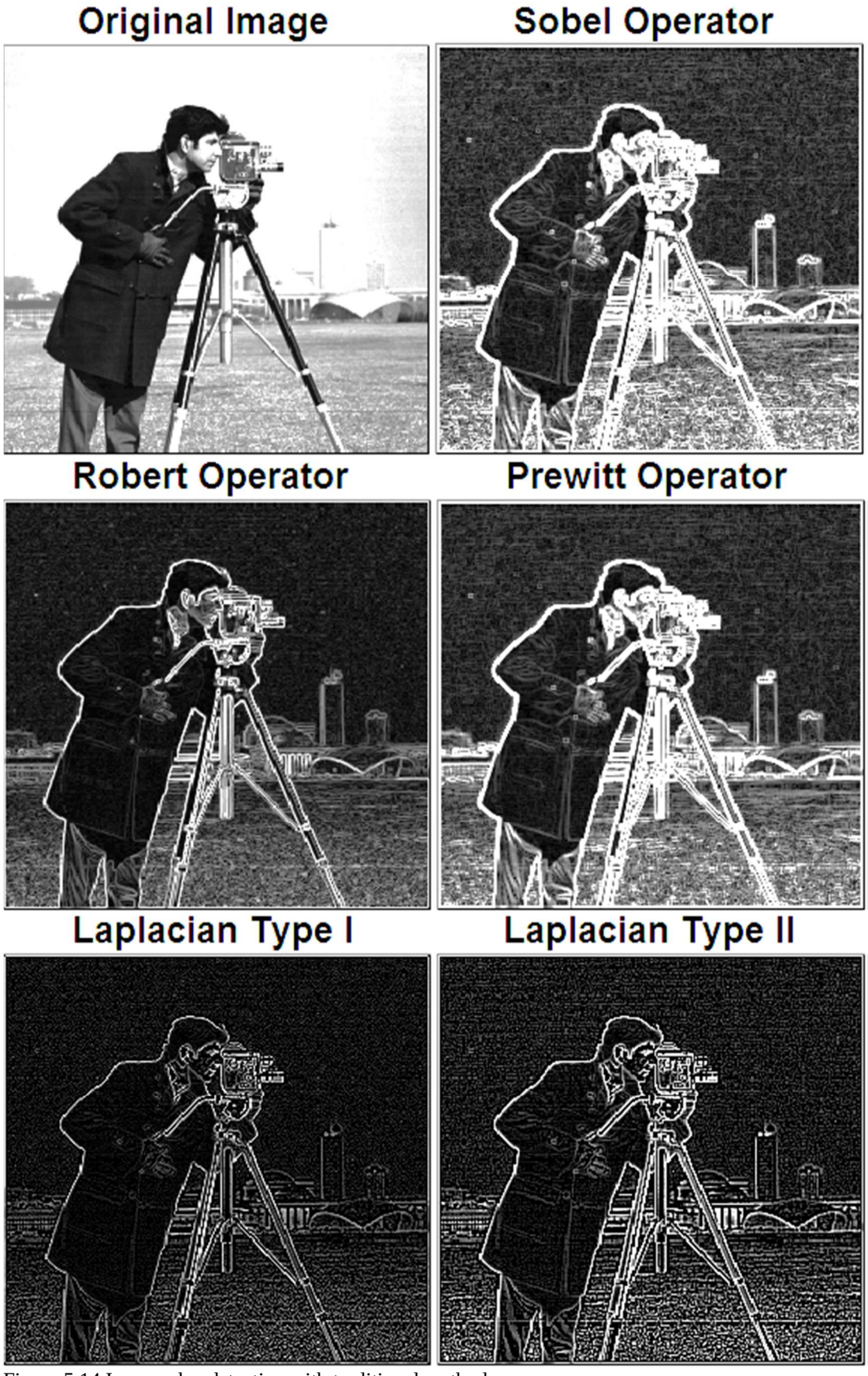


Figure 5.14 Image edge detection with traditional methods

Sobel and Prewitt operators create results that resemble each other very much, but in general Sobel operator creates cleaner edges. Robert operator seeks for the edges running 45^{o} at the pixel grid, so it produces a good result in cameraman example. When compared with other operators, Laplacian operators produce sharper edges, this can be clearly seen on the cameraman example; even the grass on the ground produces some edges.

5.2.3 Wavelet Based Multi Resolution Edge Detection

Extracting relevant features is a key step when signals and images are analyzed and interpreted. Signals and images are characterized by local features, such as peaks, edges, and breakdown points. The wavelet transform-based methods are typically useful when the target features consist of rapid changes, such as the sound caused by engine knocking or edges in an image. Wavelet signal processing is suitable for extracting the local features of signals because wavelets are localized in both the time and frequency domains.

Wavelet analysis is a local analysis and is suitable for time-frequency analysis. In wavelet edge detection method, the transform used is DWT and the filter is one which searches for the local maxima in a wavelet domain. The wavelet transform offers a multi-scale analysis, which can be applied to edge detection.

The concept of DWT was first introduced by Stromberg from the Littlewood-Paley decomposition of operators a function. The two dimensional DWT descends from the Laplacian pyramid scheme of Burt and Adelson as shown in figure 5.15:

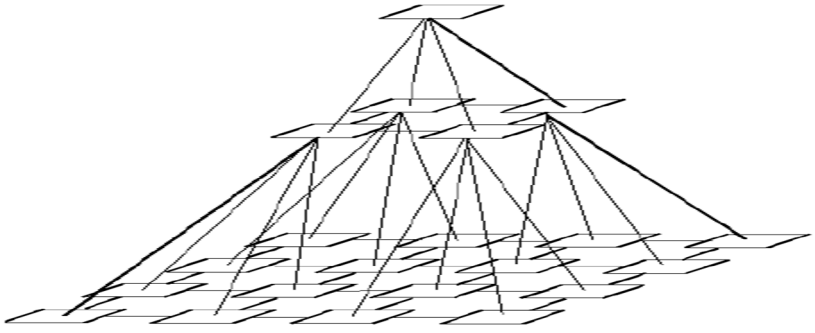


Figure 5.15 Descend of 2D DWT from Laplacian pyramid (Chaganti, 2005)

If the function being expanded is a sequence of numbers, like samples of a continuous function f(x), the resulting coefficients are called the discrete wavelet transform (DWT) of f(x). DWT transforms a discrete time signal to a discrete wavelet representation.

The DWT is not calculated in terms of matrices due to the issue of storage; instead filters are used to compute the DWT.

Let a = {.,a_{-1} , a_0, a_1,.} be a sequence and let's consider two filter impulses h and g.

The two convolutions are defined as in equation 5.22:

$$(Ha)_k = \sum_l h_{l-2k} a_k \qquad (Ga)_k = \sum_l g_{l-2k} a_k \qquad 5.22$$

where H and G corresponds to one step of the DWT (up or down)

In the decomposition, the function is successively convolved with the two filters H (low frequencies) and G (high frequencies). Each resulting function is decimated by suppression of one sample out of two. The high frequency signal is left, and same iteration is done with the low frequency signal.

In the reconstruction, the sampling is restored by inserting a 0 between each sample, then it is convolved with the conjugate filters H~ and G~, the resulting functions are added and the result is multiplied by 2. Iteration is repeated up to the smallest scale (Chaganti, 2005).

The 2D algorithm is based on separate variables leading to prioritizing of x and y directions. The scaling function is defined by equation 5.23:

$$\phi(x, y) = \phi(x)\, \phi(y) \qquad 5.23$$

The detail signal is obtained from three wavelets using equations 5.24, 5.25 and 5.26:

- A vertical wavelet: $\Psi^1(x, y) = \phi(x)\, \phi(y)$ 5.24

- A horizontal wavelet: $\Psi^2(x,y) = \Psi(x)\,\phi(y)$ 5.25
- A diagonal wavelet: $\Psi^3(x,y) = \Psi(x)\,\Psi(y)$ 5.26

which leads to three sub-images in each of the decomposition levels as shown in figure 5.16:

<table>
<tr><td>...</td><td>H.D.
j = 2</td><td rowspan="2">Horizontal
Details
j = 1</td><td rowspan="2">Horizontal Details
j = 0</td></tr>
<tr><td>V.D.
j = 2</td><td>D.D.
j = 2</td></tr>
<tr><td colspan="2">Vertical
Details
j = 1</td><td>Diagonal
Details
j = 1</td></tr>
<tr><td colspan="3">Vertical Details
j = 0</td><td>Diagonal Details
j = 0</td></tr>
</table>

Figure 5.16 Derivation of sub-images in each wavelet decomposition level (Chaganti, 2005)

Significant intensity changes in an image normally occur at different spatial resolution or scales. Traditional edge detectors select a special spatial mask that detects edges at a particular resolution. A small mask edge detector is susceptible to noise and produces spurious edges. In contrast, to that a large mask edge detector is relatively robust to noise, but distorts the edges and may not detect some finer details. Thus it is very difficult to detect edges with a single spatial edge mask.

The edge preserved denoising lends itself to a wavelet-based procedure for edge detection. The edge in a signal gives rise to peaks in the high pass filter outputs or the detail sub-bands at concordant locations. This is a characteristic of the DWT. In other words, edges give rise to peaks across several levels of details at coordinate

values that moves to the left by a factor of one half at every transition from a finer scale to a coarser scale. The stronger the edge, the higher are the peaks in the DWT. Consequently, an edge can be found from the wavelet transform by identifying peaks at concordant locations.

Each pixel in an image has 8 other neighboring pixels, two in the horizontal direction, two in the vertical direction and four in the diagonal direction. If at a given location, the variation of the pixel value is small in the vertical direction but large in the horizontal direction, then the pixel is a vertical edge point of the image. Similarly if the variations in the horizontal direction pixel values are smaller and the vertical direction pixel vales are larger, then the pixel is a horizontal edge point. If the variations in the pixel values of both the horizontal and the vertical are large, then the pixel is a diagonal edge point of the image. Since the DHT (Discrete Haar Transform) of matrices involves computing averages and differences of adjacent pixel values in various combinations, the edges in an image can be found by using the DHT (Chaganti, 2005).

Biorthogonal wavelets are symmetric wavelets with compact support unlike Daubechies wavelets which are asymmetric. Biorthogonal wavelets provide perfect reconstruction of the signal, without redundancy, using two sets of wavelets. One set of wavelets is used for the decomposition and the other is used for reconstruction.

Biorthogonal wavelets have the smallest supported set. They are capable of detecting singular data. It has the two way capability, one is to smooth data with a B-spline scaling function and another is to detect distributed data with biorthogonal wavelets. So this specific property provides the foundation for image edge detection based on the biorthogonal wavelet transform. Three degree biorthogonal wavelets are used to detect edges in an image. The wavelet bases constructed by the biorthogonal wavelet functions translation and dilation are semi-orthogonal wavelet bases. Therefore these wavelets are orthogonal to scaling functions at the same level but not orthogonal to each other.

In the process of finding edges by biorthogonal wavelets, the image is decomposed into four weights at scale 2^j:

Discrete approximation signal of image: $S_{2^j} f(x,y)$

Discrete detail signal of image: $W_{2^j}^h f(x,y),\ W_{2^j}^v f(x,y),\ W_{2^j}^d f(x,y)$

The local maxima in the position of horizontal detail image $W_{2^j}^h f(x,y)$ and vertical detail image $W_{2^j}^v f(x,y)$ are corresponding to the image edge point in horizontal and vertical directions.

At the scale 2^j, the value of $M_{2^j} f(x,y)$ is given by equation 5.27:

$$M_{2^j} f(x,y) = \sqrt{\left|W_{2^j}^v f(x,y)\right|^2 + \left|W_{2^j}^h f(x,y)\right|^2} \qquad 5.27$$

The local maxima point means that this point value $M_{2^j} f(x,y)$ is larger than of these closed points. The edge of the image can be detected at every scale. When the scale of the biorthogonal wavelet increases, the ability of reducing the noise also increases (Chaganti, 2005).

In the thesis, biorthogonal and Haar wavelets are used to detect edges in the image. Usually, the modulus-maxima of the detail coefficients of signal correspond to the edge of signal**.** Main wavelet edge detection function used in LabVIEW completes the following steps to implement the multi-scale edge detection:

- Performs the undecimated wavelet transform with chosen wavelet on the rows and the columns, respectively, at each level. Obtains the detail coefficients along the row direction $W_l^r(i,j)$ and the column direction $W_l^c(i,j)$. (i, j) denotes the row and column indexes of the matrix, respectively.
- Computes a new matrix R_l using equation 5.28:

$$R_l(i,j) = \sqrt{|W_l^r(i,j)|^2 + \left|W_l^c(i,j)\right|^2} \qquad 5.28$$

- Finds the local maxima of R_l as the edge points at level l.

LabVIEW presents some ready to use functions to the user like MATLAB does. In this case, the ready functions in LabVIEW contain some error-preventing mechanisms which slow down the code execution speed, so edge detection algorithm and its relative sub-functions are completely re-arranged to maximize the code execution speed.

Figure 5.17 and figure 5.18 show two original images together with wavelet based edge detection applied those original images. A comprehensive LabVIEW application is re-written to perform these operations.

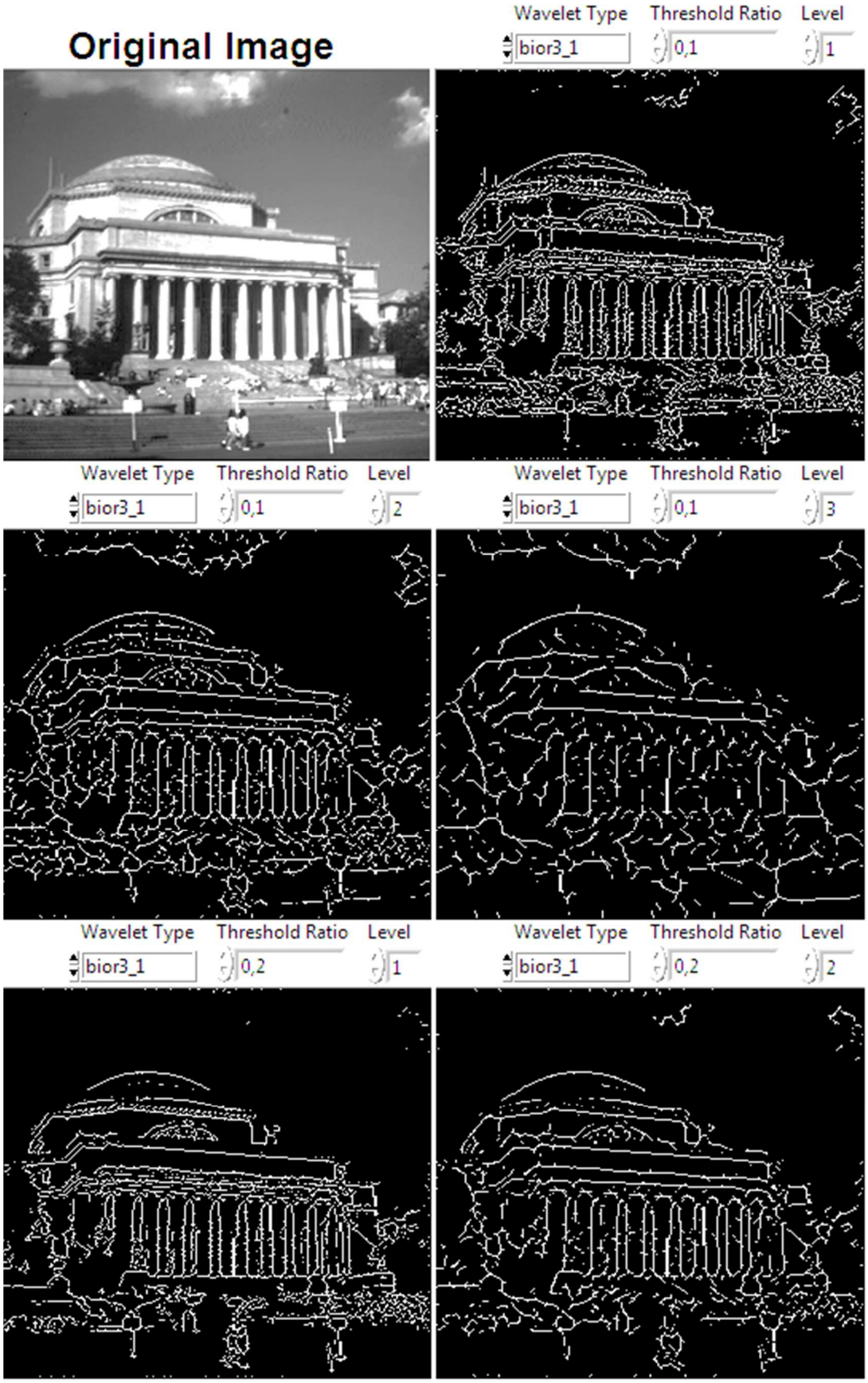


Figure 5.17 Wavelet based image edge detection with biorthogonal wavelet

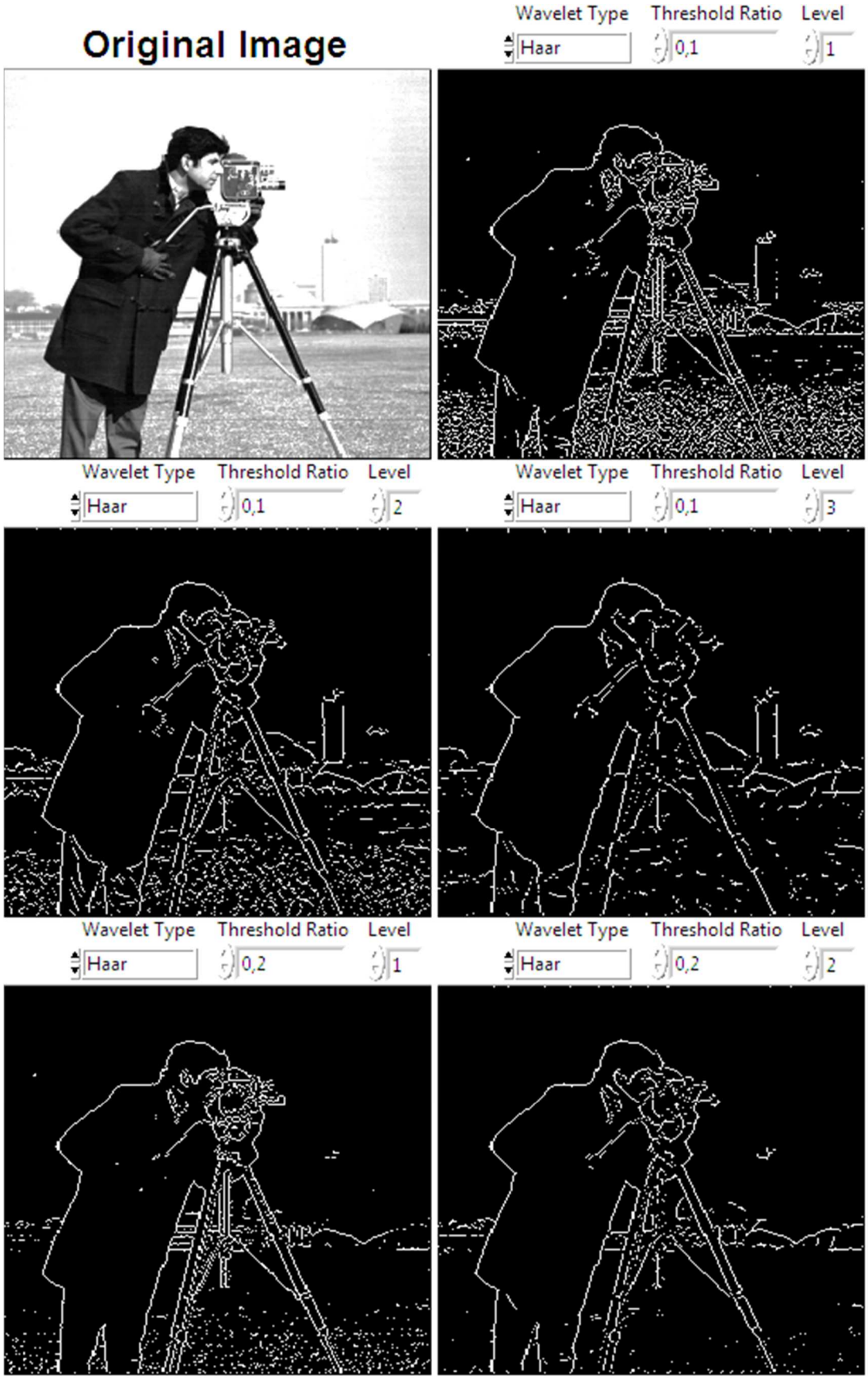


Figure 5.18 Wavelet based image edge detection with Haar wavelet

Figure 5.17 and figure 5.18 show two images and their associated edge maps detected at different levels of resolutions using the wavelet transform-based method. Conventional methods process an image at a single resolution and return a binary edge map. The wavelet transform-based method processes an image at multiple levels of resolution and returns a series of grey-level edge maps at different resolutions.

A large level value corresponds to an edge map with low resolution. The global profile of the image can be obtained in a low-resolution edge map and the detailed texture of the image can be obtained in a high-resolution edge map. A multi-resolution edge detection method can also be formed by examining the edge maps from the low resolution to the high resolution. With the multi-resolution edge detection method, an object of interest can be located in the image reliably and accurately, even under noisy conditions.

In wavelet based edge detection there are two important parameters, the threshold value and the level of scale. The threshold value determines the threshold level for finding local maxima and minima. As previously mentioned wavelet based edge detection is performed in multi-scale. The level of scale determines the scale of wavelet decomposition of the original image.

As can be seen from figure 5.17 and figure 5.18, increase in threshold ratio in same scale reduces number of unwanted edges. Increase in the scale level also increases the area of minimal edge detection regions. In other words this can also be perceived like someone is looking at the image to see the edges in the image from a further distance.

Edge detection is a very important problem in image processing. It is clear that wavelets are more functional and better edge detectors both theoretically and experimentally. For these reasons, wavelet based image edge detection method is used in the thesis for image edge detection.

Although the wavelet transform is a good de-correlator for images, the wavelet coefficients of natural images exhibit strong dependencies both across scales and between neighbor coefficients within a sub-band, especially around image edges. This gave rise to several successful joint statistical models in the wavelet domain as well as improved image compression schemes. The major drawback for wavelets in two-dimensions is their limited ability in capturing directional information such as curvature structures in real-world images.

5.3 PSNR: A Fast Image Comparison Algorithm

Image quality measuring plays an important role in a variety of image processing applications. Very often the quality of an image needs to be quantified. This can be done by subjective testing sessions, or by objective-computational metrics. The aim of the objective metrics is to predict how much of the distortion is observed by user. For this reason many metrics based on different principles have been developed .

Subjective methods for digital image quality assessment are defined in ITU-R Rec. BT.500-11. More methods are still being developed; some of them are included in ITU-R Report BT.1082. The principle of subjective methods is that groups of assessors (or even a single assessor) judge the quality of an image being presented to them. Subjective methods are the most accurate in determining “how much” of image distortion can be perceived, and thus can be a measure of the performance of objective assessment methods. The disadvantage of these subjective methods is clear: they are expensive and impossible to be included in automatic systems (e.g. setting parameters of a system according to the instant output image video quality).

In this part of the thesis, only objective computational methods are discussed. In other words, there is always a “perfect” image at hand to compare the quality of a distorted one.

The simplest and fastest objective assessment methods are statistically defined mean squared error (MSE) and peak signal to noise ratio (PSNR) (Slanina & Ricny,

n.d.). These methods are pixel-based, i.e. the distorted picture and the reference are compared pixel-by-pixel. The MSE is computed using equation 5.29:

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - K(i,j)\,]^2 \qquad 5.29$$

Where m by n is the size of the image and I(i,j) and K(i,j) are the luminance values of the reference and the distorted image, respectively. PSNR can easily be computed using equation 5.30 (Slanina & Ricny, n.d.):

$$PSNR = 10log_{10}\frac{M^2}{MSE}\,[dB] \qquad 5.30$$

where M is the maximum pixel value (e.g. 255 for 8-bit images).

Another different approach to PSNR is that: as the human visual system is not equally sensitive to all spatial frequencies, a contrast sensitivity function (CSF) is taken into account. The CSF is simulated by a low-pass or band-pass frequency filter.

First of all, the difference of the reference and the distorted image is computed. Then the difference is transformed into frequency domain using 2D Fast Fourier transform. The obtained error spectrum is weighted by the CSF resulting in weighted error spectrum. The last thing to do is to compute the power of the weighted error spectrum and the power of the signal (also transformed into frequency domain). This method is called as weighted signal to noise ratio (WSNR) (Slanina & Ricny, n.d.).

Although it is clear that WSNR is an advanced version of PSNR, this advancement comes in more complexity in calculation. One of the most important reason in selection of PSNR as picture quality assessment method is that it has less computational complexity when compared with WSNR and other picture quality assessment methods, but still produces satisfactory results.

Application of PSNR algorithm is shown in figure 5.19. The image under test is derived from the reference image by smoothing the reference image by a 3x3 Gaussian filter with given deviations.

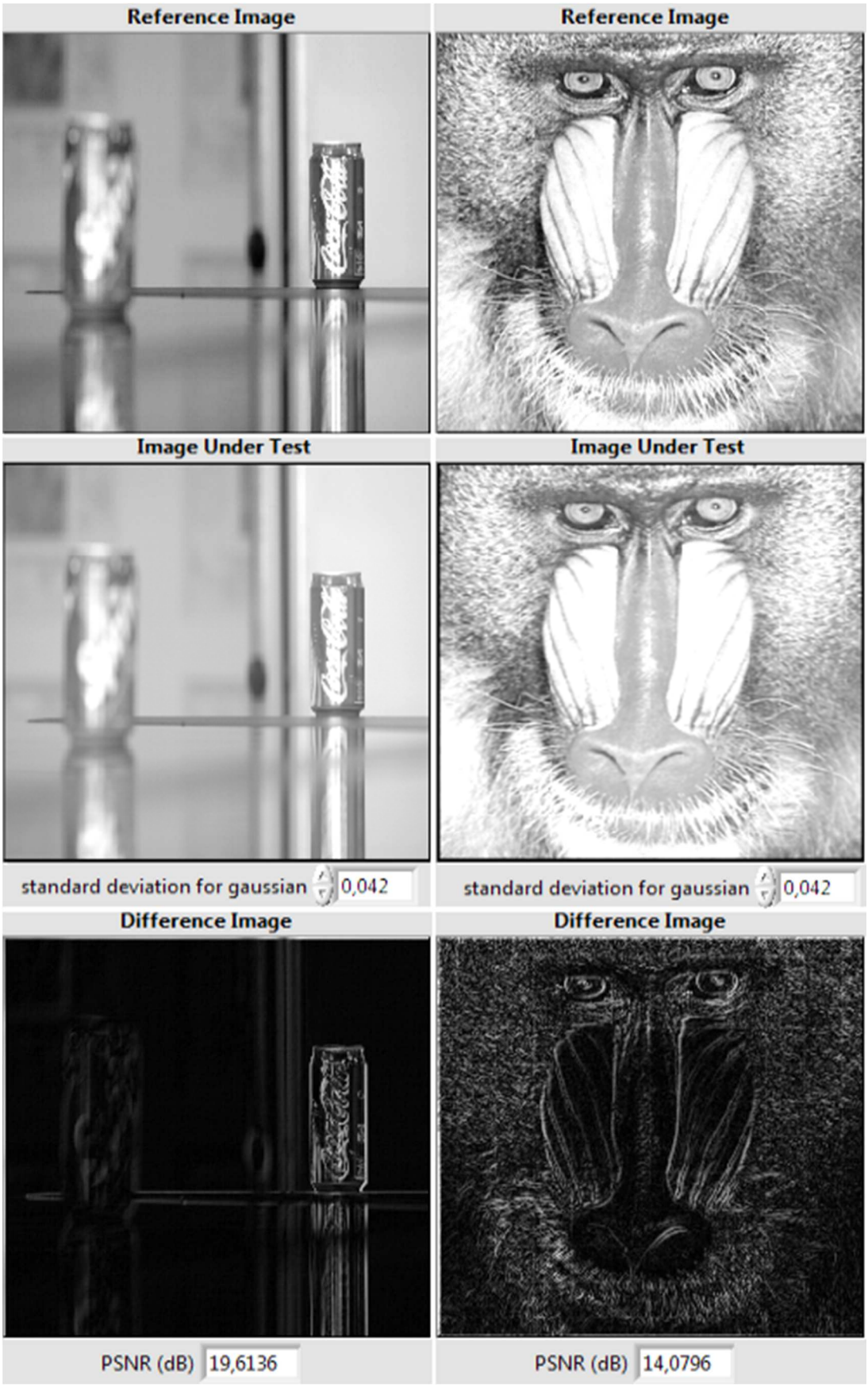


Figure 5.19 Image comparison with PSNR algorithm

At first glance difference between reference image and image under test is just seen as a brightness difference, but actually this is not the truth. The images under test are smoothed versions of reference images, and this is clearly seen in difference images. In such a scenario PSNR results indicate that compared pictures are different from each other and this is not just a brightness difference. PSNR value is expressed in dB, and if two images are identical the PSNR value is infinity, but this is not usually the case especially for compression algorithms.

5.4 Software Application: Image Processing on LVDS Data

In previous sections of this chapter various image processing algorithms are discussed and some examples are presented. In image processing part of the thesis PSNR (Peak Signal to Noise Ratio) algorithm and wavelet based image edge detection are implemented in software, but so far performance of these algorithms are not evaluated.

Performance of image processing algorithms is also important together with their abilities in current application area. To evaluate the performance metrics of the algorithms discussed in this chapter, a series of benchmarks are performed with some sample images. For this purpose, mean execution time of the algorithms are measured programmatically and they are shown at table 5.2. It must be kept in mind that the results of these measurements highly depend on the processing power of the PC that runs the benchmark and to minimize the non-deterministic behavior of the PC operating system, benchmark programs are run at time critical priority level and five measurements are taken and averaged for each test item.

Table 5.2 Comparison of mean execution times for different image processing algorithms

PC Specification	Intel Core 2 Duo 2.0 GHz processor with 4MB L2 cache 667MHz DDR2 RAM		
Picture Size	**Image Processing Methods and Mean Execution Times (milliseconds)**		
	Haar Wavelet 1st Level	*Haar Wavelet 2nd Level*	*Haar Wavelet 3rd Level*
256 x 256	48	101	162
280 x 420	82	172	272
	Biorthogonal 3_1 1st Level	*Biorthogonal 3_1 2nd Level*	*Biorthogonal 3_1 3rd Level*
256 x 256	49	104	168
280 x 420	86	177	282
	Laplacian Type I/II	*Traditional Methods*	*PSNR*
256 x 256	28	70	28
280 x 420	92	144	94

Another important thing to note here is that the second level cache memory size of the computer CPU. At table 5.2, picture sizes are small, and the entire picture can fit into the L2 (Level 2) cache memory of the CPU, but when the picture size exceeds the size of this memory an instant increase in calculation times are observed.

Change in threshold values for wavelet analysis does not affect the execution time of the process, so it is kept constant while performing benchmarks. For traditional methods a 3x3 convolution kernel is used, so they all produce the same computational complexity for the computer CPU.

When table 5.2 is examined, it is clearly seen that the first level decomposition of wavelets for image edge detection yields better execution time results than traditional methods except Laplacian, although wavelet analysis is more complicated than traditional methods it computes faster than traditional methods, this is most probably the result of better utilization of extended command sets of the CPU. Second level wavelet decomposition still offers an acceptable performance when compared with

traditional methods and the outcome of wavelet analysis always yields visually better results.

PSNR calculation stands in a different place, it is capable of comparing two images pixel by pixel. It is a very simple, fast and powerful tool as can be depicted from previous examples and table 5.2.

As a result, wavelet based edge detection method and PSNR calculation are chosen as two different image processing methods in the thesis.

A screenshot from picture compare (PSNR) tab of the PC application software is shown in figure 5.20. Upper left picture is the captured image from the TV mainboard while the lower left picture is taken from a golden TV mainboard sample. A scart signal generator is used in this case as signal input to the TV, so it is marked as SD content as can be seen. The first step in PSNR analysis is to separate the original color images into their Y, Cb and Cr components, here ITU standard defines different coefficients for SD and HD contents, so the selection must be done correctly. After that Y, Cb and Cr components of captured image is subtracted from Y, Cb, and Cr components of reference image, and images at the right hand side are got. Then PSNR calculation is performed on between Y components of captured and reference image (because human eye is most sensitive to the luma component), and measurement value is found as dB. For example in figure 5.20 test limit threshold is set as 60 dB, result is around 10 dB, hence the test is failed. The first noticed difference between two images at left is the info bar at the bottom, but this is not the only difference as can be seen from difference pictures, in this case PSNR is a really useful calculation method to describe these differences in a quantitative way.

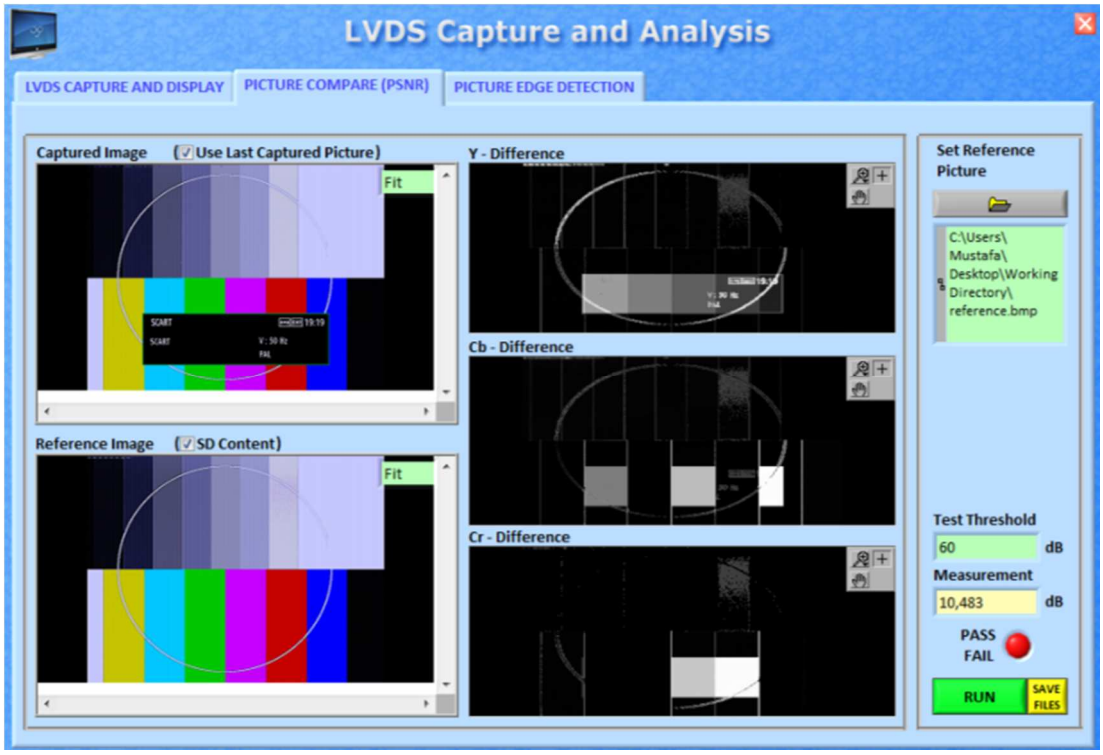


Figure 5.20 Picture compare (PSNR) tab of the PC application software

A screenshot from picture edge detection tab of the PC application software is shown in figure 5.21. Upper left picture is the captured image from the TV mainboard while the lower left picture is taken from a golden TV mainboard sample. A scart signal generator is used in this case as signal input to the TV, so it is marked as SD content as can be seen. The first step in edge detection analysis is to separate the original color images into their Y components, here ITU standard defines different coefficients for SD and HD contents, so the selection must be done correctly. After that wavelet based edge detection algorithm is calculated over Y components of reference and captured image and the results are displayed at right hand side. Here Haar or Biorthogonal 3_1 wavelet can be selected. Wavelet threshold and decomposition level must be determined specifically for different reference images. Then total number of different edge points between reference and captured image is found and displayed. As the PSNR calculation shows in figure 5.20, there is a visible difference around the center circle between two images; also info bar at the bottom is clearly showed as a difference. Here there are 27,230 different edge points between two images. The first noticed difference between two images at left is the info bar at the bottom, but this is not the only difference as can

be seen from difference picture, in this case edge detection presents another useful calculation method to describe these differences in a quantitative way.

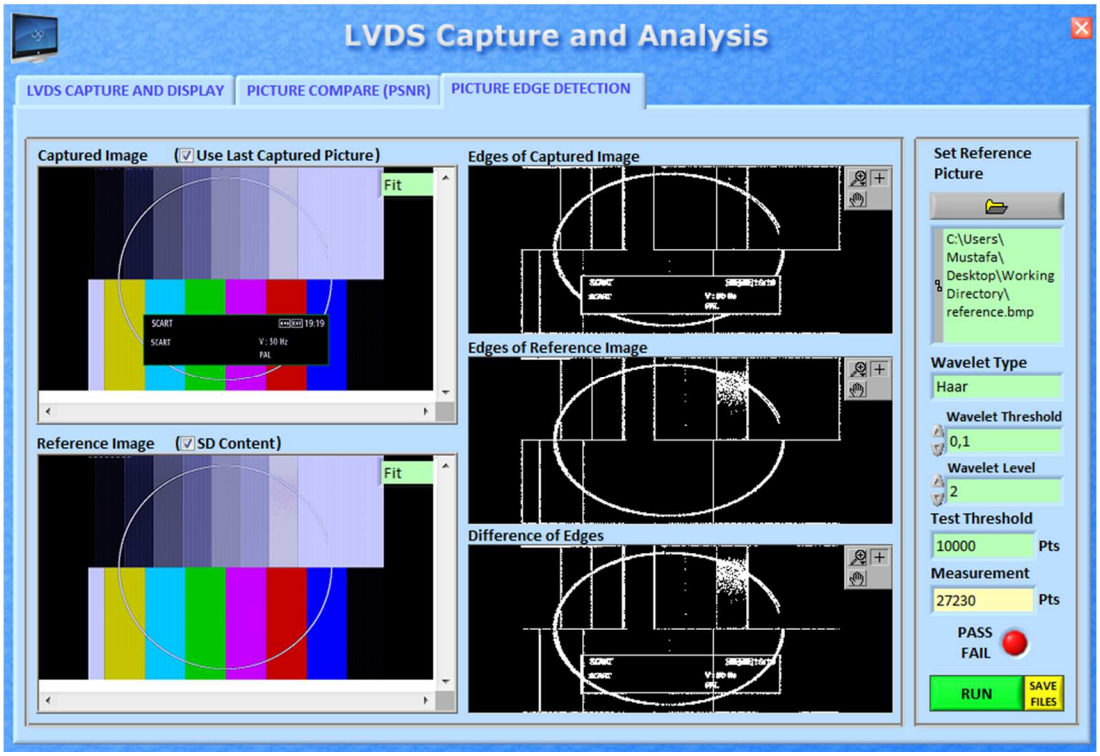


Figure 5.21 Picture edge detection tab of the PC application software

In common, resultant pictures of both analysis methods can be saved for later examination, and comparison algorithms can be run with different reference pictures without capturing from the TV mainboard again.

CHAPTER SIX
SAMPLE RUNS ON REAL TELEVISIONS

In this section some sample runs are performed on different size (HD Ready and Full HD) TVs. Up to this point it is never mentioned that but in this section a design verification check is also explained.

The system always captures an image from TV mainboard but how one can know that the captured image is 100 percent correct? This can be proven by using a pre-defined static pattern generated by a highly accurate digital pattern generator.

An Astro branded VG-849 model HDMI pattern generator is used as a verification device. This device is capable of generating and transmitting user defined static pattern data in a wide range of resolutions with a great accuracy due to the digital nature of HDMI interface. Color bar pattern in figure 6.1 is used as a verification pattern in HD Ready (1366 x 768) and Full HD (1920 x 1080) resolutions. This pattern is formerly loaded into the signal generator as a test program, so it was already in device. These eight different colors are full power combinations of basic RGB colors, so $2^3 = 8$ different colors are produced. Before testing the system on a different TV, this pattern is applied to the TV, and a few captures are taken. Firstly, all captured images must be binary identical to each other, this can be easily checked with a regular file comparison program, like Beyond Compare. Secondly, one of these files is fully controlled with pre-written software, because all color values are known. During these sample runs no problems are observed related to correctness of the captured data.

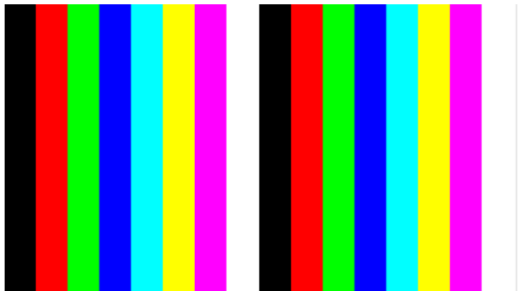


Figure 6.1 Color bar pattern used for picture grabbing card hardware verification

Test setup in figure 6.2 is prepared to perform testing. The setup consists of following components:

- The picture grabbing card designed in the thesis work
- A PC which contains the application software designed in the thesis work
- A TV whose back cover is open to reach its LVDS output
- A USB flash drive with sample patterns in it
- Power supply to the grabbing card
- Ethernet cable which connects grabbing card to the PC
- An LVDS cable which connects LVDS output of TV to the grabbing card
- An RS-232 cable which is necessary to upload embedded software to the grabbing card for first time after auto insertion of components on the grabbing card, the software of the grabbing card can be updated at any time via this interface without de-soldering any component

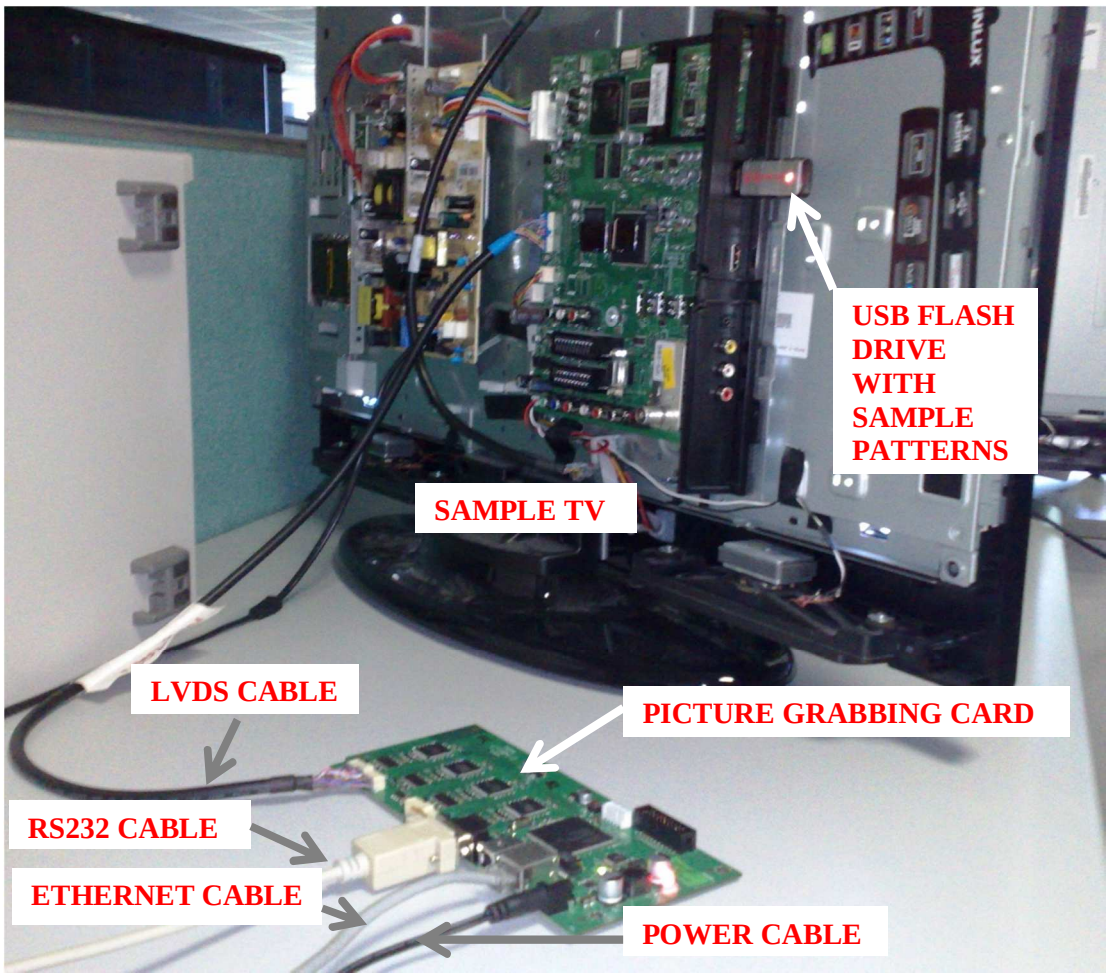


Figure 6.2 Test setup to grab images from TV mainboard

Examples in figure 6.3, figure 6.4 and figure 6.5 are taken while PIP (Picture In Picture) mode is enabled in TV. The main video source is analog RF (Radio Frequency) input signal, and the source of smaller video window in main picture is HDMI. Although it is not easily visible, both analog video (RF input) and digital video (HDMI input) in reference and captured pictures are different from each other.

Figure 6.3 shows the PSNR based processing. The difference between two pictures is roughly visible, but their HDMI inputs are truly different from each other, but difference between color bar is also visible.

Figure 6.3 Application of PSNR based image comparison on sample images

In figure 6.4, image edge detection is performed on same pictures with a threshold value of 0.01. In this case the difference between color bar signals is clearly visible, but there is much more difference between small windows, nearly 18,000 different edge points are present.

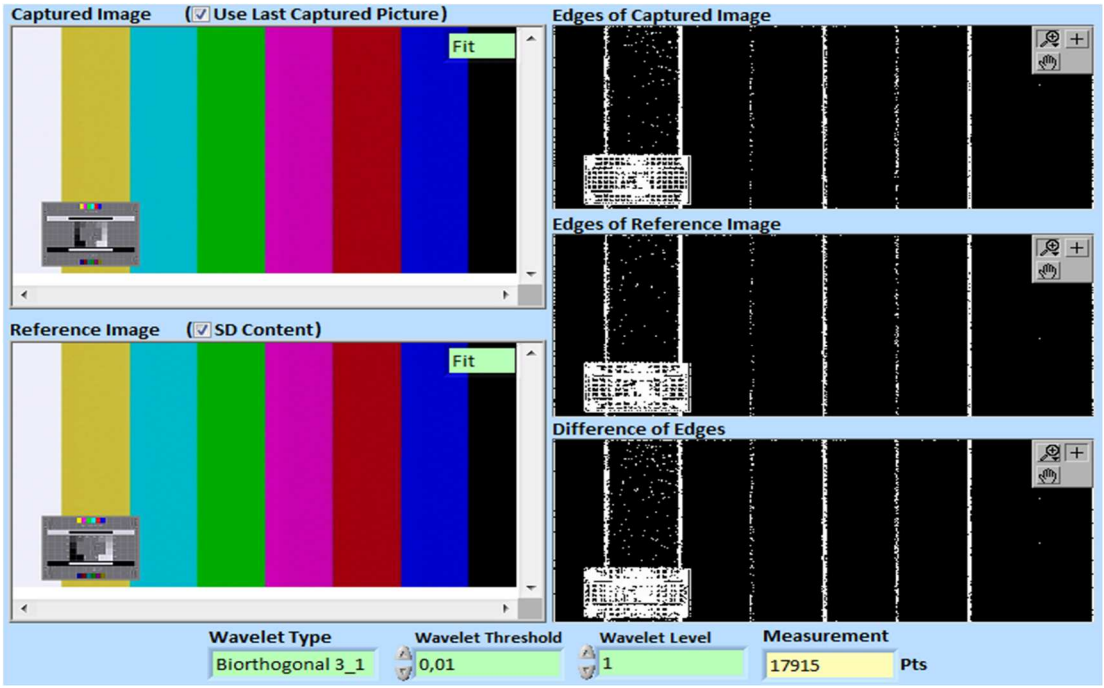


Figure 6.4 Application of wavelet based image edge detection on sample images

The question is if the user only concern with small window, how could the pattern signal be ignored? This is possible with wavelet based edge detection as shown in figure 6.5. This time wavelet threshold is set to 0.1, which is ten multiple of the previous value, and difference between color bar signals is lost and total number of different edge points drops to nearly 8000.

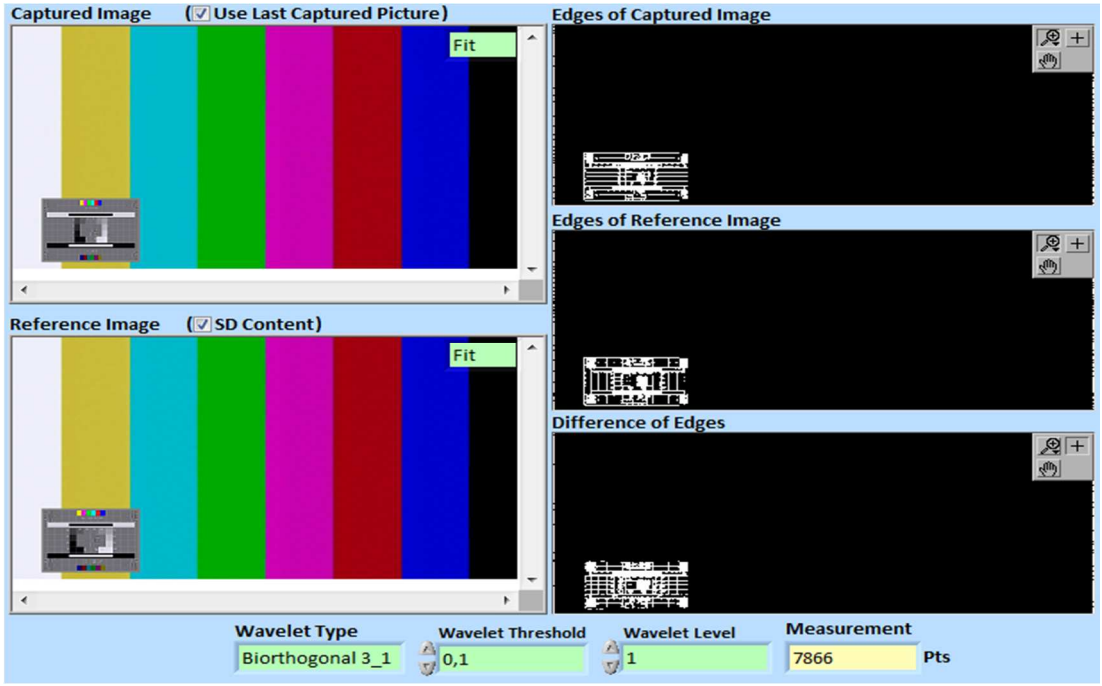


Figure 6.5 Application of wavelet based image edge detection on sample images

CHAPTER SEVEN
CONCLUSION AND FUTURE WORK

The main goal of this thesis work is to design an embedded system which is capable of receiving LVDS data output of an LCD TV, decoding and transmitting it to the PC for analysis.

The most overwhelming task in the thesis work is the hardware development and implementation part. It is designed from the bottom without any reference about the system structure. The embedded system block diagram, full hardware schematics and PCB design is given at appendix for reference. The PCB has four layers and designed in a way that only one side of that is used for component insertion which makes its production easier.

The second challenging part in the design is embedded software development. Although NXP's LPC2468 is a RISC machine, it has a highly complex structure which makes the software development quite difficult. The embedded code is written as single-process hence there is no operating system running on the base to speed up the software development. Size of the embedded software reaches nearly 3000 lines of C code.

The third part in the thesis is to develop a PC application to communicate with the designed hardware. This part is easier when compared with other two parts, but this time the processing speed of the PC must stay in some acceptable range, for this reason some benchmarks are performed to measure the performance.

Currently the picture grabbing card captures a 1366 x 768 image approximately in 1 second and it takes around 2 seconds to capture a 1920 x 1080 image. LVDS standard of the panel (VESA or JEIDA) also affects capturing time at around 15 percent. Hardware of the card is designed in accordance with VESA standard, so capturing images from JEIDA LVDS output increases total time due to internal calculations on microcontroller. For Research & Design purposes a complete picture

capture is necessary to see all data in detail, but for mass production purposes only some portion of data is analyzed to speed up the production process.

The system designed in the thesis can be used in R&D, production and quality assurance departments of an LCD TV manufacturer. Low cost of the system (currently less than €50) makes it suitable for multiple units testing conducted by single PC. In this way various functionalities of an LCD TV can be verified without using human labor and LCD panels. The system is really sensitive to any kind of distortion on resultant images. As it is shown in previous chapters these distortions in the images are hard to detect with bare human eyes. Connection between the capture card and the PC is established by using Ethernet which is a very flexible communication method and permits long distances between picture grabbing card and controlling PC.

The future work regarding the subject can be the real time transfer and analysis of data on the PC. An HD Ready panel (1366 x 768) whose vertical refresh rate is 60 Hz displays 60 images on the screen in one second. Each of these images is approximately 3 MB in size, so in every second 180 MB of data flows through the LVDS connection between the TV mainboard and LCD panel. This is roughly a 1.5 Gbps data link. Such a wide bandwidth cannot be established through Ethernet (maximum data rate is 1 Gbps) or USB (maximum data rate is 480 Mbps) connection. Although 10 Gbps Ethernet is available, they only present in some server computers. Also USB3.0 is a newly emerging technology, but currently none of the embedded systems support USB3.0.

Above speed limitations on these interfaces make on-board data manipulation mandatory. In this case a DSP (Digital Signal Processor) or an FPGA based option can be regarded as a solution, a lossless compression algorithm can be implemented inside of DSP or FPGA and this compressed stream can be delivered to the PC, but here another question arises: how can the PC analyze this much of data in real time?

The PC application software in the thesis performs PSNR calculation approximately in 2 seconds and first level wavelet based edge calculation approximately in 4 seconds for an HD Ready image. Even real time image grabbing is achieved in hardware, processing of data on PC requires a lot of time or an offline analysis is required. In this case another alternative computing unit in the PC can aid to decrease computation times, the graphics processing unit and CUDA (Compute Unified Device Architecture).

CUDA is a parallel computing architecture developed by NVIDIA Corporation. CUDA is the computing engine in NVIDIA graphics processing units (GPUs) that is accessible to software developers through industry standard programming languages. Programmers use 'C for CUDA' (C with NVIDIA extensions), compiled through a PathScale Open64 C compiler to code algorithms for execution on the GPU. Third party wrappers are also available for Python, FORTRAN, Java and MATLAB.

CUDA gives developers access to the native instruction set and memory of the parallel computational elements in CUDA GPUs. Using CUDA, the latest NVIDIA GPUs effectively become open architectures like CPUs. Unlike CPUs however, GPUs have parallel "many-core" architecture, each core capable of running thousands of threads simultaneously - if an application is suited to this kind of architecture, the GPU can offer large performance benefits. This approach of solving general purpose problems on GPUs is known as GPGPU. Such a solution can be adapted to image processing task in the system design and high processing capabilities (especially on matrix operations) of a modern GPU can be fully utilized to overcome calculation speed issues in a distributed parallel computing architecture.

REFERENCES

Altera (July 2000). *Board design guidelines for LVDS systems.* Retrieved January 20, 2009, from http://www.altera.co.jp/literature/wp/wp_lvdsboard.pdf.

Chaganti, V. R. (April 2005). *M.Sc. Thesis: Edge detection of noisy images using 2-d discrete wavelet transform.*

Chan, G. (n.d.). *"HD" versus "SD" color space.* Retrieved March 3, 2010, from http://www.glennchan.info/articles/technical/hd-versus-sd-color-space/hd-versus-sd-color-space.

Chunghwa (February 27, 2006). *CLAA320WB02 TFT LCD panel technical specification*. Retrieved February 2, 2009, from http://beyondinfinite.com/lcd/Library/Chungwa/CLAA320WB02.pdf.

Corbet, J., & Rubini, A. (2001). Chapter 15: Memory mapping and DMA. In *Linux device drivers* (2nd ed.) (412-463). USA: O'Reilly Media.

Direct memory access (n.d.). Retrieved December 22, 2009, from http://en.wikipedia.org/wiki/Direct_memory_access.

DMA and interrupt handling (n.d.). Retrieved November 16, 2009, from http://www.eventhelix.com/RealtimeMantra/FaultHandling/dma_interrupt_handling.

EIA (May 2002). *EIA standard EIA/CEA-861-B: A DTV profile for uncompressed high speed digital interfaces.* VA: Electronic Industries Alliance.

Fairchild Semiconductor (June 2005). *LVDS fundamentals.* Retrieved January 25, 2009, from http://www.fairchildsemi.com/an/AN/AN-5017.pdf.

Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2003). *Laplacian/Laplacian of Gaussian.* Retrieved February 8, 2010, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.

Harvey, A. F. (1991). *DMA fundamentals on various PC platforms.* Retrieved May 22, 2009, from http://physweb.bgu.ac.il/COURSES/SignalNoise/DMA.pdf.

JEIDA (February 1999). *JEIDA standard 59-1999: Digital interface standards for monitor*. Japan: Japan Electronic Industry Development Association.

June, F. (June 2006). *Color spaces, from RGB to YCbCr and vice versa.* Retrieved February 5, 2010, from http://www.webkinesia.com/games/vcompress2.

Katz, D., & Gentile, R. (November 1, 2007). *Using direct memory access effectively in media-based embedded applications - part 1*. Retrieved March 28, 2010, from http://www.eetimes.com/design/automotive-design/4006782/Using-Direct-Memory-Access-effectively-in-media-based-embedded-applications--Part-1.

Laplacian based edge detection (n.d.). Retrieved March 9, 2010 from http://www.ii.metu.edu.tr/~ion528/demo/lectures/6/3/index.

LG Display (November 8, 2008). *LC420WUN TFT LCD panel technical specification.* Retrieved February 5, 2009, from Vestel Electronics Inc. R&D Department component specification database.

LG Display (December 19, 2008). *Lx156WH1 TFT LCD panel technical specification.* Retrieved February 5, 2009, from Vestel Electronics Inc. R&D Department component specification database.

Li, J. (August 2003). *M.Sc. Thesis: A wavelet approach to edge detection.*

LPC 2300 easyweb (n.d.). Retrieved February 15, 2009, from http://www.keil.com/download/docs/295.

Martin, T. (2007). *The insider's guide to the NXP LPC2300/2400 based microcontrollers: An engineer's introduction to the LPC2300 & LPC2400 series*. Coventry: Hitex (UK) Ltd.

Media independent interface (n.d.). Retrieved August 12, 2009, from http://en.wikipedia.org/wiki/Media_Independent_Interface.

Monolithic Power Systems (June 2004). *MP1583 3A, 23V step down converter*. Retrieved November 15, 2009, from http://www.cypex.com.cn/upload/ 20097910135684864.pdf.

National Instruments (June 2008). *LabVIEW advanced signal processing toolkit: wavelet analysis tools user manual.* Retrieved January 5, 2010, from LabVIEW Advanced Signal Processing Toolkit software documentation directory.

National Instruments (June 2008). *LabVIEW advanced signal processing toolkit: time frequency analysis tools user manual.* Retrieved January 5, 2010, from LabVIEW Advanced Signal Processing Toolkit software documentation directory.

National Instruments (June 2008). *LabVIEW advanced signal processing toolkit: time series analysis tools user manual.* Retrieved January 5, 2010, from LabVIEW Advanced Signal Processing Toolkit software documentation directory.

National Instruments (July 8, 2009). *Understanding LVDS for digital test systems.* Retrieved August 25, 2009, from http://zone.ni.com/devzone/cda/tut/p/id/4441.

National Semiconductor (2008). *LVDS owner's manual.* Retrieved January 13, 2009, from http://www.national.com/appinfo/lvds/files/National_LVDS_Owners_ Manual_4th_Edition_2008.pdf.

National Semiconductor (May 2008). *DP83848C PHYTER – commercial temperature single port 10/100 Mb/s Ethernet physical layer transceiver data sheet.* Retrieved November 16, 2009, from http://www.national.com/ds/DP/DP83848C.pdf.

NXP (October 2007). *LPC2300/LPC2400 TCP/IP overview.* Retrieved June 5, 2009, from http://ics.nxp.com/literature/presentations/microcontrollers/pdf/lpc23xx.lpc24xx.arm.ethernet.tcpip.training.pdf.

NXP (July 2008). *LPC24XX user manual.* Retrieved March 27, 2009, from http://www.nxp.com/documents/user_manual/UM10237.pdf.

Oki Semiconductor (May 2007). *FEDSMS81V32322-01 1,114,112-Word x 32-Bit field memory technical specification.* Retrieved October 12, 2009, from http://www.datasheetpro.com/900095_download_MS81V32322_datasheet.

Other methods of edge detection (n.d.). Retrieved March 5, 2010, from http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/moredge.

Potter, D. (n.d.). *Using ethernet for industrial I/O and data acquisition.* Retrieved December 24, 2009, from http://www.ni.com/pdf/misc/us/usingethernet.pdf.

PSNR quality analysis methodology (April 18, 2009). Retrieved February 12, 2010, from http://techblog.cineform.com/?p=752.

Rangarajan, S. (n.d.). A*lgorithms for edge detection.* Retrieved March 12, 2010, from http://www.uweb.ucsb.edu/~shahnam/AfED.doc.

Reduced media independent interface (n.d.). Retrieved August 12, 2009, from http://en.wikipedia.org/wiki/Reduced_Media_Independent_Interface.

Rodriguez, E. (May 17, 2004). *TCP vs. UDP*. Retrieved July 15, 2009, from http://www.skullbox.net/tcpudp.

Rongen, H. (n.d.). *Introduction to PC-based data acquisition systems.* Retrieved November 16, 2009, from http://www.fz-juelich.de/zel/datapool/page/160/DAQ.pdf.

Slanina, M., & Ricny, V. (n.d.). *A comparison of full-reference image quality assessment methods*. Retrieved January 5, 2010, from http://www.urel.feec.vutbr.cz/ra2007/archive/ra2006/abstracts/045.pdf.

Sloss, A. N., Symes, D., & Wright, C. (2004). *ARM system developer's guide: Designing and optimizing system software*. USA: Morgan Kaufmann Publications.

Stasonis, R. (n.d.). *How to implement functional test in an automated environment.* Retrieved December 15, 2009, from http://www.ddmconsulting.com/Design_Guides/functest.pdf.

Texas Instruments (November 2002). *LVDS application and data handbook.* Retrieved January 15, 2009, from http://focus.ti.com/lit/ug/slld009/slld009.pdf.

Thine Electronics Inc. (n.d.). *THC63LVDF84B LVDS 24-bit color host-LCD panel interface receiver technical specification.* Retrieved October 13, 2009, from http://www.thine.co.jp/products/LVDS/pdf/thc63lvdf84b_rev320_e_datasheet.pdf

Transmission control protocol (n.d.). Retrieved August 12, 2009, from http://en.wikipedia.org/wiki/Transmission_Control_Protocol.

VESA (n.d.). *"Display port": A new digital display interface standard*. Retrieved March 12, 2010, from http://www.gnss.com/technology/DisplayPort%20-%20Overview%20-%20VESA.pdf.

VESA (August 2003). *VESA standard: VESA and industry standards and guidelines for computer display monitor timing (DMT).* CA: Video Electronics Standards Association.

VESA (February 2008). *VESA standard: TV panels standard.* CA: Video Electronics Standards Association.

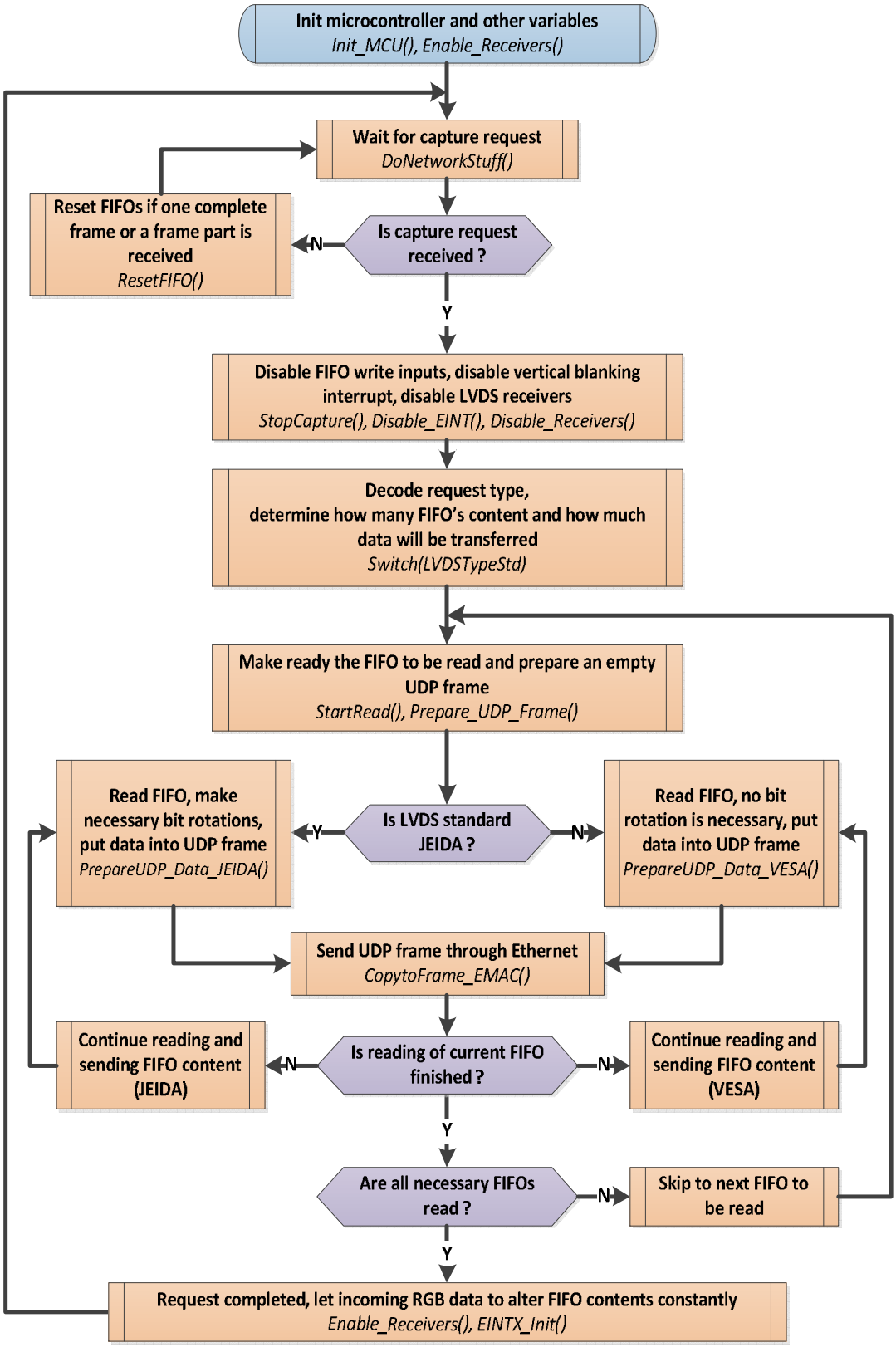
Video quality (n.d.). Retrieved March 25, 2010, from http://en.wikipedia.org/wiki/Video_quality.

What is data acquisition and data logging? (n.d.). Retrieved February 5, 2010, from http://home.datataker.com/acquisition.

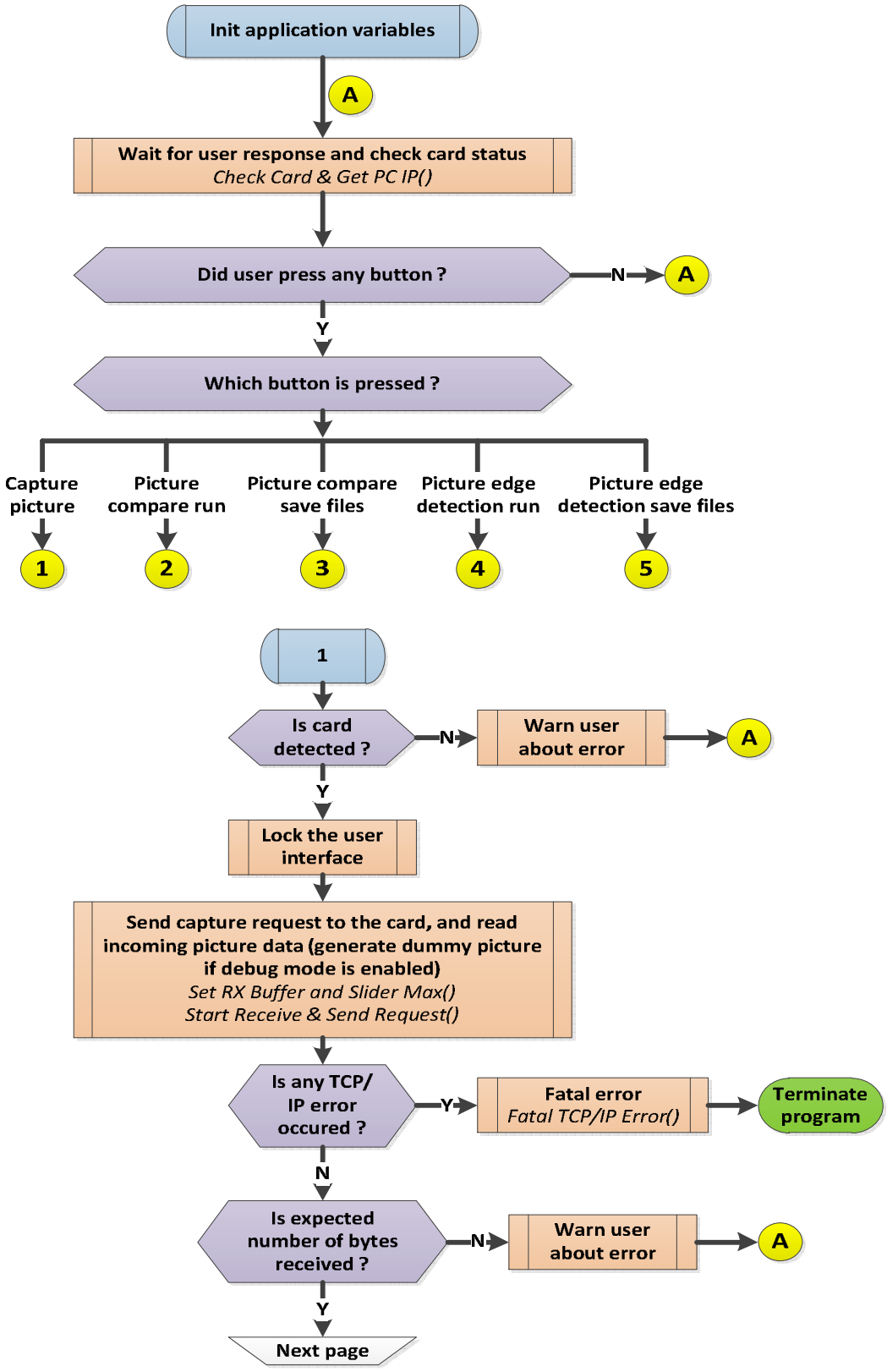
YUV (n.d.). Retrieved February 15, 2010, from http://en.wikipedia.org/wiki/YUV.

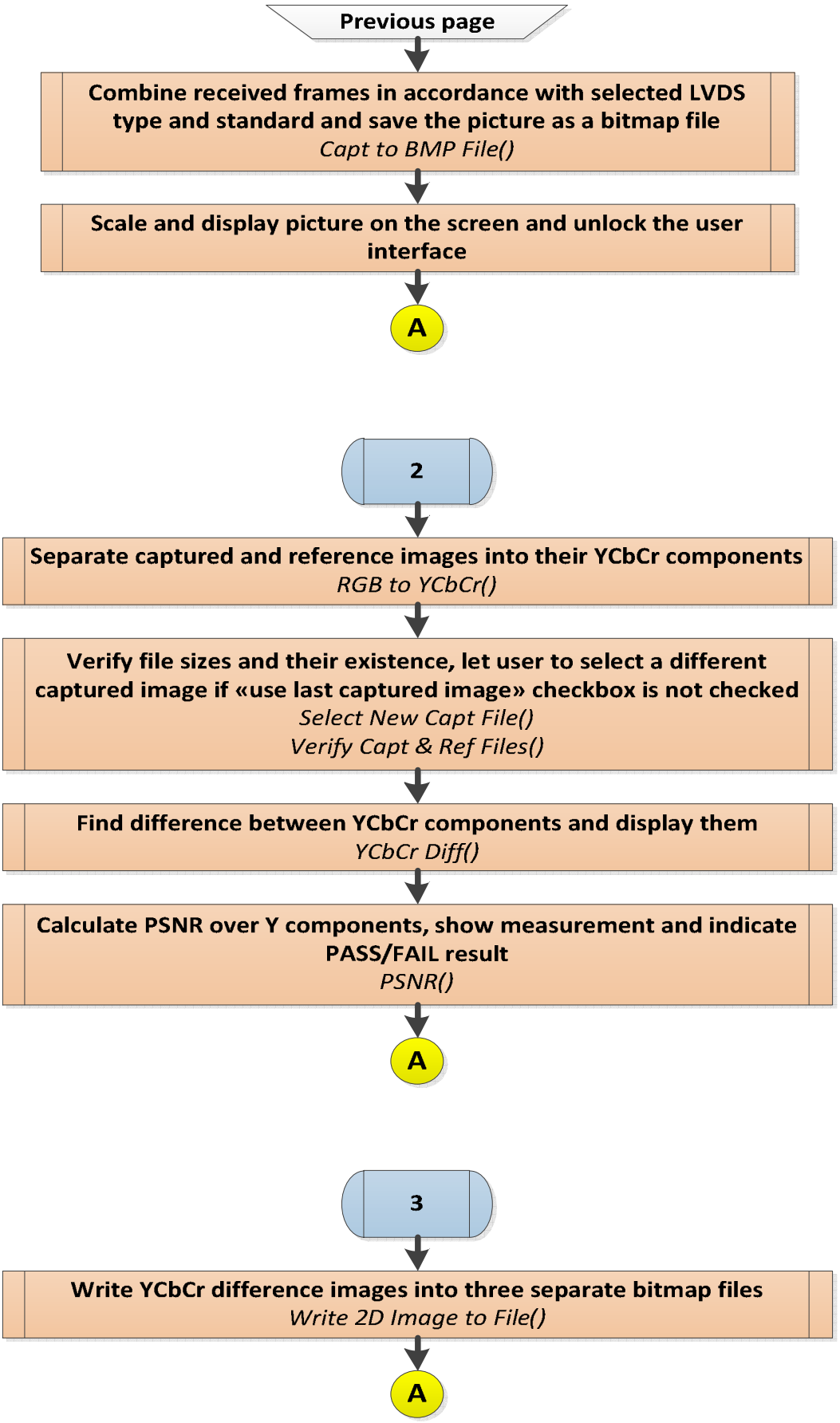
APPENDICES

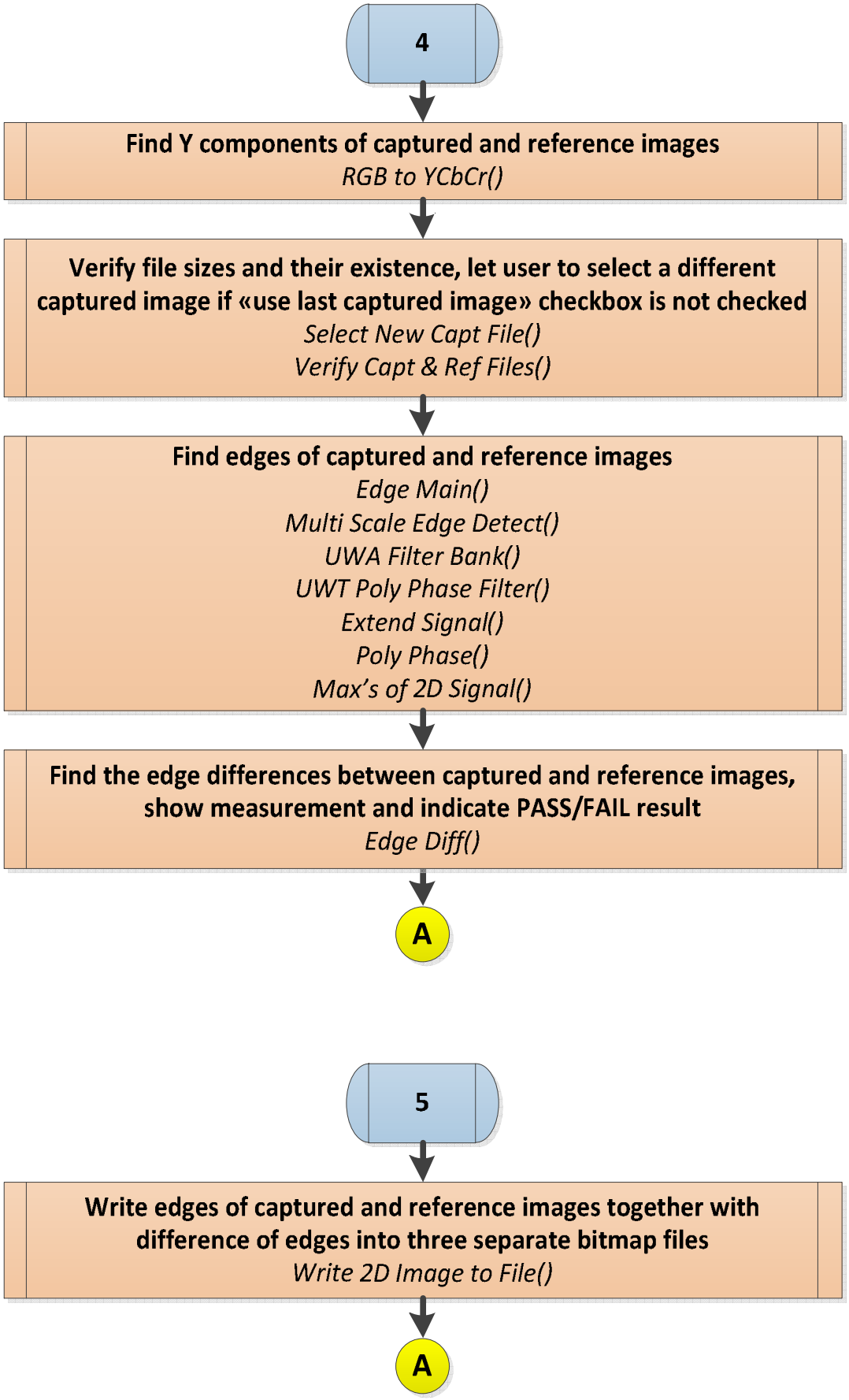
A1 Flowchart of Embedded Software with Core Functions



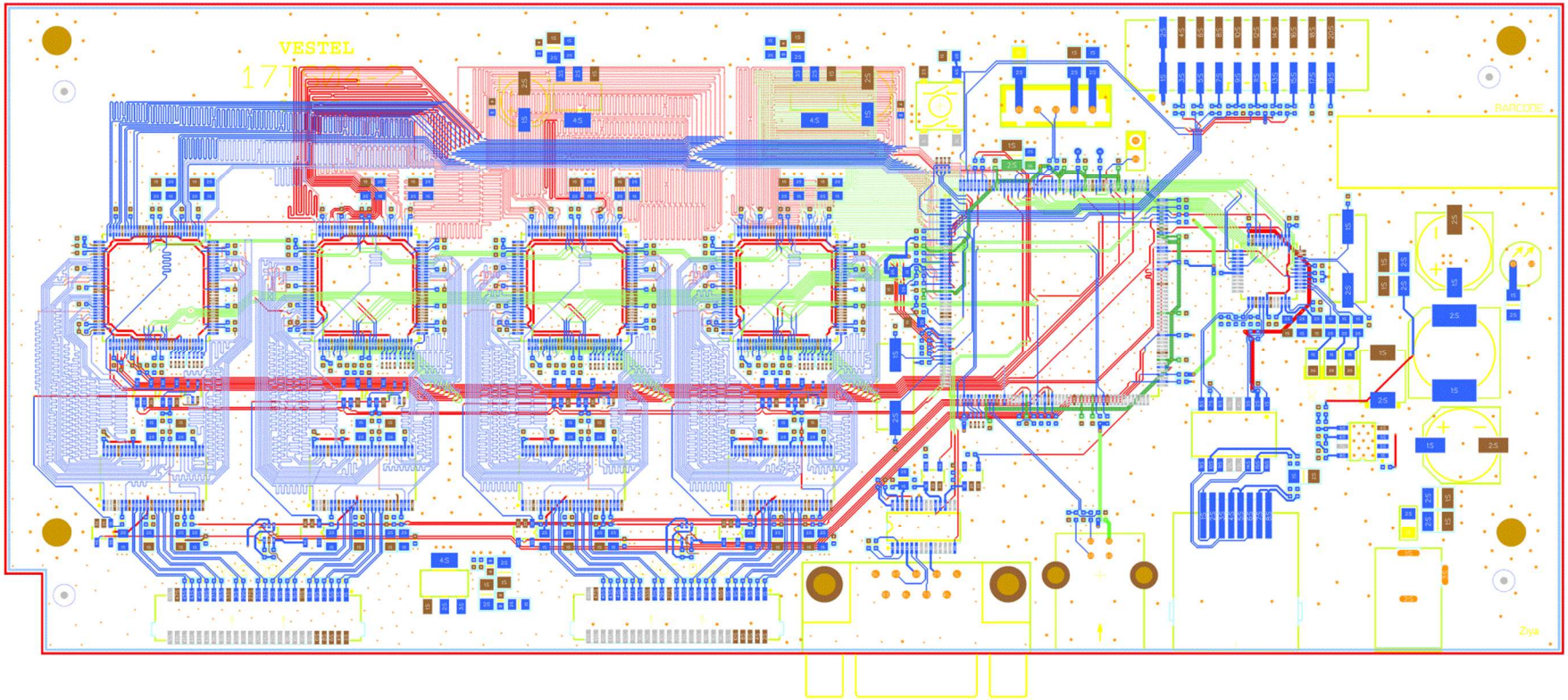
A2 Flowchart of PC Application Software with Core Functions



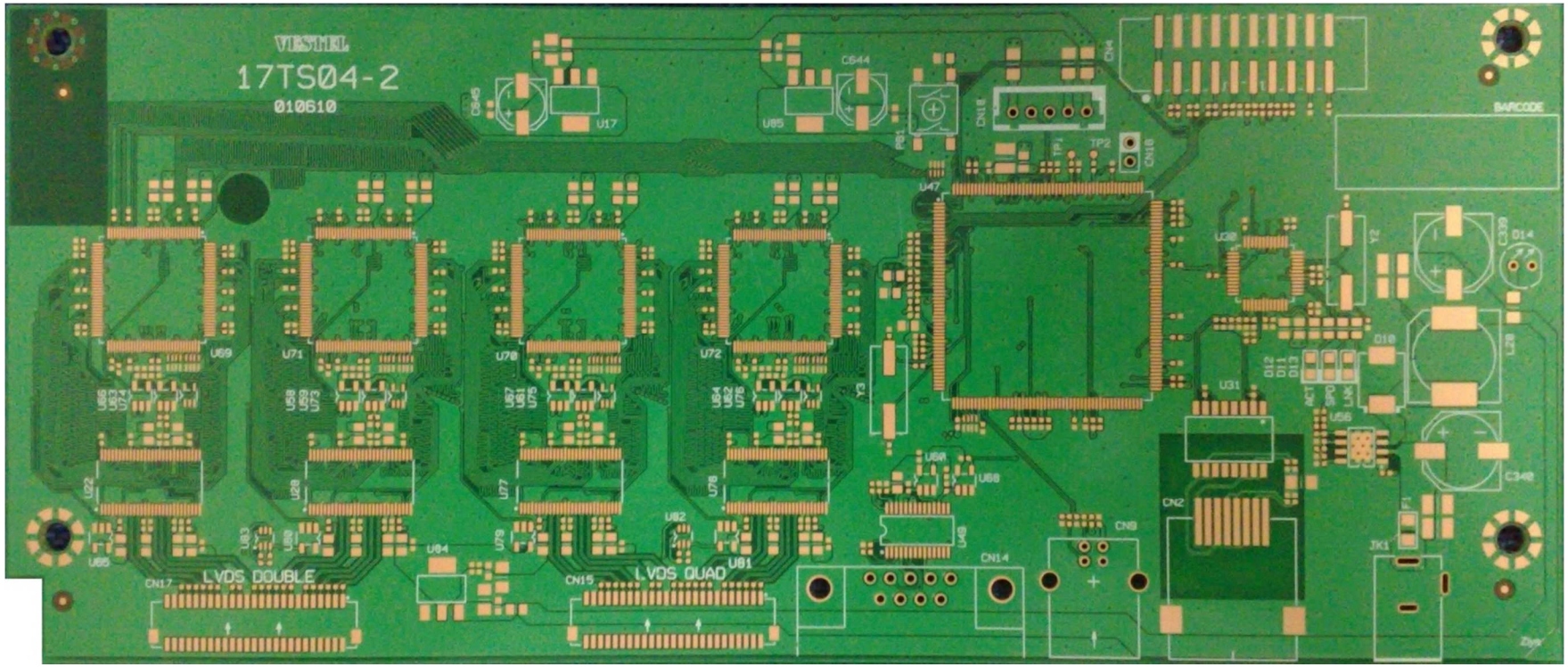

Previous page
Combine received frames in accordance with selected LVDS type and standard and save the picture as a bitmap file
Capt to BMP File()
Scale and display picture on the screen and unlock the user interface
A
2
Separate captured and reference images into their YCbCr components
RGB to YCbCr()
Verify file sizes and their existence, let user to select a different captured image if «use last captured image» checkbox is not checked
Select New Capt File()
Verify Capt & Ref Files()
Find difference between YCbCr components and display them
YCbCr Diff()
Calculate PSNR over Y components, show measurement and indicate PASS/FAIL result
PSNR()
A
3
Write YCbCr difference images into three separate bitmap files
Write 2D Image to File()
A


4
Find Y components of captured and reference images
RGB to YCbCr()
Verify file sizes and their existence, let user to select a different captured image if «use last captured image» checkbox is not checked
Select New Capt File()
Verify Capt & Ref Files()
Find edges of captured and reference images
Edge Main()
Multi Scale Edge Detect()
UWA Filter Bank()
UWT Poly Phase Filter()
Extend Signal()
Poly Phase()
Max's of 2D Signal()
Find the edge differences between captured and reference images, show measurement and indicate PASS/FAIL result
Edge Diff()
A
5
Write edges of captured and reference images together with difference of edges into three separate bitmap files
Write 2D Image to File()
A

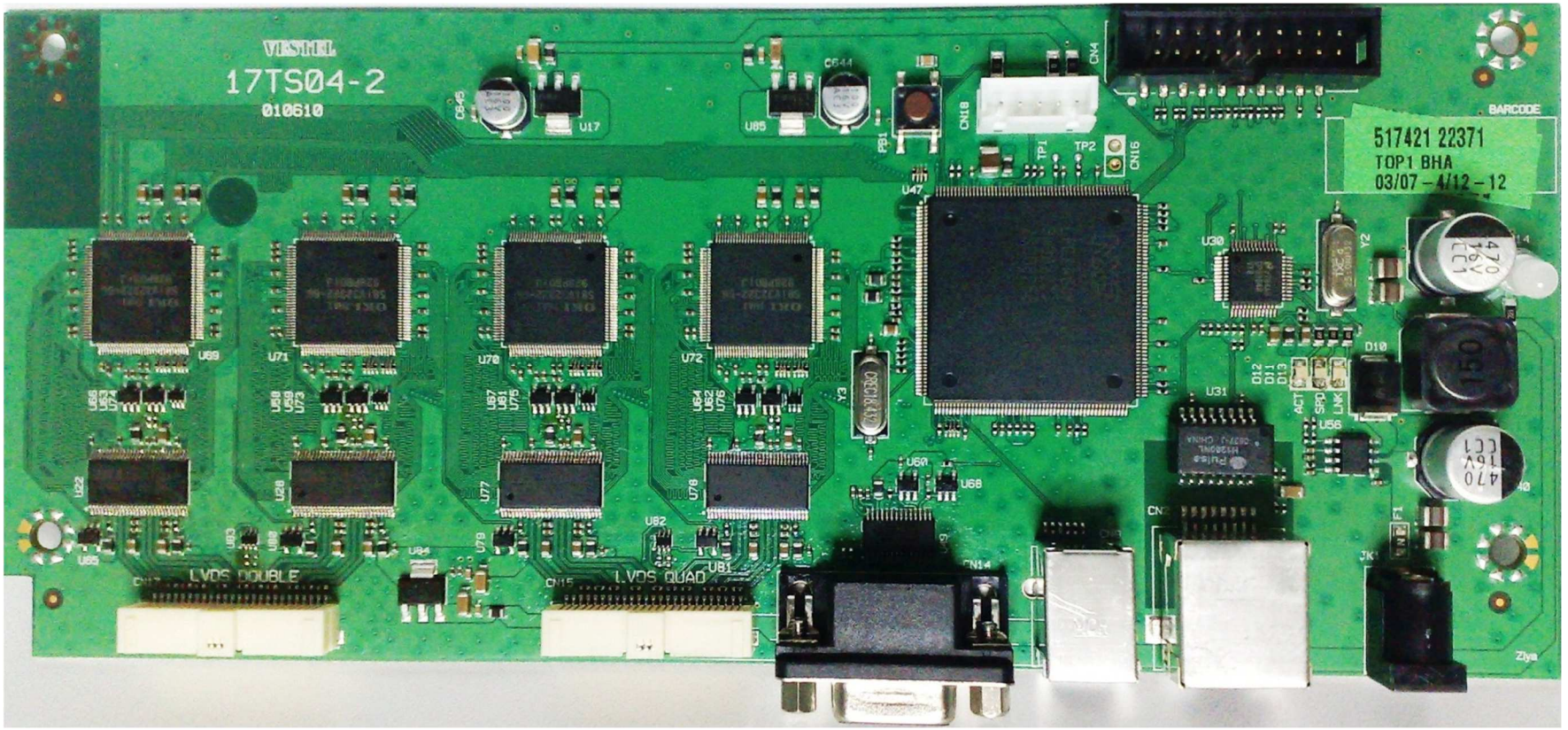
A3 Printed Circuit Board Design



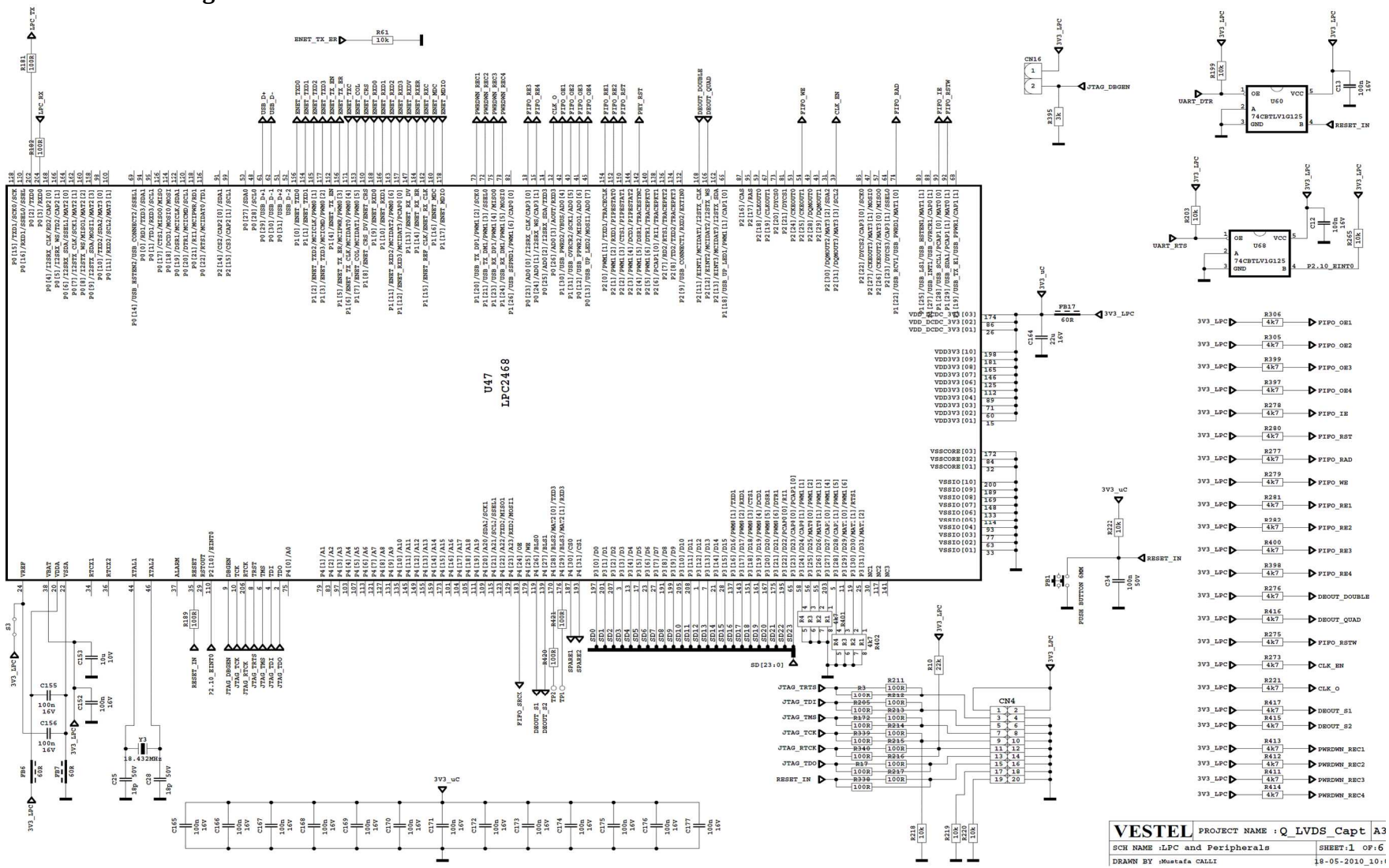
A4 Top Side of Empty Printed Circuit Board

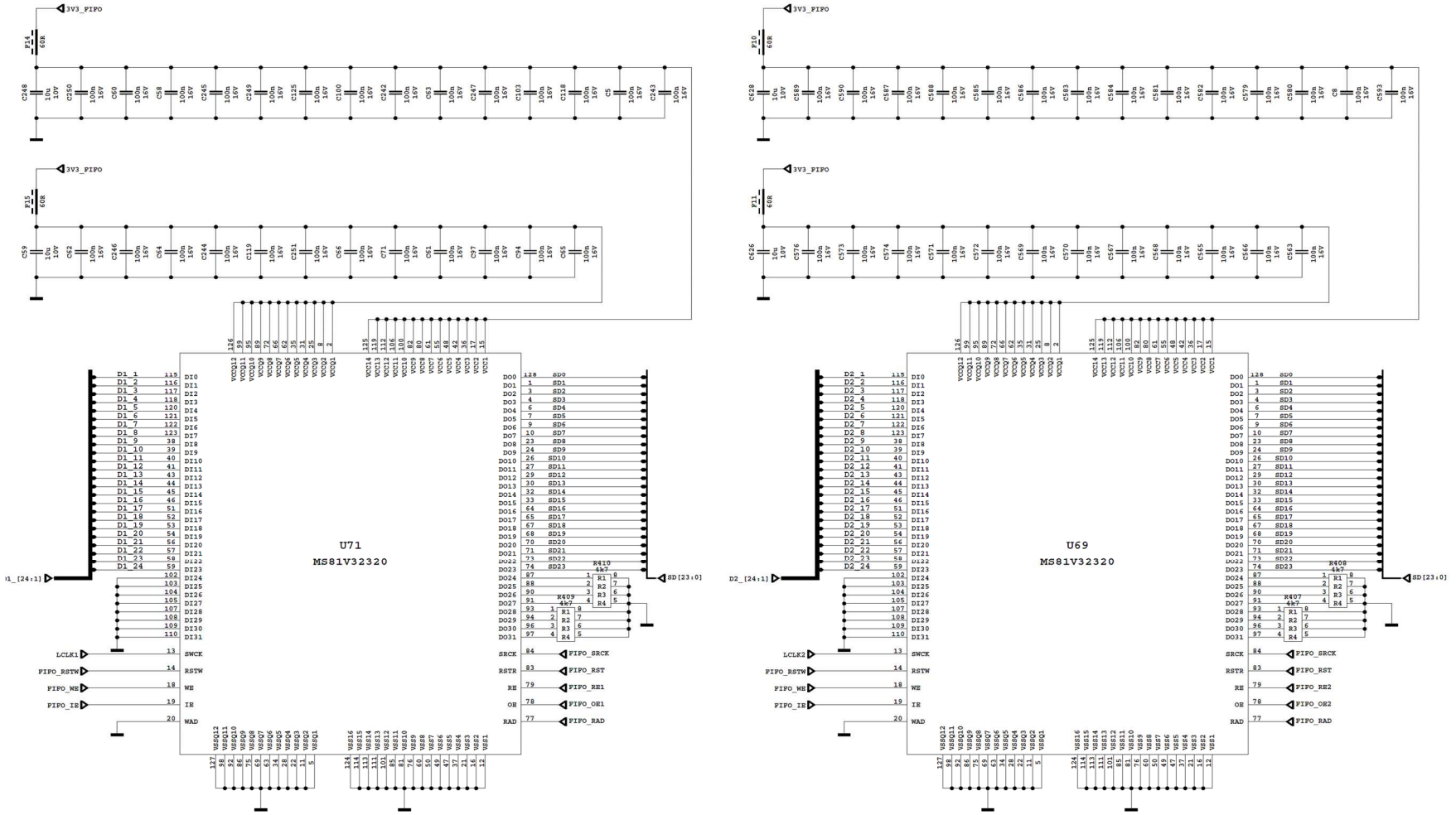


A5 Top Side of Printed Circuit Board After Auto Insertion

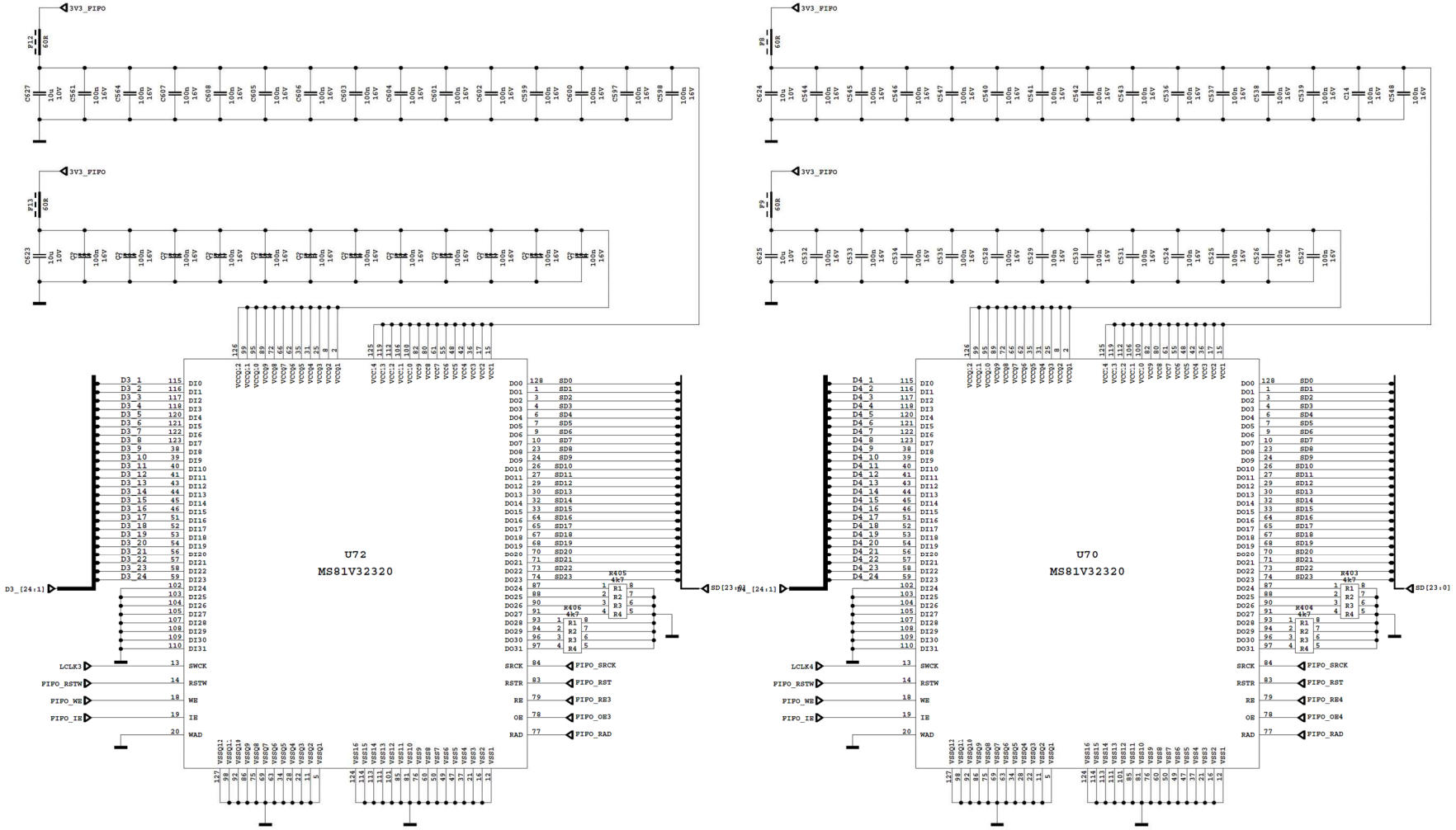


A6 Hardware Design Schematics

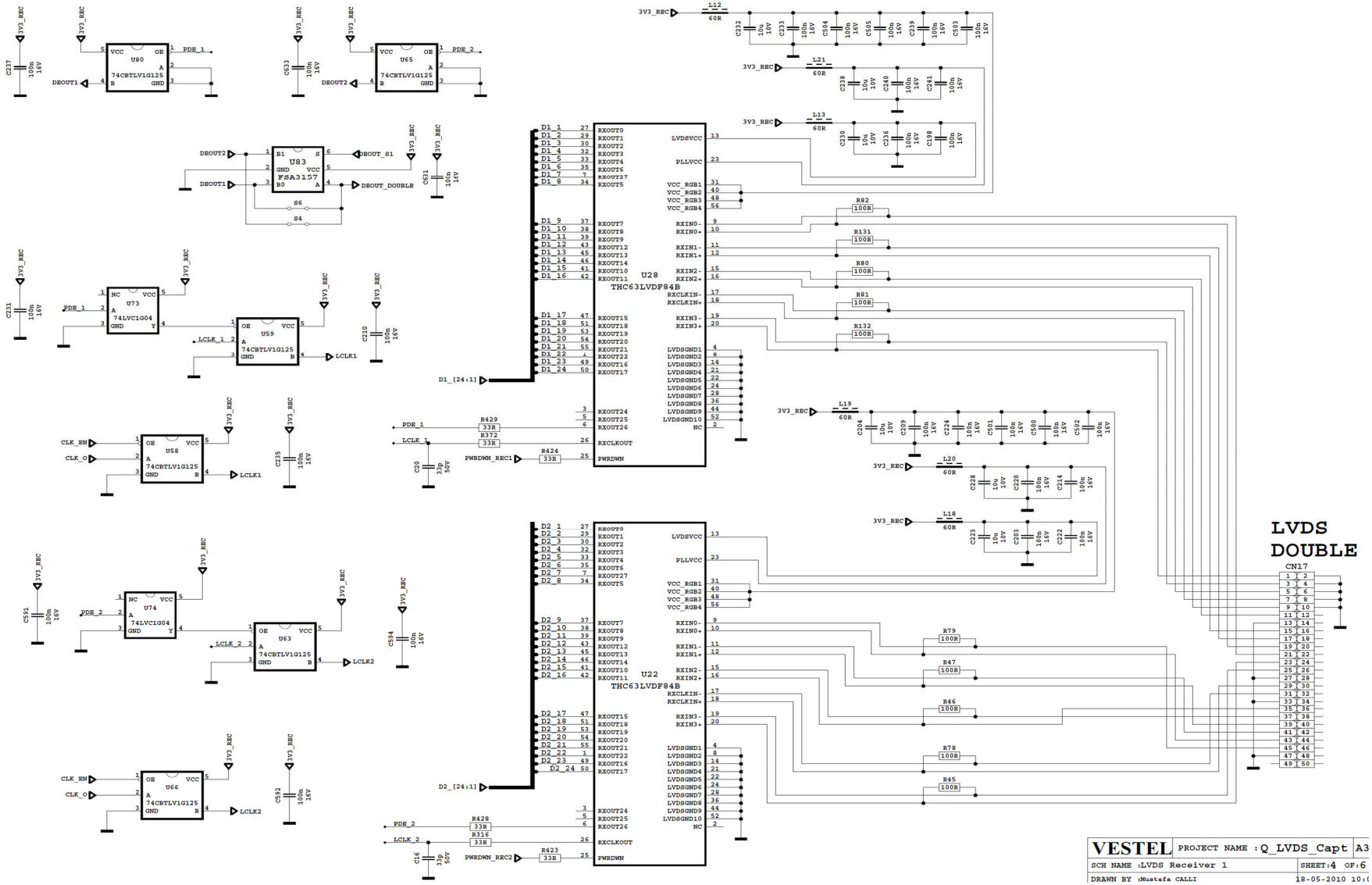


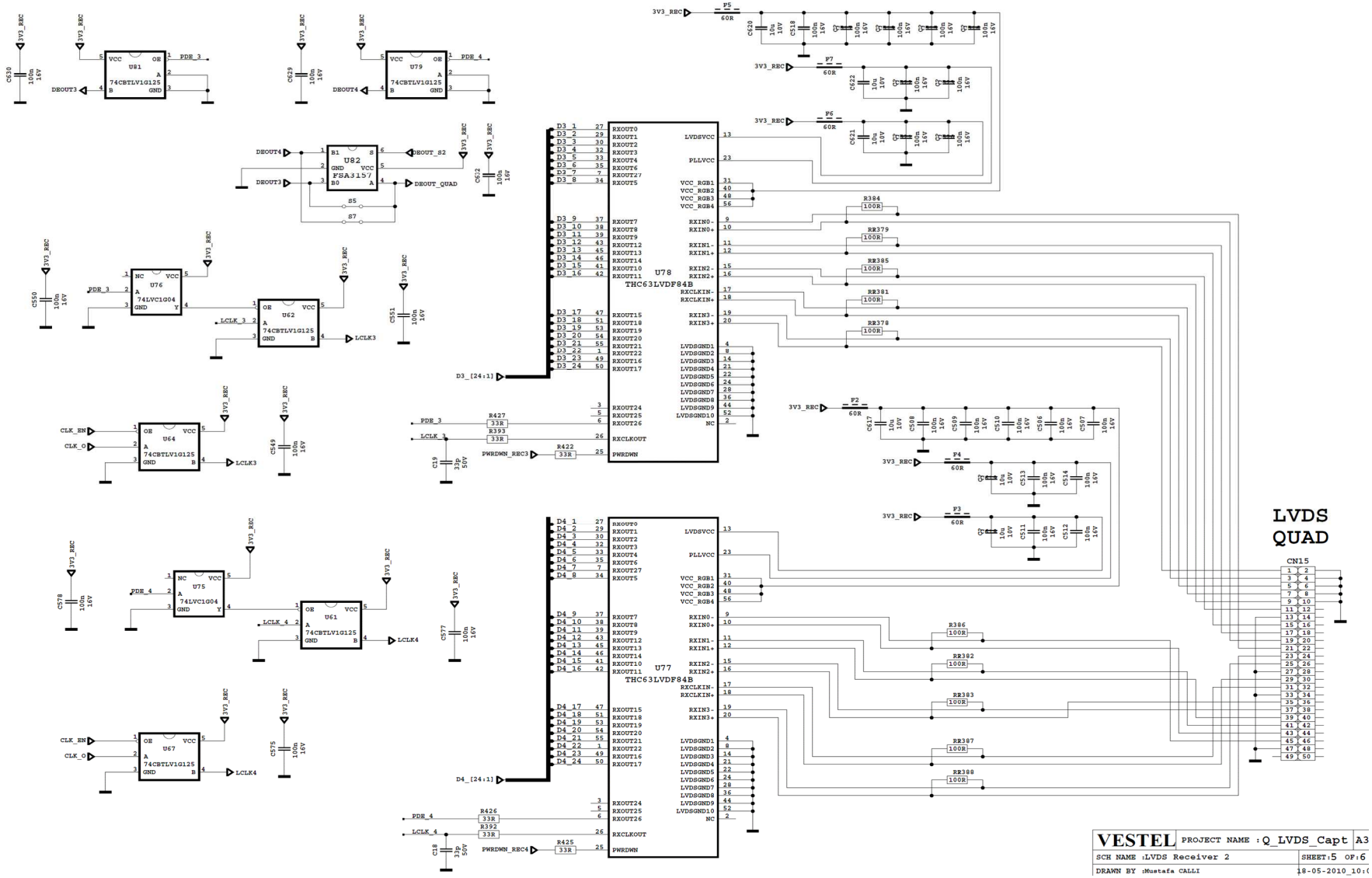


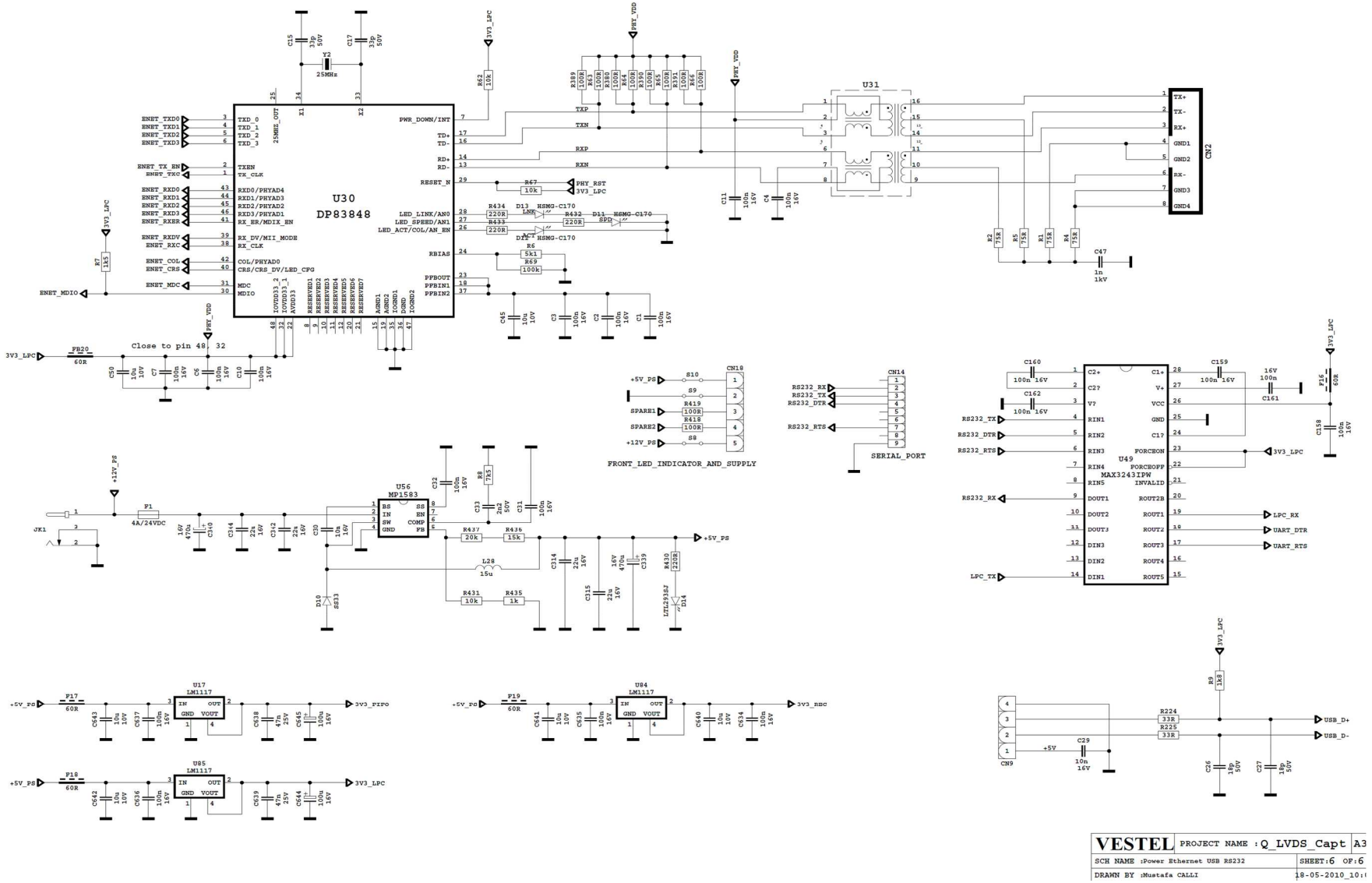
VESTEL	PROJECT NAME : Q_LVDS_Capt	A3
SCH NAME : FIFO Group 1		SHEET: 2 OF: 6
DRAWN BY : Mustafa CALLI		18-05-2010_10:



VESTEL	PROJECT NAME : Q_LVDS_Capt	A3
SCH NAME :FIFO Group 2	SHEET:3 OF:6	
DRAWN BY :Mustafa CALLI	18-05-2010_10:(	


LVDS
DOUBLE
VESTEL
PROJECT NAME : Q_LVDS_Capt
SCH NAME :LVDS Receiver 1
SHEET:4 OF:6
DRAWN BY :Mustafa CALLI
U28
THC63LVDF84B
U22
THC63LVDF84B
U80
74CBTLV1G125
U65
74CBTLV1G125
U83
FSA3157
U73
74LVC1G04
U59
74CBTLV1G125
U58
74CBTLV1G125
U74
74LVC1G04
U63
74CBTLV1G125
U66
74CBTLV1G125
CN17
3V3_REC
DEOUT1
DEOUT2
DEOUT_S1
DEOUT_DOUBLE
PDE_1
PDE_2
LCLK_1
LCLK_2
LCLK1
LCLK2
CLK_EN
CLK_O
D1_[24:1]
D2_[24:1]
PWRDWN_REC1
PWRDWN_REC2


LVDS
QUAD
U78
THC63LVDF84B
U77
THC63LVDF84B
U82
FSA3157
U81
74CBTLV1G125
U79
74CBTLV1G125
U76
74LVC1G04
U75
74LVC1G04
U64
74CBTLV1G125
U62
74CBTLV1G125
U67
74CBTLV1G125
U61
74CBTLV1G125
CN15
VESTEL
PROJECT NAME :Q_LVDS_Capt
SCH NAME :LVDS Receiver 2
DRAWN BY :Mustafa CALLI


U30
DP83848
U31
CN2
U56
MP1583
U49
MAX3243IPW
CN14
SERIAL_PORT
CN18
FRONT_LED_INDICATOR_AND_SUPPLY
U17
LM1117
U84
LM1117
U85
LM1117
CN9
Close to pin 48, 32
VESTEL
PROJECT NAME : Q_LVDS_Capt
SCH NAME :Power Ethernet USB RS232
SHEET:6 OF:6
DRAWN BY :Mustafa CALLI

A7 Embedded System Block Diagram