

MOBILE VIDEO SURVEILLANCE



By

NC Zainab Khalid

NC Zunaira Sajjad

NC Samoona Aslam

NC Jovaria Maqsood

Project Supervisor

Lec Intisar Rizwan-i-Haque

Submitted to the Faculty of Electrical Engineering, Military College of Signals
National University of Sciences and Technology, Rawalpindi in partial fulfillment for
the requirements of a B.E. Degree in Electrical (Telecommunication) Engineering

JUNE 2014

ABSTRACT

MOBILE VIDEO SURVEILLANCE

Surveillance systems have been going through an era of development. This has been mainly because of the increasing need of security. Real-time surveillance has gained rising popularity due to their automatic surveillance and security actions. This mobile video surveillance is a real time intelligent system based on FPGA (Field Programmable Gate Array). It provides the functionality of monitoring, security and control in order to improve efficiency of security system with respect to timing constraints. There are two units of the system: the first being FPGA which is connected with motion sensors, IP camera and GPRS/GSM module and the second unit is the mobile phone. Sensors monitor the target area including the entrance as well as windows. The video of the target area is continuously captured by the IP camera. In case of any abnormal situation, alarm messages are sent to the user's mobile through GPRS/GSM module. The video is then transmitted to the user through Wi-Fi module. The user also has the option to arm/disarm the system.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or at elsewhere.

Copyright by

Zainab Khalid

Zunaira Sajjad

Samoon Aslam

Jovaria Maqsood

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent.

To our parents, without whose constant support and unstinting cooperation and assistance
a work of this magnitude would not have been possible.

ACKNOWLEDGEMENTS

All praises for Allah Almighty who gave us the strength to accomplish this mighty task despite numerous difficulties and hardships on our way.

We offer our utmost gratitude to our Supervisor, Lec Intisar Rizwan-i-Haque for his constant guidance and encouragement. He gave us hope and boosted up our confidence during the tough phases of our project. The project wouldn't have been possible without his expert advice, unfailing patience and invaluable efforts.

We are also grateful to our parents and family for their admirable support and last but not the least, our auspicious university for making us what we are today.

Table of Contents

Chapter	Page no.
Chapter 1: Introduction	1
1.1. Project Overview	1
1.2. Problem Statement	1
1.3. Project Scope	1
1.4. Objectives	2
1.5. Deliverables	2
1.6. Conclusion	2
Chapter 2: Literature Review	3
2.1. Surveillance	3
2.2. Video Surveillance	3
2.3. FPGA	4
2.4. Verilog	5
2.5. IP Cores	5
2.6. Virtex-6 embedded tri-mode Ethernet MAC wrapper	6
2.7. Conclusion	6
Chapter 3: Technological Requirements	7
3.1. Hardware Requirements	8
3.1.1. Virtex-6 XC6VLX240T-1FFG1156	8

3.1.2. SIM900 Module	10
3.1.3. IP Camera.....	11
3.1.4. Laptop and Mobile phone	11
3.1.5. Wi-Fi Router	12
3.1.6. Switch.....	12
3.1.7. Motion Sensor	12
3.2. Software Requirements	13
3.2.1. Xilinx ISE Design suite	13
3.2.2. iSpy	14
3.2.3. IP cam viewer.....	14
3.2.4. Wireshark	15
3.2.5. Docklight.....	16
3.3. Conclusion	16
Chapter 4: Design and Development	17
4.1. System Overview	17
4.1.1. System Block Diagram	18
4.2. Operating Frequencies	19
4.3. Hardware Integration	19
4.3.1. Interfacing Motion Sensor	20
4.3.2. GSM Module	20

4.3.3. Ethernet	23
4.4. Video Storage	27
4.4.1. Camera Settings	29
4.4.2. Video Storage On Detection	29
4.4.3. Assigned Directory	30
4.5. Conclusion	31
Chapter 5: Project Analysis and Evaluation	32
5.1. Top Module	32
5.2. GSM Command Transmitter Module	33
5.3. GSM Command Receiver Module	35
5.4. GSM Interfacing Module	36
5.5. Ethernet Core Generation	40
5.5.1. Ethernet Module-Eth 0 (Copper Interface)	40
5.5.2. SFP Module Eth1 (Optical Interface)	42
5.6. Conclusion	45
Chapter 6: Future Work and Conclusion	46
6.1. Future Work	46
6.1.1. H.264 Video Compression	46
6.1.2. Encryption and Decryption of Video	48
6.1.3. Face detection and fingerprint algorithms	48

6.1.4. Object Classification.....	48
6.1.5. Increase in the coverage area	49
6.2. Conclusion	49
Chapter 7: Bibliography	50
APPENDIX A-1	51
APPENDIX A-2	59
APPENDIX A-3	64
APPENDIX A-4	76

LIST OF TABLES

Table	Page no.
Table 3-1: Features of Virtex-6 XC6VLX240T-1FFG1156.....	9
Table 4-1: Operating frequencies.....	19
Table 4-2: Sensor pin assignment on FPGA.....	20
Table 4-3: SIM900 module and FPGA interfacing.....	22
Table 4-4: Communication between FPGA and user via SIM900 module	23
Table 4-5: Number of bits for frame	27
Table 5-1: Top module design summary	38

LIST OF FIGURES

Figure	Page no.
Figure 3-1: Breakdown of technological requirements	7
Figure 3-2: Block diagram of Virtex-6.....	8
Figure 3-3: Virtex-6 board	9
Figure 3-4: SIM900 module.....	11
Figure 3-5: IP camera.....	11
Figure 3-6: Live video transmission on mobile phone	12
Figure 3-7: Motion sensor.....	13
Figure 4-1: System layout.....	18
Figure 4-2: System block diagram	19
Figure 4-3: GSM module	21
Figure 4-4: System interfacing.....	22
Figure 4-5: FPGA interfacing with FPGA and router	24
Figure 4-6: Frame processing	26
Figure 4-7: Camera settings	29
Figure 4-8: Video storage on detection.....	30
Figure 4-9: Stored clips in software	30
Figure 5-1: Top module	32
Figure 5-2: Top module design in Xilinx	33
Figure 5-3: GSM command transmitter module	33
Figure 5-4: GSM command receiver module	35
Figure 5-5: GSM interfacing module.....	37

Figure 5-6: RTL schematic of Top module.....	39
Figure 5-7: Ethernet core generation	41
Figure 5-8: BANK33	42
Figure 5-9: SFP module core generation	43
Figure 5-10: BANK 116	44
Figure 6-1: Video coding	47

LIST OF ABBREVIATIONS

ACE	Advanced Configuration Environment
AVI	Audio Video Interleaved
BPI	Byte Peripheral Interface
CF	Compact Flash
CCTV	Closed-circuit television
CMOS	Complementary metal–oxide–semiconductor
CAT 5	Category 5
DDR3	Double Data Rate Type Three
FPGA	Field-programmable gate array
GPIO	General-Purpose Input/Output
GPRS	General packet radio service
GMII	Gigabit Media Independent Interface
HDL	Hardware description language
I/O	Input/output
IP	Internet Protocol
IR	Infrared
JTAG	Joint Test Action Group
JPEG	Joint Photographic Experts Group
LED	Light Emitting Diode
LCD	Liquid Crystal Display
MAC	Media Access Control
M2M	Machine-to-Machine

MJPEG	Motion Joint Photographic Experts Group
PHY	Physical Layer
PCS	Physical Coding Sub-Layer
PMA	Physical Media Access
RGMI	Reduced Media Independent Interface
RTL	Register Transfer Level
SODIMM	Small Outline Dual In-Line Memory Module
SMA	SubMiniature Version A
SGMII	Serial Gigabit Media Independent Interface
SFP	Small Form-Factor Pluggable
TCP	Transport Control Protocol
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver/Transmitter
UCF	User Constraint File
VGA	Video Graphics Array
Wi-Fi	Wireless Fidelity

Chapter 1: Introduction

This chapter will give a brief description of the design and will explain why there was a need to develop an effective anti-theft system. It will then explain the objectives, scope and deliverables of the project.

1.1. Project Overview

This mobile video surveillance system is a real time intelligent system based on FPGA. It provides the functionality of monitoring, security and control with the help of IP camera, motion sensor and GPRS/GSM module.

1.2. Problem Statement

In light of the worsening crime situations, the need for efficient security system has increased. People lead busy lives, come in and out of work, go on vacations, and run errands so they need to access a live view of their security cameras when they are away from their home or business. Thus, most of the time one is not actually able to be on the site where cameras are installed. Therefore, a real time mobile video surveillance system based on FPGA has been designed. One of the primary benefits of this system is that the user can view the target area on his mobile anywhere provided that there is internet access.

1.3. Project Scope

The surveillance system can serve as a handy tool for monitoring indoor environment. The intruder can be detected within a range of 12m.

1.4. Objectives

The objective is to design an efficient security system using FPGA. Due to the increasing threats people want security systems that can warn them instantly and provide them access to the live video of the target area even if they are not on the site (where the camera is installed). The following user requirements form the target objectives of our project:

- Reception of alarm messages on mobile phone in case of an intrusion that detected by motion sensor
- Knowledge of the location through which the intruder has entered
- Access to the video through internet
- Video can be monitored remotely through mobile
- Backup storage

1.5. Deliverables

The end goal is to have an embedded real time interactive video transmission and control system to meet the actual needs for home security, monitoring the target area remotely, and to prevent car-theft.

1.6. Conclusion

Mobile video surveillance is an intelligent security system which has the functionality of providing access to the video of the target area remotely to the user.

Chapter 2: Literature Review

This chapter will include the literature review conducted from the start of the project and will give brief description of the important concepts related to the project.

Throughout the project, literature review was an ongoing process. In order to gain more knowledge and learn the necessary skills required to complete this project, it was very essential to refer to the variety of sources.

2.1. Surveillance

To monitor the activities, changing information and behavior, mostly regarding people is surveillance. Usually government organizations observe a particular suspected group or individuals.

2.2. Video Surveillance

Over the years, video surveillance has been a well-known security tool since it allows the user to monitor the target area remotely. Due to advancement in technology, security cameras have shown a great deal of improvement. It is a commonly deployed surveillance technique in contrast to the other types of surveillance. Banks, offices, markets and others are highly dependent on these effective surveillance systems.

CCTV (Closed Circuit Television) analog systems were used in the past, they can be expensive because they require constant maintenance. Cost effective, simple to operate and flexibility are some praise worthy features of digital technology. In order to match

user's specific needs, Security systems deploying IP (Internet Protocol) cameras are a good option since they are easy to install and maintain.

2.3. FPGA

FPGA stands for "Field Programmable Gate Array". FPGA essentially is a huge array of gates, which can be programmed and reconfigured anytime anywhere. FPGAs are manufactured by companies like Xilinx, Altera, Actel etc. We implemented different tutorials on Spartan-3E regarding VGA, LCD display, instantiation etc initially. [1]

For basic understanding of FPGA some tutorials were implemented using Spartan 3-E Starter Kit. The software used was Xilinx ISE 12.3. Brief description of these tutorials is given below:

- **Simple introductory project:** This tutorial consisted of two parts. The first was linked to the design of the program and circuit implementation. The second step was the test bench simulation in order to test the project. The results were simulated by giving the inputs and observing the outputs. [2]
- **Multiple instantiations of same component:** In this tutorial, a single component was created and simulated. Replicating the components three times was another task. [3]
- **IP cores:** In order to incorporate an IP core into the project, it was learnt how to use Xilinx's CORE Generator System by creating a multiplier. [4]
- **Image generation on the Digilent Spartan 3-E board:** In this tutorial, images were generated on the VGA (Video Graphics Array) monitors by implementing a VGA

controller. An encryption algorithm was written to show this implementation. To display the video on the screen of the monitor, VGA display port was used. [5]

- **Serial RS-232:** RS-232 port was used control the frequency of a simple 4-bit counter using hyperterminal in which five different frequencies were used. A laptop was connected with an FPGA kit using available RS-232 port on the board. [6]

2.4. Verilog

The hardware designers in industry and academia use Verilog as one of the major Hardware Description Languages (HDL). The second one being VHDL. It is easier to learn Verilog as compared to VHDL. Verilog HDL uses four different levels of abstraction including architectural or behavioral level as well as gate and switch levels. Three important levels of abstractions are as follows:

- **Behavioral Level:** Concurrent algorithms are used to describe this level. There are a set of instructions which are executed sequentially. Main elements include functions, tasks and always block.
- **Register-Transfer Level:** Transfer of data between the registers and characteristics of circuits by operations are specified by register transfer level.
- **Gate Level:** Logical links and timing properties describe the characteristics of the system. Discrete Signals can have definite logic values. [7]

2.5. IP Cores

Xilinx Targeted Design Platforms key building blocks consist of Intellectual Property (IP) cores. To easily create Plug-and-Play IP, Xilinx FPGA tools and designs play an important role. The market and general specific needs of the end users can easily be met

with the help of an extensive catalog of cores. This provides with a competitive advantage. [8]

2.6. Virtex-6 embedded tri-mode Ethernet MAC wrapper

For the generation of HDL wrapper files for the Ethernet core in Virtex-6, the CORE Generator Virtex-6 Embedded Tri-mode Ethernet Media Access Controller (MAC) Wrapper is used. Depending on the user requirements, simulation scripts, HDL wrappers and test benches are generated automatically through core generation functionality of Xilinx ISE. Its striking features include:

- A physical interface that is user configurable
- The connection between the PHY and Ethernet MAC is simplified
- SGMII or 1000base-X interfaces with single bit stream
- 2.5 Gbps as the upper limit for 1000base-X interface
- In GMII/MII or RGMII modes, Clock Enable inputs are available

2.7. Conclusion

The literature review carried out was mainly focused on the tutorials performed on Spartan-3E starter kit, learning Verilog, features of Virtex-6 and learning about various components used in the project for e.g. IP camera, motion sensors etc.

Chapter 3: Technological Requirements

This chapter will provide a brief overview of the technological requirements for the project “Mobile video surveillance”.

Mobile Video Surveillance provides three functionalities i.e. monitoring, security and control. For monitoring purposes, an IP camera is required to be interfaced with FPGA kit. A Wi-Fi router is required so that the captured video can be viewed anywhere on the mobile phone provided that there is internet access.

For storage purposes, a backup is required which can save the video clip whenever an intruder is detected so that incase the user didn't view the message or was unable to view the video, he can still view the video clip later through backup.

The technological requirements are sub-divided into hardware and software and are discussed in the following sub-sections. Figure 3-1 shows valid breakdown of Technological Requirements.

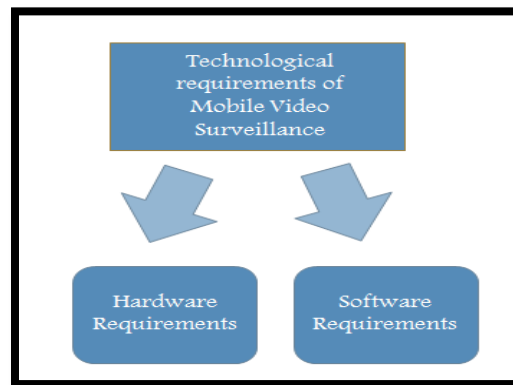


Figure 3-1: Break down of technological requirements

3.1. Hardware Requirements

The Hardware required for the implementation of the project includes:

3.1.1. Virtex-6 XC6VLX240T-1FFG1156

Virtex-6 board was selected because two Ethernet ports were required for the project. As it can be seen in Figure 3-3 that virtex-6 comprises of both these ports.

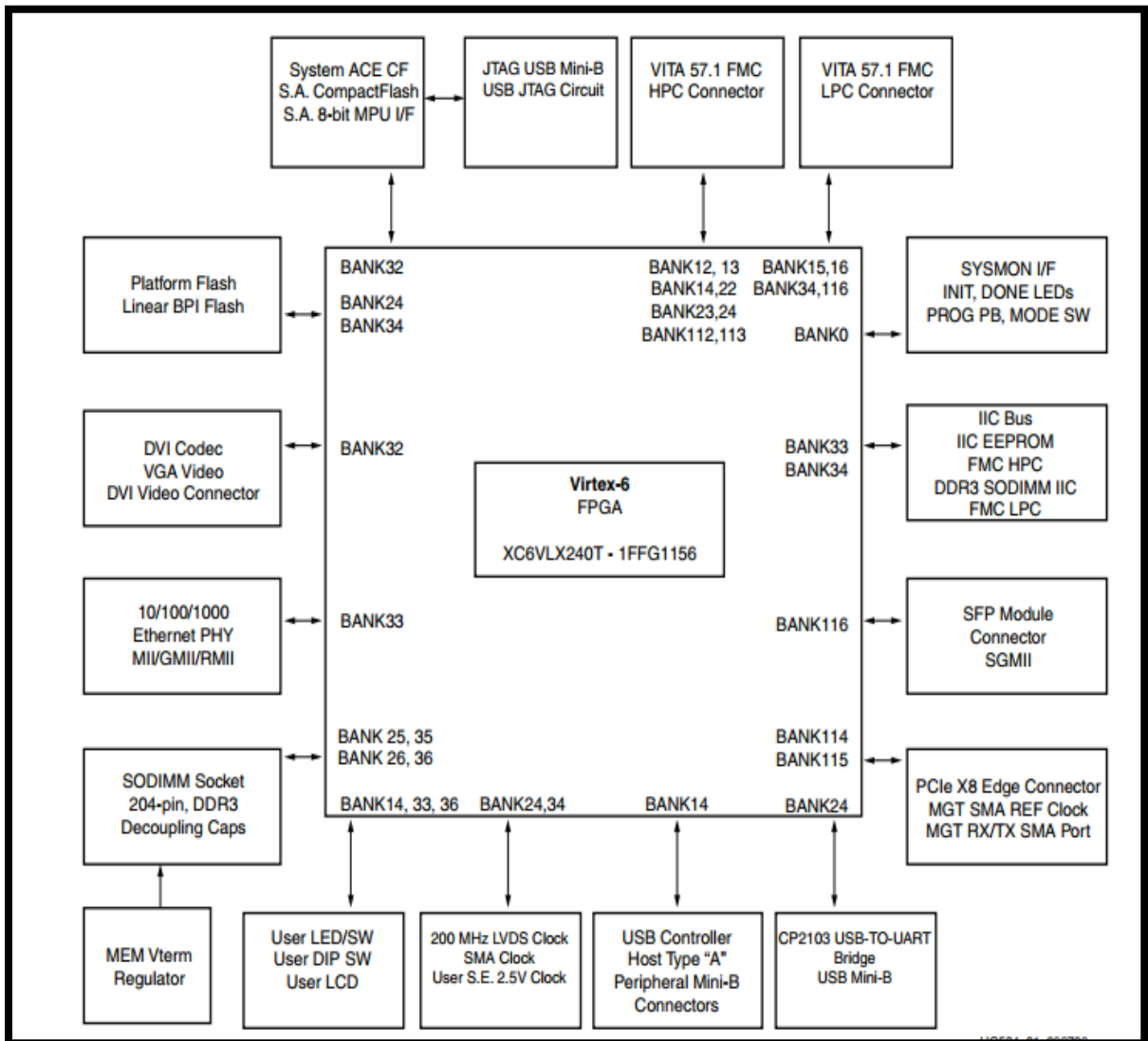


Figure 3-2: Block diagram of Virtex-6

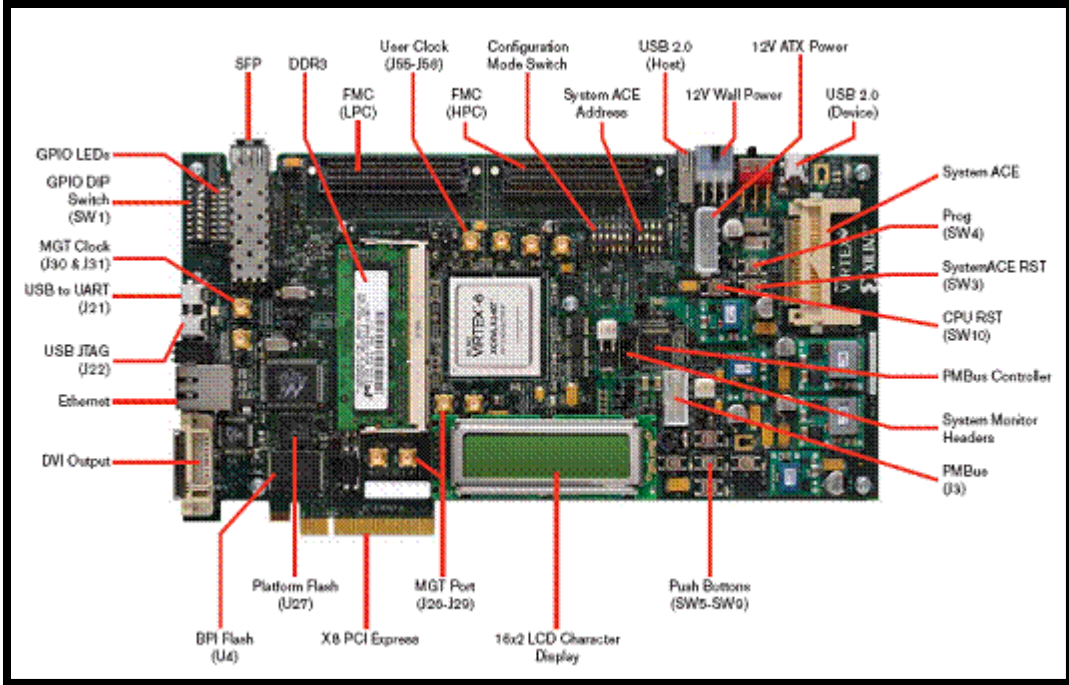


Figure 3-3: Virtex-6 board

Few important features of this board are discussed in Table 3-1.

Table 3-1: Features of Virtex-6 XC6VLX240T-1FFG1156

No.	Feature	Notes
1	FPGA Virtex-6	XC6VLX240T-1FFG1156
2	DDR3 SODIMM	Micron 512 MB MT4JSF6464HY-1G1
3	128 MB Platform Flash	Xilinx XCF128X-FTG64C
4	Linear BPI Flash	Numonyx JS28F256P30T95
5	System ACE CF Controller	Xilinx XCCACE-TQ1441 (bottom of board)
6	JTAG Cable Connector (USB Mini-B)	USB JTAG circuit
7	Clock generation	200 MHz OSC, SMA connectors
	a. 200 MHz oscillator (on backside)	sitime 200 MHz 2.5V LVDS OSC
	b. oscillator socket, single-ended	MMD Components 66 MHz 2.5 V
	c. SMA connectors	SMA pair

	d. MGT REFCLK SMA connectors	SMA pair
8	SFP connector	AMP 136073-1
9	Ethernet (10/100/1000) with SGMII	Marvel M88E1111 EPHY
10	USB Mini-B, USB-to-UART	Silicon Labs CP2103GM bridge
11	USB-A Host, USB Mini-B peripheral connectors	Cypress CY7C67300-100AXI
12	Video- DVI connector	Chrontel CH7301C-TF codec
13	Status LEDs	
	a. Ethernet status	Right-angle link rate and direction LEDs
	b. FPGA INIT, DONE	Init (red), done (green)
	c. system ACECF status	Status (green), Error (red)
14	User I/O	
	a. User LEDs, green(8)	User I/O (active-High)
	b. User pushbuttons, N.O. momentary (5)	User I/O (active-High)
	c. User LEDs, green (5)	User I/O (active-High)
	d. User DIP switch (8-pole)	User I/O (active-High)
	e. User GPIO SMA connectors	SMA pair
	f. LCD 16 Character x 2 line display	Displaytech S162D BA BC
15	Switches	
	a. Power ON/OFF	Slide switch
	b. FPGA –PROG- B Push button	Active-low

3.1.2. SIM900 Module

SIM900 Module works on a number of frequencies including 850 MHz, 900 MHz, 1800 MHz and 1900 MHz. To use as plug in GSM modem is really easy to use and portable. The PC Serial port can be directly connected with the RS232 Level converter on the modem. The AT commands can be used to configure the baud rate where by initially it is in Auto baud mode. To enable the user to connect with internet via GPRS, the GSM/GPRS RS232 modem has an internal TCP/IP stack. It can be used for data transfer as well as SMS applications.



Figure 3-4: SIM900 module

3.1.3. IP Camera

IP camera used provides Day/Night high-resolution image. The Compression format used is Motion-JPEG. Its resolution is CMOS 300,000 pixels. Pan Horizontal rotation is 0-270° whereas tilt vertical rotation is 0-90°. Its range is up to 10m. Multiple users can access it at the same time and it can be protected with the help of a password.



Figure 3-5: IP camera

3.1.4. Laptop and Mobile phone

Laptops are required to create backup storage. In case the user doesn't has internet access and is unable to receive live transmission of video, he still can watch the video clip later

through backup storage. Mobile phone is required to view the video captured by the IP camera.



Figure 3-6: Live video transmission on mobile phone

3.1.5. Wi-Fi Router

Wi-Fi router is needed to assist wireless video transmission on mobile phone. The user can monitor his home even when he is far away.

3.1.6. Switch

The TP-LINK with 5 ports can be used as a switch to convert to Gigabit Ethernet.

3.1.6.1. Features

- Five 10/100/1000Mbps RJ-45 port
- Auto-Negotiation, Auto MDI/MDIX, Half/Full duplex
- IEEE 802.3 flow control
- Plug and play
- 15K jumbo frame

3.1.7. Motion Sensor

Motion sensors are deployed at multiple locations like windows, doors etc. on a single site. The model required for the project is HVR-SA 1200. It can easily detect movement

within an area of 40 feet with a detection angle of 180°. Its sensitivity can be increased and decreased according to the requirements of the user. Day/night settings can also be adjusted.



Figure 3-7: Motion sensor

3.2. Software Requirements

The software(s) required for the implementation of the project includes:

3.2.1. Xilinx ISE Design suite

Xilinx ISE 13.4 was used for the coding of the design. Xilinx has produced Xilinx ISE (Integrated Software Environment) software tool that is used for synthesis and analysis of HDL designs that enables the developer the following:

- The design can be synthesized
- Timing analysis can be performed
- Register-transfer level diagrams examination
- A design's reaction to different stimuli can be simulated

- The programmer can configure the target device

3.2.2. iSpy

The libraries and programs needed for handling multimedia data are provided by iSpy connect which is open source software. To detect and record movement or sound and provides features of monitoring, alerting, surveillance, and security, iSpy uses webcams and microphones. All the captured media is made available securely over the web. It can be run on many computers together [9]. Following are the functions that it performs:

- When motion is detected it automatically captures images
- Motion detection trigger level that is adjustable
- 99 cameras can be supported
- Capability of DVR card
- Capability of multiplexing
- Sensitivity level of image can be adjusted
- Image Archiving (1,000s of images)

3.2.3. IP cam viewer

Real time video from IP or USB cameras can be viewed on mobile phone by using IP Camera Viewer. Wherever security is required, any USB or IP camera can be used, be it your home, office or parking area.

Up to 4 camera feeds can be controlled by the user simultaneously. By using this lightweight application, the user can get a live preview from multiple cameras. By using

centralized camera and layout management, the user can view cameras from multiple remote locations on a single screen. To cater the security needs, the user can change the arrangement and preview layout of the cameras. [10]

The orientation of camera preview can also be adjusted. By supporting many PTZ (Pan/Tilt/Zoom) enabled network cameras, it helps in adjusting the coverage area. IP Camera Viewer provides an additional functionality of digital zoom irrespective of the fact whether it is supported or not.

3.2.4. Wireshark

This software is an open-source and free packet analyzer. Analysis, communications protocol development, network trouble shooting are some of its features [11].

It understands the encapsulation procedure of different networking protocols. Together with the meanings, it can display the fields and parse as specified by different networking protocols. Following are the features of Wireshark:

- Hundreds of protocols can be inspected
- Can perform offline analysis and live capture
- Three-pane packet browser as the standard
- A Multi-platform software supported on Linux, Windows, Solaris, OS X, netbsd, and FreeBSD
- Display filters that are the most powerful
- Rich VoIP analysis
- Captured gzip compressed files can be decompressed on the fly

- Allows live data reading from IEEE 802.11, PPP/HDLC, Ethernet, Bluetooth, ATM, Ring, Frame Relay, Token, USB and FDDI

3.2.5. Docklight

For serial communication protocol, Docklight is an analysis, simulation and testing tool. It can be used for testing the serial communication of a single device and for monitoring the communication between two serial devices. Communications automotive, automation and control, consumer products and equipment manufacturers are some industry areas where productivity can be increased using Docklight. Lastly it is very easy to use.[12]

3.3. Conclusion

The hardware part of the project consists of Virtex-6 kit, SIM900 module, IP camera, motion sensors, switches and router. Xilinx Design Suite 13.4, Docklight, Wireshark and IP cam Viewer are the softwares that were used to carry out different tests.

Chapter 4: Design and Development

This chapter will provide information regarding the methodology and approach leading to the development of this project.

4.1. System Overview

There are two units of the system: the first being FPGA which is connected with the sensors and IP camera along with the SIM900 module and the second unit consists of mobile phone. Target area will be monitored by sensors installed at certain locations including the entrance as well as the windows. The video of the target area will be continuously captured by the rotatable IP camera that is connected to the Ethernet port of FPGA via switch. When an intruder will enter, the system will send 'INT AT LOC X' (Intruder at location x) message to the mobile user via SIM900. The user can then view the video of the target area in his mobile phone by providing the correct username and password. In a highly secure place, the number of police station can also be added so in case an intruder enters, the message will be sent to them as well. If the user fails to read the message at that time, the user can view the video later that will be stored using video storage software. The user can also ARM/DISARM the system. When the user wants to stop the system, he will send a 'DIS' (Disarm) message to FPGA via SIM900.

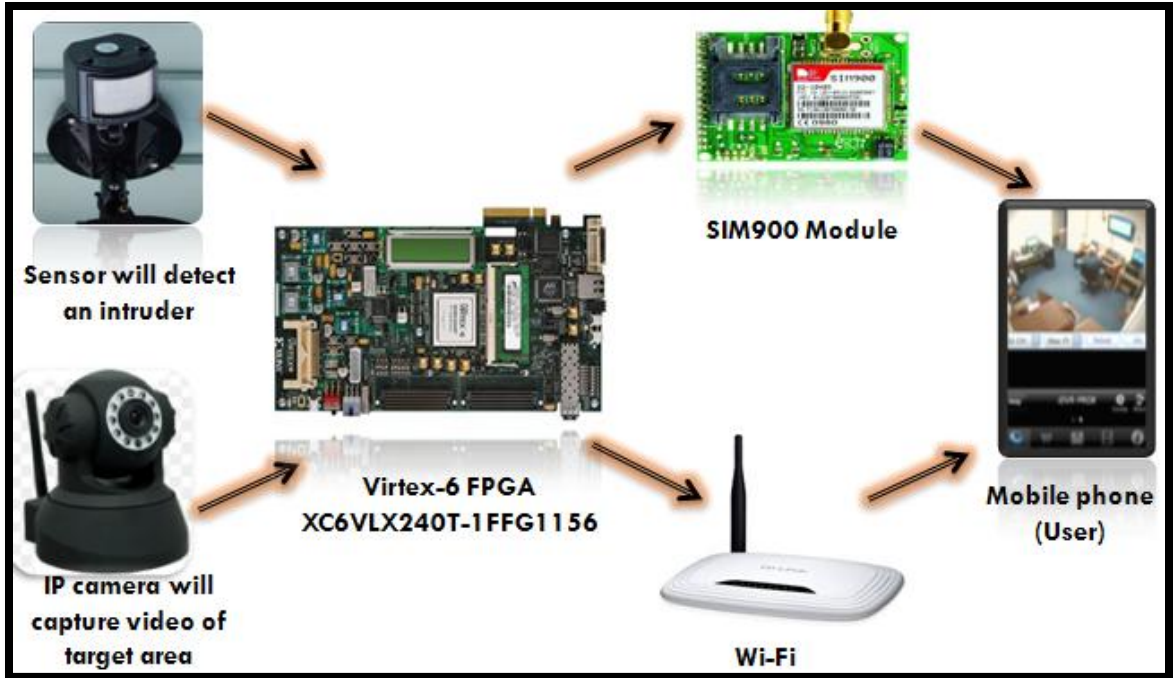


Figure 4-1: System layout

4.1.1. System Block Diagram

Figure 4-2 shows the system block diagram of Mobile Video Surveillance. The sensor is connected to the FPGA through J62 pins and IP camera is connected to the Ethernet port on FPGA. The SIM900 GPRS/GSM module is also connected to the FPGA kit through J62 pins which acts as the communication medium between the FPGA and the mobile station. By using this unit, information is sent from the FPGA to the mobile station and the instructions are sent from the mobile station to the FPGA. FPGA executes any instruction sent by the user from the mobile station. Any mobile phone that can send and receive messages can be used for this purpose. The instructions that are sent to FPGA and any alert from the FPGA is received in form of a message. The video that is captured through the IP camera is transmitted through Wi-Fi. Wi-Fi device is connected to the switch which is further connected to the SFP port on FPGA.

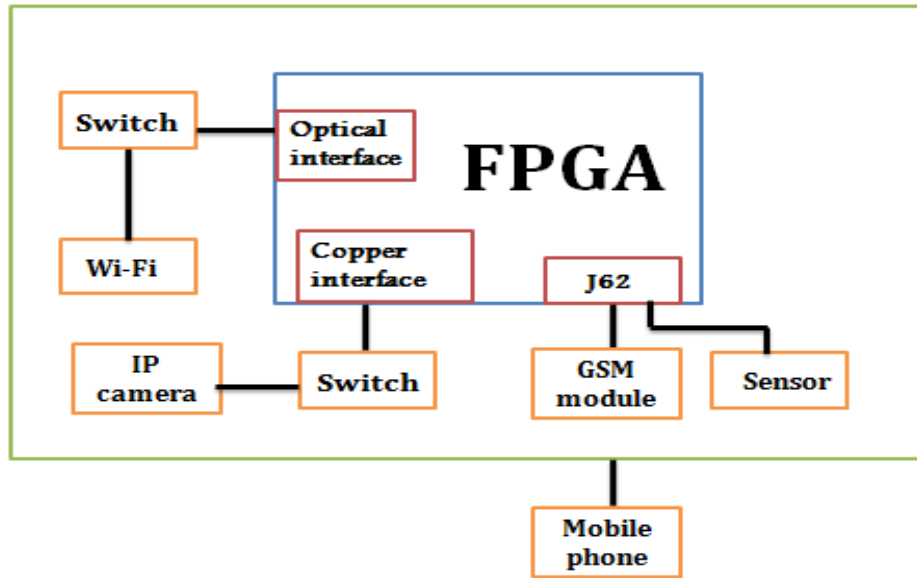


Figure 4-2: System block diagram

4.2. Operating Frequencies

Table 4-1 gives the operating frequencies of various modules of the system.

Table 4-1: Operating frequencies

Module		Operating Frequency
Eth 1 (optical interface)		125 MHz
Eth 0 (copper interface)		125 MHz
GSM	Tx	1200 Hz
	Rx	16 x 1200 Hz

4.3. Hardware Integration

The hardware development of Mobile Video Surveillance revolves around Virtex-6 XC6VLX240T-1FFG1156 FPGA. All the modules including motion sensor, SIM900 and IP camera are interfaced with Virtex-6 XC6VLX240T-1FFG1156 FPGA.

4.3.1. Interfacing Motion Sensor

Sensors will be installed at various locations in the target area. When an intruder will enter, the signal will be cut and a message “INT AT LOC X” will be sent from FPGA to the user via SIM900 module. Motion sensors are assigned on the dip switches on FPGA. Their assignment is shown in Table 4-2

Table 4-2: Sensor pin assignment on FPGA

U1 FPGA Pin	Schematic Net Name	Pin	Function assigned
L20	GPIO_DIP_SW 4	DIP Switch Pin SW1.4	Sensor at location 1
L21	GPIO_DIP_SW 3	DIP Switch Pin SW1.3	Sensor at location 2

Schematic Net name is used to search the general purpose input output Dip switch FPGA pin on the schematics. U1 FPGA pin number is used in the UCF (User Constraint File). FPGA pin L20 and L21 are assigned the functionality of sensor at location 1 and location 2. When the Dip switch 4/3 is high indicating that the signal is cut, a message that an intruder has entered from location 1/2 is sent to the user.

4.3.2. GSM Module

GSM command transmitter module is used to send messages. The five commands that are Test connect, Set number, Set text, Send message and Delete all messages are executed sequentially. GSM command receiver module is used to receive messages. The messages that are received by the system are ARM and DISARM. GSM command transmitter and

receiver together comprise the GSM interfacing module which is part of the Top module in FPGA design. The top module also includes the Ethernet module and the SFP module.

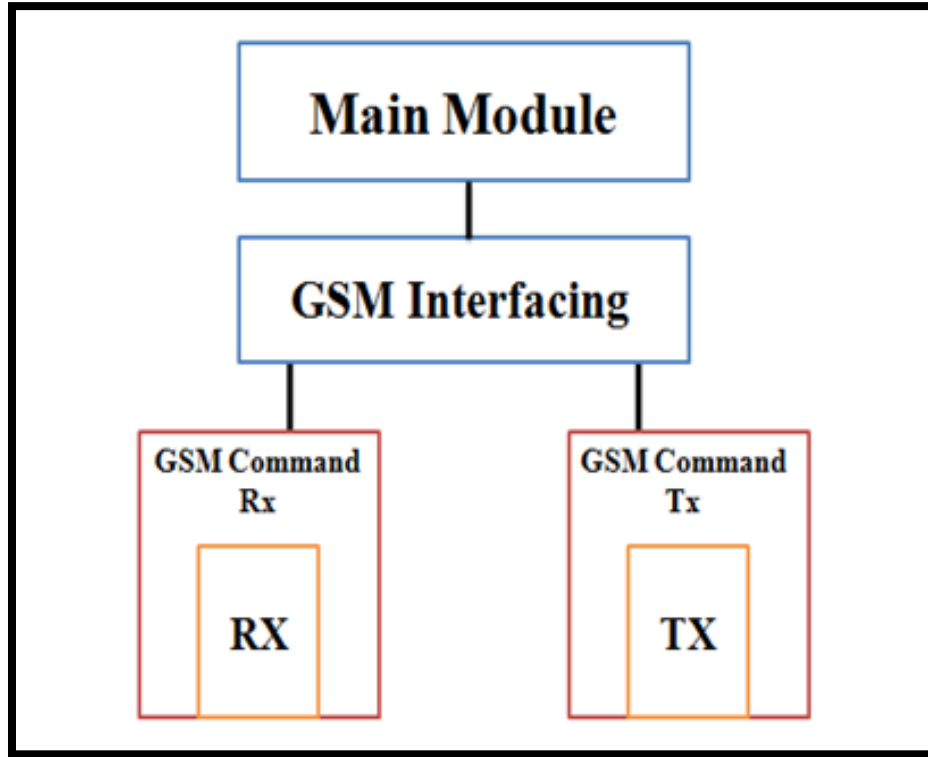


Figure 4-3: GSM module

4.3.2.1. Interfacing SIM900 module

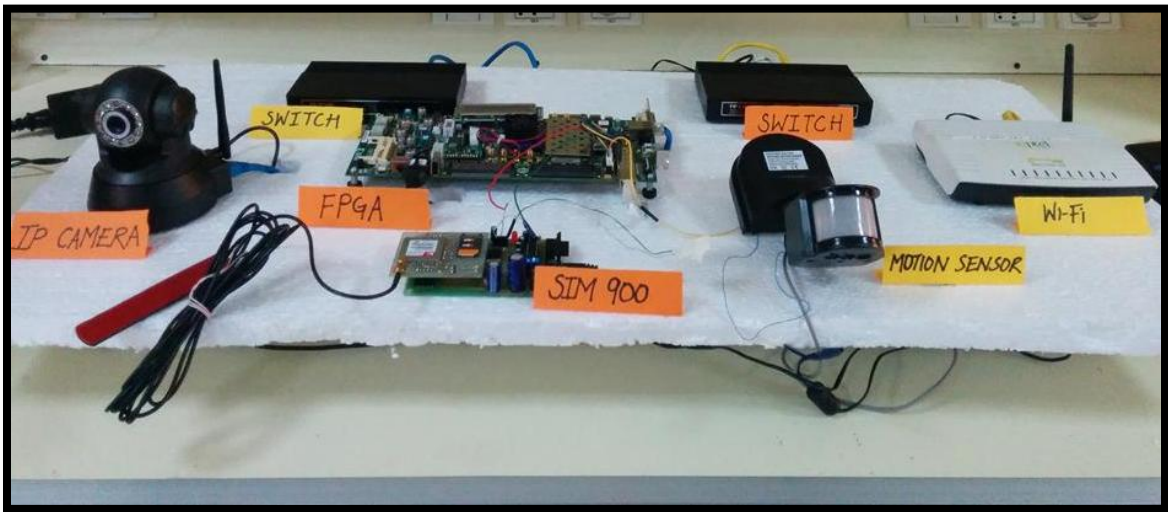


Figure 4-4: System interfacing

J62 pins of FPGA are used to interface SIM900 module with FPGA. The connections are explained in Table 4-3.

Table 4-3: SIM900 module and FPGA interfacing

UI FPGA Pin	Schematic Net Name	Pin	Controlled LED	Function assigned
D22	GPIO_DIP_SW1	DIP Switch Pin SW1.1	-	SIM900 enable
G26	GPIO_SW_C	Pushbutton Switch Pin SW9.2	-	Reset
AC22	GPIO_LED_0	GPIO J62 Pin1	DS12	SIM900 Receiver
AD24	GPIO_LED_7	GPIO J62 Pin 8	DS21	SIM900 Transmitter

The pin D22 is assigned the functionality of enabling SIM900 module. G26 pin resets the system. A wire from pin AC22 (FPGA transmitter) is connected to SIM900 receiver slot. It performs the functionality of transmitting the messages from FPGA to the user. A wire from AD24 (FPGA receiver) is connected to SIM900 transmitter slot. It performs the functionality of receiving the messages from the user to the FPGA.

4.3.2.2. Communication between FPGA and User via SIM900 Module

The SIM900 GPRS/GSM module acts as the communication medium between the FPGA and the mobile station. By using this unit, information is sent from the FPGA to the mobile station and the instructions are sent from the mobile station to the FPGA. This communication is done in form of text messages.

Table 4-4: Communication between FPGA and user via SIM900 module

Action	Message transmitted	Transmitter	Receiver
System powers ON	"POWER"	FPGA	User
When the signal is cut (sensor)	"INT AT LOC X"	FPGA	User
To disable transmission of video	"DIS"	User	FPGA
To enable transmission of video	"ARM"	User	FPGA

4.3.3. Ethernet

IP Camera is used to capture the video of the target area which is connected to Copper interface (Eth 0) of FPGA via switch. The IP camera operates on 100 Mbps. The Copper

interface of FPGA can work on 10/100/1000Mbps but due to its complexity, 1000Mbps have been used, so there was a need to use a switch between the camera and copper interface. A camera with Gbps interface could also be used but due to its unavailability it was not used. The switch performs auto-negotiation (converts from Mbps (IP camera) to Gbps (FPGA)). This video is then transmitted from optical interface of FPGA to the user via switch.

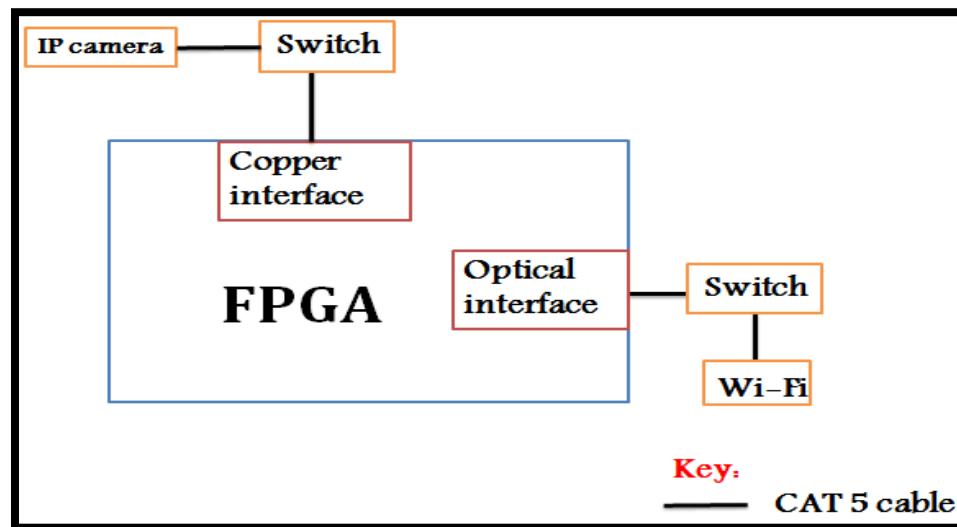


Figure 4-5: FPGA interfacing with IP camera and router

The following ports will be physically connected through CAT 5 cable with RJ-45 connectors at both the ends.

- IP camera and switch
- Switch and Eth 0 (Copper interface FPGA)
- Eth 1 (Optical interface FPGA) and switch
- Switch and router

4.3.3.1. Ethernet Core Design and Configuration

Xilinx's Core Generator tool was used to design a Virtex-6 Embedded Tri-mode Ethernet MAC Wrapper to enable the Ethernet ports of FPGA board. The wrapper that we designed through the software was for standard Virtex-6 FPGA board (ML605), therefore it had to be configured according to the FPGA board available, i.e. Virtex 6 (xc6vlx240t, ff1156) to enable its Ethernet ports. Therefore the entire UCF of the project was changed according to the development board. The changes that were made are shown in the Figure 5-6 and 5-8.

Changing the UCF was quite a challenging task. BANK33 and BANK116 were used from schematics of Virtex-6 (ML605) in order to change the UCF. The following options for the interfaces were used during the core generation in Xilinx ISE 13.4.

Eth0 (copper interface):

- Virtex-6 Tri-Mode Ethernet MAC Wrapper 1.5
- PHY interface: GMII
- BANK 33

Eth1 (Optical interface):

- Virtex-6 Tri-Mode Ethernet MAC Wrapper 2.2
- PHY interface: 1000BASE X PCS PMA
- BANK 116

4.3.3.2. Frame Processing

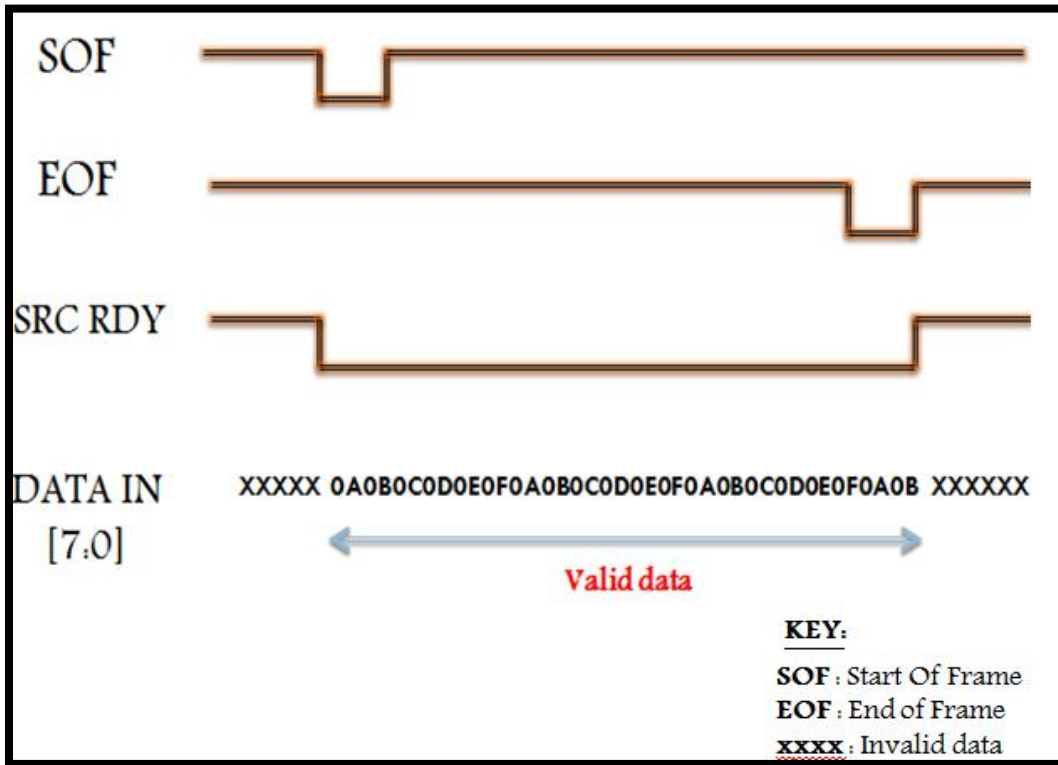


Figure 4-6: Frame processing

At the SOF (start of the frame), as shown in the Figure 4-6, the signal gets low for one cycle. As soon as that happens, the src ready signal also gets low and the reception of valid data starts. 'xxxx' indicates invalid data. The data received during the time the source ready signal is low is the valid data. As soon as the signal gets high during one clock cycle, the src ready signals gets high and it indicates that this is the end of frame. Valid data is recieved for a number of cycles and starts when the frame starts and ends when the frame ends.

The bits utilized by each signal are shown in Table 4-5.

Table 4-5: Number of bits for frame

SIGNALS	BITS
SOF (Start of frame)	1
EOF (End of frame)	1
SRC RDY (Source Ready)	1
DATA IN (Data Input)	8

4.4. Video Storage

Being expensive, closed source, lacking a number of features, restricted data usage, poor control over sensitivity and poor audio control are a number of problems that are linked with the current security and surveillance software. Using iSpy you can do pretty much anything, protecting your business and home with an open source software. iSpy is sensitive to movement and sound so it makes an awesome camera security system. Cameras and microphones can be added, configured and monitored- displaying live video and audio from a variety of network sources. You can switch cameras and microphones on or off, trigger recordings, switch on alerts etc. Following are some important features of iSpy:

- When motion is detected it automatically captures images
- Motion detection trigger level that is adjustable
- 99 cameras can be supported

- Capability of DVR card
- Capability of multiplexing
- Sensitivity level of image can be adjusted
- Image Archiving (1000s of images)

In the project, whenever an intruder enters, motion detector which is installed at entrances detects an intruder and sends alarming messages to user with details of the particular location through which the intruder has entered. The user can then view the live video of the target area in his mobile phone. In case the user fails to view the video at that particular time due to some reasons, he can view the video later which is stored using the iSpy software.

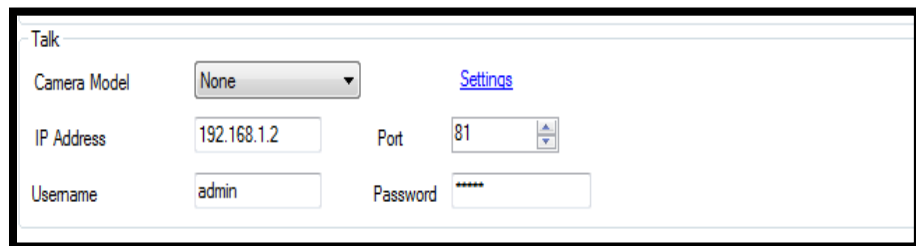
The video storage works on 'Record on detect' i.e. whenever some movement is detected, it starts to record the video and it keeps on recording the video until there is no movement. It will record up to 8 seconds after no movement is detected. The duration for which the video is recorded after no movement is detected can be set according to the user's requirement. After that it will stop recording automatically. These settings are done in order to increase the efficiency since there is no use of continuous storage. It is just wastage of space. The video that is recorded is automatically stored in a local directory as per the user's choice.

Initially cameras are added to the software and camera settings are made according to the camera model and user requirement.

4.4.1. Camera Settings

In the camera settings, following settings need to be changed:

- **Motion detection:** Motion detection sensitivity, display style and number of frames processed can be changed
- **Alerts:** The alert can be enabled on movement or no movement. The alert interval and the action performed upon alert can be set.
- **Recording:** The recording mode whether to detect on movement or without movement and inactivity recording duration can be set.
- **Storage:** By using this option, the user can set the storage directory and the video storage folder size.



Talk			
Camera Model	None		Settings
IP Address	192.168.1.2	Port	81
Username	admin	Password	*****

Figure 4-7: Camera settings

4.4.2. Video Storage On Detection

When movement is detected, iSpy starts recording the video. Blue rectangles start appearing around the border of any object that is detected. Under no movement condition, no video is recorded.



Figure 4-8: Video storage on detection

The blue boxes in Figure 4-8 shows that it has detected an intruder as explained above and red dot on top right shows recording has started.

4.4.3. Assigned Directory

The recorded clips are stored in the local user assigned directory.

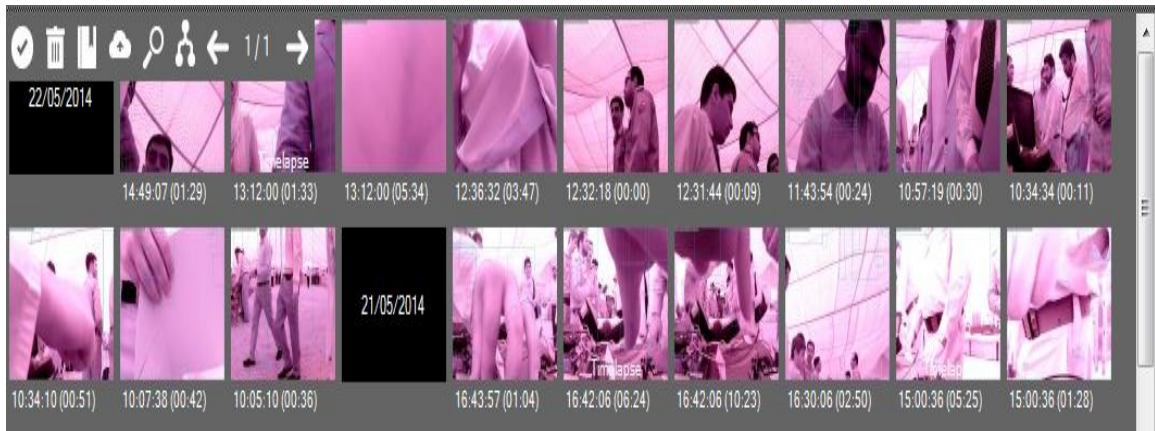


Figure 4-9: Stored clips in software

4.5. Conclusion

The hardware integration comprises of interfacing SIM900 module, IP camera and motion sensor with FPGA. Video storage has been done using iSpy which is mainly the backup storage.

Chapter 5: Project Analysis and Evaluation

In this chapter, simulations will be discussed that were used to evaluate the performance of the project design.

5.1. Top Module

The top module of the project is composed of three sub modules as shown in Figure 5-1.

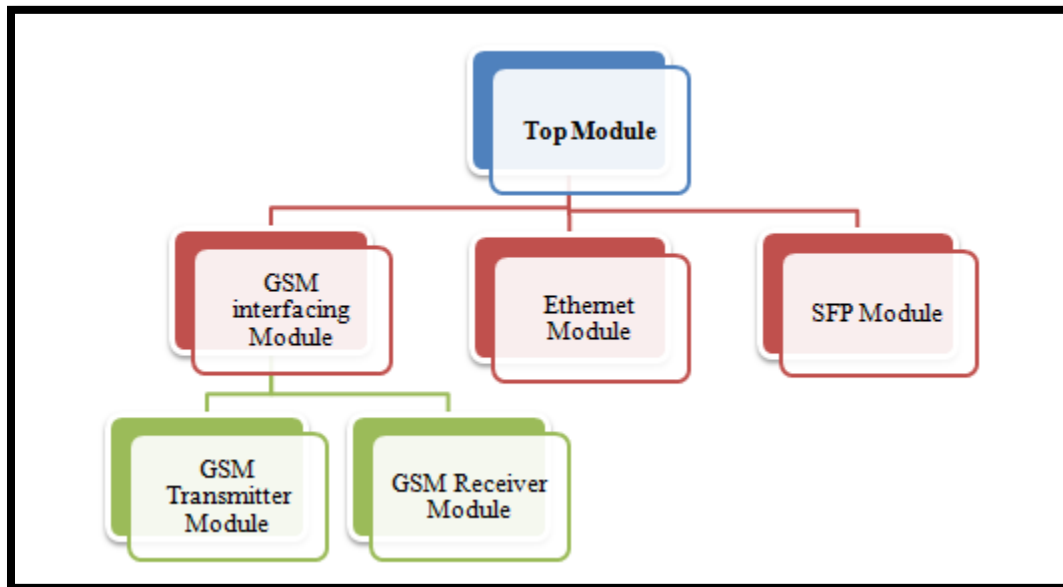


Figure 5-1: Top module

Figure 5-2 shows the design of Top module in Xilinx. The algorithm designed for the project was synthesized, implemented and then its programming file was generated using Xilinx's Project Navigator.

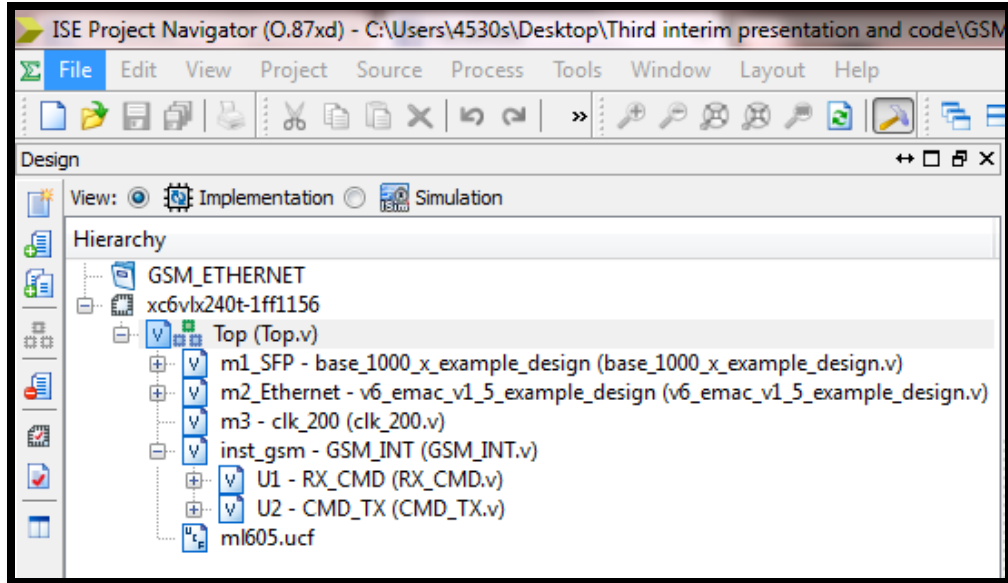


Figure 5-2: Top module design in Xilinx

In order to check whether the designed algorithm was valid or not, initially a test bench was created. Inputs for the test bench were defined within the code. The simulator used was Xilinx's simulator version 13.4, ISim.

5.2. GSM Command Transmitter Module

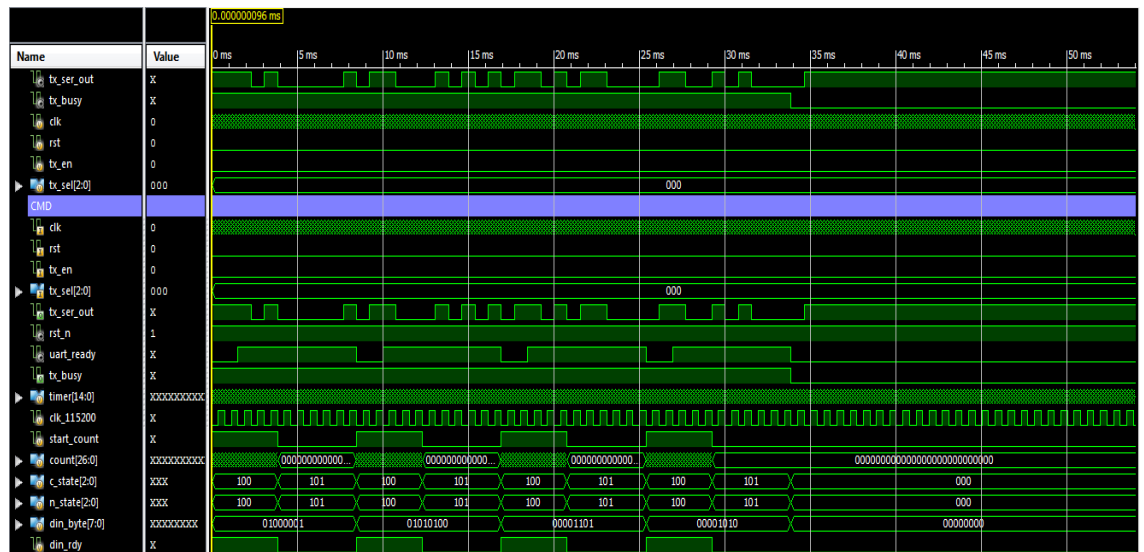


Figure 5-3: GSM command transmitter module

Figure 5-3 shows the timing diagram of GSM command transmitter module. The input of this module is a `go_signal` that indicates the system is ready to start and commands can be executed one by one which is received from the main module. If the state of this signal is 0 then it stays at its initial position but if the state is 1 then the signal passes through the selection function. It can be seen in Figure 5-3 that at the positive edge of `din_rdy` i.e. when the data is ready to be transmitted, eight bits of `din_byte` are transmitted during one cycle. The data is stored in the buffer and most significant eight bits of the data are transmitted during one cycle. When all the data has been transmitted, the `din_rdy` signal becomes low indicating that there is no data to be transmitted. `c_state` and `n_state` indicates the current state and the next state of the state machine respectively. There are five different commands that are executed sequentially following the same procedure.

- **Test connect:** This command makes sure that GSM module is connected to the system.
- **Set number:** This command sets the number of the mobile user with whom the communication is done to control the system.
- **Message:** This command sets the message that has to be sent.
- **Send message:** This command sends the text to the user to let him know about the state of the system.
- **Delete all messages:** This command deletes all the previous inbox messages.

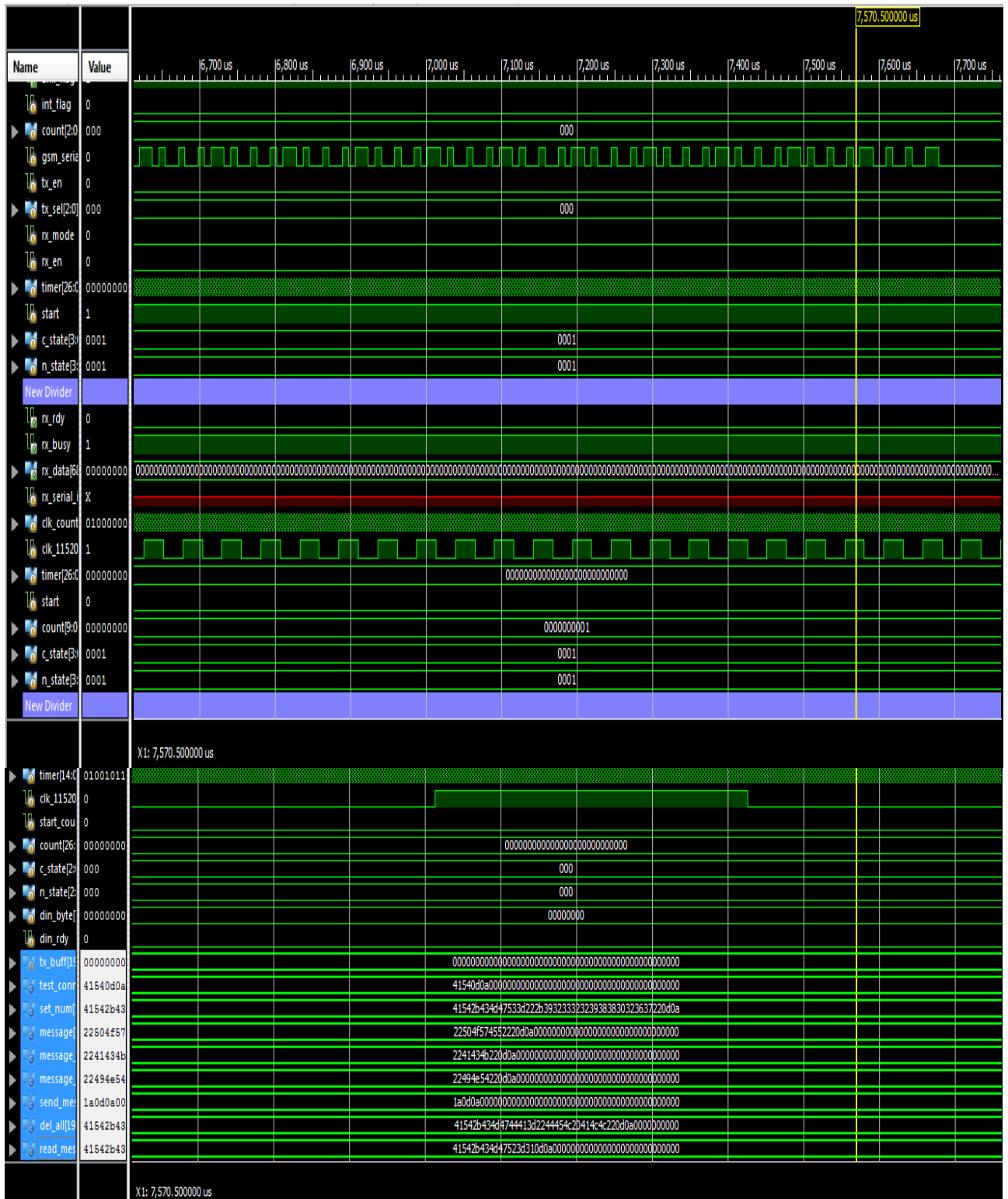


Figure 5-5: GSM interfacing module

Figure 5-5 shows the simulation for GSM interfacing module. This module is formed by combining the GSM command transmitter module and GSM command receiver module. It performs the functionality of both the modules. The sensor functionality is added here. In case an intruder enters, the sensor is cut. Three bits are assigned to the sensor showing which sensor is cut.

The design was synthesized for hardware testing. The design summary generated is shown in Table 5-1.

Table 5-1: Top module design summary

Device Utilization Summary			
Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1,566	301,440	1%
Number used as Flip Flops	1,556		
Number used as Latches	10		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	1,194	150,720	1%
Number used as logic	1,117	150,720	1%
Number using O6 output only	502		
Number using O5 output only	66		
Number using O5 and O6	549		
Number used as ROM	0		
Number used as Memory	24	58,400	1%
Number of occupied Slices	474	37,680	1%
Number of LUT Flip Flop pairs used	1,451		
Number with an unused Flip Flop	258	1,451	17%
Number with an unused LUT	257	1,451	17%
Number of fully used LUT-FF pairs	936	1,451	64%
Number of unique control sets	86		
Number of slice register sites lost to control set restrictions	378	301,440	1%
Number of bonded IOBs	35	600	5%
Number of LOCed IOBs	34	35	97%
IOB Flip Flops	21		
Number of bonded IPADs	4		

Table 5-1 shows the device utilization summary. Number of slice registers available are 301,440 while only 1566 are utilized. In the same way, only 1556 flip flops and only 1% of the memory is utilized.

After the synthesis phase of the synthesis process, a schematic representation of the design was displayed. Figure 5-4 shows the RTL schematic view of the various sub modules of the top module.

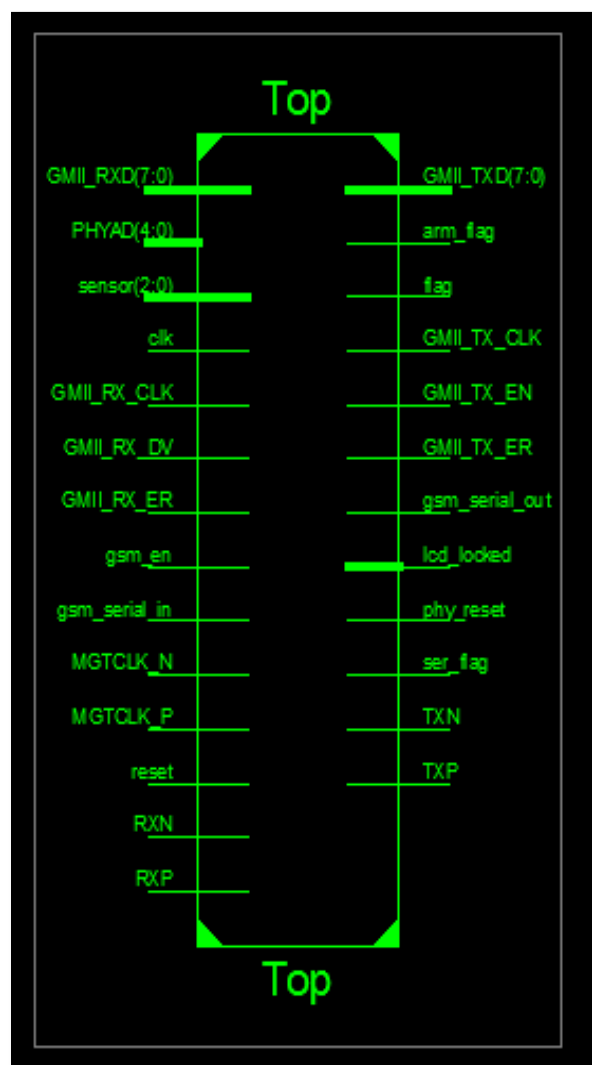


Figure 5-6: RTL schematic of Top module

Figure 5-6 shows the external Transfer of data between the registers and characteristics of circuits by operations are specified by register transfer level.

After design implementation, a .bit file was generated. This is basically used to link the code with the hardware. The programming file was ported on the FPGA using via USB Port available on the FPGA. When it was verified that the code was working, the MCS file was burnt on EPROM.

5.5. Ethernet Core Generation

Xilinx's Core Generator tool was used to design a Virtex-6 Tri-mode Ethernet MAC Wrapper to enable the Ethernet ports of Virtex-6 FPGA board.

5.5.1. Ethernet Module-Eth 0 (Copper Interface)

The steps done to design a Virtex-6 Tri-mode Ethernet MAC Wrapper are:

Tools → Core generator → File → New project → Project options → Part

The following options were selected:

Family: Virtex6

Virtex-6 was used since it has two Ethernet ports which is requirement of our project.

Device: xc6vlx240t

This device type contains features which satisfy our requirements.

Package: ff1156

ff indicates package type and 1156 indicates number of pins.

Speed Grade: -1

Speed grade -1 indicates that the slope of performance is low.

Then we selected Communication and networking → Networking → Virtex-6 Tri-Mode Ethernet MAC Wrapper 1.5

In PHY interface, GMII was selected and core was generated.

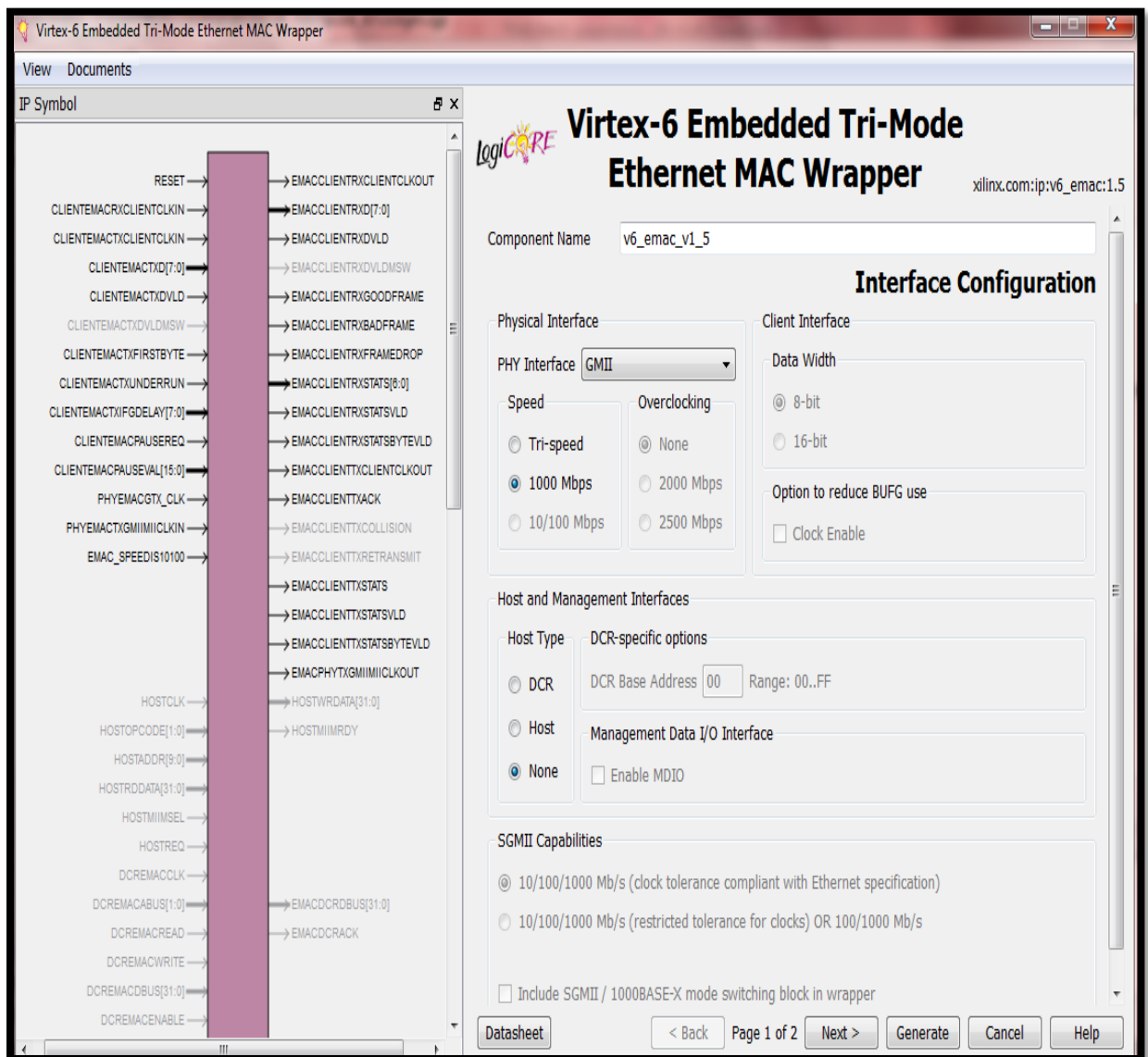


Figure 5-7: Ethernet core generation

Communication and networking → Networking → Virtex-6 Tri-Mode Ethernet MAC
Wrapper 2.2

In PHY interface, 1000BASE X PCS PMA was selected and core was generated.

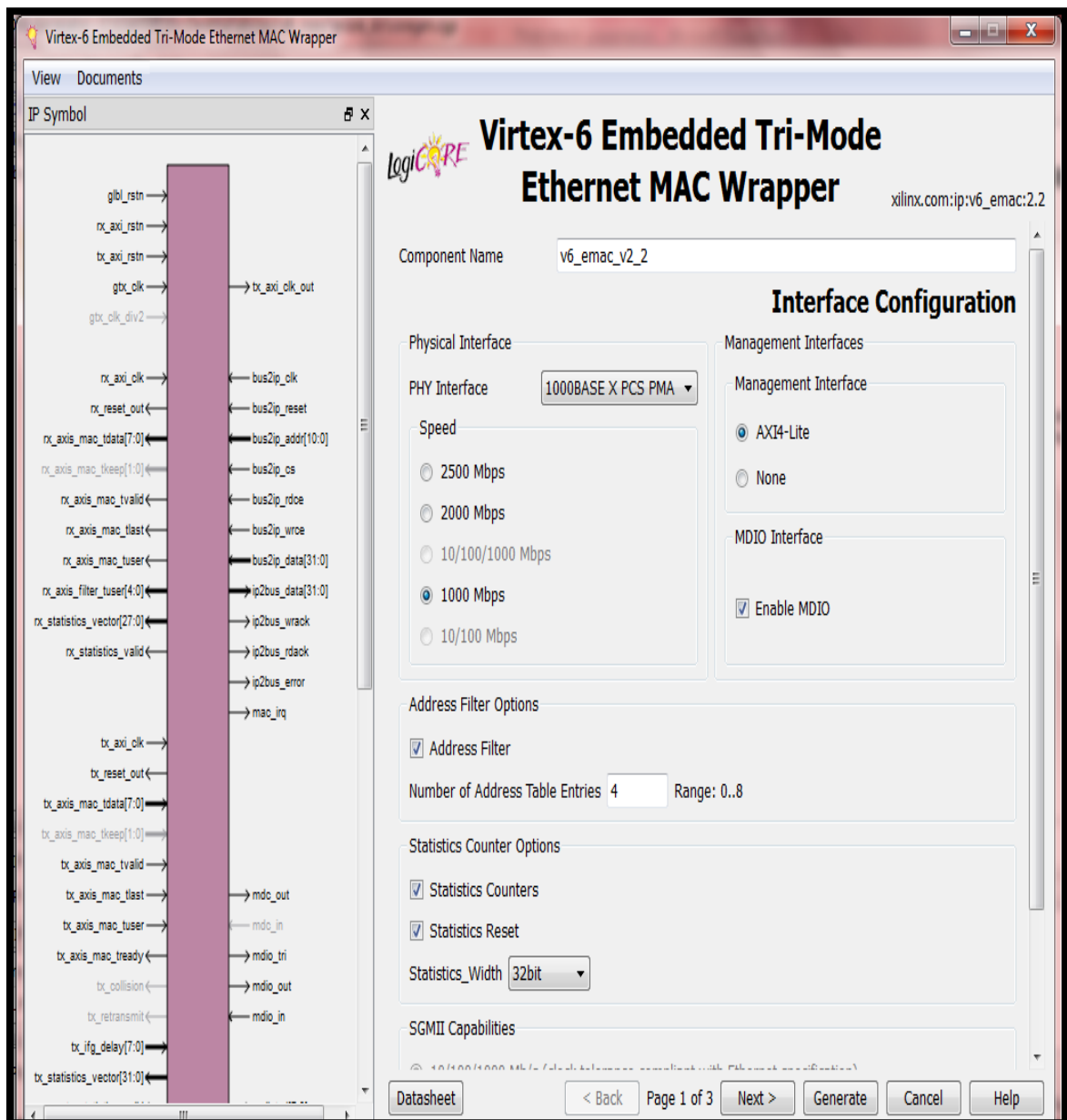


Figure 5-9: SFP module core generation

After the generation of the core, all the files from Virtex-6 embedded Tri-Mode example's design folder were added step by step. We changed the entire UCF of the project for our development board using BANK 116 from schematics of ML-605.

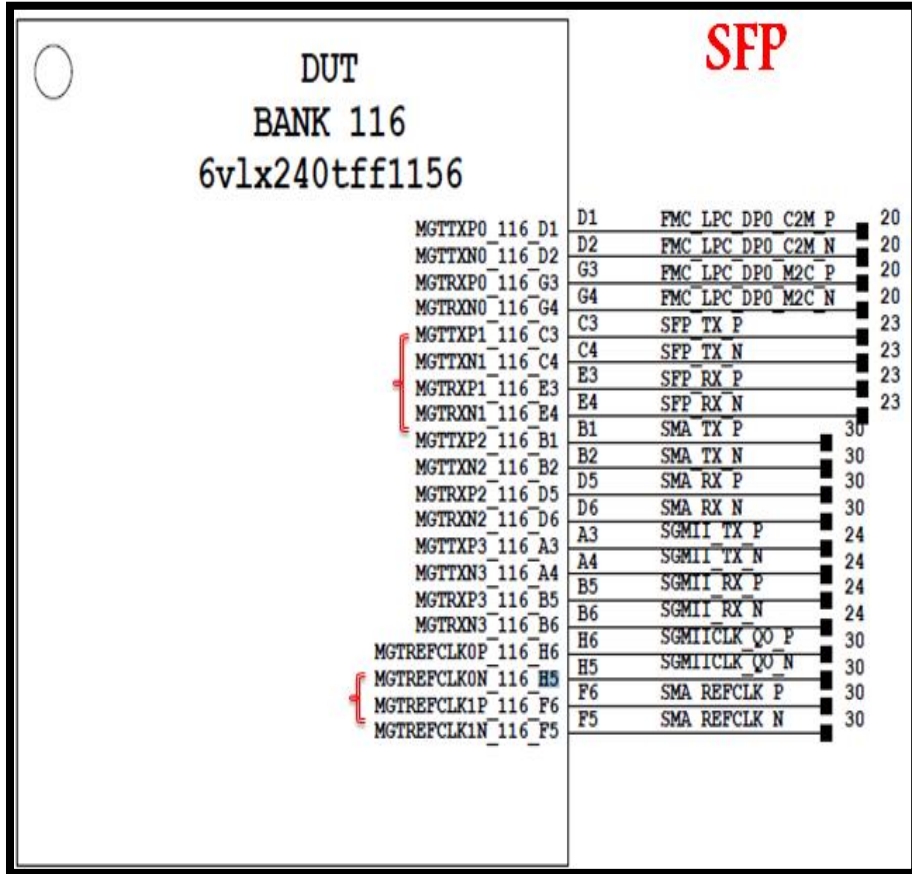


Figure 5-10: BANK 116

Figure 5-10 shows BANK 116 which was used to change the generated UCF. The marked portion indicates the pins that were incorporated in the UCF to configure it according to the requirements.

5.6. Conclusion

Multiple test benches were designed to analyze and evaluate the performance of the project in Xilinx ISE 13.4. The GSM transmitter and receiver modules were tested using these test benches. Ethernet core (Eth0 and Eth1) was generated by using core generator in Xilinx ISE 13.4.

Chapter 6: Future Work and Conclusion

In this chapter, the aspects for further developments in the proposed project will be discussed. Moreover, a summarized view of the entire project will also be provided which highlights Mobile Video Surveillance's core processes and at the end the conclusion will be the discussed.

6.1. Future Work

A project like Mobile Video Surveillance can be customized easily to demand of the user. Some of the enhancements are discussed in this chapter that can be made to the project in future to enhance its features.

6.1.1. H.264 Video Compression

To effectively send video files over a network and store them on computer disks, video compression technologies are used that reduce and remove redundant video data. A significant reduction in file size can be achieved without much effect on the visual quality by using these techniques.

The latest MPEG standard for video encoding is H.264. As compared to Motion JPEG, H.264 encoder can effectively reduce the size of a digital video file by more than 80% and 50% more as compared to the MPEG-4 standard. Thus requiring lesser storage space and network bandwidth for a video file. Similarly given the same bit rate, a much higher video quality can be achieved.

Digital television, mobile TV, internet video streaming, DVD-Video and video conferencing are some of the applications of video compression technology. The products

from different manufacturers including storage media, encoders and decoders can inter-operate due to standardization of video compression. A video can be converted into a compressed format by using an encoder and it can uncompressed by using a decoder. H.264 can be implemented in a video surveillance system by first encoding the video from the camera into a bit stream and then sending it to a decoder which reconstructs the original video.

6.1.1.1. Working of H.264

A compressed H.264 bit stream can be produced by carrying out prediction, transforming and encoding processes. Inverse processes are performed by H.264 video decoder including decoding, inverse transform and reconstruction to produce the original video. As shown in the figure below, initially a sequence of original video frames are encoded into the H.264 format, a series of bits that represents the video in compressed form which are then stored or transmitted and can be decoded to reconstruct the video sequence. It must be kept in mind that the original sequence is not identical since this is a lossy compression technique which does not retain the original quality and picture. [13]

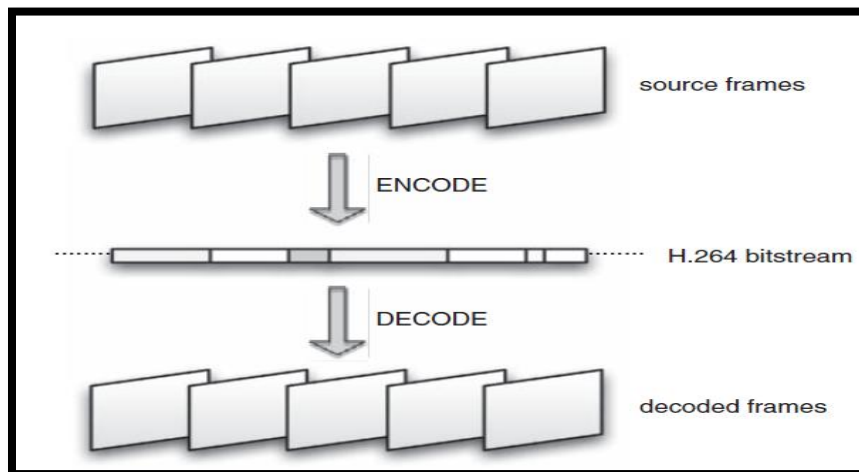


Figure 6-1: Video coding

This H.264 compression can be added to our project. The video from the IP camera can be compressed using FPGA and then this video can be sent to the user's mobile phone.

6.1.2. Encryption and Decryption of Video

FPGA Implementation of Real Time Encryption Engine for Real Time Video Encryption can be done to implement crypto cores for three different algorithms viz. AES, 3DES and Twofish in order to achieve real time encryption and decryption of video data from a real time source.

6.1.3. Face detection and fingerprint algorithms

The locations and sizes of human faces in digital images can be located using a technology known as face detection. Objects such as buildings, trees and bodies are rejected there by detecting only faces. Face detection can be regarded as a more general case of face localization. The locations and sizes of a number of faces (usually one) that are known is found through face localization. Face processing and bitwise matching with the underlying face image that is stored in the database is carried out in face detection. There are a number of face detection applications including human computer interface, video surveillance, biometrics and image database management.

The automated method of verifying a match between two human fingerprints is known as fingerprint recognition or fingerprint authentication. A number of individuals can be identified and their identity can be verified by using this method.

6.1.4. Object Classification

Object classification can be added to our project that will help in differentiating vehicles and humans.

6.1.5. Increase in the coverage area

In our project we used one camera and one motion sensor and other two sensors were assigned on the DIP switches. To secure bigger areas we can add more number of cameras and sensors.

Other syndicates can work on the above mentioned areas to make this project more secure and by integrating our project with one of the above techniques would produce a very useful project for the industry.

6.2. Conclusion

Hence, our mobile video surveillance provides completely digital solution. It can enable efficient surveillance with the help of many security features. Many people have worked on individual modules in the past. We have customized the modules according to our requirements. Whereas we have added many security features from our own and integrated them together into a complete system.

Programming has been done in verilog using Xilinx and implemented on FPGA. Interfacing of camera, sensors, SIM 900 and the motion sensor has been done with FPGA. FPGA controls all the processing i-e the sending and receiving of messages, video transmission and the intruder detection part by using motion sensor.

In a highly secure place, the number of police station can also be added so incase an intruder enters, the message will be sent to them as well. This security system can be implemented in banks or highly secured areas.

Chapter 7: Bibliography

- [1] www.ecs.csun.edu/~smirzaei/site2014/fpga_fundamentals.pdf
- [2] http://www.cosmiac.org/tutorial_1.html
- [3] http://www.cosmiac.org/tutorial_2.html
- [4] http://www.cosmiac.org/tutorial_8.html
- [5] http://www.cosmiac.org/tutorial_9.html
- [6] http://www.cosmiac.org/tutorial_10.html
- [7] hdlplanet.tripod.com/verilog/verilog-manual.htm
- [8] www.xilinx.com/publications/archives/xcell/Xcell48.pdf
- [9] www.ispyconnect.com/
- [10] www.deskshare.com/ip-camera-viewer.aspx
- [11] www.wireshark.org/
- [12] www.docklight.de/pdf/docklight_manual.pdf
- [13] www.slideshare.net/vcodex/h264-video-compression-an-overview

APPENDIX A-1

Code for Top Module

```
`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////

module Top(

    MGTCLK_P,

    MGTCLK_N,

    reset,

    TXP,

    TXN,

    RXP,

    RXN,

    PHYAD,

    GMII_TXD,

    GMII_TX_EN,

    GMII_TX_ER,

    GMII_TX_CLK,

    GMII_RXD,

    GMII_RX_DV,

    GMII_RX_ER,

    GMII_RX_CLK,

    phy_reset,

    lcd_locked,

    clk,

    // rst,

    sensor,

    gsm_en,

    gsm_serial_in,
```

```

gsm_serial_out,
flag,
arm_flag,
ser_flag );
input MGTCLK_P;
input MGTCLK_N;
input reset;
output TXP;
output TXN;
input RXP;
input RXN;
input [4:0]PHYAD;
output [7:0] GMII_TXD;
output GMII_TX_EN;
output GMII_TX_ER;
output GMII_TX_CLK;
input [7:0] GMII_RXD;
input GMII_RX_DV;
input GMII_RX_ER;
input GMII_RX_CLK;
output phy_reset;
output lcd_locked;

//=====

input          clk;
input          [2:0] sensor;
input          gsm_en;
input          gsm_serial_in;
output         gsm_serial_out;

```

```

output          flag;

output          arm_flag;

output          ser_flag;

//-----wires-----//

wire  wire_tx_clk_1;

wire  wire_clk_125;

wire  ref_clk;

wire  locked;

wire  reset_1;

wire  [7:0] wire_in_tx_ll_data_i_0;

wire  wire_in_tx_ll_sof_n_i_0;

wire  wire_in_tx_ll_eof_n_i_0;

wire  wire_in_tx_ll_src_rdy_n_i_0;

wire  [7:0] wire_tx_ll_data_i_0;

wire  wire_tx_ll_sof_n_i_0;

wire  wire_tx_ll_eof_n_i_0;

wire  wire_tx_ll_src_rdy_n_i_0;

wire  kl;

reg  reset_flag = 1'b0;

reg  lock_delay1,lock_delay2;

reg[11:0]phy_reg;

reg  Phy_rst;

wire  phy_reset_wire;

////-----key signal-----

base_1000_x_example_design  m1_SFP

( // Client receiver interface

```

```

.EMACCLIENTRXDVL          (),
.EMACCLIENTRXFRAMEDROP   (),
.EMACCLIENTRXSTATS       (),
.EMACCLIENTRXSTATSVLD    (),
.EMACCLIENTRXSTATSBYTEVLD (),
// Client transmitter interface
.CLIENTEMACTXIFGDELAY      (),
.EMACCLIENTTXSTATS       (),
.EMACCLIENTTXSTATSVLD    (),
.EMACCLIENTTXSTATSBYTEVLD (),
// MAC control interface
.CLIENTEMACPAUSEREQ       (),
.CLIENTEMACPAUSEVAL       (),

// EMAC-transceiver link status
.EMACCLIENTSYNCAQSTATUS   (),
.EMACANINTERRUPT          (),
// 1000BASE-X PCS/PMA interface
.TXP                       (TXP),
.TXN                       (TXN),
.RXP                       (RXP),
.RXN                       (RXN),
.PHYAD                     (),
// 1000BASE-X PCS/PMA clock buffer input
/-- MGTCLK_N,
/-- MGTCLK_P,
// Asynchronous reset
.RESET                     (reset_1),
.clk_ds1                   (clk_ds),
.clk_125                   (wire_clk_125),

```

```

.in_tx_ll_data_i_0      (wire_in_tx_ll_data_i_0),
.in_tx_ll_sof_n_i_0    (wire_in_tx_ll_sof_n_i_0),
.in_tx_ll_eof_n_i_0    (wire_in_tx_ll_eof_n_i_0),
.in_tx_ll_src_rdy_n_i_0 (wire_in_tx_ll_src_rdy_n_i_0),
.tx_ll_data_i_0        (wire_tx_ll_data_i_0),
.tx_ll_sof_n_i_0       (wire_tx_ll_sof_n_i_0),
.tx_ll_eof_n_i_0       (wire_tx_ll_eof_n_i_0),
.tx_ll_src_rdy_n_i_0   (wire_tx_ll_src_rdy_n_i_0)

//-----
);

v6_emac_v1_5_example_design m2_Ethernet
( // Client receiver interface
.EMACCLIENTRXDVL      (),
.EMACCLIENTRXFRAMEDROP  (),
.EMACCLIENTRXSTATS    (),
.EMACCLIENTRXSTATSVLD  (),
.EMACCLIENTRXSTATSBYTEVLD  (),
// Client transmitter interface
.CLIENTEMACTXIFGDELAY  (),
.EMACCLIENTTXSTATS    (),
.EMACCLIENTTXSTATSVLD  (),
.EMACCLIENTTXSTATSBYTEVLD  (),
// MAC control interface
.CLIENTEMACPAUSEREQ    (),
.CLIENTEMACPAUSEVAL    (),

// Clock signal
/--GTX_CLK,

// GMII interface

```

```

.GMII_TXD          (GMII_TXD),

.GMII_TX_EN        (GMII_TX_EN),

.GMII_TX_ER        (GMII_TX_ER),

.GMII_TX_CLK       (GMII_TX_CLK),

.GMII_RXD          (GMII_RXD),

.GMII_RX_DV        (GMII_RX_DV),

.GMII_RX_ER        (GMII_RX_ER),

.GMII_RX_CLK       (GMII_RX_CLK),

// Reference clock for IODELA Ys

.REFCLK            (ref_clk),

// Asynchronous reset

.RESET            (reset_1),

.tx_clk_1          (wire_tx_clk_1),

.in_tx_ll_data_i_1 (wire_tx_ll_data_i_0),

.in_tx_ll_sof_n_i_1 (wire_tx_ll_sof_n_i_0),

.in_tx_ll_eof_n_i_1 (wire_tx_ll_eof_n_i_0),

.in_tx_ll_src_rdy_n_i_1 (wire_tx_ll_src_rdy_n_i_0),

.tx_ll_data_i_1    (wire_in_tx_ll_data_i_0),

.tx_ll_sof_n_i_1   (wire_in_tx_ll_sof_n_i_0),

.tx_ll_eof_n_i_1   (wire_in_tx_ll_eof_n_i_0),

.tx_ll_src_rdy_n_i_1 (wire_in_tx_ll_src_rdy_n_i_0)

    ///-----

);

// IBUF reset_ibuf (

//   .I(reset),

//   .O(reset_1)

// );

IBUFDS_GTXE1 ckingen (

    .I (MGTCLK_P),

```

```

        .IB (MGTCLK_N),
        .CEB (1'b0),
        .O (clk_ds),
        .ODIV2 ()
    );
    BUFG bufg_tx (
        .I (wire_clk_125),
        .O (wire_tx_clk_1)
    );

    clk_200 m3
    (
        .CLK_IN1(wire_clk_125),
        .CLK_OUT1(ref_clk),
        .LOCKED(locked)
    );
    //assign phy_reset = !reset_1;
    assign lcd_locked = locked;
    //-----auto reset-----//
    assign phy_reset_wire = (!(lock_delay2 ^ locked)) || reset_flag; //switch;
    always @ (posedge wire_tx_clk_1)
    begin
        lock_delay 1 <= locked;
        lock_delay 2 <= lock_delay 1;
    end
    always @ (phy_reset)
    begin
        if(phy_reset == 0)
            reset_flag = 1'b1;
    end
end

```



```

assign reset_1 = !phy_reset | reset | (~arm_flag);

always@(posedge wire_tx_clk_1)

begin

if(phy_reset_wire==1'b0)

begin

phy_reg<=12'b1100_0000_0000;

Phy_rst<=1'b1;

end

else

begin

phy_reg<=1'b1;

phy_reg[11:1]<=phy_reg[10:0];

Phy_rst<=phy_reg[11];

end

end

assign phy_reset =Phy_rst;

//=====GSM=====//
GSM_INT inst_gsm(

        .clk(clk),

        .rst(reset),

        .sensor(sensor),

        .gsm_en(gsm_en),

        .gsm_serial_in(gsm_serial_in),

        .gsm_serial_out(gsm_serial_out),

        .flag(flag),

        .arm_flag(arm_flag),

        .ser_flag(ser_flag)

        //flag_1

);

endmodule

```

APPENDIX A-2

Code for Clock:

```
//-----  
// "Output Output Phase Duty Pk-to-Pk Phase"  
// "Clock Freq (MHz) (degrees) Cycle (%) Jitter (ps) Error (ps)"  
//-----  
// CLK_OUT1__200.000_____0.000_____50.0_____109.241_____96.948//  
//-----  
// "Input Clock Freq (MHz) Input Jitter (UI)"  
//-----  
// __primary_____125.000_____0.010  
  
`timescale 1ps/1ps  
  
(* CORE_GENERATION_INFO =  
"clk_200,clk_wiz_v3_2,{component_name=clk_200,use_phase_alignment=true,use_min_o_jitter=false,us  
e_max_i_jitter=false,use_dyn_phase_shift=false,use_inclk_switchover=false,use_dyn_reconfig=false,feedb  
ack_source=FDBK_AUTO,pritype_sel=MMCM_ADV,num_out_clk=1,clkin1_period=8.000,clkin2_peri  
od=10.000,use_power_down=false,use_reset=false,use_locked=true,use_inclk_stopped=false,use_status=f  
alse,use_freeze=false,use_clk_valid=false,feedback_type=SINGLE,clock_mgr_type=MANUAL>manual_o  
verride=false}" *)  
  
module clk_200  
  
  (// Clock in ports  
  
   input   CLK_IN1,  
  
   // Clock out ports  
  
   output  CLK_OUT1,  
  
   // Status and control signals  
  
   output  LOCKED  
  
  );  
  
  // Input buffering  
  
  //-----  
  
  // IBUFG clkin1_buf
```

```

// (.O (clkin1),
// .I (CLK_IN1));

// Clocking primitive
//-----
// Instantiation of the MMCM primitive
// * Unused inputs are tied off
// * Unused outputs are labeled unused
wire [15:0] do_unused;
wire      drdy_unused;
wire      psdone_unused;
wire      clkfbout;
wire      clkfbout_buf;
wire      clkfboutb_unused;
wire      clkout0b_unused;
wire      clkout1_unused;
wire      clkout1b_unused;
wire      clkout2_unused;
wire      clkout2b_unused;
wire      clkout3_unused;
wire      clkout3b_unused;
wire      clkout4_unused;
wire      clkout5_unused;
wire      clkout6_unused;
wire      clkfbstopped_unused;
wire      clkinstopped_unused;

MMCM_ADV
#(.BANDWIDTH      ("OPTIMIZED"),
  .CLKOUT4_CASCADE ("FALSE"),

```

```

.CLOCK_HOLD      ("FALSE"),
.COMPENSATION     ("ZHOLD"),
.STARTUP_WAIT     ("FALSE"),
.DIVCLK_DIVIDE   (1),
.CLKFBOUT_MULT_F (8.000),
.CLKFBOUT_PHASE  (0.000),
.CLKFBOUT_USE_FINE_PS ("FALSE"),
.CLKOUT0_DIVIDE_F (5.000),
.CLKOUT0_PHASE   (0.000),
.CLKOUT0_DUTY_CYCLE (0.500),
.CLKOUT0_USE_FINE_PS ("FALSE"),
.CLKIN1_PERIOD   (8.000),
.REF_JITTER1     (0.010))
mcm_adv_inst
// Output clocks
(.CLKFBOUT      (clkfbout),
.CLKFBOUTB     (clkfboutb_unused),
.CLKOUT0       (clkout0),
.CLKOUT0B      (clkout0b_unused),
.CLKOUT1       (clkout1_unused),
.CLKOUT1B      (clkout1b_unused),
.CLKOUT2       (clkout2_unused),
.CLKOUT2B      (clkout2b_unused),
.CLKOUT3       (clkout3_unused),
.CLKOUT3B      (clkout3b_unused),
.CLKOUT4       (clkout4_unused),
.CLKOUT5       (clkout5_unused),
.CLKOUT6       (clkout6_unused),
// Input clock control
.CLKFBIN       (clkfbout_buf),

```

```

.CLKIN1      (CLK_IN1),
.CLKIN2      (1'b0),
// Tied to always select the primary input clock
.CLKINSEL    (1'b1),
// Ports for dynamic reconfiguration
.DADDR       (7'h0),
.DCLK        (1'b0),
.DEN         (1'b0),
.DI          (16'h0),
.DO          (do_unused),
.DRDY        (drdy_unused),
.DWE         (1'b0),
// Ports .for dynamic phase shift
.PSCLK       (1'b 0),
.PSEN        (1'b 0),
.PSINCDEC    (1'b 0),
.PSDONE      (psdone_unused),
// Other control status signals
.LOCKED      (LOCKED),
.CLKINSTOPPED (clkinstopped_unused),
.CLKFBSTOPPED (clkfbstopped_unused),
.PWRDWN      (1'b0),
.RST         (1'b0));

// Output buffering
//-----
BUFG clk_buf
(.O (clkfbout_buf),
.I (clkfbout));

```

```
    BUFG clkout1_buf
    (.O (CLK_OUT1),
     .I (clkout0));
endmodule
```

APPENDIX A-3

UCF

The xc6vlx240tff1156-1 part is chosen for the example design.

This value should be modified to match the device.

CONFIG PART = xc6vlx240tff1156-1;

Locate the Tri-Mode Ethernet MAC instance

INST "m2_Ethernet/*v6_emac" LOC = "TEMAC_X0Y0";

#INST "m1/*v6_emac" LOC = "TEMAC_X0Y1";

INST "m1_SFP/*gtx0_v6_gtxwizard_i?gtxe1_i" LOC = "GTXE1_X0Y17";

#####

CLOCK CONSTRAINTS

The following constraints are required. If you choose to not use example.

design level of wrapper hierarchy, the net names should be translated to

match the design.

#####

Ethernet GTX_ CLK high 125 MHz reference clock

#NET "*GTX_CLK" TNM_NET = "ref_gtx_clk";

#TIMEGRP "v6_emac_v1_5_clk_ref_gtx" = "ref_gtx_clk";

#TIMESPEC "TS_v6_emac_v1_5_clk_ref_gtx" = PERIOD "v6_emc_v1_5_clk_ref_gtx" 8 ns HIGH 50 %;

Ethernet GTX_ CLK high quality 125 MHz reference clock

NET "WIRE_TX_CLK_1" TNM_NET = "ref_gtx_clk";

TIMEGRP "v6_emac_v1_5_clk_ref_gtx" = "ref_gtx_clk";

TIMESPEC "TS_v6_emac_v1_5_clk_ref_gtx" = PERIOD "v6_emac_v1_5_clk_ref_gtx" 8 ns HIGH 50 %;

```

# Ethernet GMII PHY-side receive clock

NET "*GMII_RX_CLK" TNM_NET = "phy_clk_rx";
TIMEGRP "v6_emac_v1_5_clk_phy_rx" = "phy_clk_rx";
TIMESPEC "TS_v6_emac_v1_5_clk_phy_rx" = PERIOD "v6_emac_v1_5_clk_phy_rx" 7.5 ns HIGH 50
%;

# Ethernet MAC reference clock driven by transceiver

NET "*clk125_o" TNM_NET = "clk_gt_clk";
TIMEGRP "base_1000_x_gt_clk" = "clk_gt_clk";
TIMESPEC "TS_base_1000_x_gt_clk" = PERIOD "base_1000_x_gt_clk" 8 ns HIGH 50 %;

## IDELAYCTRL 200 MHz reference clock
#NET "REFCLK" TNM_NET = "clk_ref_clk";
#TIMEGRP "ref_clk" = "clk_ref_clk";
#TIMESPEC "TS_ref_clk" = PERIOD "ref_clk" 5 ns HIGH 50 %;

#####

# PHYSICAL INTERFACE CONSTRAINTS

# The following constraints are necessary for proper operation, and are tuned
# for this example design. They should be modified to suit your design.

#####

# GMII physical interface constraints
# -----

# Set the IDELAY values on the PHY inputs, tuned for this example design.
# These values should be modified to suit your design.
INST "*gmii?ideldv" IDELAY_VALUE = 26;
INST "*gmii?ideld0" IDELAY_VALUE = 26;
INST "*gmii?ideld1" IDELAY_VALUE = 26;

```



```

INST "*gmii?ideld2" IDELAY_VALUE = 26;
INST "*gmii?ideld3" IDELAY_VALUE = 26;
INST "*gmii?ideld4" IDELAY_VALUE = 26;
INST "*gmii?ideld5" IDELAY_VALUE = 26;
INST "*gmii?ideld6" IDELAY_VALUE = 26;
INST "*gmii?ideld7" IDELAY_VALUE = 26;
INST "*gmii?ideler" IDELAY_VALUE = 26;
INST "*gmii_rxc_delay" IDELAY_VALUE = 0;
INST "*gmii_rxc_delay" SIGNAL_PATTERN = CLOCK;

# Group all IDELAY-related blocks to use a single IDELAYCTRL
INST "*dlyctrl" IODELAY_GROUP = gmii_idelay;
INST "*ideld?" IODELAY_GROUP = gmii_idelay;
INST "*ideldv" IODELAY_GROUP = gmii_idelay;
INST "*ideler" IODELAY_GROUP = gmii_idelay;
INST "*gmii_rxc_delay" IODELAY_GROUP = gmii_idelay;

# The following constraints work in conjunction with IDELAY_VALUE settings to
# check that the GMII receive bus remains in alignment with the rising edge of
# GMII_RX_CLK, to within 2ns setup time and 0 hold time.
INST "GMII_RXD<?>" TNM = "gmii_rx";
INST "GMII_RX_DV" TNM = "gmii_rx";
INST "GMII_RX_ER" TNM = "gmii_rx";
TIMEGRP "gmii_rx" OFFSET = IN 2 ns VALID 2 ns BEFORE "GMII_RX_CLK" RISING;

# Constrain the GMII physical interface flip-flops to IOBs
INST "*gmii?RXD_TO_MAC*" IOB = true;
INST "*gmii?RX_DV_TO_MAC" IOB = true;
INST "*gmii?RX_ER_TO_MAC" IOB = true;
INST "*gmii?GMII_TXD_?" IOB = true;

```

```
INST "*gmii?GMII_TX_EN" IOB = true;
```

```
INST "*gmii?GMII_TX_ER" IOB = true;
```

```
# Location constraints are chosen for this example design.
```

```
# These values should be modified to suit your design.
```

```
# * Note that regional clocking imposes certain requirements
```

```
# on the location of the physical interface pins and the TEMAC instance.
```

```
# Please refer to the Virtex-6 FPGA Embedded Tri-Mode Ethernet MAC
```

```
# User Guide for additional details. *
```

```
#####-----PHY_SIGNAL-----#####
```

```
#####
```

```
#NET "PHY_COL" LOC = "AK13"; ## 114 on U80
```

```
#NET "PHY_CRS" LOC = "AL13"; ## 115 on U80
```

```
#NET "PHY_INT" LOC = "AH14"; ## 32 on U80
```

```
#NET "PHY_MDC" LOC = "AP14"; ## 35 on U80
```

```
#NET "PHY_MDIO" LOC = "AN14"; ## 33 on U80
```

```
#NET "PHY_RESET" LOC = "AH13"; ## 36 on U80
```

```
#NET "PHY_RXCLK" LOC = "AP11"; ## 7 on U80
```

```
#NET "PHY_RXCTL_RXDV" LOC = "AM13"; ## 4 on U80
```

```
#NET "PHY_RXD0" LOC = "AN13"; ## 3 on U80
```

```
#NET "PHY_RXD1" LOC = "AF14"; ## 128 on U80
```

```
#NET "PHY_RXD2" LOC = "AE14"; ## 126 on U80
```

```
#NET "PHY_RXD3" LOC = "AN12"; ## 125 on U80
```

```
#NET "PHY_RXD4" LOC = "AM12"; ## 124 on U80
```

```
#NET "PHY_RXD5" LOC = "AD11"; ## 123 on U80
```

```
#NET "PHY_RXD6" LOC = "AC12"; ## 121 on U80
```

```
#NET "PHY_RXD7" LOC = "AC13"; ## 120 on U80
```

```
#NET "PHY_RXER" LOC = "AG12"; ## 9 on U80
```

```
#NET "PHY_TXCLK" LOC = "AD12"; ## 10 on U80
```

```

#NET "PHY_TXCTL_TXEN"    LOC = "AJ10"; ## 16 on U80
#NET "PHY_TXC_GTXCLK"   LOC = "AH12"; ## 14 on U80
#NET "PHY_TXD0"         LOC = "AM11"; ## 18 on U80
#NET "PHY_TXD1"         LOC = "AL11"; ## 19 on U80
#NET "PHY_TXD2"         LOC = "AG10"; ## 20 on U80
#NET "PHY_TXD3"         LOC = "AG11"; ## 24 on U80
#NET "PHY_TXD4"         LOC = "AL10"; ## 25 on U80
#NET "PHY_TXD5"         LOC = "AM10"; ## 26 on U80
#NET "PHY_TXD6"         LOC = "AE11"; ## 28 on U80
#NET "PHY_TXD7"         LOC = "AF11"; ## 29 on U80
#NET "PHY_TXER"         LOC = "AH10"; ## 13 on U80

```

```
#####//-----
```

```

# Locate the GMII physical interface pins
INST "GMII_TXD<0>" LOC = "AM11";
INST "GMII_TXD<1>" LOC = "AL11";
INST "GMII_TXD<2>" LOC = "AG10";
INST "GMII_TXD<3>" LOC = "AG11";
INST "GMII_TXD<4>" LOC = "AL10";
INST "GMII_TXD<5>" LOC = "AM10";
INST "GMII_TXD<6>" LOC = "AE11";
INST "GMII_TXD<7>" LOC = "AF11";
INST "GMII_TX_EN"  LOC = "AJ10";
INST "GMII_TX_ER"  LOC = "AH10";
INST "GMII_TX_CLK" LOC = "AH12";
INST "GMII_RXD<0>" LOC = "AN13";
INST "GMII_RXD<1>" LOC = "AF14";
INST "GMII_RXD<2>" LOC = "AE14";
INST "GMII_RXD<3>" LOC = "AN12";
INST "GMII_RXD<4>" LOC = "AM12";

```

```

INST "GMII_RXD<5>" LOC = "AD11";
INST "GMII_RXD<6>" LOC = "AC12";
INST "GMII_RXD<7>" LOC = "AC13";
INST "GMII_RX_DV" LOC = "AM13";
INST "GMII_RX_ER" LOC = "AG12";
INST "GMII_RX_CLK" LOC = "AP11";

# Locate the 125 MHz reference clock buffer
INST "bufg_tx" LOC = "BUFGCTRL_X0Y28"; //y6
#
## Locate the 200 MHz delay controller clock buffer
#INST "refclk_bufg" LOC = "BUFGCTRL_X0Y7";

#####
# LOCALLINK FIFO CONSTRAINTS
# The following constraints are necessary for proper operation of the LocalLink
# FIFO. If you choose to not use the LocalLink level of wrapper hierarchy,
# these constraints should be removed.
#####

# LocalLink client FIFO transmit-side constraints
# -----

# Group the clock crossing signals into timing groups
INST "*client_side_FIFO_1?tx_fifo_i?rd_tran_frame_tog" TNM = "tx_fifo_rd_to_wr_1";
INST "*client_side_FIFO_1?tx_fifo_i?rd_retran_frame_tog" TNM = "tx_fifo_rd_to_wr_1";
INST "*client_side_FIFO_1?tx_fifo_i?rd_col_window_pipe_1" TNM = "tx_fifo_rd_to_wr_1";
INST "*client_side_FIFO_1?tx_fifo_i?rd_addr_txfer*" TNM = "tx_fifo_rd_to_wr_1";
INST "*client_side_FIFO_1?tx_fifo_i?rd_txfer_tog" TNM = "tx_fifo_rd_to_wr_1";
INST "*client_side_FIFO_1?tx_fifo_i?wr_frame_in_fifo" TNM = "tx_fifo_wr_to_rd_1";

```

```
TIMESPEC "TS_tx_fifo_rd_to_wr_1" = FROM "tx_fifo_rd_to_wr_1" TO "v6_emac_v1_5_clk_ref_gtx" 8
ns DATAPATHONLY;
```

```
TIMESPEC "TS_tx_fifo_wr_to_rd_1" = FROM "tx_fifo_wr_to_rd_1" TO "v6_emac_v1_5_clk_ref_gtx" 8
ns DATAPATHONLY;
```

```
# Reduce clock period to allow for metastability settling time
```

```
INST "*client_side_FIFO_?tx_fifo_i?wr_tran_frame_tog" TNM = "tx_metastable_1";
```

```
INST "*client_side_FIFO_?tx_fifo_i?wr_rd_addr*" TNM = "tx_metastable_1";
```

```
INST "*client_side_FIFO_?tx_fifo_i?wr_txfer_tog" TNM = "tx_metastable_1";
```

```
INST "*client_side_FIFO_?tx_fifo_i?frame_in_fifo" TNM = "tx_metastable_1";
```

```
INST "*client_side_FIFO_?tx_fifo_i?wr_retran_frame_tog*" TNM = "tx_metastable_1";
```

```
INST "*client_side_FIFO_?tx_fifo_i?wr_col_window_pipe_0" TNM = "tx_metastable_1";
```

```
TIMESPEC "TS_tx_meta_protect_" = FROM "tx_metastable_1" 5 ns DATAPATHONLY;
```

```
# Transmit-side client FIFO address bus timing
```

```
INST "*client_side_FIFO_?tx_fifo_i?rd_addr_tfer*" TNM = "tx_addr_rd_1";
```

```
INST "*client_side_FIFO_?tx_fifo_i?wr_rd_addr*" TNM = "tx_addr_wr_1";
```

```
TIMESPEC "TS_tx_fifo_addr_" = FROM "tx_addr_rd_1" TO "tx_addr_wr_1" 10 ns;
```

```
# LocalLink client FIFO receive-side constraints
```

```
# -----
```

```
## Group the crossing signals into timing groups
```

```
#INST "*client_side_FIFO_1?rx_fifo_i?wr_store_frame_tog" TNM = "rx_fifo_wr_to_rd_1";
```

```
#INST "*client_side_FIFO_1?rx_fifo_i?rd_addr_gray*" TNM = "rx_fifo_rd_to_wr_1";
```

```
#
```

```
#TIMESPEC "TS_rx_fifo_wr_to_rd_" = FROM "rx_fifo_wr_to_rd_1" TO "v6_emac_v_5_clk_ref_gtx" 8
ns DATAPATHONLY;
```

```
#TIMESPEC "TS_rx_fifo_rd_to_wr_" = FROM "rx_fifo_rd_to_wr_1" TO "v6_emac_v_5_clk_phy_rx" 8
ns DATAPATHONLY;
```

```
## Reduce clock period to allow for metastability settling time
```

```
#INST "*client_side_FIFO_1?rx_fifo_i?wr_rd_addr_gray_sync*" TNM = "rx_metastable_1";
```

```
#INST "*client_side_FIFO_1?rx_fifo_i?rd_store_frame_tog" TNM = "rx_metastable_1";
```

```

#TIMESPEC "TS_rx_meta_protect_1" = FROM "rx_metastable_1" 5 ns;

#####

## LOCALLINK sfp FIFO CONSTRAINTS

## The following constraints are necessary for proper operation of the LocalLink
## FIFO. If you choose to not use the LocalLink level of wrapper hierarchy,
## these constraints should be removed.

#####

#

## LocalLink client FIFO transmit-side constraints

## -----

#

## Group the clock signals into timing groups

INST "*client_side_FIFO_?tx_fifo_i?rd_tran_frame_tog" TNM = "tx_fifo_rd_to_wr_0";
INST "*client_side_FIFO_?tx_fifo_i?rd_retran_frame_tog" TNM = "tx_fifo_rd_to_wr_0";
INST "*client_side_FIFO_?tx_fifo_i?rd_col_window_pipe_1" TNM = "tx_fifo_rd_to_wr_0";
INST "*client_side_FIFO_?tx_fifo_i?rd_addr_txfer*" TNM = "tx_fifo_rd_to_wr_0";
INST "*client_side_FIFO_?tx_fifo_i?rd_txfer_tog" TNM = "tx_fifo_rd_to_wr_0";
INST "*client_side_FIFO_?tx_fifo_i?wr_frame_in_fifo" TNM = "tx_fifo_wr_to_rd_0";

TIMESPEC "TS_tx_fifo_rd_to_wr_" = FROM "tx_fifo_rd_to_wr_0" TO "base_1000_x_gt_clk" 8 ns
DATAPATHONLY;

TIMESPEC "TS_tx_fifo_wr_to_rd_0" = FROM "tx_fifo_wr_to_rd_0" TO "base_1000_x_gt_clk" 8 ns
DATAPATHONLY;

# Reduce clock period to allow for metastability settling time

INST "*client_side_FIFO_?tx_fifo_i?wr_tran_frame_tog" TNM = "tx_metastable_0";
INST "*client_side_FIFO_?tx_fifo_i?wr_rd_addr*" TNM = "tx_metastable_0";
INST "*client_side_FIFO_?tx_fifo_i?wr_txfer_tog" TNM = "tx_metastable_0";
INST "*client_side_FIFO_?tx_fifo_i?frame_in_fifo" TNM = "tx_metastable_0";
INST "*client_side_FIFO_?tx_fifo_i?wr_retran_frame_tog*" TNM = "tx_metastable_0";

```

```

INST "*client_side_FIFO_?tx_fifo_i?wr_col_window_pipe_0" TNM = "tx_metastable_0";
TIMESPEC "TS_tx_meta_protect_0" = FROM "tx_metastable_0" 5 ns DATAPATHONLY;

# Transmit-side client FIFO address bus timing
INST "*client_side_FIFO_?tx_fifo_i?rd_addr_txfer*" TNM = "tx_adr_rd_0";
INST "*client_side_FIFO_?tx_fifo_i?wr_rd_addr*" TNM = "tx_adr_wr_0";
TIMESPEC "TS_tx_fifo_addr_" = FROM "tx_adr_rd_" TO "tx_adr_wr_0" 10 ns;

# LocalLink client FIFO receive-side constraints
# -----

## Group the clock crossing signals into timing groups
#INST "*client_side_FIFO_0?rx_fifo_i?wr_store_frame_tog" TNM = "rx_fifo_wr_to_rd_0";
#INST "*client_side_FIFO_0?rx_fifo_i?rd_addr_gray*" TNM = "rx_fifo_rd_to_wr_0";
#
#TIMESPEC "TS_rx_fifo_wr_to_rd_0" = FROM "rx_fifo_wr_to_rd_0" TO "base_1000_x_gt_clk" 8 ns
DATAPATHONLY;
#TIMESPEC "TS_rx_fifo_rd_to_wr_0" = FROM "rx_fifo_rd_to_wr_0" TO "base_1000_x_gt_clk" 8 ns
DATAPATHONLY;
#
## Reduce clock period to allow for metastability settling time
#INST "*client_side_FIFO_0?rx_fifo_i?wr_rd_addr_gray_sync*" TNM = "rx_metastable_0";
#INST "*client_side_FIFO_0?rx_fifo_i?rd_store_frame_tog" TNM = "rx_metastable_0";
#TIMESPEC "TS_rx_meta_protect_0" = FROM "rx_metastable_0" 5 ns;
## Locate the Tri-Mode Ethernet MAC instance
#INST "*v6_emac" LOC = "TEMAC_X0Y0";

## Group design elements around Ethernet MAC
## closure in this example design. These values may be modified
## removed to best suit your design.
#INST "*" AREA_GROUP = "AG_example_design";

```

```

#AREA_GROUP "AG_example_design" RANGE = CLOCKREGION_X1Y0:CLOCKREGION_X1Y3;
#INST "*client_side_FIFO_0?tx_fifo_i?ramgen" LOC = "RAMB36_X7Y16";
#INST "*client_side_FIFO_0?rx_fifo_i?ramgen" LOC = "RAMB36_X7Y17";

#####

# CLOCK CONSTRAINTS

# The following constraints are required. If you choose to not use the example
# design level of wrapper hierarchy, the net names should be translated to
# match your design.

#####

## Ethernet MAC reference clock driven by transceiver
#NET "clk125_o" TNM_NET = "clk_gt_clk";
#TIMEGRP "base_1000_x_gt_clk" = "clk_gt_clk";
#TIMESPEC "TS_base_1000_x_gt_clk" = PERIOD "base_1000_x_gt_clk" 8 ns HIGH 50 %;

#####

# PHYSICAL INTERFACE CONSTRAINTS

# The following constraints are necessary for proper operation, and are tuned
# for this example design. They should be modified to suit your design.

#####

# 1000BASE-X physical interface constraints
# -----
##### SFP #####
////-----

#NET "SFP_LOS"          LOC = "V23";  ## 8  on P4
#NET "SFP_RX_N"        LOC = "E4";   ## 12 on P4
#NET "SFP_RX_P"        LOC = "E3";   ## 13 on P4
#NET "SFP_TX_DISABLE_FPGA" LOC = "AP12"; ## 1  on Q22

```



```

#NET "SFP_TX_N"          LOC = "C4";  ## 19 on P4
#NET "SFP_TX_P"          LOC = "C3";  ## 18 on P4
##
#NET "SGMIICLK_QO_N"     LOC = "H5";  ## 2 on series C55 0.1uF
#NET "SGMIICLK_QO_P"     LOC = "H6";  ## 2 on series C56 0.1uF

# Place the transceiver components, chosen for this example design.
# These values should be modified according to your specific design.
INST "MGTCLK_N" LOC = "H5";
INST "MGTCLK_P" LOC = "H6";
INST "TXP"     LOC = "C3";
INST "TXN"     LOC = "C4";
INST "RXP"     LOC = "E3";
INST "RXN"     LOC = "E4";

##### Local Signal#####
NET "reset" LOC = "G26";
NET "phy_reset" LOC = "AH13";
#NET "lcd_locked" LOC = "AC22";
#=====
NET "arm_flag" LOC = AE23;
NET "flag" LOC = AE22;
#NET "rst" LOC = G26;
NET "clk" LOC = U23;

NET "gsm_en" LOC = D22;
NET "gsm_serial_out" LOC = AC22;
NET "gsm_serial_in" LOC = AD24;
NET "sensor[0]" LOC = L20;
NET "sensor[1]" LOC = L21;

```

NET "sensor[2]" LOC = AE24;

#####3

NET "ser_flag" LOC = J25;

USER MANUAL

1. READING INSTRUCTIONS

This Manual is a guide to the system “Mobile Video Surveillance”. It contains essential instructions for setup and operations.

2. INTERFACING OF THE SYSTEM

2.1. Interfacing Of Motion Sensor With FPGA

Install the motion sensor at the target area such that when there is some motion in the target area the motion sensor detects the motion. Assign the sensors on the dip switches on FPGA. Their assignment is following:

- Sensor 1 is assigned to U1 FPGA Pin L20
- Sensor 2 is assigned to U1 FPGA Pin L21
- Sensor 3 is assigned to U1 FPGA Pin AE24

2.2. Testing The Voltage Of Sim900 Module

Before interfacing the SIM900 module with FPGA check the voltage of the module with the help of DMM. It should be in the range of 3.4V to 4.5V.

2.3. Interfacing Of Sim900 Module With FPGA

Interface SIM900 module with J62 pins of FPGA. The connections are as following:

- SIM900 is enabled by GPIO DIP SW1 (D22)
- The system is reset by GPIO SW C (G26)
- SIM900 receiver is connected to GPIO LED 0 (AC22) on FPGA
- SIM900 transmitter is connected to GPIO LED 7 (AD24) on FPGA

2.4. CAT-5 Cable Connections

Connect the following ports through CAT 5 cable with RJ-45 connectors at both the ends.

- IP camera and switch
- Switch and Eth 0 (Copper interface FPGA)
- Eth 1 (Optical interface FPGA) and switch
- Switch and router

3. OPERATING THE SYSTEM

After making all the connections, first of all turn on the router, IP camera and switches.

Then turn on the SIM900 module, FPGA kit and motion sensor.

After the SIM900 module is turned on, a message of “POWER” is received at the mobile.

When there is some movement in front of the motion sensor, a message of “INT AT LOC 3” will be received at the user’s mobile.

4. VIEWING THE VIDEO ON MOBILE

In order to view the video on mobile phone open the “IP cam viewer” application installed on the mobile phone. IP camera can also be rotated with the help of this application.

5. VIDEO STORAGE ON iSPY SOFTWARE

In order to view the stored video open the iSpy software installed on the laptop. The video storage works on ‘Record on detect’ i.e. whenever some movement is detected, it starts to record the video and it keeps on recording the video until there is no movement. It will record up to 8 seconds after no movement is detected. After that it will stop recording automatically.

5.1. Camera Settings

In the camera settings, following settings need to be changed:

- Motion detection
- Alerts
- Recording
- PTZ
- Cloud storage
- Scheduling
- Storage

5.2. Video Storage on Detection

When movement is detected, iSpy starts recording the video. The blue boxes appear on detection of an intruder and red dot appears on top right showing recording has started.

The recorded clips are stored in the local user assigned directory.

6. CD Contents:

- Project presentation with voice narration
- Project Video
- Project thesis (pdf + word document)
- Codes and executables