POLITECNICO DI TORINO

Faculty of Engineering

Master of Science in Electronic Engineering

Master Thesis

Development of an embedded system for networking applications

Advisor:
prof. Danilo Demarchi

Candidate:
Francesco Gramazio

Company tutor
Telsy SPA
Ing. Fabrizio Vacca

December 2018

Summary

Nowadays, people in the world get in touch with about 100 embedded systems per day, that not only have the aim to realize specific functions but also are a way to introduce innovations. According to the World Trade Statistics, in 2009 the 98% of all programmable devices were embedded, and if in 2010 there were about 16 billion of embedded systems, in 2020 it is estimated that this number will increase, astonishingly reaching 40 billion units. With this capillary spreading of devices, privacy and information are in constant danger: according to Cybercrime Report 2016, cyber-attacks grow of the 350% every year. It is of the foremost importance, therefore, to secure communications through embedded systems, especially for companies (and in particular military companies). National defence agencies must deal with confidential information, thus they have to develop solid devices for security: clear data has to be encrypted prior to the transmission and decrypted after the reception, to ensure that every information that comes out into the external world is safe, with no possibilities to externally monitor the traffic network and to go back to the original clear information.

In this work, we propose the design of a cipher IP board called ENA (Embedded Networked Appliance), aimed to ensure a complete security of IP traffic exchanged on strategic networks for military applications, in order to guarantee a safe exchange of information from one host to another. The device is intended to be installed both at the transmitter and the receiver side, to hide information during the transmission. The Encrypting IP box is available for both optical and copper-based wired Ethernet Interfaces. It is designed for real time applications; the device can be used for VoIP transmissions and for time-sensitive applications, for this reason the UDP protocol is used for transmission. The device is composed by two different parts: a carrier board, that contains components to provide power to the whole system, and the main module, which is a proprietary company board. The main focus of the study is the design of the specific carrier board, with proper interfaces, for the realization of the IP cipher, and the development of the microprocessor and microcontroller software, to execute the different tasks (e.g. download of the FPGA bitfile into the Flash Memory, configuration of the FPGA and the microprocessor application for Ethernet frame manipulation and encryption).

At the end of the design the system is tested with an external traffic analyser to analyse the throughput, and the working conditions are specified. The minimum frame size that allows a correct encrypting of the information through Ethernet has been discovered to

be 1280byte. Thanks to this work, it has been possible to highlight the limits of the microprocessor and, although with the current restrictions the ENA could not be suitable for being an enterprise product, it represents a good proof of concept of how deeply military companies have to deal with the topic of security and confidentiality.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this thesis is the design of an Ecrypting IP Box. The scope of an Ecrypting box is to ensure a complete security of IP traffic exchanged on strategic networks for military applications.
Nowadays the privacy and the informations are in constant danger; according to Cybercrime Report 2016 [4], the cyber attacks annually grow of 350%. So it is evident how the companies and in particular military companies have to develop solid devices for security.

The scope of this device is to guarantee secure exchanging of information from one host to another host. The clear data before being transmitted has to be encrypted to ensure that each information that comes out to the external world, is safe with no possibility to monitor externally the traffic network and to go back to the original clear information. At the other side another Encrypting IP Box has to be install to decrypt the informations that arrive. The Encrypting IP box is available both for optical and copper based wire Ethernet Interfaces. It is designed for real time applications; so the device could be use for Voip transmissions and for time-sensitive applications; for this reason UDP protocol is used for the transmissions (appendix D).

The thesis is divided into five chapters: in the first chapter a description is given related to the problem of security, the use of cipher equipment and the summery of each chapter. In the second chapter a detail description of the embedded system world is made and the reason why, nowadays, the embedded systems are dominating the entire consumer electronics and non-consumer landscape. Moreover, a detailed description of the different embedded systems that are present today and the main components that form a complete embedded system is given. A brief overview of the design embedded system (ENA - Embedded Networked appliance) is also present and where it is positioned into the different functional categories of embedded systems. The third chapter deals with the main project steps for the design of the Encrypting IP Box and a detailed description of each step is present. In the fourth chapter, the Ethernet interfaces of the design device and of a similar commercial device are analysed initially, then theoretical performance and graphs are described to perform critical analysis on the device. Then performance tests are carried

out to measure the throughput of the device; comparisons are performed to comment the different results. In the fifth chapter a summary of the entire work is performed and conclusions are made.
Thanks to this work it has been possible to analyze the performance of the device and it has been possible to decree the final specifications and if it would be possible a future use of the device for enterprise realizations.

Chapter 2

System overview

2.1 Embedded Systems

2.1.1 Introduction

Nowadays, most processing systems are not personal computers, but they are devices that communicate with the external environment; they have a specific function and, in spite of what we think, in most cases, they don't require to open programs or to have interfaces with mouse and keyboard. These systems are called embedded and they dominate the market all over the world even if they are not so known as the general purpose systems. At the moment, it is considered that a person in the world gets in touch daily with about 100 embedded systems. It isn't a surprise if it is thought that a lot of devices have almost one microprocessor like smartphone, cash machine, washing machine, dishwasher, credit card, ink-jet printer, scanner, up to the automotive domain where cars today contain dozens of embedded systems, such as transmission control, cruise control, etc; any kind of device that runs on electric power already has a computing system or will soon have a computing system embedded on it.
The embedded systems not only have the aim to realize specific functions but they are also a way to introduce new innovations. In history we have witnessed the advent of different industrial revolutions. Starting from the 60s, a second industrial revolution broke out which led to the development and diffusion of digital technologies. We have moved from big computers, that occupied entire rooms, to personal computers, to today's huge spread of laptops and smartphones, helped by the possibility to connect all over the world through the network and the internet. Of course this diffusion has a heavy impact on the embedded system market which has a market share of more than 90% of the total electronic sector. According to the World Trade Statistics, in 2009 98% of programmable devices were embedded and if in 2010 there were about 16 billion embedded systems, in 2020 it is estimated that 40 billion units will be reached. [2]
According to an article of Zion Research titled "Embedded Systems Market", the global demand of embedded systems market was valued USD 159.00 billion in 2015 and is expected to become 225.34 billion by the end of 2021 (figure 2.1).
This market growth is driven also by the huge demand, in the last years, of medical

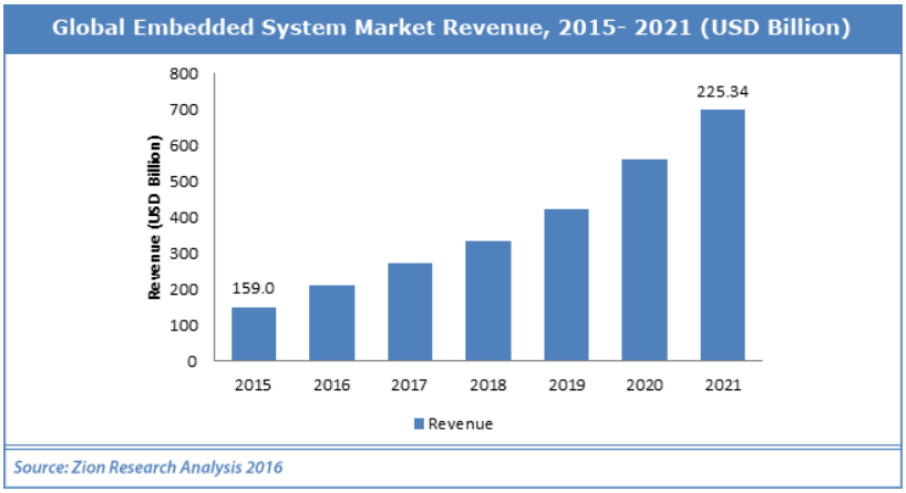

Figure 2.1: Global Embedded System Market Revenue in USD Billion [5]

devices such as ECG embedded systems, heart rate monitors and glucose level monitors. The exponential growth of embedded systems, the advent of the internet and its enormous development, has allowed the growth of what is known as the Internet of Things. It is a new era where not only people are connected to the network through PCs, smartphones and tablets but also through any kind of object.
A question that could arise spontaneously is why this huge diffusion of embedded systems compared to that general purpose systems. The answer is easy: the main benefits of embedded systems is the low power consumption, the very small size and low cost per-unit, due to the possibility to reduce the complexity of the circuits that have to perform few dedicated functions and the possibility to integrated it in every object that surround us daily.

An embedded system is a system where different areas work together; it integrates hardware circuitry with software programming techniques. The software used in embedded system is a set of instructions that perform specific tasks. A general scheme of an embedded system is represented in figure 2.2.
The core of an embedded system is the electronic hardware that is on the Printed Circuit Board. Broadly, the typical structure of an embedded system consists of a device that interacts with the external world through Input/Output interfaces that can be A/D Converters(which convert an analog physical signal sent, for instance, by a sensor to a digital signal) or UARTs interface or infrared ports. All the information is processed in a Central Processing Unit and is stored in a memory.

2.1.2 Embedded Hardware

Hardware is fundamental, it is the physical core that contains different particular components, depending on the requirement and specification. Some of them are:

- General Purpose Microprocessors: They are single chip devices that contain an Arithmetic & Logic Unit, a Program Counter, a Stack Pointer(SP), registers, interrupts circuits, different clocks all integrated on a single chip. It is necessary to add a memory(ROM and RAM), memory decoder, oscillators, serial and parallel ports externally. It is used to perform a huge amount of computations and to provide many applications and tasks that required a great complexity compared to a microcontroller. Summing up it has a high cost, a high power consumption, a large memory size, flash and cache, external bus interface with a memory management unit to handle a huge amount of read/write operations and a greater memory usage.

- Microcontrollers: It is a computer system-on-chip; so it contains an integrated processor, a memory, a small amount of RAM, peripheral devices(timers, DAC, DAC and serial communication devices), all on one chip. Preferably used in small applications with precise calculation. So it is a very compact and low power chip. Of course, it has physical limits like a limited amount of ram , less flexibility due to its not expandability, less reliable and low performance. Nevertheless it is low cost, low power and very small size. A microcontroller also provides a set of pins that allow the use of sensors, actuators and transfer of data to other devices.

- DSPs/ASIP: An Application Specific Instruction set Processor (ASIP) is used for specific applications like digital-signal processing, telecommunications and embedded control. The advantage of the ASIP in an embedded system is the flexibility notwithstanding good performance, power and size. A Digital-Signal Processor(DSP) is a class of ASIP; it is a single chip designed to have very high performance, numerically intensive tasks (like multiply, add, shift).

- ASICs: Application Specific Integrated Circuits are designed for a specific application for example Digital to Audio Converter or Mpeg2 Decoder. They have a very high performance but a very high cost and they are not flexible.

- FPGA/CPLDs: Field Programmable Gate Array is a fully programmable customized chip. The big advantage is the cost and reliability. It is a bidimensional array of logic blocks and flip-flops that allow the user to configure different interconnections between blocks. It has the possibility to design a processor, a ROM, a RAM, a DSP and any kind of block onto a single chip.
  There are different types of FPGA according to its physical structure:

  – Reprogrammable FPGA(SRAM Based) that can be reprogrammed endlessly when need. It is very flexible but has a higher cost.
  – One time programmable FPGA based on antifuse: it can be reprogrammed only once. Usually used for particular applications like aerospace, satellite and very high security app.

  The Complex Programmable Logic Devices differs from FPGA mainly due to its architecture. It consists of more programmable sum of product logic arrays with small numbers of clocked registers. So they are less flexible but have the advantage

of predictable timing delays and have higher logic interconnection ratio. Differently The FPGA architecture is dominated by different interconnections. A CPLD contains an embedded flash which stores the configuration, whereas an FPGA has to be configured each time it is switched on.

- System on Chip (SoC): It contains a CPU, Peripheral devices, Power Management Circuits, Communicaion interfaces(UART, SPI, I^2C) on a single Integrated Circuit. It can include different programmable processors. For exampe a SoC can contain an ARM Cortex+ Custom GPU + DSP + FPGA.
- Input Devices: They take input from the outside world and route the signal into the different blocks. Inputs can be different types of sensors, switches, etc.
- Output Devices: They are the result of the different operations that occur in the microcontroller. Different examples of outputs can be LED, LCD, Actuators, Motors,Relays, etc.
- Bus Controllers: They handle all the communications, and the transfers among the different components inside the embedded system. Some examples of bus controllers are Serial Buses (SPI, I_2C,etc) RS232, RS485 and Universal Serial Buses.
- Memories: They are used to store data. In an embedded system a Non-Volatile RAM, Volatile RAM, DRAM,etc are usually present.

It is possible to compare the structure in figure 2.2 with the well-known Desktop Computer. In an embedded system the primary memory, central processing unit, and all the peripheral components are built on a single chip that is called Microcontrollers. On the other hand a desktop computer has to handle larger data dimension compared to an embedded system. Personal computer has to elaborate huge amounts of data and transfer it faster between CPU and memory, CPU and Input/Output devices. To store such a so huge amount of information, secondary memories like Hard Disks or CD-Rom are needed moreover it implements different methods like direct memory access or multi-level cache that are not necessary in embedded system.

2.1.3 Embedded Software

Software is essential for any type of embedded systems. It provides all the functionality to the system. Due to the different nature of the tasks that an embedded system has to perform, different languages are used; for example one language can be used to obtain a good and precise control-dominated application but isn't the best choice for signal processing applications or for network interfaces applications. Four types of different languages are employed:

- Software languages: use instructions to describe the sequences to be executed. There are different types of software languages that depend on the abstract level: assembly and high level. An assembly language uses a set of instruction written in symbolic form and perform very simple operations on registers and memories. High level

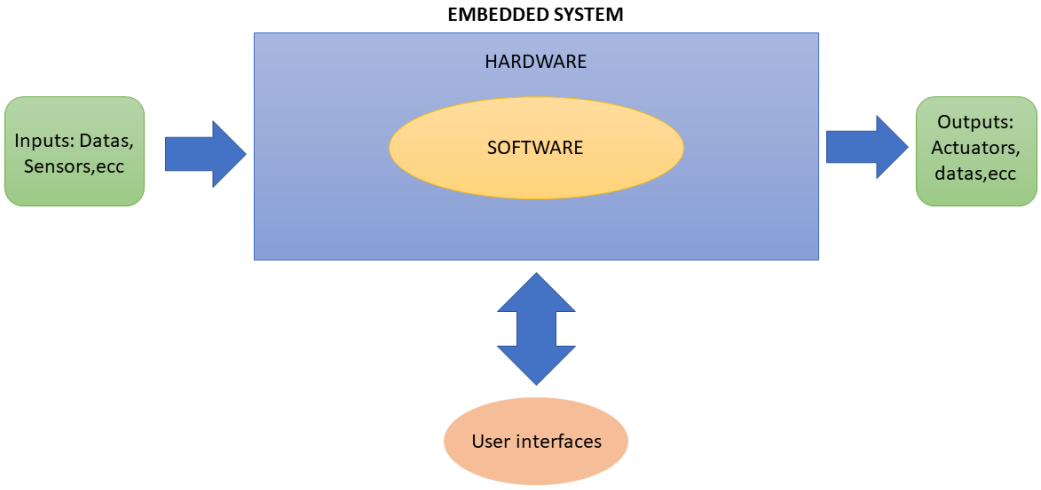

Figure 2.2: General scheme of an Embedded System

languages like C, C++, ADA, etc are used to construct more complex functions, loops, arrays etc.

- Hardware languages: Verilog and VHDL(Very high speed integrated circuits Hardware Description Language) are the most popular hardware languages. They are used to describe the system with discrete event semantics and structural hierarchy. They are also used to simulate digital integrated circuits.
- Dataflow languages: They are used to describe systems with processes that run concurrently and communicate through queues. They are composed by nodes, arcs and data. Typically they are used for signal processing applications.
- Hybrid languages: this language combines different type of other languages like Esterel that combines hardware semantics with software language.

2.1.4 Types of Embedded Systems

Embedded systems can also be classified into different types based on performance of the microcontroller, functional requirements and performance. They are divided into four different functional categories:

- Stand alone embedded systems: This type of systems does not require an host like a PC but they work by themselves. They take the input from analog or digital sources, they process the information, compute calculation and convert the data; in

the end they give the results (outputs) that are used to control motors, switches or show information onto the display. Examples of stand alone embedded systems are: digital cameras, microwave ovens, dish washes, videogame consoles, etc.

- Real time embedded systems: Real-time systems are systems that monitor and control an external environment. Sensors, input interfaces and actuators are used to connect the environment to the systems. So they have the ability to react when an event happens; for example vehicle systems for cars, aircrafts, radio communications, are examples of real time embedded systems.
- Networked embedded systems: These systems handle different networks to access the resources. They can use LAN or internet to connect. The connection of course can be wired or wireless. An example of networked embedded systems can be a home security systems that is connected with protocol TCP/IP or any kind of system that is connected to a web server and can be controlled by a web browser. The designed Encrypting IP Box system is a networked embedded system.
- Mobile embedded systems: they are used in smartphones, digital cameras, etc.

2.2 System Architecture of the designed embedded system

In figure 2.3 a general block diagram of the designed system is represented. The hardware embedded is composed by:

- A microcontroller(STM Family): It has the main role to download the bitfile of FPGA from the USB Cik-Token to the Flash Memory and every time the system is switched on, it sends the bitfile to the FPGA. The microcontroller has also some Status LED. The user has the possibility to erase the entire internal microcontroller flash memory and the external memory with a button. It has internal buses to communicate with the microprocessor and the FPGA. It has also the task to switch on and off the microprocessor and the FPGA.

- A microprocessor(IMX Family): It handles all the network communications with the outside world. The packets that arrive through the Ethernet are processed inside the microprocessor and are sent to the FPGA. It has two Ethernet interfaces that manage the network traffic; a USB OTG to connect the IMX to an external Computer and some Status LEDs. The microprocessor is programmed through an internal connector.

- An FPGA: It has the task to process the internet packets, if they are not encoded they are encrypted by the FPGA and viceversa. The FPGA has to be programmed each time the system is switched on because the chosen FPGA is SRAM based.

- A flash Memory (QSPI): The Bitfile and other informations (like logs) are saved inside the flash memory.

- Inputs/Outputs devices: The main Inputs are the Cik-Token USB and the two SFP Ethernet modules that have the role of both Input and Output. The different output LEDs are used as status indicators that are used to inform the user about the status of the machine. The system has a reset bottom to reset all the machine.

All the most important components used are analysed and a brief explanation of the features are given focusing mainly on the most important characteristics that are used to achieve the technical project specifications.

2.2.1 Microcontroller STM32F756 IGK6

A microcontroller is a self-contained system. The big advantage of which is the flexibility of use. In fact it is possible to change its work very easily only by programming it again. It contains peripherals, integrated memories, registers and a processor. In the designed device the microcontroller is the main component that powers on all the other devices and manages all the other components. When the device is powered, only the mictrocontroller is activated and it handles all the principal tasks that involve the other parts.
The used microcontroller STM32F756 is based on ARM 32-bit Cortex -M7 RISC core with a maximum frequency of 216 MHz. An ARM CPU is a RISC (Reduced Instruction

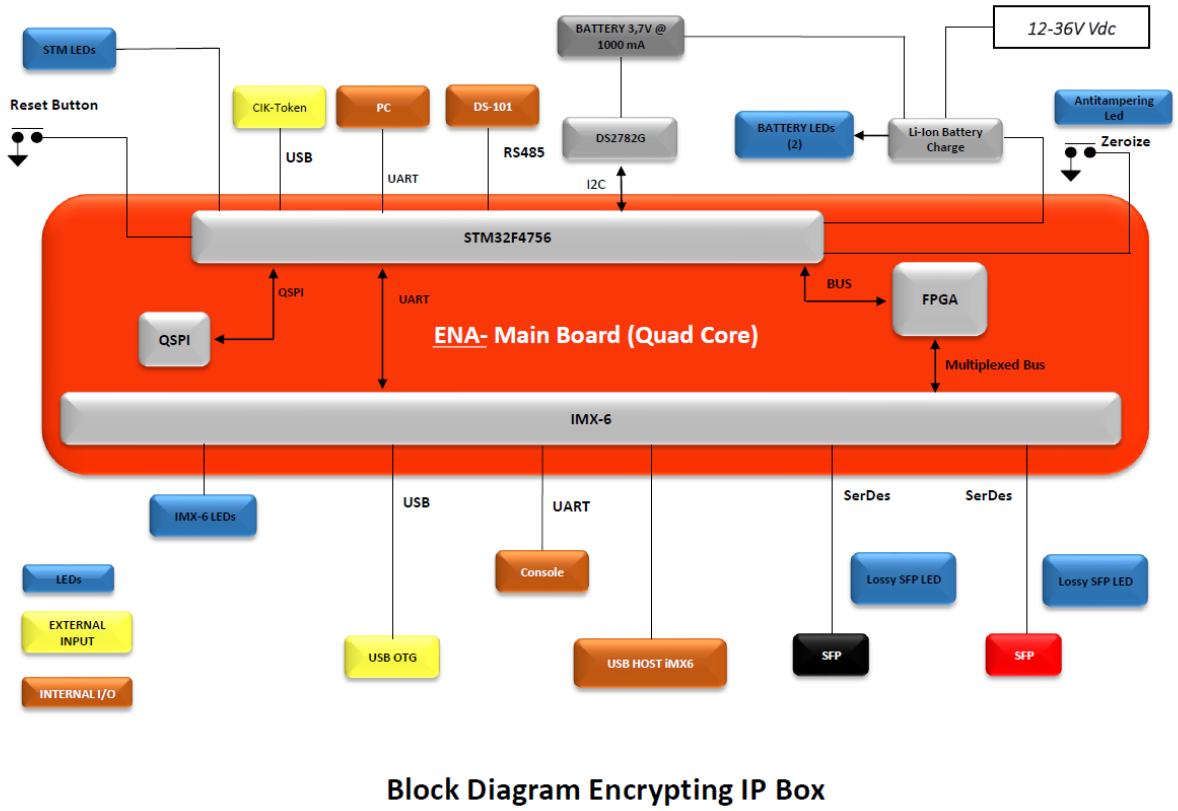

Figure 2.3: System Block Diagram

Set Computing) machine and this is the first big difference from the Intel CPU (CISC); so the instructions in RISC CPU are smaller and allow to achieve a linear and simpler architecture compared to the CISC machine. RISC architectures are defined "load-store" because they allow to access the memory with simple specific instructions that are used to read and write the data in the registers of the microprocessor.
For its intrinsic simpler architecture, an ARM CPU has a low power consumption and a better heat dissipation that are the main features required in portable devices like smartphones, tablets and embedded systems.
The Cortex-M family are ARM microprocessor cores that are designed for use as dedicated microcontroller chip. The Cortex-M7 core has a single floating point unit (SFPU) which supports all the data-processing instructions; it also implements a full set of DSP instructions. It features a six-stage pipeline with branch speculation that tries to guess which way a branch will go before this is known; it improves the flow in the instruction pipeline to achieve high effective performance in the microprocessor.
The STM32F756 embeds up to 1MB of Flash memory, 320kB of SRAM and has a

FMC(Flexible External Memory Controller) with up to 32-bit data bus. The FMC includes three memory controllers: NOR/PSRAM, NAND and Synchronous DRAM. In the project the SRAM memory controller is used to program the integrated FPGA in the board. It embeds, also, a Quad SPI memory interface that is used to interface the microcontroller with a QSPI flash memory that stores the bitfile of FPGA. In low-power, three different modes can be used: sleep, stop and Standby modes. It offers also a true random number hardware generator(RNG) and a cryptographic acceleration cell for AES 128, 192, 256, HASH(SHA-1, SHA-2) and HMAC.
In the following subsections the main features that are used in the design of the device are analized.

Embedded Flash memory(FLASH)

The Flash Memory interface manages Cortex-M7 and TCM accesses to the Flash memory. The TCM(Tightly-coupled memory) has the purpose to provide low-latency memory respect the unpredictability of the cache. In fact the Tightly coupled memory has deterministic access time. The accesses through the cache are not deterministic since the data can be in the cache(hit) or the data must be fetched from the main memory(miss). So the TCM provides a more efficient memory accesses. The Flash memory interface implements the erase and program Flash memory operations and it has the capacity up to 1Mbyte.In figure 2.4 the flash memory interface connection inside the microcontroller is shown. The embedded flash has three interfaces:

- 64-bits ITCM interface that is connected to the ITCM bus of Cortex and it is used for instruction and data read access; the write accesses is not supported on ITCM;
- 64-bits AHB interface that is connected th the AXI bus of Cortex through the AHB matric and is used for code execution, read and write accesses. The AHB interface supports the DMA data transfer;
- 32-bits of AHB register that is used for control and status register accesses.

It is possible to see that after the flash interface, the flash bus is 256 bits.
The flash memory has a main block that is divided into 4 sectors of 32 Kbytes, 1 sector of 128 Kbytes and 3 sectors of 256 Kbytes. In table(2.1) all the information about the memory that is required to write correctly the program code and to place the code in the right part of the memory is summarized.

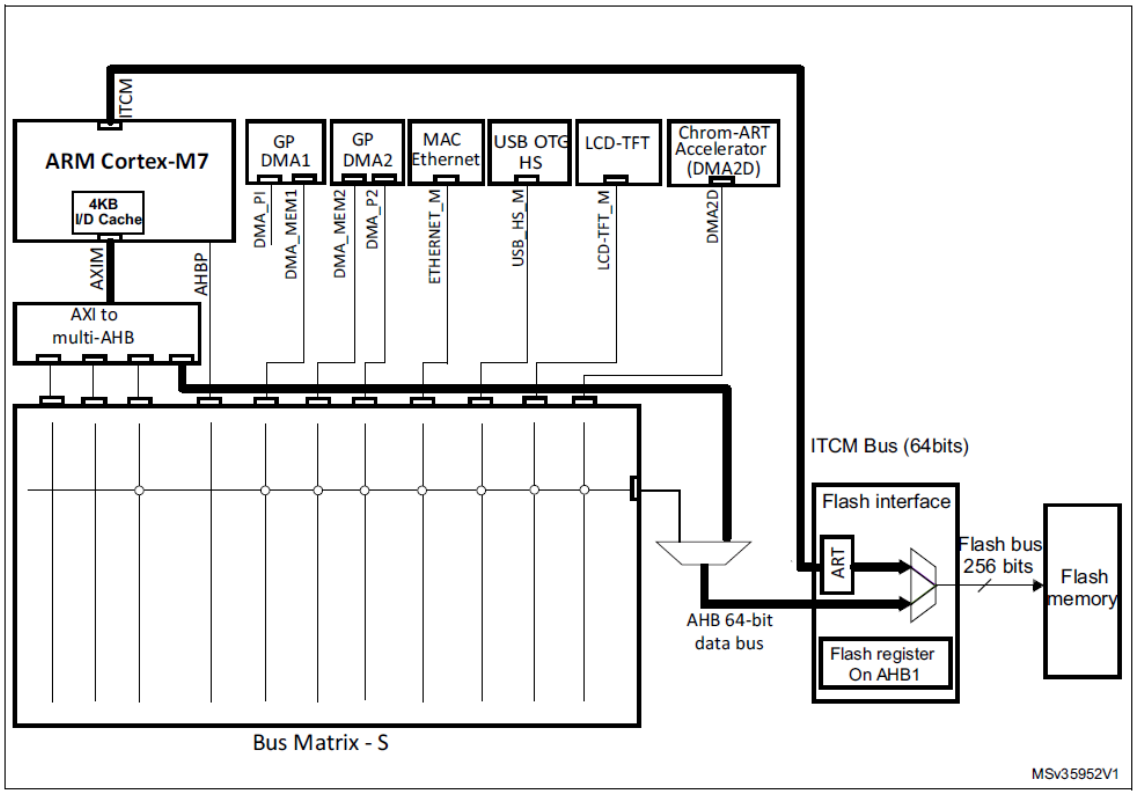

Figure 2.4: Flash memory interface connection inside microcontroller [21]

Table 2.1: Embedded Flash memory organization

Name	Block base address	Sector size
Sector 0	0x0800 0000 - 0x0800 7FFF	32 KB
Sector 1	0x0800 8000 - 0x0800 FFFF	32 KB
Sector 2	0x0801 0000 - 0x0801 7FFF	32 KB
Sector 3	0x0801 8000 - 0x0801 FFFF	32 KB
Sector 4	0x0802 0000 - 0x0803 FFFF	128 KB
Sector 5	0x0804 0000 - 0x0807 FFFF	256 KB
Sector 6	0x0808 0000 - 0x080B FFFF	256 KB
Sector 7	0x080C 0000 - 0x080F FFFF	256 KB

Quad SPI Interface

The STM microcontroller has a dedicated Quad SPI interface that is specialized to communicate with Quad SPI Flash memories. The Quad SPI Flash Memories are Serial Memories that use the SPI(Serial Peripheral Interface) with four wire for data. The main benefit of using QSPI is higher speed. Compared to the traditional SPI Flash interface that has four

lines (CLK, CS, MISO, MOSI), the QSPI uses 6 lines. From figure 2.5 the QSPI has:

- CLK - the clock output;
- DQ[3:0] - the serial I/O bidirectional signals that are used to transfer address, data and command information;
- nCS - the chip select output;

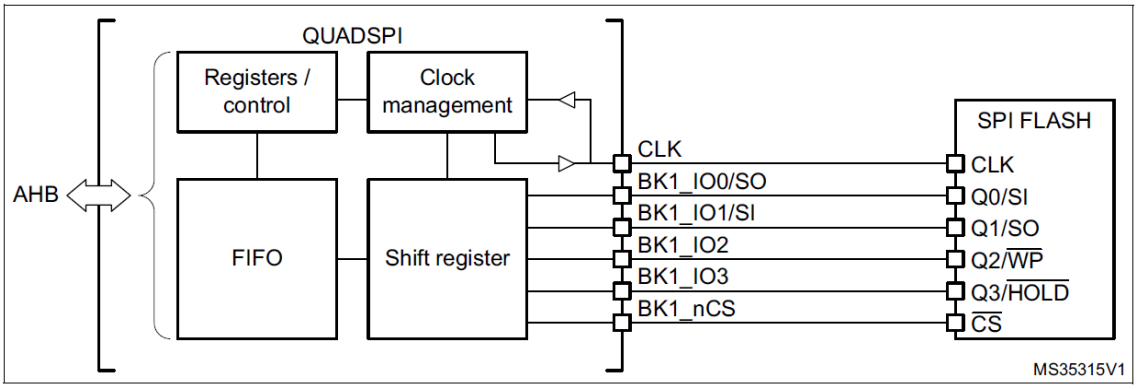

Figure 2.5: The scheme of the QuadSPI interface in STM microcontroller [21]

The QSPI communicates with the Flash memory sending commands. In figure 2.6 the format of a complete command is represented. Each command is formed at least by one instruction phase, the address phase, the alternate phase and the data phase.

- Instruction phase: an 8-bits of instruction is sent once every clock cycle on DQ0 line that specifying the type of operation;
- Address phase : 4 bytes are sent to the Flash memory to select the address. The bits are sent 4 bits every clock cycle using the 4 serial line DQ[0:3];
- Alternate phase: 1 to 4 bytes are sent to control the mode of operation.
- Data phase: during this phase any number of bytes can be sent or received from the Flash memory. The data is sent 4 bit every clock cycle using the four I/O pins.

Instruction phase: 8 bits	Address phase : 4 Bytes	Alternate phase: 1-4 bytes	Data phase: 64 bytes

Figure 2.6: Complete command that is send to the Flash Memory

Clock-out Capability

The microcontroller has the possibility to use one Pin(PC9) as special clock output. It is possible to configure it with the prescaler the frequency clock of the pin. This pin is used as clock for the FPGA.

Flexible memory controller (FMC)

The STM microcontroller has to write the bitfile, stored in the QSPI, in the FPGA. Before the read/write operations it is necessary to configure the memory controllers. The FMC consists of four main blocks as shown in figure 2.7:

- AHB interface
- NOR Flash/PSRAM/SRAM controller
- SDRAM controller
- NAND controller

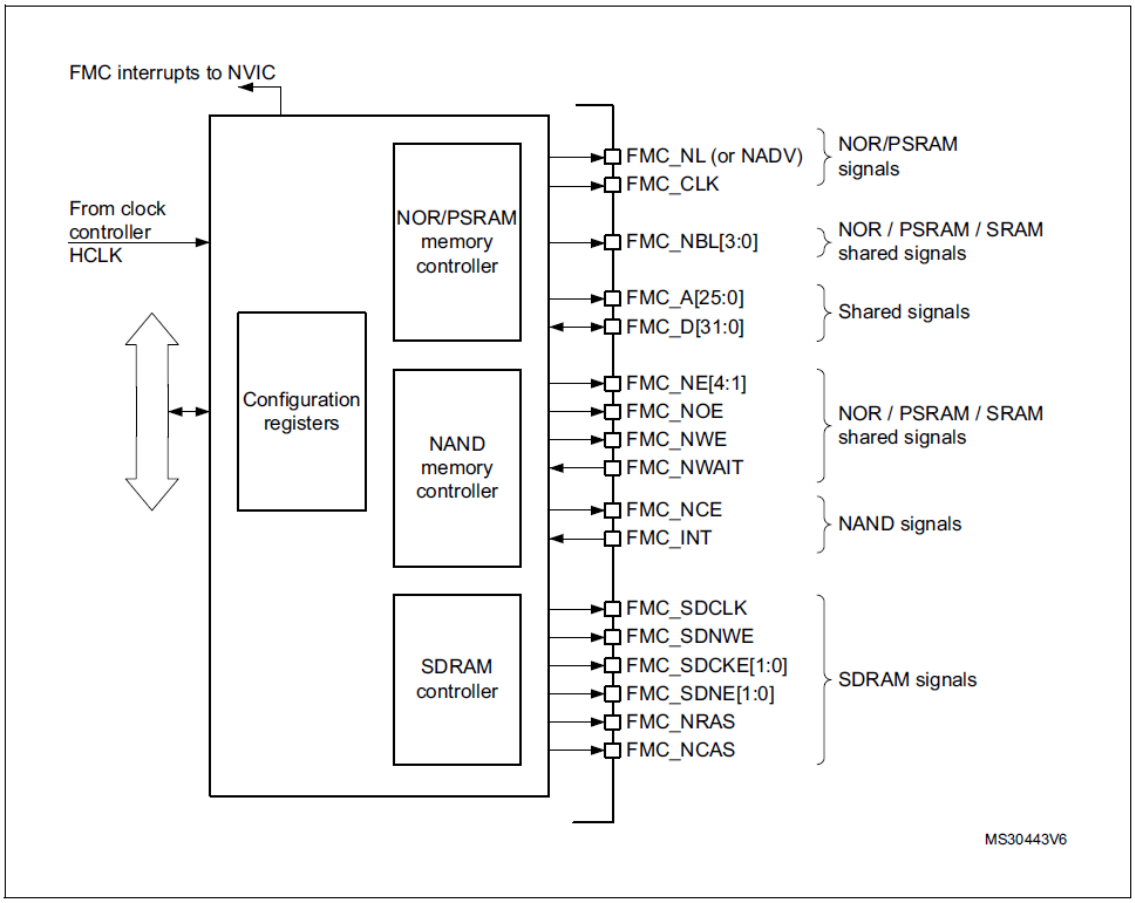

Figure 2.7: FMC blocks diagram [21]

The FPGA contains an SRAM cell where the bitfile have to be stored so it is necessary to configure one of the four banks of the FMC. It is chosen to use bank 3 to perform the operations of configuration and bank 0 to perform check write/read operations between

FPGA and microcontroller; this division is chosen to have relaxed timing for configuration(done only once) and better performance in read/write register operations. The FMC generates the appropriate signal timings to drive the different types of memories and in the case of SRAM a non-multiplexed mode and normal asynchronous mode is selected. In this mode, the following parameters that depend on the memory datasheet have to be computed and set:

- ADDSET: address setup time
- DATAST: data setup time
- ACCMOD: access mode

These parameters give the FMC the flexibility to access static memories. There are four extended access modes(A,B,C and D) that allow the possibility to change the timing. With the ADDSET and DATAST it is possible to modify the timing to read and write operations. The two variables are multiple of the clock core of the micro. In figures 2.8 and 2.9 it is possible to see the timing for read and write operation and how the two parameters modify the timing.

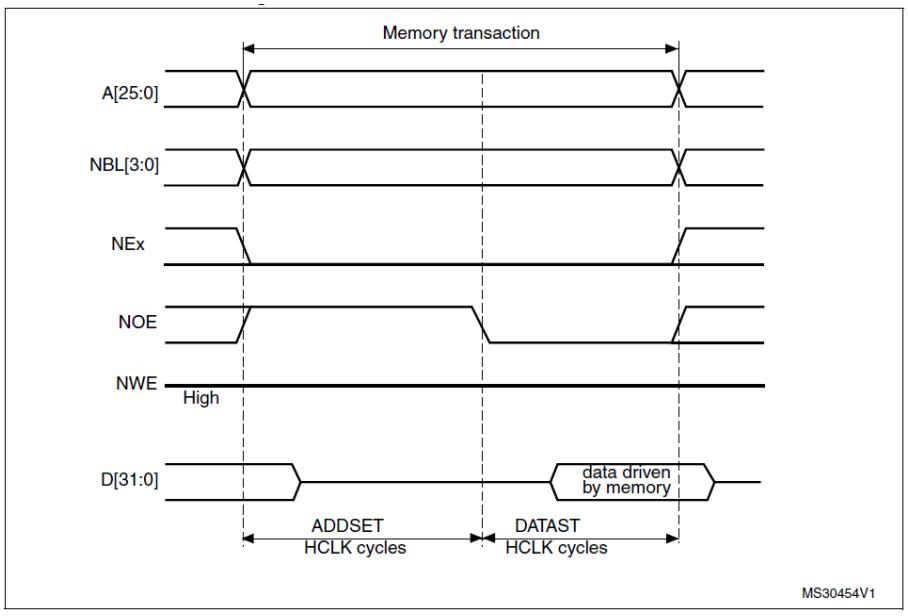

Figure 2.8: SRAM asynchronous read access timing in extended mode [21]

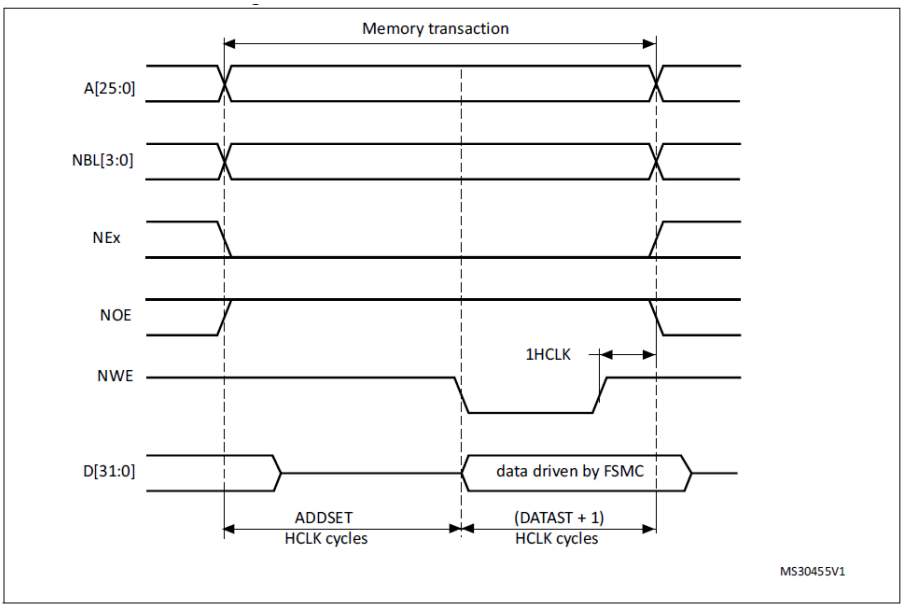

Figure 2.9: SRAM asynchronous write access timing in extended mode [21]

Universal asynchronous receiver transmitter (UART)

An embedded system often communicates with the external world. It could be to send and receive commands, or for debugging purposes or to transfer data to another device. One of the most used interfaces is the UART. It is used for different purposes, one of them is to get the debug console functional and to receive information after a command is sent. It supports synchronous and asynchronous half-duplex and full-duplex communications; it's also possible to use DMA(direct memory access) for multibuffer cofiguration. A bidirectional communication requires two pins minimum: Receive data (RX) and Transmit data (TX).
The serial data that are transmitted and received through the pins are composed by:

- An Idle Line prior for the priority;
- A start bit;
- A data word(7 or 8 or 9 bits) with the least significant bit first;
- 1 or 2 stop bits;
- A status register;
- Data registers(receive and transmit)
- A baud rate register

2.2.2 QSPI - NOR Flash Memory

The MT25Q is a multiple input/output serial Nor Flash memory device manifactured in 45nm. It has a security protection where each sector can be locked independently. To access the full memory storage, the device includes an extended address register; the base address is 3-Byte address and can only access 128MB of memory; with the extended address register (3 bits [2:0]) it is possible to select one of the eight 128MB segments of the memory. In the device, this mode is enabled to store all the bitfile of FPGA and to have enough space for the audit file.

2.2.3 IMX6

The iMX 6Quad is a 32-bit processor that feature implementation of the quad ARM Cortex -A9 core, which operates at speed up to 800MHz. The Cortex-A family compared to the Cortex-M family (used in STM microcontroller) is the only that includes a memory management unit(MMU); the MMU is harware component that handles all memory operations associated with the processor. So in other words it is responsable for all the memory management and its main feature is to perform the translation of virtual memory addresses to physical adresses. It includes a General Interrupt Controller(GIC), 32kB of L1 Instruction cache, 32kB of L1 Data cache and 1MB L2 cache, shared by four cores. The IMX6 is the core of the device; it handles the communications through the Ethernet. The received data are sent to the FPGA with a 16 bit protocol communication.

External Interface Module(EIM)

The External Interface Module manages the interface to the external devices. It includes the generation of the chip select, clock, and the control of all the external signals. It is possible to use asynchronnous communication to access device in SRAM-like mode and synchronous access. The interface with the FPGA is a multiplexed Address/Data mode , so address and data bits are sent on the same pin. This reduces the number of wires but the speed is lower. The pins used are EIM_DA[15:0]. The writing operation is performed in syhchronous access; when the address is on the line, the Chip Select enable, write enable and LBA are pulled down, so at the next rising edge of the clock the address is sampled, then the LBA is pulled up and the Data are written. For the reading operation, instead, an asynchronous access is performed. The Chip select is pulled down, then the LBA is pulled down when the Address is ready on the line and when the Output enable has a transition from high to low, the data can be read. After one data is read, the chip select is released. In figures 2.10 and 2.11the timing of the operations in details are present.

MAC-NET

The MAC-NET core handles the process of the different networking protocols, such as IP protocol(Appendix B) and TCP. It also implements a hardware acceleration block to optimize the performace of network controllers. The IMX6 supports different speed configurations (10/100-Mbit/s) and gigabit full-duplex operations. The Ethernet interface

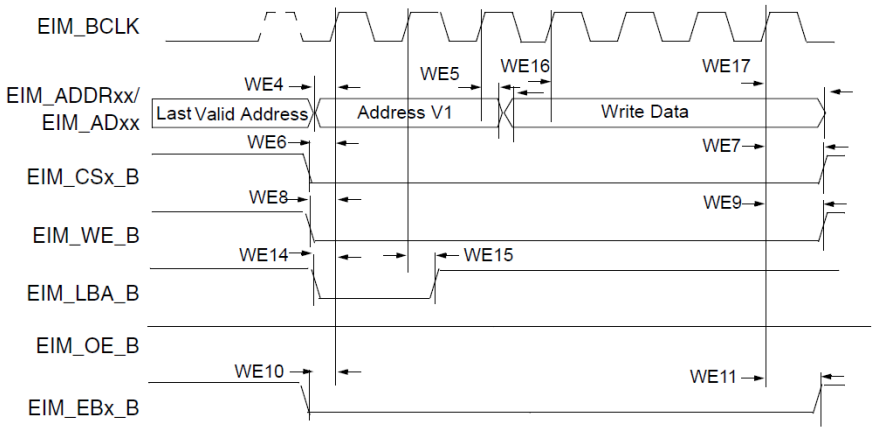

Figure 2.10: Synchronous Muxed A/D write access [20]

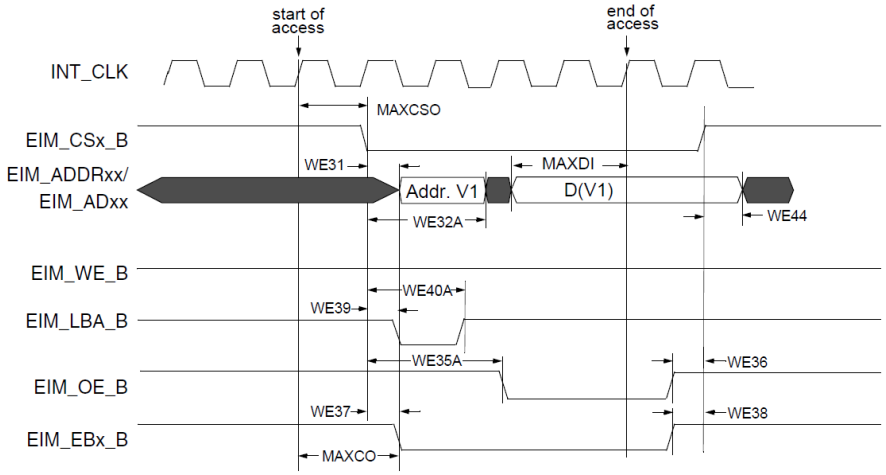

Figure 2.11: Asynchronous Muxed A/D read access [20]

used is a SFP transceiver. The small form-factor pluggable is a compact transceiver that is used both for telecommunication and data communications applications. It interfaces a network device to a fiber optic. It is also compatible with different communication starndards and has a transmission rate ranging from 100Mbps up to 10 Gbit/s. All the communications needed a standard model to transmit informations, so different standards were born.

One of the most famous technologies to connect computers, routers, printers all over the world is Ethernet. In Ethernet, devices wait for a free time slot to communicate in the network and the waiting device transmits when there is no transmitting data. The medium though the data travel, can be a coaxial cable, a twisted pair cable, or a fiber optics(like in

this device). Ethernet technology defines technical specifications at the physical lever and at the MAC level of the ISO/OSI model network (Appendix A). The system divides the data into shorter pieces called frames. The basic structure of an Ethernet packet(frame) is recived by the datalink layer. The main elements of a frame are:

- Preamble: The starter is a sequence of 7 bytes of alternating 1 and 0. This sequence of bits is used to "wake up" the receiver, to synchronize the clocks of the transmitter and receiver.
- Start Frame Delimiter(SFD): The eight byte indicates the beginning of the ethernet frame.It is immediately followed by the MAC address.
- Destination MAC address: It is an uniquely number that contains the LAN address of the destination and it is composed by 6 bytes.
- Source MAC address: It is the address of the source.
- EtherType: It is composed by 2 bytes and indicates the type of protocols or the length of the data.
- Payload: It contains the real data that are transmitted. It has a minimum lentgh of 48 bytes and if the minimum length is not achieved a padding is added. If the length is above the maximum value, the data are split into different packets.
- Frame Check Sequence(FCS): It is a CRC(Cyclic redundancy check) that allows to check the presence of transmission errors. The receiver computes the CRC with an algorithm and compares it with the CRC that is received.

2.2.4 Cyclone IV

A Field Programmable Gate Array (FPGA) is an integrated circuit that has the possibility to be configured by a designer after manufacturing. In fact, it contains an array of programmable logic blocks. It is possible to realize complex logic functions with a very high scalability.It also includes memory elements, that can be simple flip-flop or more complex block memory like SRAM. Altera Cyclone IV is a low power FPGA. It features 6K to 150K logic elements, up to 6.3Mb of embedded memory, up to 360 18x18 multipliers for DSP processing applicatons and data rates up to 3.125 Gbps. It includes up to 30 global clock (GCLK) networks and up to eight PLLs with five outputs. It is possible to dinamically reconfigure the PLLs. It also supports SDR, DDR, DDR2, SDRAM and QDRII SRAM interfaces and supports the use of error correction coding bits on DDR and DDR2 SDRAM interfaces.
The Cyclone IV uses SRAM cells to store the configuration data. Due to the volatile type of memory, the configuration data must be download in the FPGA each time the device powers up. There are different way to configure the device: AS, Ap, FPP and JTAG configuration schemes. In this device a fast passive parallel(FPP) configuration is performed using the STM32F756 microcontroller. The configuration data that is stored in the QSPI is transferred to the FPGA on the *DATA[7..0]* pins. The configuration data is transferred

one byte per clock cycle. After the configuration , the registers and I/O pins must be initialized, then the device enters in user mode. To perform the configuration a Cycle State Machine is to be designed following the cycle waveform shown in the data sheet. A fast passive parallel(FPP) configuration is performed; in this way 8 bits are latched into the FPGA on every rising edge clock. In figure 2.12 it is shown the configuration cycle waveform. After the device is configured, its registers, I/O pins have to be initialized. The configuration phase consists of 3 stages: reset, configuration and initialization. When nCONFIG is low, the device is in reset mode, with the transition low-to-high of nCONFIG, the FPGA starts the configuration. When the nCONFIG goes from 0 to 1, it releases the open-drain nSTATUS pin, so it is pulled high by the pull-up resistor and now it is possible to configure the FPGA. After the FPGA has recived all the configuration data, it release the CONF DONE pin and a transition low-to-high is performed. When the initialization is done and the FPGA is in user mode, the INIT DONE pin is pulled up. Now the FPGA is ready to perform its operations.

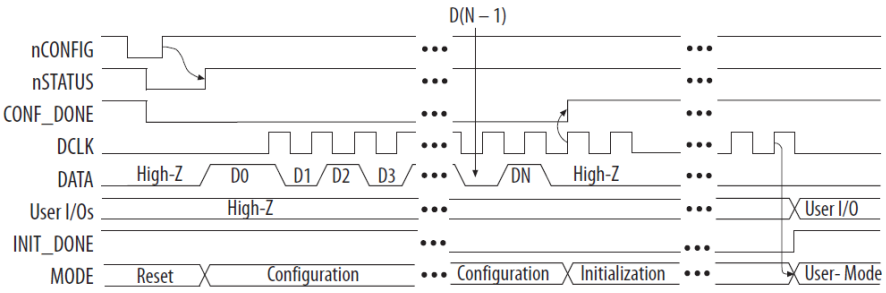

Figure 2.12: Configuration Cycle Waveform [1]

Chapter 3

System Architecture

In this chapter a detailed description of the system architecture and all the design steps is done. In the figure 2.3 a general block diagram of the designed system is represented. The system is composed mainly by two big subsystems:

- the carrier board that contains the components that provide power to the whole system and the interfaces to the external world like USB TOKEN, LEDs,reset button,Zeroize button, USB OTG and SFP connectors.
- the main module that contains all the active components that manage the informations that arrive from the external world like the microcontroller, the microprocessor, memories, buses,ecc.The main module is called ENA (Embedded Networked Appliance)

The ENA is put in contact with the carrier board through an Express Connector.

The used carrier board is a developer carrier board that was designed by Telsy with different modules that can be used for development of different applications. The main scope of this thesis is the design of the system and the evaluation and measurement of the performance of the ENA. At the end, if the performance are satisfactory a custom designed carrier board will be produced and in the section 3.1.1, the custom carrier board details are analysed. A flow chart of the general designed steps is made in figure 3.1 and in the following sections all the steps are described in details. It is possible to notice from the general block scheme (2.3) that the microcontroller has the rule to manage all the other components so when the ENA is switched on, the STM is turned on; all the other components are still off. For this reason, first, the STM software is developed to perform all the tasks needed for the correct download of the bitfile into the FPGA and to handle the correct reading of the USB Token. After that,if the FPGA works correctly, it is turned on the microprocessor IMX6 and the network transmission can be handled; a detailed description is in section 3.1.2. The last step is related to the tests and measurements to provide a good precise features and to establish performance; the details are discussed in the chapter 4.

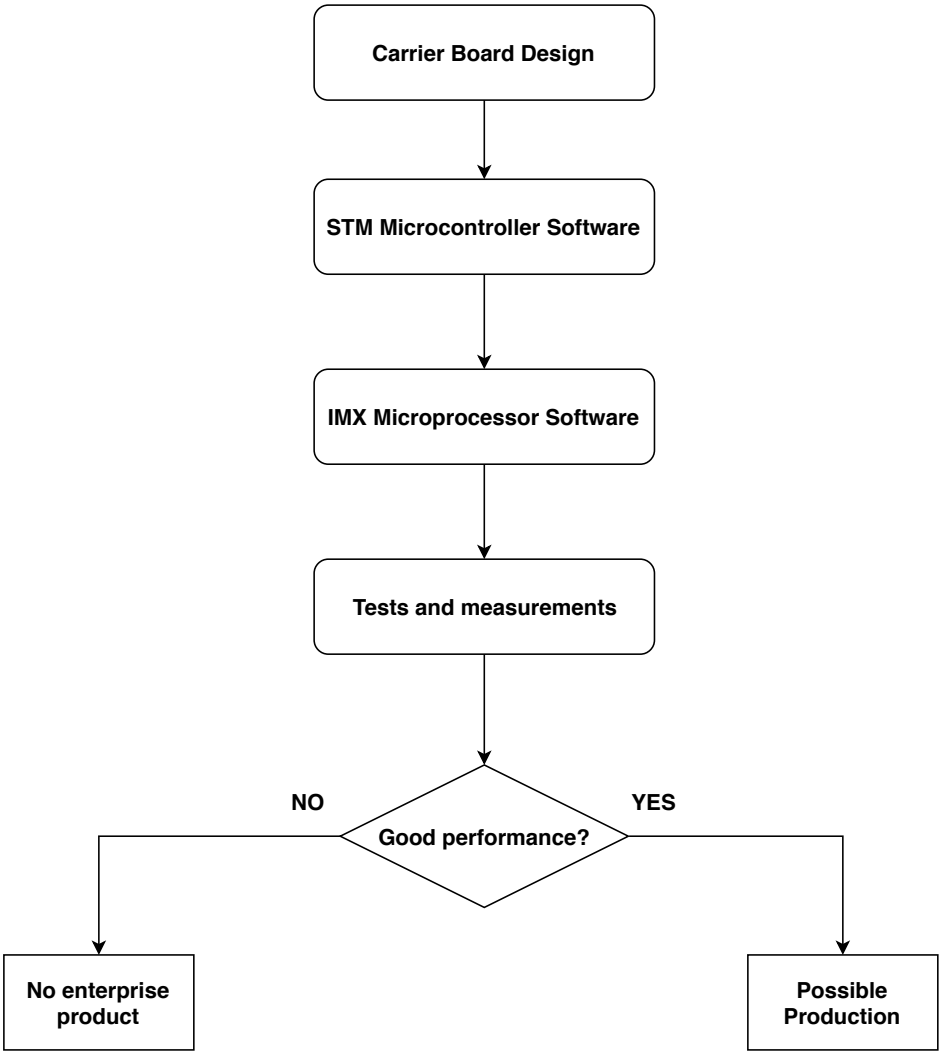

Figure 3.1: General Design Flow Chart

3.1 Carrier Board Design and Custom Module Structure

In this section it is analysed the custom designed carrier board with its schematic pages and the pins of the express connector. The ENA, instead, has already been realized for other applications and it is covered by copyright, so only a brief description of the hardware and pins is made.

3.1.1 Carrier Board Design

The main play role of the carrier board is to provide power supply to the different parts of the system. The main power source is the 230V electric power that is reduced to the range (12-36)V with an external power supply. The system includes also a Li-Ion Battery with a charger, so that, when there is a blackout, the device continues to work. The external inputs/outputs integrated into the carrier board are the following:

- Plug for 12-36V Vdc;
- 2 LEDs Battery;
- 2 SFP Optical Connector
- 2 LEDs for Lossy in SFP Connection
- Zeroize Button
- Red LED for Antitampering
- Reset Button
- USB Token
- OTG Micro-USB
- 3 LEDs for IMX6 information
- 2 LEDs for STM information

A list with the used components with the relative explanation is done in the following:

- Dual Input Li-Ion Battery Charger LTC4078. It is a linear charger that is able to charging a single-cell Li-Ion battery from wall adapter. It has a maximum 22V rating for wall adapter and the charging stops if the power source exceeds the over-voltage limit. To avoid high increasing of temperature the LTC4078 has an internal feedback regulator that maintains a constant die temperature also during high power operations. Two pins of the charger battery provide charge informations. They are connected to two leds. The pin CHRG is activated(pulled down) when the device is charging, and when the cycle is completed, the LED turns off. The PWR pin is pulled down, so the LED turns on when there is a valid input charging(i.e. when the input supply is greater than the undervoltage and less then the overvoltage) and it turns off when the wall adapter is removed.

- Stand-Alone Fuel Gauge IC. The DS2782G+ is an integrated circuit that can measure voltage, temperature, current and can estimate the capacity of the rechargable lithium battery. It gives also information of capacity estimation remaining and the percentage. The calculations are stored into an EEPROM chip. It is used to understand if there is a voltage onto the battery, the temperature, if the current passes and charges the battery correctly. It is possible to program it with an I^2C. Due to the fact that the voltage of I^2C from the STM is too low (1.8V) a voltage level translator is used;
- Voltage-Level Translator TCA9406 is a 2-bits bidirectional I^2C voltage-level translator. This allows the device to interface between higher logic signal levels(The DS2782G+ needs 3.3V) and lower logic levels(1.8V of STM);
- LT8614 is a Synchronous Step-Down Switcher regulator. The LT8614 is used to mantain a very stable voltage also at high frequency. It minimizes the EMI emissions and mantains a very stable voltage up to 2MHZ. It has also a very low quiescent current to have high efficiency with smal load current. The device is used to deliver 3.3V to the other components and to provide independent voltage 3.3V to the two SFP connector.
- LD1117. It is a Low drop-out voltage regulator needed to provide a fixed and stable voltage output 1.8V.
- A single inverter buffer/Driver with open-Drain output is used to connect the switch button to the microcontroller. A "standard" switch is normally open and when it is pushed, it closes, but the Zeroize button is normally closed and when it is pushed has to be opened. To do this, an inverter is used.
- Two SRV05 integreted circuits are used. They are a 10A diode array to protect the USBs connector against ESD and high surge events;
- Noise suppression filters that are applied to all the USB connectors to suppress noise for differential signal line without distortion in high speed transmission due to the high coupling.
- LEDs. The anode of the LEDs is connected to the main board through the COM-Express and the cathode is connected to the ground. So when a high logic level is asserted, the Led switches on.

In the following, it is included the schematic of the custom carrier board with some notes for the PCB Designer. Some components are not used in this project but a space for a possible welding of the chip is considered for future company applications. This components are marked into the schematics with NF.

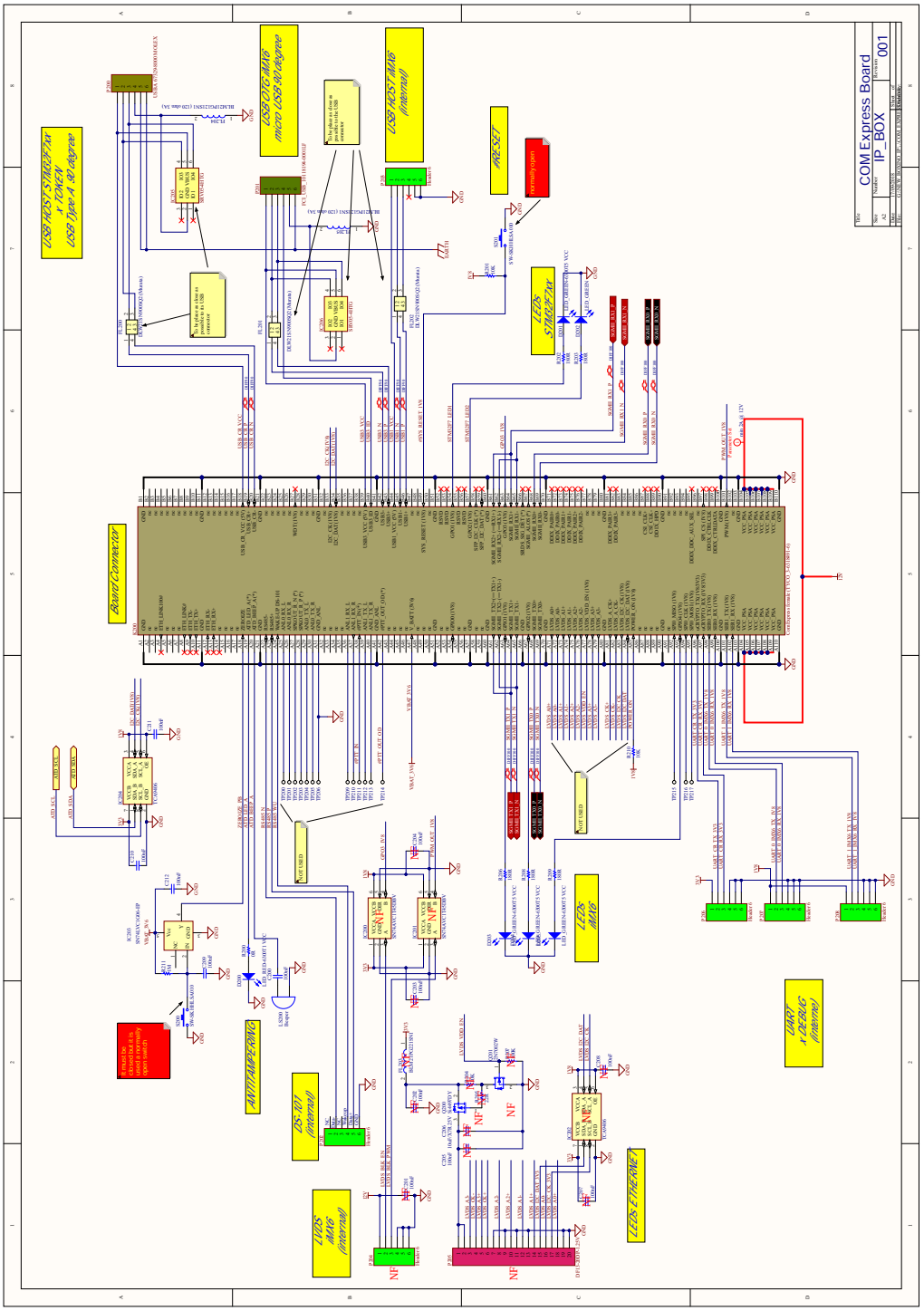

Figure 3.2: Carrier Board Schematic: Express Connector Page

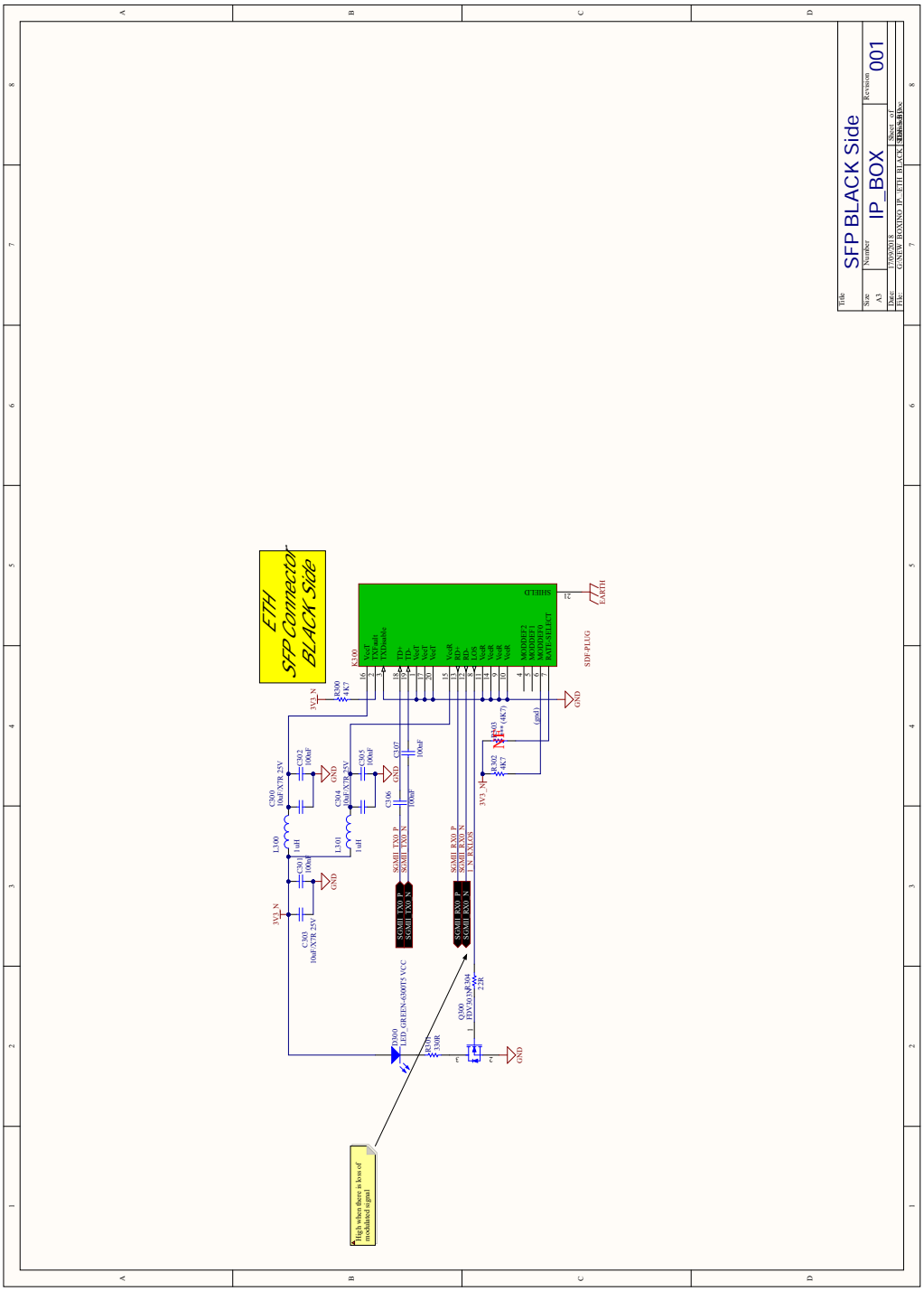

Figure 3.3: Carrier Board Schematic: SFP Black Optical Module Page

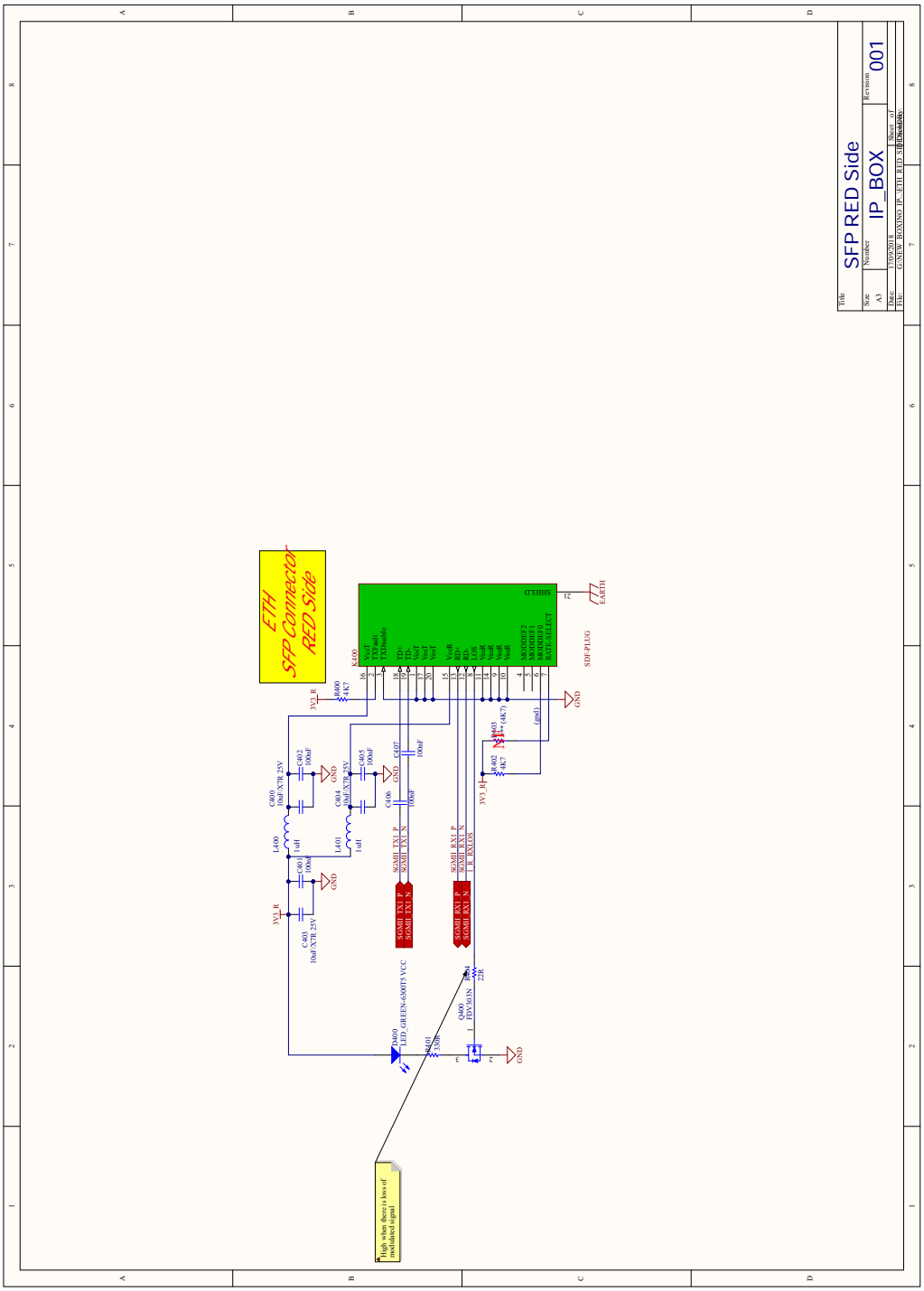

Figure 3.4: Carrier Board Schematic: SFP Red Optical Module Page

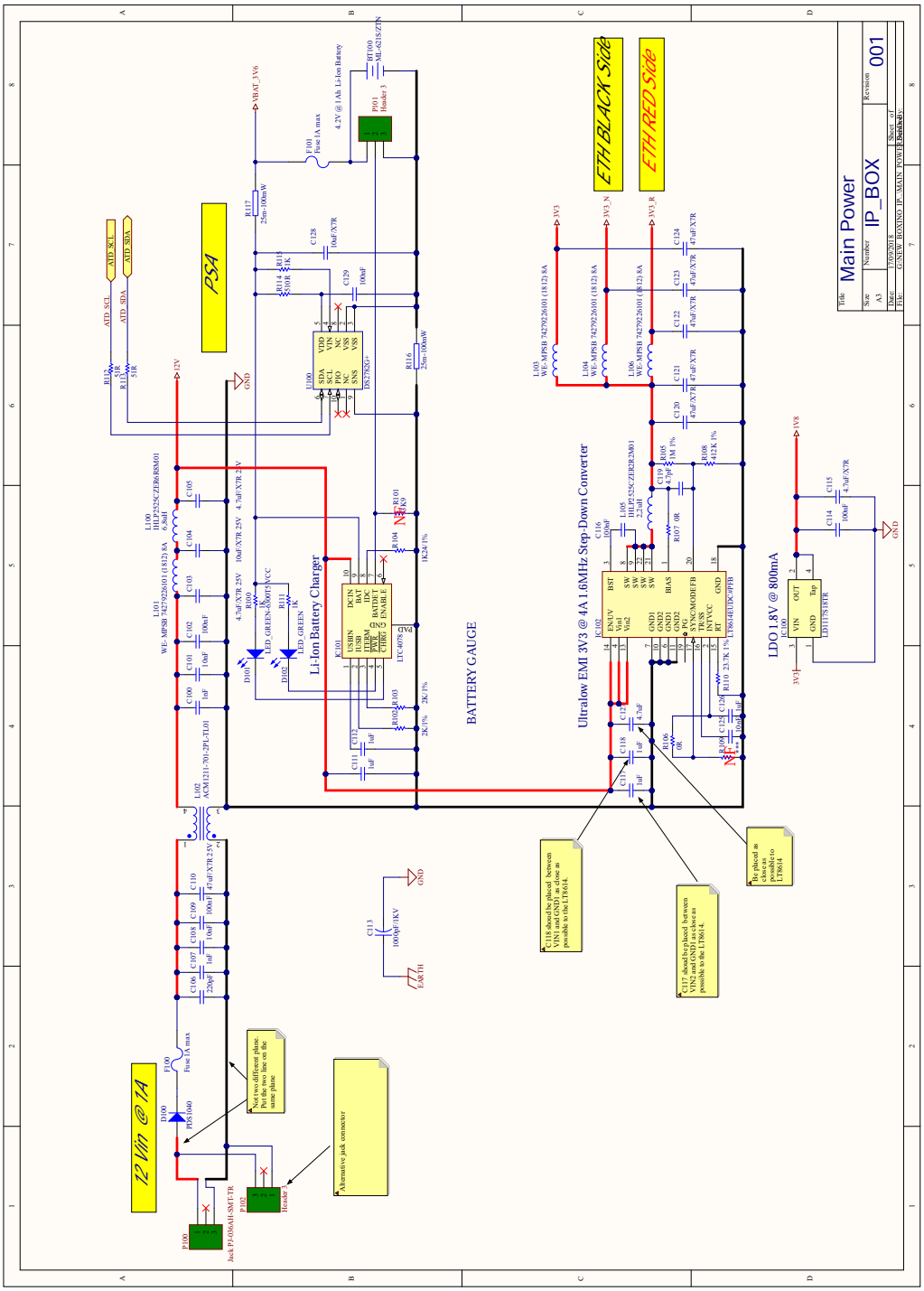

Figure 3.5: Carrier Board Schematic: Main Power Page

Express Connector

The motherboard integrated a COM-Express female type TYCO_3-63184916 with 220pins but not all the pins are used. A list with the used pins is done in the tables 3.1, 3.2, 3.3 and 3.4. The COM Express provides to the main board the 12V, the Voltage battery (3.6V) and the physical ground.

Table 3.1: COM Express A-pins I

PIN	Signal	Note
A1	GND	Ground
A11	GND	Ground
A18	ZEROIZE	It connects the Zeroize botton to the STM
A19	ATD_LED_A	Antitampering Led
A20	ATD_BEEP_A	Antitampering Beeper
A21	GND	Ground
A22	RS485-	DS-101 Data -
A23	RS485+	DS-101 Data+
A24	WAKE-UP DS-101	Wake up
A47	V_BATT(3V6)	Voltage from the Li-Battery
A51	GND	Ground
A57	GND	Ground
A60	GND	Ground
A61	SGMII_TX2+	SGMII Tx Line connects to the Red Switch SFP
A62	SGMII_TX2-	SGMII Tx Line connects to the Red Switch SFP

Table 3.2: COM Express A-pins II

PIN	Signal	Note
A63	GPIO(1V8)	It connects a LED to the IMX
A64	SGMII_TX1+	Connected to the A61
A65	SGMII_TX1-	Connected to the A62
A66	GND	Ground
A67	GPIO2(1V8)	It connects a LED to the IMX
A68	SGMII_TX0+	SGMII Tx Line connects to the Black Switch SFP
A69	SGMII_TX0-	SGMII Tx Line connects to the Black Switch SFP
A70	GND	Ground
A85	POWER_ON(1V8)	Connected to 1.8V
A93	GPIO4(1V8)	It connects a LED to the IMX
A98	SER0_TX(1V8)	UART Transmitter IMX
A99	SER0_RX(1V8)	UART Reciver IMX
A100	GND	Ground
A101	SER1_TX(1V8)	UART 1 Transmitter IMX
A102	SER1_RX(1V8)	UART 1 Reciver IMX
A103	GND	Ground
A104-A109	VCC	Power Input (+5V or +12V) default value=12V
A110	GND	Ground

3.1.2 The ENA

The Main custom module is the core of the device; it contains the principal components: the FPGA, the microcontroller, the microprocessor, the QSPI and all the used buses. In the figure 3.6 a Block Diagram of the main board is shown. For each main component a brief description of the pins and their functions is made for the explanation of the schematic.

STM32F756

In this subsection the pinout of the STM is done, focusing the attention on the main used pins. The pins are divided into different ports. In the tables 3.5,3.6 and 3.7,the pinouts and the pin descriptions are present. The STM is connected to the FPGA with a Bus of 16 bits; the FPGA stores the configuration data in SRAM cells, so the Bus is composed by a Data Bus of 16 bits where the data travel, a 10 bit Address bus where the address of the registers is passed to the FPGA during the read and write operations and the signal control pins (write enable, output enable, chip select). The STM has also the task to power on/off the FPGA and to manage the interrupt from FPGA with the Power On control Signal and Interrupt signal. It powers on/off the microprocessor IMX6 and sends the BOOT parameters. The STM uses an UART to communicate with the IMX6 receiving informations and sending command. Lastly, it shares a bus with the QSPI memory to read and write data. The JTAG Interface is used to program the STM.

Table 3.3: COM Express B-Pins I

PIN	Signal	Note
B1	GND	Ground
B11	GND	Ground
B18	USB_CR_VCC(5V)	USB Power +5V
B19	USB_CR+	USB Differential Line+
B20	USB_CR-	USB Differential Line-
B21	GND	Ground
B31	GND	Ground
B33	I2C_CK(1V8)	I2C IMX CLK for Battery Managment Chip
B34	I2C_DAT(1V8)	I2C IMX DAT for Battery Managment Chip
B39	USB3_VCC	Micro USB OTG IMX6 Power 5V
B40	USB3_ID	Micro USB OTG IMX6 ID
B41	GND	Ground
B42	USB3-	Micro USB OTG IMX6 -
B43	USB3+	Micro USB OTG IMX6+
B44	USB1_VCC	USB Host IMX6 Power (internal to the device)
B45	USB1-	USB Host IMX6 - (internal to the device)
B46	USB1+	USB Host IMX6 + (internal to the device)
B49	SYS_RESET(1V8)	Reset Button

Table 3.4: COM Express B-Pins II

PIN	Signal	Note
B51	GND	Ground
B54	GPIO1(1V8)	LED1 STM
B57	GPIO2(1V8)	LED2 STM
B60	GND	Ground
B61	SGMII_RX2+	SGMII Rx Line connects to the Red Switch SFP
B62	SGMII_RX2-	SGMII Rx Line connects to the Red Switch SFP
B64	SGMII_RX1+	Connected to B61
B65	SGMII_RX1-	Connected to B62
B68	SGMII_RX0+	SGMII Rx Line connects to the Black Switch SFP
B69	SGMII_RX0-	SGMII Rx Line connects to the Black Switch SFP
B70	GND	Ground
B80	GND	Ground
B100	GND	Ground
B103	GND	Ground
B104-B109	VCC	Power Input (+5V or +12V) default value=12V
B110	GND	Ground

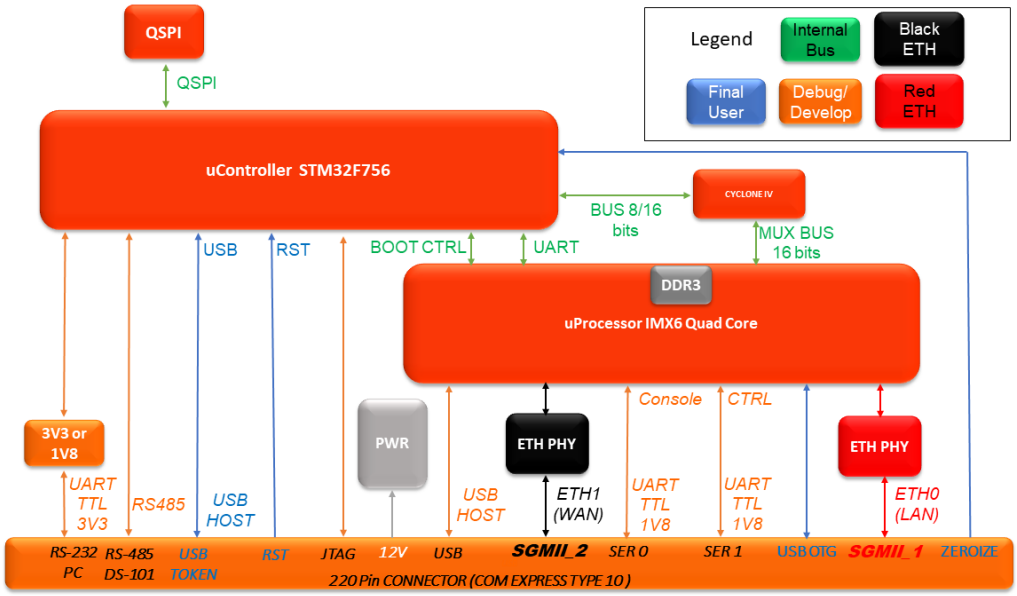

Figure 3.6: Block Diagram of the main board

FPGA-Cyclone IV

The FPGA is interfaced with the STM microcontroller and with the IMX6 microprocessor with two different buses. The STM is interfaced with the FPGA with a Bus of 16 bits of data and 10-bits address. Instead, the IMX6 is connected with a multiplexed Data/Address of 16-bit.The Pinout of FPGA is in table 3.8.

Table 3.5: PinOut of the Interface between STM and FPGA

SIGNAL	Function/Description	Connection	Direction
nEMC_BL0	Memory (Data Low)	FPGA	Out
nEMC_BL1	Memory(Data High)	FPGA	Out
EMC_OE	Memory(Output Enable)	FPGA	Out
EMC_WE	Memory(Write Enable)	FPGA	Out
EMC_CS1	Chip Select for FPGA	FPGA	Out
nINT_FPGA	Interrupt from FPGA	FPGA, Int=0;	In
nRST_FPGA	Reset FPGA	FPGA, Rst=0;	Out
EMC_A0-EMC_A5	Memory(Add 0-5)	FPGA	Out
EMC_A12-EMC_A15	Memory(Add 6-9)	FPGA	Out
EMC_D0-EMC_D1	Memory(Data 0-1)	FPGA	I/O
EMC_D2-EMC_D3	Memory(Data 2-3)	FPGA	I/O
EMC_D4-EMC_D12	Memory(Data 4-12)	FPGA	I/O
EMC_D13-EMC_D15	Memory(Data 13-15)	FPGA	I/O
FPGA_CLK	FPGA Clock	FPGA	Out
FPGA_CRC_ERROR	Crc Error from FPGA	FPGA	In
FPGA_INIT_DONE	Init. Signal from FPGA	FPGA	In
FPGA_CONFIG_DONE	Conf. Signal from FPGA	FPGA	In
FPGA_nSTATUS	FPGA Signal Status	FPGA	In
FPGA_nCONFIG	FPGA Config Signal	FPGA	Out
FPGA_PWRON	Power On Control Signal	FPGA, Off=0	Out
nFMC_NE4	CS for FPGA registers	FPGA	Out

Table 3.6: PinOut of the Interface between STM and IMX6

SIGNAL	Function/Description	Connection	Direction
IMX_BOOT_CTRL0-1	Control Signal	IMX6	Out
IMX_BOOT_CTRL2	Control Signal	IMX6	Out
IMX_BOOT_CTRL3	Control Signal	IMX6	Out
IMX_BOOT_CTRL4	Control Signal	IMX6	Out
IMX_BOOT_CTRL5-7	Control Signal	IMX6	Out
IMX_BOOT_CTRL8-9	Control Signal	IMX6	Out
IMX_BOOT_CTRL10	Control Signal	IMX6	Out
IMX_BOOT_SEL0-1	Sel. 0 and 1	IMX6	Out
nUCC_IMX6_RST_3V3	Reset for IMX6	IMX6;	Out
UCC_IMX6_PWRON_3V3	Power On IMX6	IMX6;	Out
UCC_IMX6_PW_EN	Pw Enable for IMX6	IMX6;	Out
UCC_IMX6_RXD_3V3	Rx UART from IMX6	IMX6	Out
UCC_IMX6_TXD_3V3	Tx UART to IMX6	IMX6	In

Table 3.7: PinOut of STM

SIGNAL	Function/Description	Connection	Direction
PWR_GOOD	Power Supply GOOD	ERR=0;	In
ATD_ALARM	Alarm of Antitampering	ALARM=0;	In
QSPI_CLK	Interface QSPI(Clock)	QSPI	Out
QSPI_NCS1	Interface QSPI(Chip Select)	QSPI	Out
QSPI_D0-1	Interface QSPI(D0-1)	QSPI	I/O
QSPI_D3	Interface QSPI(D3)	QSPI	I/O
QSPI_D2	Interface QSPI(D2)	QSPI	I/O
TMS/SWDIO	JTAG Interface	PC	In
TCK/SWCLK	JTAG Interface	PC	In
TDI	JTAG Interface	PC	In
TDO/SWO	JTAG Interface	PC	In
TRST	JTAG Interface	PC	In
STM32_LED1_A_3V3	LED1 STM32F756	ON=1;	Out
STM32_LED1_A_3V3	LED1 STM32F756	ON=1;	Out
UART7_RX_3V3	Rx UART from PC	PC	In
UART7_TX_3V3	Tx UART to PC	PC	Out
USB_FS_N	USB- Interface	USB TOKEN	I/O
USB_FS_P	USB+ Interface	USB TOKEN	I/O

3.2 STM Microcontroller Software

In this section the role of the STM, its behaviour and the different functions of the code are analized. In figure 3.7, the flow chart of the STM set up is represented. After the power on of the ENA (so the power on of STM), the STM has to be configured to open in a right way the USB Token. Then, two STM Leds are switched on to communicate that the USB configuration is done and it is possible to plug in the USB Token. The USB Token has to be identify; after that the bitfile is write into the flash memory. If the process ends without errors, it is possible to configure the FPGA. At the end of the FPGA configuration, the microprocessor IMX6 is switched on.

3.2.1 Initialization TOKEN

After the power on of the STM, it is necessary to configure the USB host. A USB Token is used to download the FPGA bitfile into the board. The Token is a USB pen-drive with a dedicated chip and the possibility to plug in a micro SD and a SDCard. It contains two file systems: one is the Virtual System USB and the second is a FatFs folder called DISC_1. The DISC_1 is the folder of the microSD. In DISC_1 a folder is present named with the serial number of the device. Each device has an unique serial number, so in this way, every single USB token is associated to its device.

Table 3.8: Pinout of FPGA Cyclone IV

SIGNAL	Function/Description	Connection	Direction
EMC_BL0(IO/-BLE)	Memory(Data Low)	STM	In
EMC_BL1(IO/-BLE)	Memory(Data High)	STM	In
EMC_OE	Memory(Output Enable)	STM	In
EMC_WE	Memory(Write Enable)	STM	In
FPGA_CS1	Memory(Chip Select)	STM	In
FPGA_INT	Interrupt to STM	STM	Out
RTS_FPGA	Reset of FPGA	STM	In
EMC_A0-A9	Memory(Address 0-9)	STM	I/O
EMC_D0-D15	Memory(Data 0-15)	STM	I/O
FPGA_CLK	FPGA Clock	STM	In
FPGA_CONFIG	Configuration Signal	STM	I/O
FPGA_CONFIG_DONE	Configuration Done	STM	Out
FPGA_CRC_ERROR	CRC Error Signal	STM	Out
FPGA_INIT_DONE	Init. Done Signal	STM	Out
FPGA_PWRON	Power On Controll	STM	In
FPGA_STATUS	Status Signal	STM	Out
EIM_BCLK	Clock from IMX6	IMX	In
EIM_CSN	Chip Select	IMX	In
EIM_DA0-15	Data/Address 0-15	IMX	I/O
EIM_EB0-1	Enable Signal	IMX	In
EIM_LBA	Load Signal	IMX	In
EIM_OE	Output Enable	IMX	In
EIM_RW	Read Signal	IMX	In
EIM_WAIT	Wait Signal	IMX	Out

If a different token is plugged in, it is no possibility to open the folder and to perform all the next operations. In the folder is present the bitfile of the FPGA named "fpga.rbf". From now on, if a Token USB is plugged in, an interrupt is sent to the STM.

3.2.2 LEDs 1 & 2 ON

After the Token Initialization, two LEDs are switched on. Before do this, the PIN I/O have to be configured. LEDs are output pin and they not require a high output maximum frequency. If one of the two leds remains off, some problems occur.

3.2.3 Communication TOKEN

It handles all the functions to communicate with the token, to write the Bitfile into the FPGA and to communicate with the FPGA. The STM waits until a USB is plugged in.

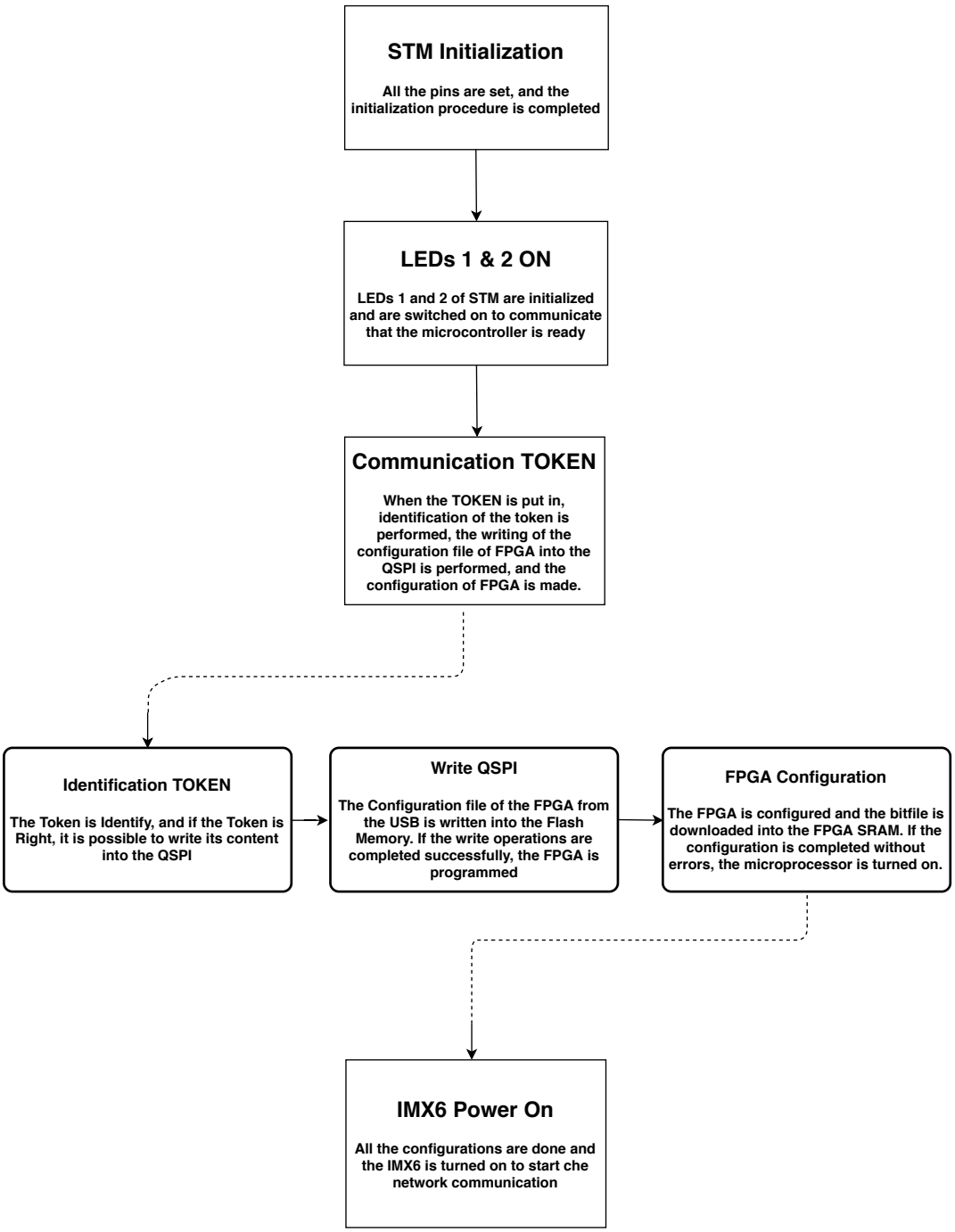

Figure 3.7: Flow Chart of STM software

When a USB is put in, an interrupt is sent to the STM, so the machine passes from a Disconnected state to a Connected state. After that, it is necessary to verify that the token has the correct serial number of the device. It is a function into an infinite loop; after that the FPGA is configured, and the IMX6 is switched on, STM does nothing. First of all, it is needed to process in loop the USB host Backgroud task, in this way when the USB TOKEN is plugged in, an interrupt is sent to the STM and the USB Class passes from DISCONNECT to HOST_USER_CLASS_ACTIVE so the application state is : APPLICATION_READY. So when the USB is put in, it is possible to verify the correctness of the Token. Then it is written the Bitfile into the QSPI. After that, even if the STM is turned off, the bitfile still remains into the Flash Memory but to configure the FPGA the TOKEN must be plugged in. Then the FPGA is configured and the bitfile is write into the SRAM of FPGA. Details of the functions are reported in the following subsections.

Identification Token

When a Token is plugged in, it is necessary to perform an identification to understand if the token is a valid token. Every single IP BOX has a serial number written into the Flash Memory of the STM. So a TOKEN has a unique identify serial number that has to correspond to the serial number of the machine. The serial number is the name of a directory into the µSD DISC_1 CARD. If the path with the serial number is correct and the directory is correctly opened, it is possible to written the bitfile contained into the directory.

Write QSPI

The Bitfile is in the folder that has the Serial Number as name, so the first thing to do is to open the folder and read the file "fpga.rbf". After that it is possible to compute the total length of the file. In the QSPI, it is not included only the bitfile but, before it, an Header is written and at the end of the bitfile a footer is included. So it is computed the total size taking into account of the total length. The QSPI Flash memory can be eresed only 512 Bytes every time, so the total number of blocks is computed. Then, all the blocks are erased. The Header contain a number called "Magic", the length of the bitfile and number of blocks. So the computed Header file is written into the QSPI. Every time a block of 512Bytes is read and write, it is verified the correctness of the operations; if there are some errors or differences between what it is written and what is read, the operation is stop immediately. Then it is verified the correctness of the written data. A reading cycle is performed and the two data are compared. During the operations a SHA-256 is computed to verify in any time the corruption of the file. At the and of all writing cycles, the final Sha-256 is computed and it is written at the end of the Bitfile into the Flash memory. A flow chart of the Download of the bitfile into the FPGA is in figures 3.8 and 3.9.

FPGA Configuration

Before the FPGA Configuration, it is necessary to power on the FPGA and set pins in properly way. All the output pins are put in default mode, in this way no conflicts are possible. The FPGA requires an external clock to perform the configuration and read the data, so with the STM it is possible to set a special pin (PC9), that is used as clock pin. A clock of $108MHz$ is selected.
Then, the bus that connects the STM with the FPGA has to be initialized and configure. The FPGA pins are placed in different STM ports, so it is necessary to configure different Port GPIO (D, E, F, G). The FPGA has an SRAM that stores the bitfile, so the FMC (subsection 2.2.1) of the STM has to be configured. It is choosen to use two configuration banks to perform different tasks. The bank 3 is used for configure the FPGA and has a relaxed time to reduce configuration errors. So it is choosen as Setup Time a value equal to 2, that is a multiple of the core clock $f_{HCLK} = 216MHz$ so a period of $T_{HCLK} = 4.63ns$. So the Setup time it is choosen to be $ADDSET = 4.63ns * 2 = 9.26ns$. From the figure 2.8, the setup time is the time needed from when the address is on the bus and the output enable is pulled down. From figure 2.9 the setup time is the time from when the address is on the bus and the write enable is acivated. The Data setup time is choosen equal to 10. In this way $DATAST = 4.63ns * 10 = 46.3ns$; during the reading process, it indicates the time duration of the output enable; so, it is the time during which it is possible to read the data on the bus. During the writing process , DATAST indicates the time during which the data is writing, so during the Configuration with long time, the FPGA has much time to save the data correctly in the SRAM, reducing this time, the FPGA has to read the data faster. The bank 0 is used for interface the STM with the registers of FPGA. They are used for perform a test to verify the correctness of the bitfile into the FPGA. After the initialization of FPGA, the configuration procedure can start.
The configuration of the FPGA is performed; after the power on of the FPGA, it is tested the integrity of the FPGA; then it is tested if the FPGA is in stand-by state. So it is set to 1 the pin $nCONFIG$, after a small delay, it is read the pin $nSTATUS$; if the pin is up, the FPGA is in STAND-BY state, otherwise there is an error ed it is not possible to perform the configuration, so the Exit Config process is performed. After that, a flag called *program_done* is set to 0; only if the configuration is correctly completed, it is asserted. Then all the errors are setted to 0; during the process, the errors are removed; this procedure guarantees the correctness of the process. At the end of the process, if one of this flag remains zero, there is some problem. During the configuration, it is possible that some errors could occur, so a reprogramming process is taken into account. The configuration, in case of error, is performed maximum twice, after that, if errors continue to occur, the FPGA is switched off. All the possible errors that can occur during the programming process are reported; on every reading cycle or writing cycle, if error occur,the FPGA is switched off. Before the writing of the bitfile into the FPGA, it is verified that the file stored into the the QSPI contains a right HEADER_BITFILE.(3.10) After the reading of the header from the QSPI, the magic number contained into the QSPI is compared with the Magic number written into the memory of the STM, if the two magic number are the same, it means that it is a secure file. Then the length contained into the header is analized

and if the file has a length equal to zero,error is returned and the process is stopped. Into the Header file is contained also the number of 512byte blocks. If this number is equal to zero an error is returned. If the HEADER_BITFILE is correct, it is possible to start the configuration 3.11. The process configuration ends when the flag program_done is set to 1 or if the counter goes to zero.
During the process, the SHA256 is computed to guarantee that the data are not corrupted. So, the SHA is initialized then the total number of byte it is set to zero. The FPGA starts the process configuration if a transition from 1 to 0 and then from 0 to 1 has to be performed from pin n_CONFIG as indicated in figure 2.12 and the FPGA "responds" moving the pin n_STATUS. If the FPGA pulls down and then up the pin, the reading and writing operations start.
An iterative process is performed for every block of 512Bytes (figure 3.12). The block is first read from the QSPI, the SHA256 is updated and an iterative cycle is done; one byte per time is write into the FPGA (figure 3.14); when all blocks are read, the pin n_STATUS is read and if it is at one the flag program_done is set to 1; else the program_done still remains zero and the exit_loop is performed (figure 3.13). When the exit_loop is performed with program_done = 0, the counter is decreased and the programming process is retried, because all the blocks are written into the FPGA but it doesn't pull down the nSTATUS flag for some errors. Every writing process of 512 bytes is concluded, the pins CONF_DONE, INIT_DONE and CRC_ERROR are checked and if one of this is pulled up, the error flag CONF_DONE_ok is removed (figure 3.14). After that all blocks are written correctly, the final SHA256 is computed. If the computed SHA256 is different from the SHA256 written at the end of the QSPI file, the bitfile written into the FPGA is corrupted, so the exit_config is performed and the FPGA is switched off. If the two SHA are equal, the bitfile is not corrupted (figure 3.13); so the *program_done* is check and if during configuration the $nSTATUS$ pin is pulled down, the flag is set, then the $Conf\_DONE\_ok$ is check and if it is equal to zero, means that the configuration is completed but contains some errors so the FPGA is switched off. If the $CONF\_DONE\_ok$ is equal to 1, the configuration doesn't contain errors. So, at this point, the initialization procedure is performed and a dummy cycles for $INIT\_CYCLE$ is done. At the end if the initialization is correctly completed, both n_STATUS pin and $CONF\_DONE$ pin are moved by the FPGA. If one of this is at 1, the initialization is finished but contains errors. If it is correct, other dummy cycles are performed. Then the check of the FPGA is performed, writing one register and after a delay, the same register is read; the result must be congruent with the operations written into the FPGA with the bitfile. If everything is completed correctly, the n_CONFIG pin is disabled and the configuration of FPGA is successful.

Power on of IMX6

After that the configuration of the FPGA and the tests, the IMX6 can be switched on and the application can start.

3.3 IMX6 Microprocessor Software

The microprocessor IMX6 is the core of the ENA; it handles all the traffic network. The scope of the board is to encrypt IP packets and decrypt them. So the IMX has to manipulates packets that arrives from one Ethernet interface, send them to the FPGA, then read them from the FPGA and forward them to the second Ethernet interface. The ENA(Embedded Networked Appliance) board is never used for this scope, so deep analysis are performed to verify the performance of the board. Different steps are done before to perform the complete task of the IMX6 in order to report all the data.

3.3.1 Kernel IP Forwarding

The IMX6 has two Ethernet interfaces one is named eth0 and the second is eth1. The first step is the verification of the correctness work of single ethernet. For this reason two PC are connected to the board with two copper ethernet wires. The two Ethernet interfaces have to be configured. The IMX6 has a USB console, so using a microUSB cable is possible to open the console of IMX6 on a PC. On the PC it is required a Linux Operating system. After the connection of the IMX6, the console is opened. In this condition the Ethernet interfaces can be configured. The network setup can be done via the $interfaces$ configuration file that is at $etc/network/interfaces$. So editing this file it is possible to set-up the network. Here it is possible to give the network card an IP address (or use the DHCP), set up the routing information, configure the IP mask, set the default routes and other options. The board has two network interfaces that have to be configured. In figure 3.15 is present the network interfaces that is created. The first interface has a Net-ID different from the other interface. The two PCs are configured with its own net ID and a gateway; through the gateway it is possible to know how to reach the destination. From the PC 1 it is sent a ping to the Board and it is verified if it works, the same think is done with the PC 2. So the two Pc are connected respectively to own network. Until now the Two PC are connected with the board but are not connected together: the two networks are different, so in order to connect one host to the other host it is needed that the board "connect" the two network; in this way each packet that arrives from the PC1 is forwarded and can be received from the PC2. To do this the IP forwarding is used. IP forwarding (or IP routing) is a process used to transfer IP packets from one network to another. So the next step is to enable the IP forwarding on the Board and verify the correct work. It is possible to enable the IP forwarding writing in the IMX6 console : $echo\ 0 > /proc/sys/net/ipv4/ip\_forward$. In this way if the configuration of the hosts and of the board is done correctly, it is possible to send a ping from the PC 1 to the PC2 and all the packets that arrives to the Board and are directed to the PC 2 are forward. It is possible in this situation to verify the performance and the throughput of the connections. Those results are the maximum throughput that the ENA can be achieve. The values are used as references for the next experimental tests.

3.3.2 User Space application for IP Forward

The second step is the design of an application at User space that handles the traffic network, manipulates every IP packet and forward it from the first Ethernet interface to the second interface. So an IP forward application is designed. There are some different possibilities to perform this task. One of this is the use of Socket Raw. A networking socket is an abstractive software that is an applicative endpoint to access to a communicative channel through a port. So a bidirectional network communication between two physical machine that are separating each other. From a software point of view, a socket is an object that has the possibility to read data or transmit data. A raw socket is a particular socket that has the possibility to read, write IP(v4) datagrams with protocols that are not handle from the kernel and it is possible to determine every section of the packet, like header and payload. Its is possible to categorize raw socket into Network Socket (L3 Socket) and Data-Link Socket(L2 Socket). In L3 socket it is possible to determine header, payload of the packet in th network layer; in L2 socket instead it is possible to set Header and payload of packet at datalink layer, so it is possible to modify and manipulate everything in packet. In the design application, the sockets are first opened, one for eth0 and the second for eth1. After that, the application waits until something arrives on the two interfaces. If the receiver of the eth0 or eth1 receives datas, all the bytes are saved into a buffer. Then the destination IP address(32 bits) is extracted from the Ethernet Frame and if the destination address isn't the IP address of the current interface, the packet is forward and it is sent to the other interface. In this way, every packets that arrives from one side, are forwarded and sent from the other interface. So sending a ping from PC1 to PC2, the IP packets arrive from the other side and a connection between the two hostes is established. In this situations the tests and the performances are computed and in the chapter 4 are discussed.

3.3.3 Encryption and Decryption of Ethernet Packets

The last step is to include in the application the FPGA scope. The packets that arrive from one channel are saved into a buffer; then are sent to the FPGA and after a number of clock cycles (that depends on the length of the Ethernet frame), if the data were clear, they are crypted from the FPGA and are read; the crypted informations are sent to the second interface. The same think happens when the ethernet data that arrives are crypted; they are sent to the FPGA and are encrypted. Those operation take time, so the throughput of the ENA decreases. So an analysis is performed in the chapter 4. Those performance are the real throughput.

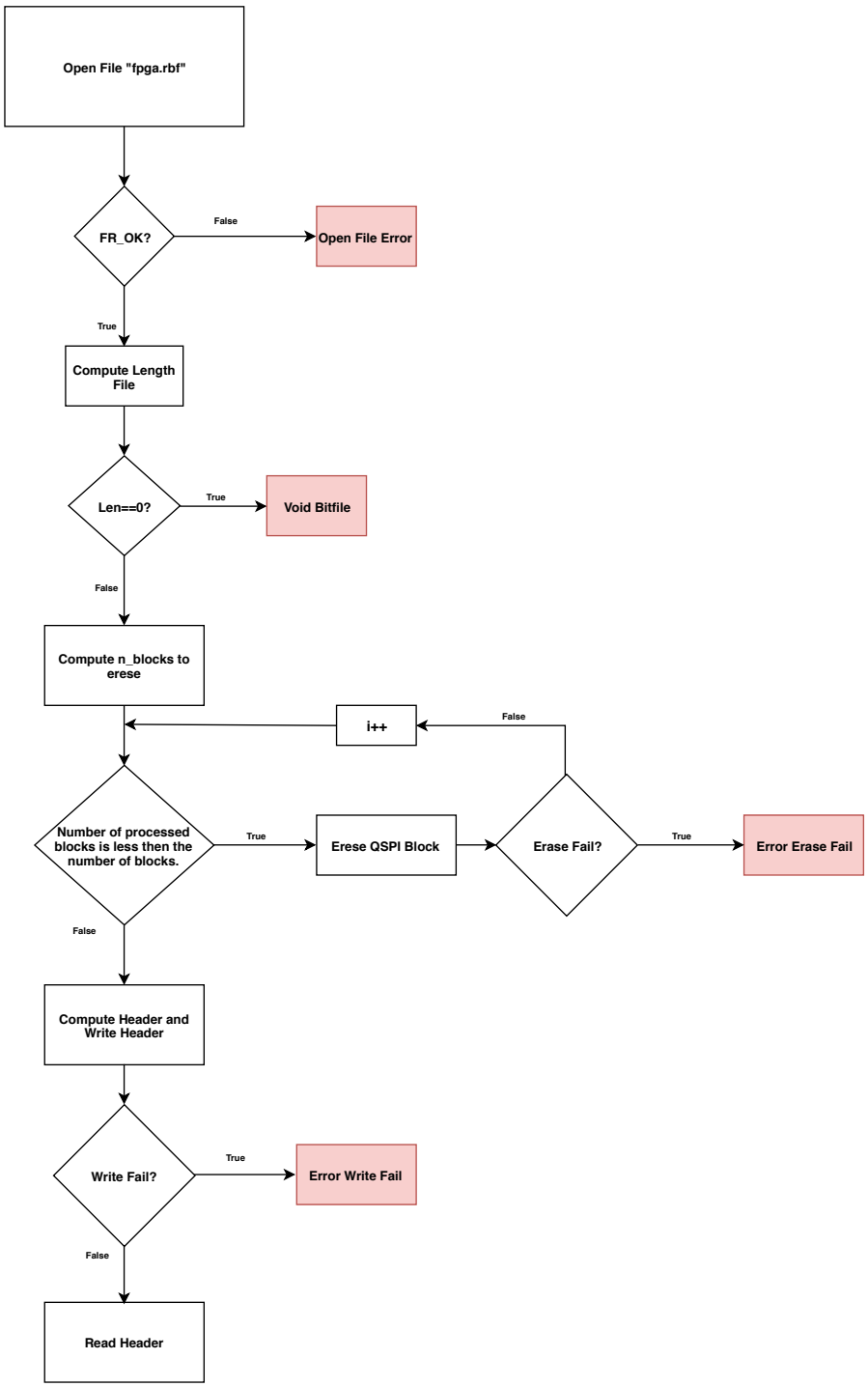

Figure 3.8: Flow Chart for QSPI writing process Part I

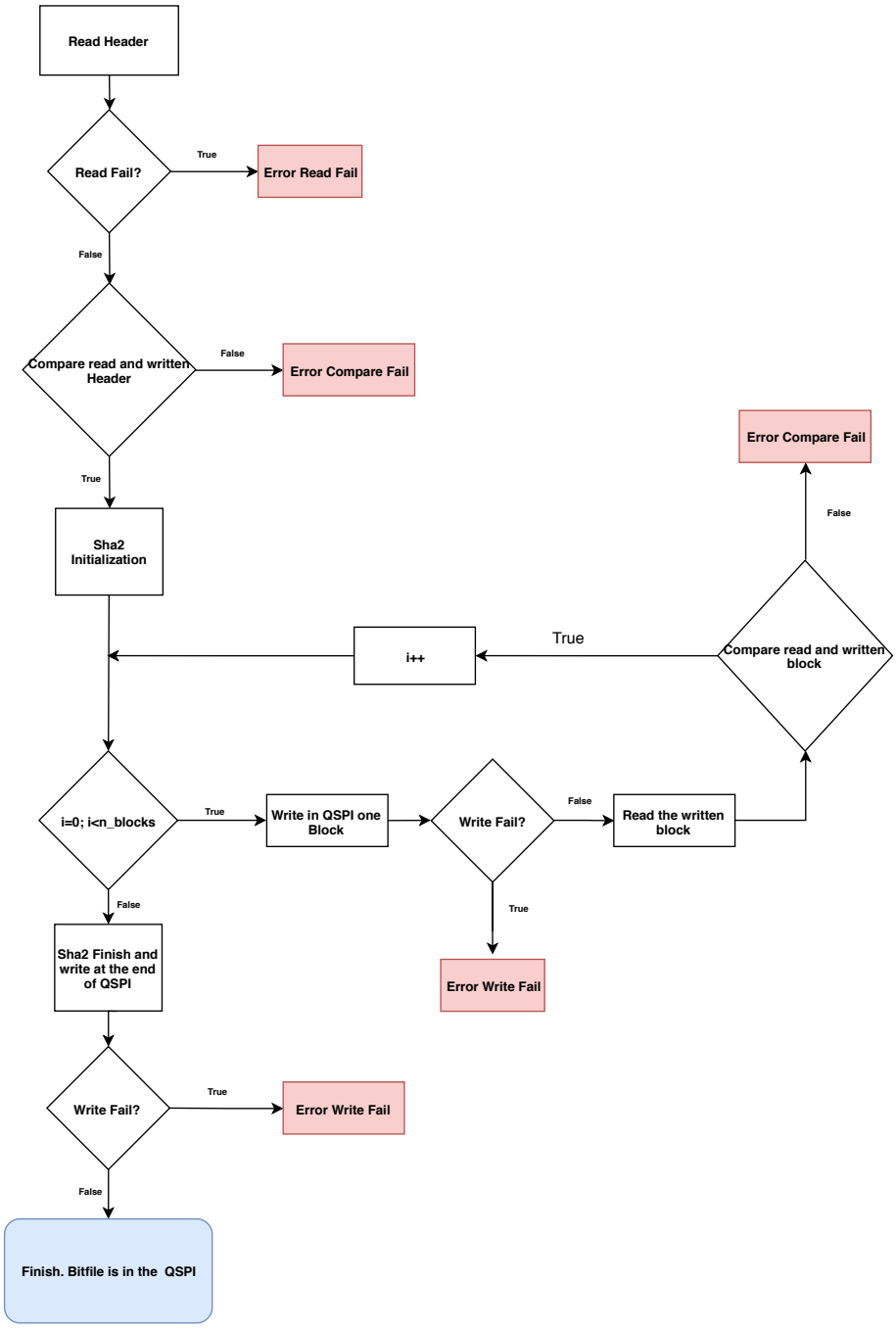

Figure 3.9: Flow Chart for QSPI writing process Part II

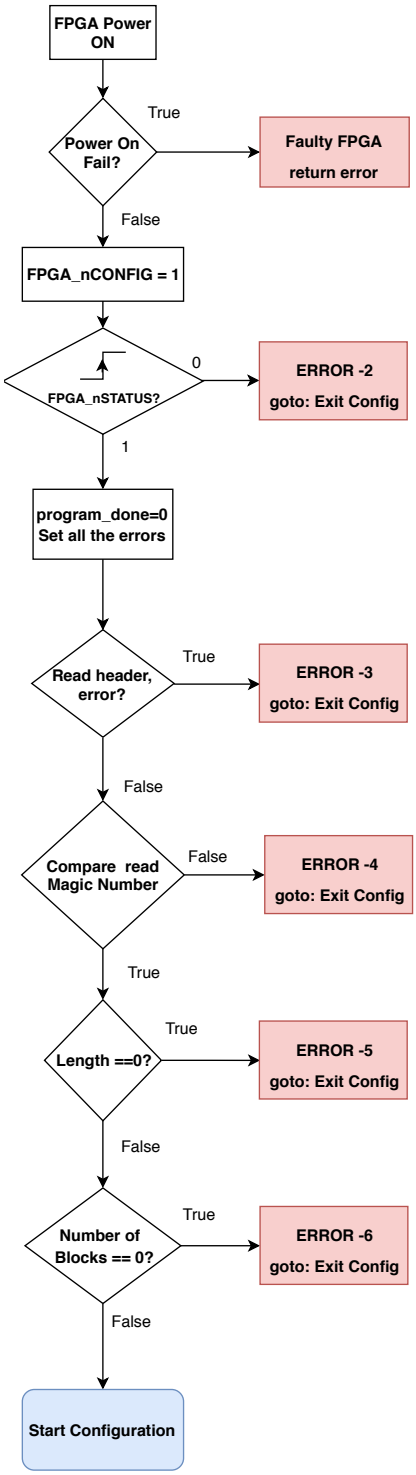

Figure 3.10: Flow Chart for FPGA Configuration Part I

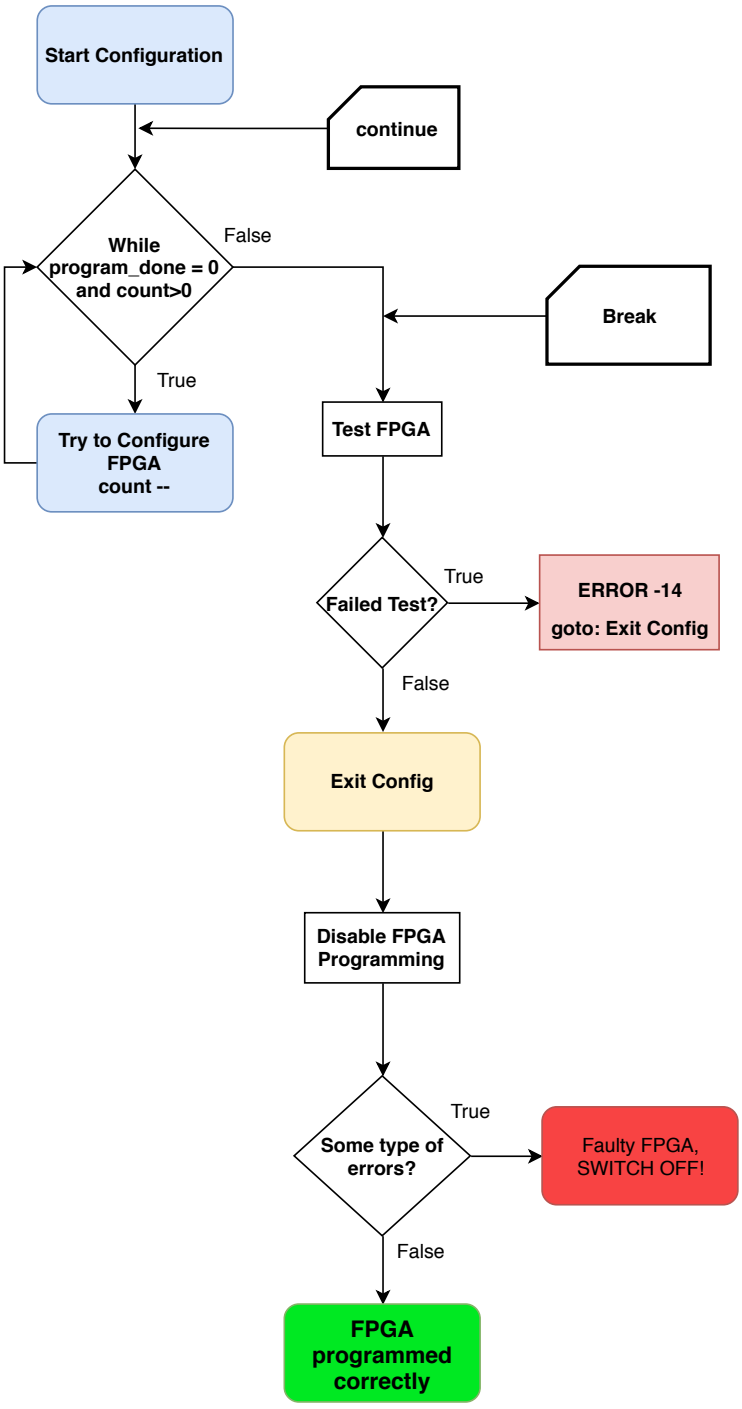

Figure 3.11: Flow Chart for FPGA Configuration Part II

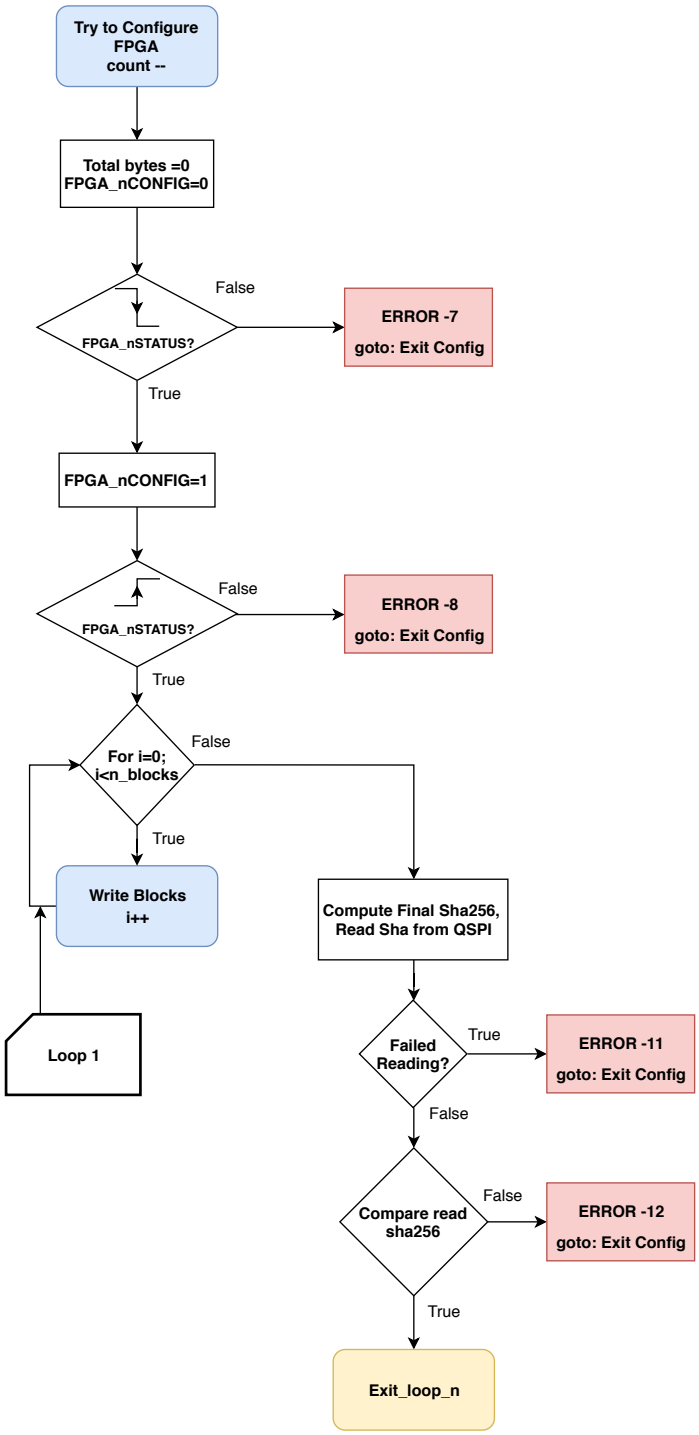

Figure 3.12: Flow Chart for FPGA Configuration Part III

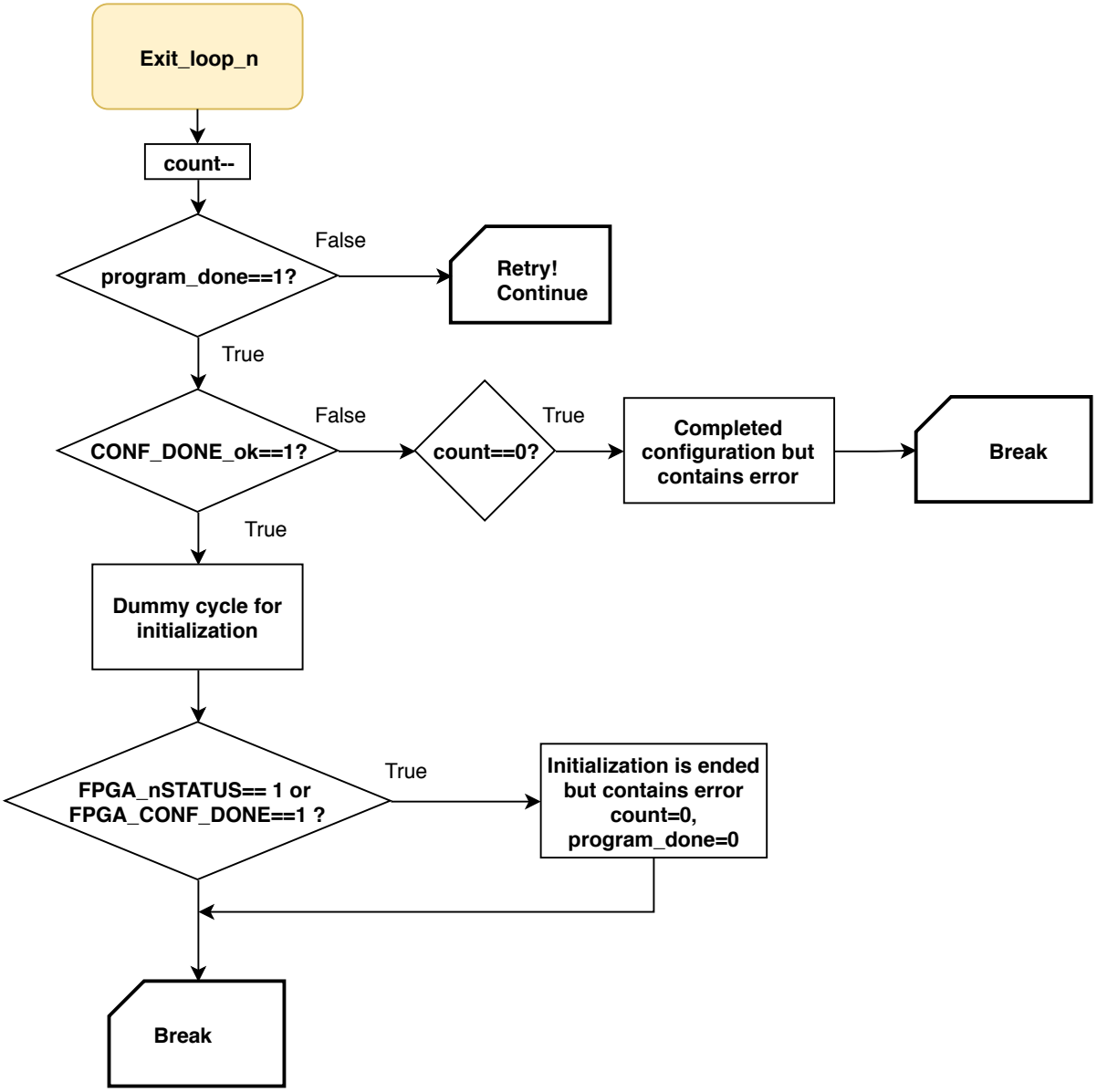

Figure 3.13: Flow Chart for FPGA Configuration Part IV

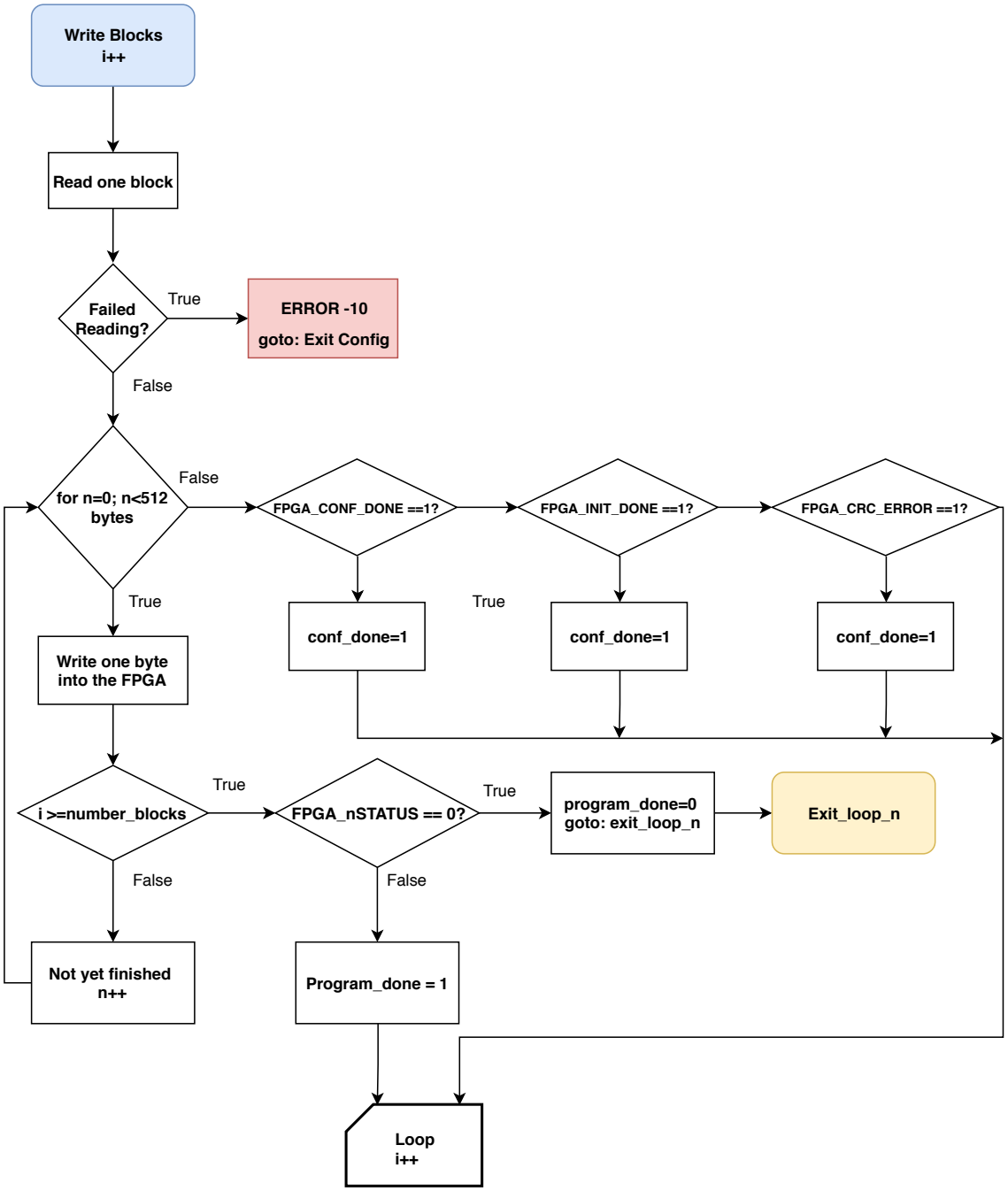

Figure 3.14: Flow Chart for FPGA Configuration Part V

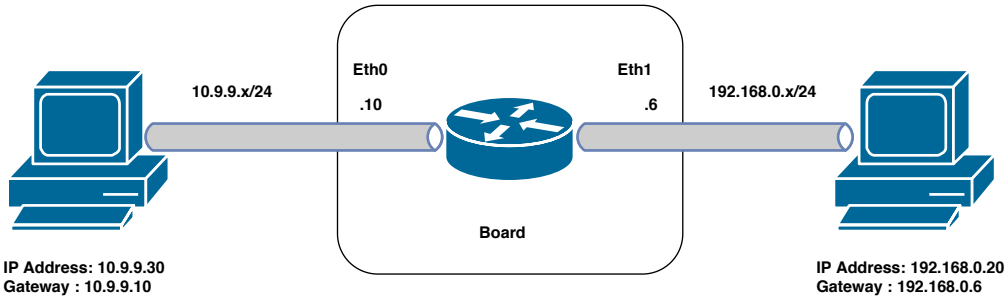

Figure 3.15: Network configuration of the Devices

Chapter 4

Throughput Tests

4.1 Description of the test devices

Before the execution of the throughput tests, a detailed description focusing the attention on the Ethernet interface architectures is performed, in order to compare in the section 4.2 devices with similar architecture. In this way, consistent tests can be performed and conclusion can be made. The throughput tests are performed on a commercial board of ROJ company called SMARC Enuc ROJ board. It is an embedded board with the same microprocessor IMX6. A deeper analysis on the hardware architecture is performed to understand if the two machine can be compared.

4.1.1 Ethernet interface hardware architecture of ENA

In figure 4.1 is present the block diagram of the ethernet interface connections between the physical connector and the IMX6 on the ENA board. The ENA Ethernet Interface is composed by two SFP Connectors used to connect Optical Fiber or RJ-45 wire (with a Converter). The Two SFP connectors are interfaced with the IMX6 with two different chips. The Intel WGI210AS is a Gigabit Ethernet Controller that offers Physical Layer Port and SGMII/SerDes port that can be connected to an external PHY. It also supports the PCI Express. So in this case it is used to connect SFP port to the Microprocessor IMX6 through PCI express. The second interface (red) is connected to a Marvell 88E1512. The Marvell Gigabit Ethernet transceiver is a physical layer device that containing a single transceiver. It supports RGMII to SGMII/Fiber, SGMII to Copper. In this case it convert a SGMII to RGMII and vice-versa. In this way it is possible to interface both SFP module with the microprocessor.

4.1.2 Ethernet interface hardware architecture of Smarc ROJ

In figure 4.2 the block diagram of the Ethernet interface architecture of the Smarc is represented. The SMARC ROJ Ethernet Interface is composed by two RJ-45 Connectors used to connect copper wires. The Two RJ-45 connectors are interfaced with the IMX6 with two different chips. The Intel WGI210AT is a Gigabit Ethernet Controller that offers

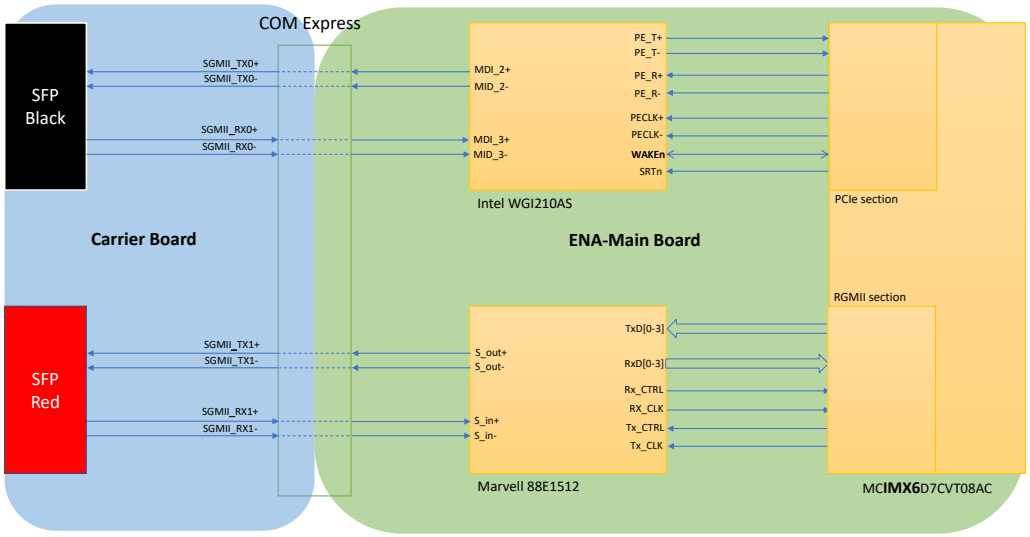

Figure 4.1: Ethernet interface architecture of ENA

Physical Layer Port and SGMII/SerDes port that can be connected to an external PHY. It also supports the PCI Express. So in this case it is used to connect the RJ-45 port to the Microprocessor IMX6 through PCI express. The second interface (red) is connected to a KSZ 9031RNXIA. The KSZ is a gigabit Ethernet Transceiver that support RGMII.It is possible to connect the RJ-45 Port to the RGMII IMX6 interface. The two boards share the same architecture interface from/to the IMX6. So, to continue the comparison is necessary to analyse the differences into the software.

4.1.3 Software comparison

The SMARC ROJ is an enterprise product and the kernel is designed and installed by the ROJ Company. The ENA kernel is designed by another external company. The first software analysis is performed writing in the console of both devices: uname -a. The two kernel version are shown:
ENA: 4.1.29 armv7l
Smarc ROJ: 4.9.11 armv7l
It is possible to compare the two kernel versions. Devices have two different kernel version 4.1 and 4.19 . Due to the fact that they share the same microprocessor the version of the arm is v7l. Other information can be collected from the lspci command that shows all the PCI devices. Writing: lspci -nnk it is possible to compare the drivers of the two Intel Gigabit controller. The two devices share the same family chip (Intel I210); they have two different codes (1533 for the SMARC and 1537 for the ENA) but has the same driver. So, no difference between the two PCI interfaces are present. The drivers of the RGMII interfaces are write into the kernels; so because of the two kernels are different, the

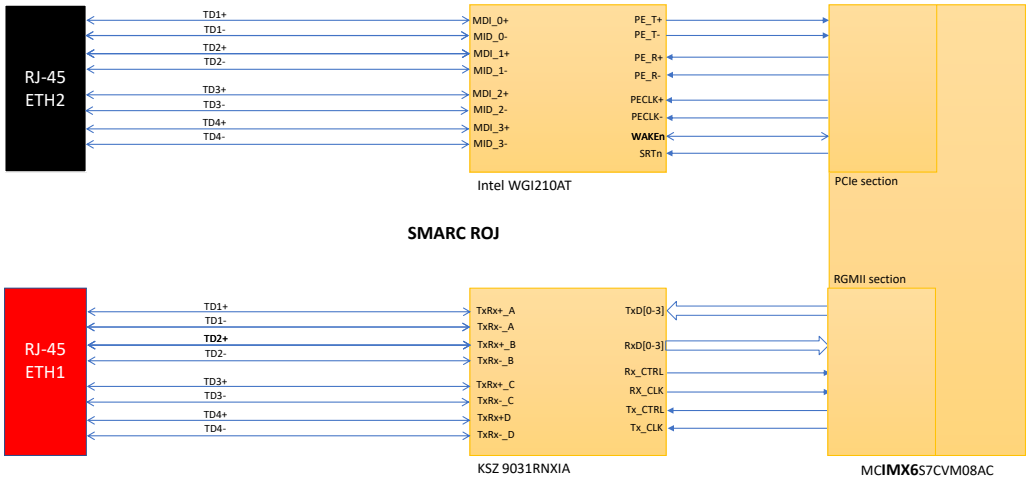

Figure 4.2: Ethernet interface architecture of Smarc ROJ

drivers of the ethernet interface connected through RGMII are different. So, in conclusion, the drivers of the two devices cannot be compared. They share the same microprocessor, the same hardware structure and the same drivers for the chip intel I210, but different drivers for the other two ethernet interfaces. Nevertheless, it is possible to compare the performances of the two devices.

4.2 Throughput performance of Kernel IP Forwarding

One of the way to determine if a device can be used as network handling machine is to perform measures of its performance. There are different indicators to measure network performances like the latency and the throughput. In the examined cases, it is performed a UDP measuring network throughput. The UDP protocol is used to carry data over IP networks; it is not influenced by the time required to deliver the packets (the latency) because the sender will send a given number of packets per second. A network traffic analyser Optiview XG is used to perform throughput analysis. The throughput is the maximum rate at which IP packets can be processed. It is measured as the number of bits that can be processed in the unit of time (bps). In UDP throughput measures, the maximum transmission rate must be chosen. In those tests a speed of 100Mbps is set; so, the local tester sends 100 million of bits in one second. The test system is composed of two test machine and a DUT (device under test). The DUT has to perform an IP forwarding, so it has to forward the packets that arrive from one port and send them through the second port. Three different devices are analysed: a general-purpose system, teh Smarc ROJ Board and the designed board (ENA). During the test different sizes of packets are

sent. With a maximum fixed throughput and fixed size, it is possible to compute the number of packets that are sent in one second. The number of theoretical frames per second are computed in equation 4.1:

$$fps = \frac{100Mbps}{8bit * size} \tag{4.1}$$

The test system is in the figure 4.3

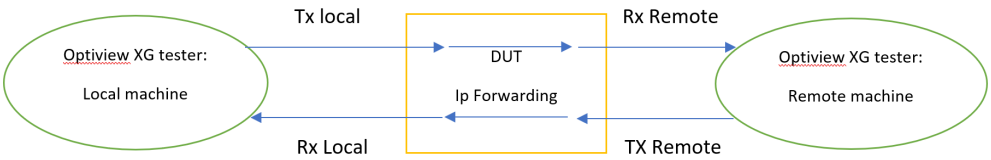

Figure 4.3: Test System

4.2.1 General Purpose Throughput test

The figure 4.4 represents the throughput rate. The local transmitter sends packet at 100Mbps; the receiver of the remote tester receives the packets at different speed rate. Then the remote transmitter sends packets at 100Mbps and the local Rx receives the packets at different rate. In the ideal case the data are sent and received at 100Mbps. The two testers theoretically send and receive at the same rate. Two aspects have to be taken into account: the symmetry and the rate. A good device for IP forwarding should have a perfect symmetry, so the received data from one side has to be the same of the received data from the other side, and it should have the maximum throughput (100Mbps in this case). Not all the devices have the same performance, so it is necessary to test the different devices. A general purpose system is the best device for perform IP forwarding so it is used as reference for the different tests. In the second graph is represented the number of packets that are sent and received. This value is strictly related to the size of the packets. Fixed the frequency, the number of packets is inversely proportional to the size of the packet. In fact if the packet is small, at 100 Mbps, it is sent higher number of packets. Also in this case an ideal device has to be symmetrical and has to send and receive the maximum number of packets. In the case of a general purpose system, the graph shows that all the curves are superimposed, so it sends packets and receives approximately the same number of packets. Data are in tables 4.1 and 4.2.From the table 4.2, smaller is the size of the packet, much packets have to be sent at the same rate, so the device has to be receive a higher value of packets in one seconds; for this reason with small packets the probability that some packets are lost is higher than the case with high dimension packets.

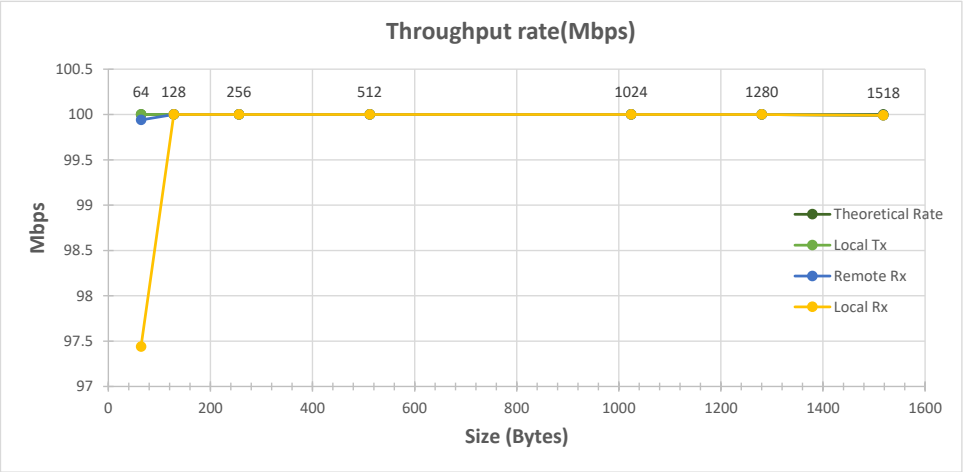

Figure 4.4: General Purpose System Throughput Test

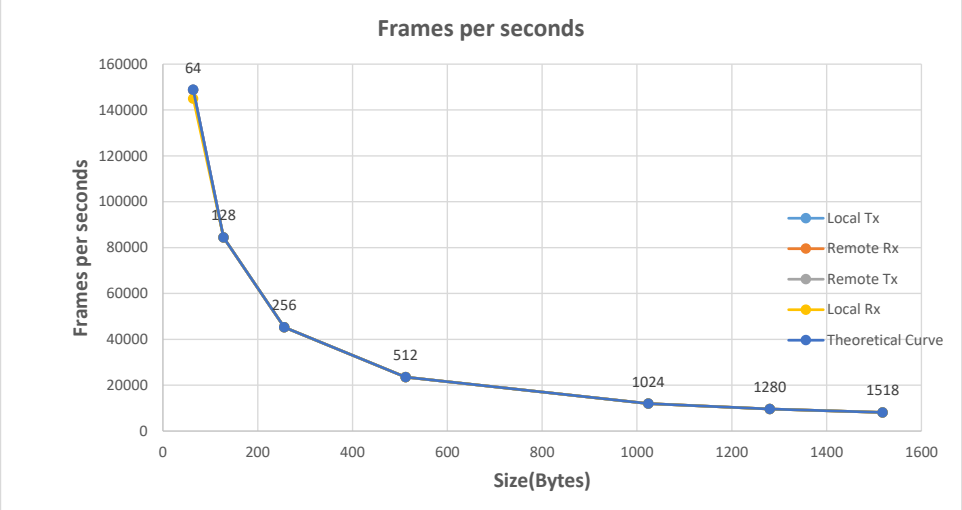

Figure 4.5: Packets vs Size Graph of General Purpose System

4.2.2 ROJ eNUC Throughput test

With the eNUC device, the throughput rate with small packets(64,128,256 Bytes) is lower respect the maximum rate; another problem is related to the symmetry of the two lines; in fact the remote device receives at higher rate respect the local receiver (figure 4.6). It is possible to analyse also the frames that are sent and received. In light blue is represented

Table 4.1: Data of Throughput of General Porpose System in Mbps

Size(Bytes)	Local Tx (Mbps)	Remote Rx (Mbps)	Remote Tx (Mpbs)	Local Rx (Mbps)
64	100.00	99.94	100.00	97.44
128	100.00	100.00	100.00	100.00
256	100.00	100.00	100.00	100.00
512	100.00	100.00	100.00	100.00
1024	100.00	100.00	100.00	100.00
1280	100.00	100.00	100.00	100.00
1518	99.99	99.99	99.99	99.99

Table 4.2: Number of frames per second of General Porpose System

Size(Bytes)	Local Tx	Remote Rx	Remote Tx	Local Rx
64	148808	148714	148810	144998
128	84458	84485	84459	84459
256	45289	45289	45289	45289
512	23496	23496	23496	23496
1024	11973	11973	11973	11973
1280	9615	9615	9615	9615
1518	8127	8127	8127	8127

the ideal curve(figure 4.7); so it is noticed that with small size packets the device is not able to receive high number of packets, so it lost a lot of packets (higher then 50%). Data are present in the tables 4.3 and 4.4. From the data is possible to notice that the performance with small size packets are very poor; packets of 64Bytes are received from the Local RX at 190 Kbps, a very low rate compared of the maximum rate. So in conclusion, the commercialized ROJ eNUC cannot be used as IP forwarder device with packets less then 512 bytes.

Table 4.3: Data of Throughput of ROJ Smarc in Mbps

Size(Bytes)	Local Tx (Mbps)	Remote Rx (Mbps)	Remote Tx (Mpbs)	Local Rx (Mbps)
64	100.00	22.14	100.00	0.19
128	100.00	43.66	100.00	1.63
256	100.00	79.07	99.99	3.88
512	100.00	99.98	100.00	97.17
1024	100.00	100.00	99.99	99.99
1280	100.00	100.00	99.99	99.99
1518	100.00	100.00	100.00	100.00

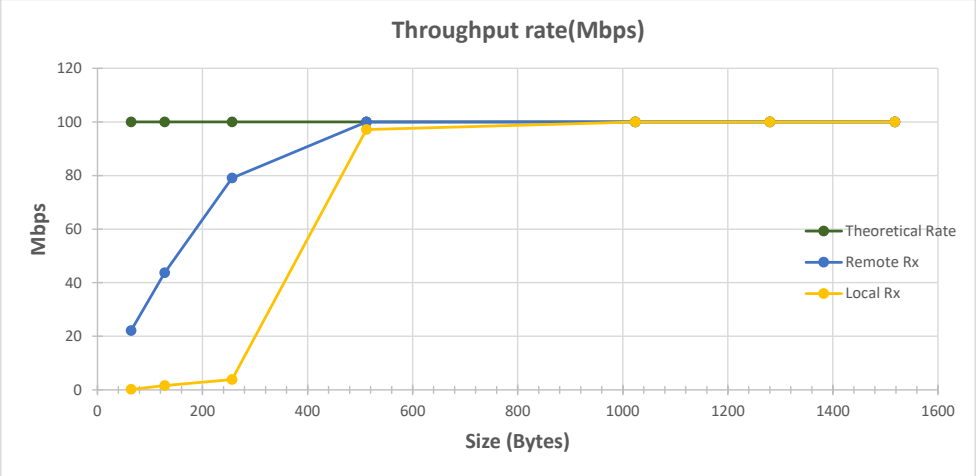

Figure 4.6: ROJ Smarc Throughput Test

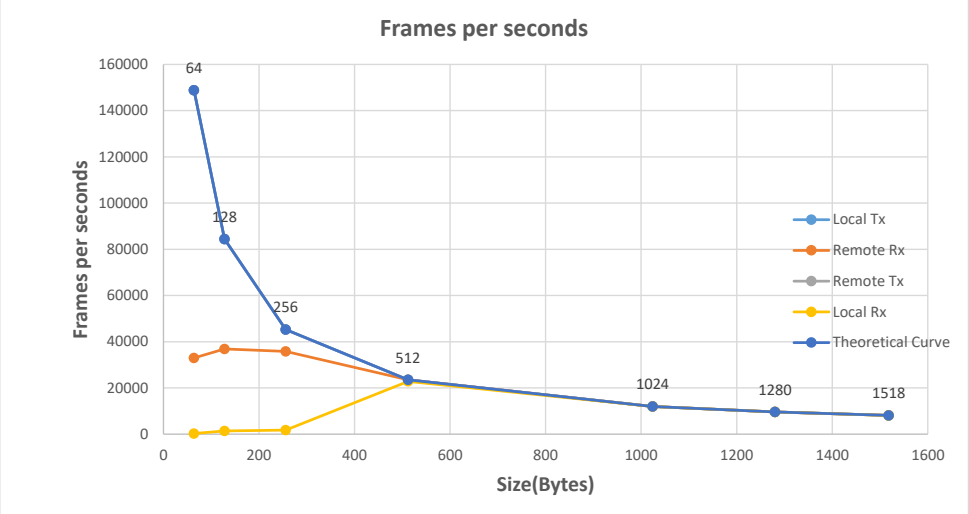

Figure 4.7: Packets vs Size Graph of ROJ Smarc

4.2.3 ENA Throughput test

The ENA device shows a very poor performance with packet dimension lower than 1024 Bytes.(figure 4.8) The device is very slow with those packets and also shows a not good symmetry. The number of packets that are received are very few respect the theoretical curve (figure 4.9). The data are in the tables 4.5 and 4.6. It is possible to notice that the ENA with packets less than 1024 , has a very poor performance; the data are received

Table 4.4: Number of frames per second of ROJ Smarc

Size(Bytes)	Local Tx	Remote Rx	Remote Tx	Local Rx
64	148813	32954	148806	290
128	84462	36877	84457	1378
256	45291	35812	45287	1759
512	23496	23492	23496	22830
1024	11973	11973	11972	11972
1280	9615	9615	9615	9615
1518	8127	8127	8127	8127

with throughput rate of Kilobit order, that is 3 order of magnitude less then the ideal performance.

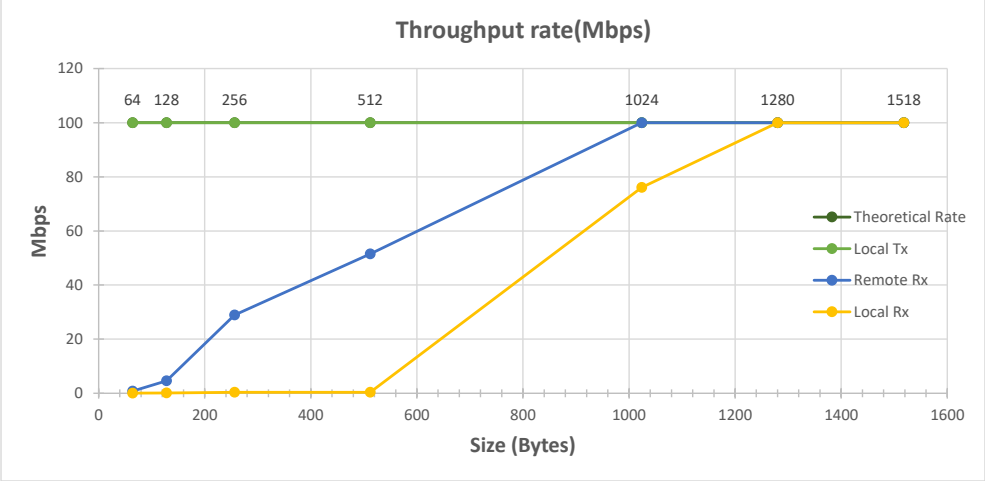

Figure 4.8: ENA Throughput Test

4.2.4 Performance Conclusions

The tests have shown different trends and performance using different devices. In the following graph, a comparison between the three devices is performed. A worst-case analysis is done to consider the worst performance.In conclusion the ENA is the worst device to perform IP forwarding operations because it shows the worst trend respect of the ideal curve. The performance of the ENA are in line with the general purpose device only with packets larger than 1280 bytes. At 1024 bytes the performance are acceptable (throughput: 76 Mbps). A similar but better trend is shown in the eNUC device. So, the causes of these performance are probably due to some bugs into the microprocessor IMX6 kernel present

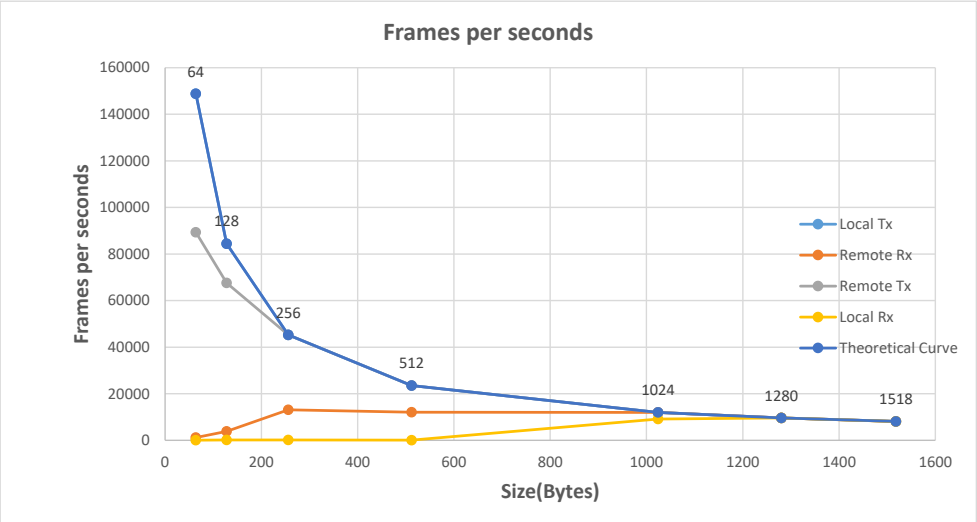

Figure 4.9: Packets vs Size Graph of ENA

Table 4.5: Data of Throughput of ENA in Mbps

Size(Bytes)	Local Tx (Mbps)	Remote Rx (Mbps)	Remote Tx (Mpbs)	Local Rx (Mbps)
64	100.00	0.80	60.03	0.0603
128	100.00	4.56	80.05	0.1407
256	100.00	28.93	100.00	0.3425
512	100.00	51.50	100.00	0.3520
1024	100.00	100.00	99.99	76.0800
1280	100.00	100.00	99.99	99.99
1518	99.99	99.99	99.99	99.99

Table 4.6: Number of frames per second of ENA

Size(Bytes)	Local Tx	Remote Rx	Remote Tx	Local Rx
64	148814	1191	89328	90
128	84460	3849	67609	119
256	45290	13104	45289	155
512	23496	12100	23496	83
1024	11973	11973	11973	9110
1280	9615	9615	9615	9615
1518	8127	8127	8127	8127

in both devices (ENA and eNUC). So from now on, it is considered to work only with packets of big size (1024,1280,1518) and relative considerations are done in this situation.

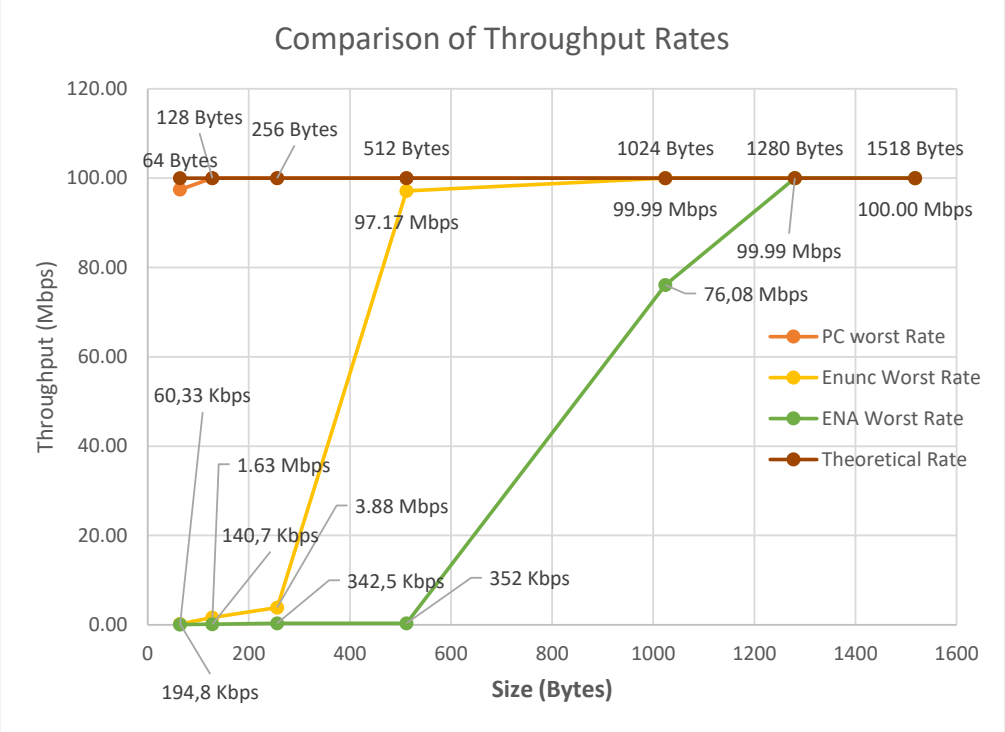

Figure 4.10: Comparison Throughput test

4.3 Throughput performance with application IP Forwarding

In this section, throughput performance are done with the user space application. The application performs an Ip Forwarding at higher level respect the Kernel level. The application open sockets at user space level and forward the packets that arrive from one interface to the other. For a consistent analysis, it is performed tests in these conditions, before adding the write and reading operations on the FPGA(for encrypt and decrypt the packets). From figure 4.12 it is possible to notice the throughput, with packet size that changes and the maximum throughput that is fixed at 100Mbps. It is possible to notice a not uniform trend. One of the cause of this trend is due to the fact that each packet that arrives has to be collected from the kernel and has to be sent to the user space level where the packet is managed and is forwarded to the second interface. These operations takes time and reduce consistently the throughput. Smaller is the packet, at 100Mbps, higher is the number of packets that the CPU has to process; in fact a better performance

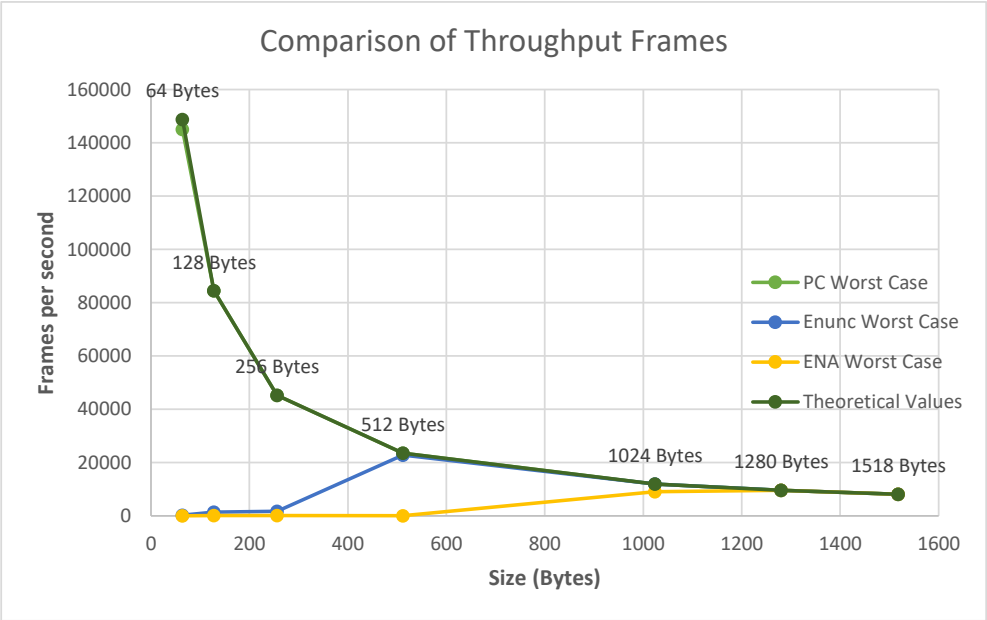

Figure 4.11: Comparison Packets vs Size Graph

are achieved when the packets are bigger. So if packets with size higher than 1024 byte are considered, a maximum throughput of 68Mbps is achieved. This means that the real throughput of the ENA cannot overcome this value, so with the FPGA that encrypts and decrypts packets, lower values are expected. From the figure 4.13 it is possible to analyse the number of packets that are received. With small packets, the number of packets that the CPU has to be handle is so high that it is not able to manage all packets , so the number of packets per second that the second interface receives, respect the number of packets that are sent, dramatically drops. Better performance are achieved with bigger packets. Here due to the dimension of packets, that is higher, the number of packets that the CPU has to be handled is lower, so the number of received packets per second is higher. In tables 4.7 and 4.8 values of the test performance are represented.

4.4 Throughput performance with Encrypting and Decrypting application

In this section the throughput of the ENA with the Encrypting and Decrypting function enabled is performed. When a packet arrives, it is collected by the kernel, it is sent to the user space level and it is encrypted by the FPGA. The FPGA has a 16 bits bus connected to the IMX so to perform all the operation a consistent number of operations the CPU and the FPGA have to perform, so, this takes time. The throughput is high influenced by this required time. In figure 4.14 is present the test throughput of the machine. With small

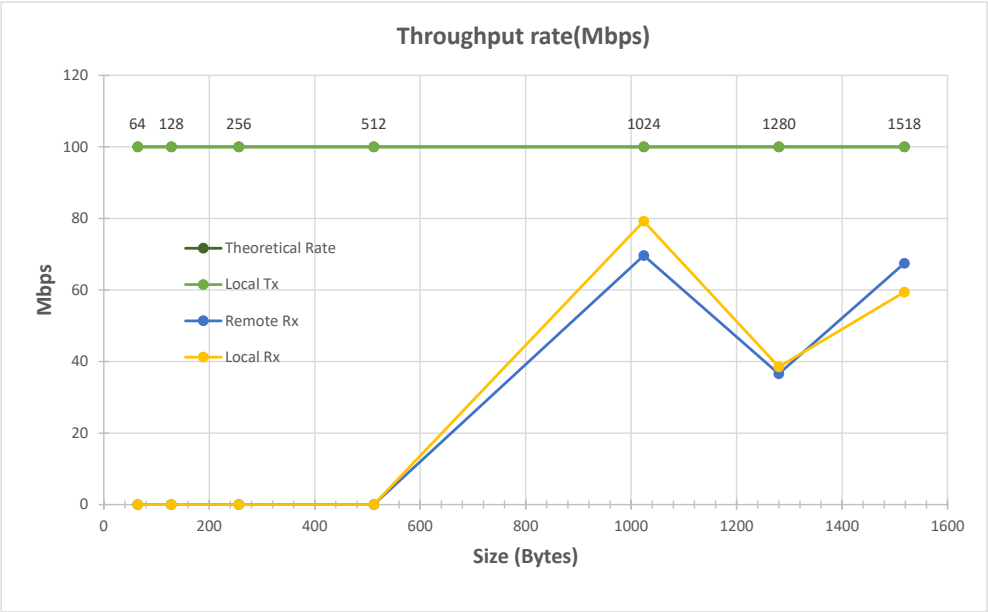

Figure 4.12: Throughput Test of ENA with IP forwarding application

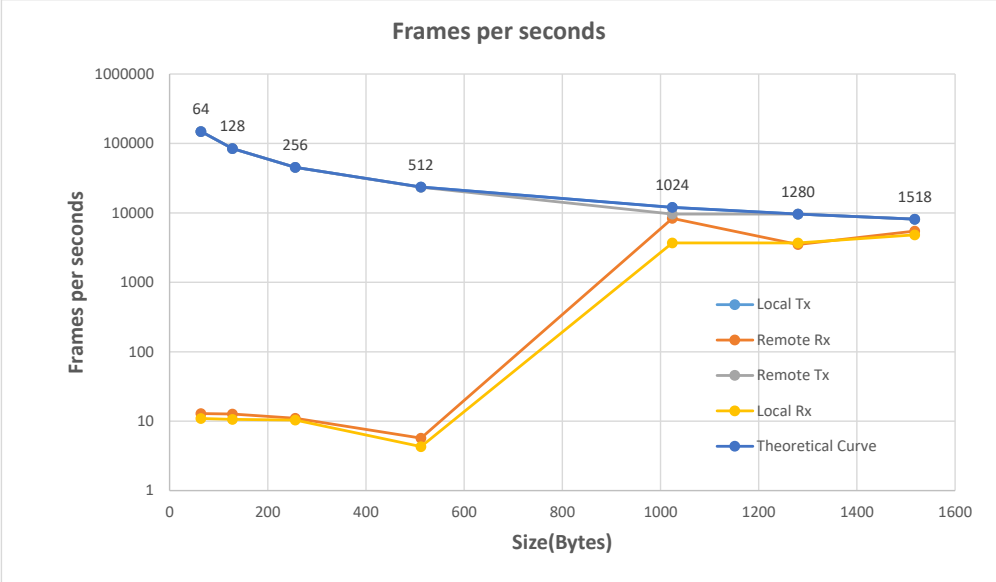

Figure 4.13: Packets vs Size Graph of ENA with IP forwarding application

Table 4.7: Data of Throughput of ENA with IP forwarding application in Mbps

Size(Bytes)	Local Tx (Mbps)	Remote Rx (Mbps)	Remote Tx (Mpbs)	Local Rx (Mbps)
64	100.00	0.0086	100.00	0.0074
128	100.00	0.0150	100.00	0.0126
256	100.00	0.0244	100.00	0.0230
512	100.00	0.0243	100.00	0.0183
1024	100.00	69.6300	100.00	79.2100
1280	100.00	36.5600	100.00	38.4900
1518	100.00	67.4500	100.00	59.3900

Table 4.8: Number of frames per second of ENA with IP forwarding application

Size(Bytes)	Local Tx	Remote Rx	Remote Tx	Local Rx
64	148813	13	148806	11
128	84477	13	84454	11
256	45292	11	45288	10
512	23497	6	23495	4
1024	11973	8337	9615	3701
1280	9615	3515	9615	3701
1518	8128	5482	8126	4827

size packets, the CPU has to handle high number of packets so the throughput drops 6 order of magnitude lower that the maximum throughput (100Mbps). If it is considered the maximum speed that is reached with 1518 byte packets, it is lower than the throughput, in the same condition, of the application without the Encrypting function. A reduction of about 88% is reached; this is caused mainly due to the high time to perform the encryption of big size packets. From the figure 4.15 it is possible to notice that the number of packets that the machine is able to process is very low respect the theoretical value(equal to the Local Tx sent packets). In tables 4.9 and 4.10 the test throughput performance are represented.

4.5 Comparison between ENA with IP Forwarding application and Encrypting/Decrypting Application

In this section a deeper analysis of the performance of the ENA is performed. Until now, it is imposed a fixed maximum throughput of 100Mbps; so the packets are sent to the ENA at 100Mbps. Now, the maximum throughput is sweeped. So changing the Throughput, it is possible to draw a graph and some final considerations can be done. For each packet size it is sweeped the throughput starting from small value (1 Mbps). A fixed throughput is selected and the received throughput is computed. The small packet size are present in

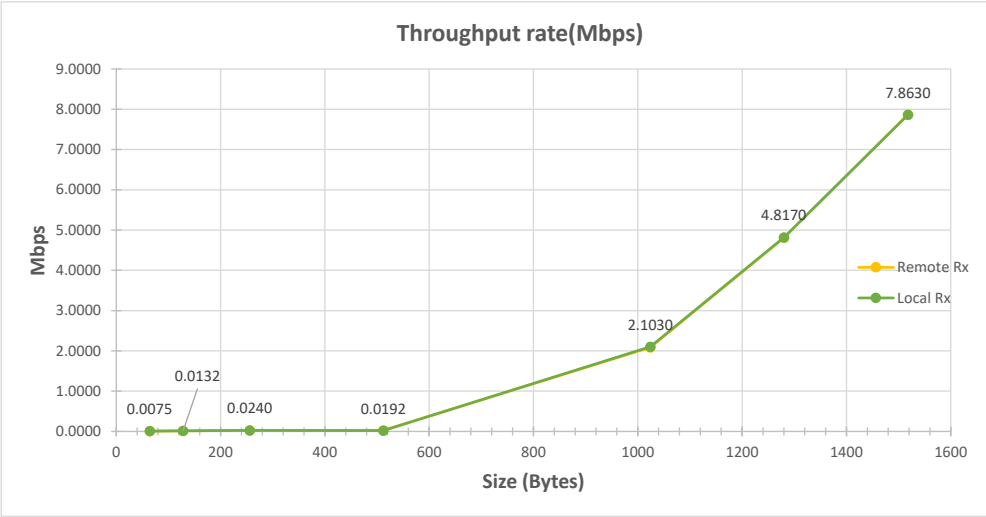

Figure 4.14: Throughput Test of ENA with Encrypting and Decrypting application

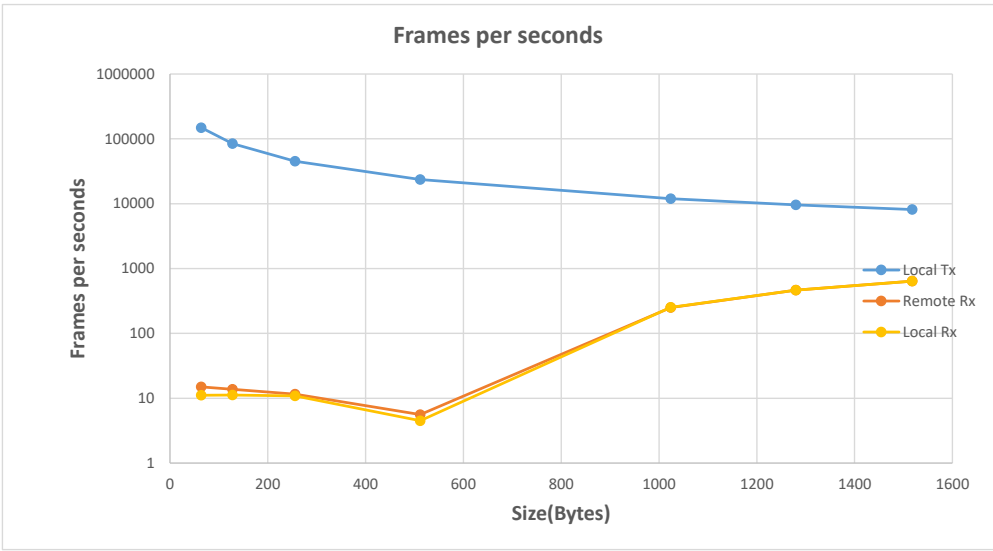

Figure 4.15: Packets vs Size Graph of ENA with Encrypting and Decrypting application

table for completeness but due to the fact that with those packet sizes the results are very poor, the analysis is deeper focused on bigger sizes (1024, 1280, 1518)bytes. When the throughput is low, the machine is able to encrypt and decrypt correctly the informations, so applying for instance, 2 Mbps the encrypted data are received at 2Mbps. So a theory line is a straight line that in a xy axis has the function $y = x$. But increasing the throughput

Table 4.9: Data of Throughput of ENA with Encrypting and Decrypting application in Mbps

Size(Bytes)	Local Tx (Mbps)	Remote Rx (Mbps)	Remote Tx (Mpbs)	Local Rx (Mbps)
64	100.00	0.0100	100.00	0.0075
128	100.00	0.0163	100.00	0.0132
256	100.00	0.0256	100.00	0.0240
512	100.00	0.0238	100.00	0.0192
1024	100.00	2.0880	100.00	2.1030
1280	100.00	4.8120	99.99	4.8170
1518	100.00	7.8670	99.99	7.8630

Table 4.10: Number of frames per second of ENA with encrypting and decrypting application

Size(Bytes)	Local Tx	Remote Rx	Remote Tx	Local Rx
64	4465975	448	4464142	334
128	2533847	412	2533677	335
256	1358749	348	1358657	326
512	704924	168	74876	135
1024	359211	7499	359192	7553
1280	288459	13880	288451	13894
1518	243817	19181	243799	19171

the number of packets that are sent (fixing the size) increasing too; furthermore, with small packet dimension, the number of packets that has to be processed, is higher respect the number of packets of bigger packet size. So the performances of small packet is very poor. Considering for instance, a size of 128 Bytes from the figure 4.16 is possible to notice that at very low throughput the system is able to encrypt the data but above 4Mbps, the performance start to drop and after about linear decrease, the system is able to receive very small packets per second(in order of units). Increasing the size of packets, the situation gets better. With the biggest packet size (1518 byte) the maximum throughput peak is at about 15Mbps but the maximum throughput is 20Mbps, so a loss of 22.3% of packets is shown. This loss is too high so the maximum acceptable throughput is at about 15Mbps where the losses are less then 2.5%. After a peak, each curve drops; in theory, after the peak, the curve has to saturate at the peak value because even if the throughput increases, the CPU of IMX6 is able to handle only packets with the maximum peak throughput, but this doesn't happen. The causes are different and other analysis and graphs are realized.

One of the possible causes of this decreasing trend, can be attributed to the necessary time to perform the Encrypting operation through the FPGA, so an analysis with the sweep of throughput also with the IP Forwarding application without the Encrypting and Decrypting application is performed and a comparison is made. In figure 4.17 the Sweep

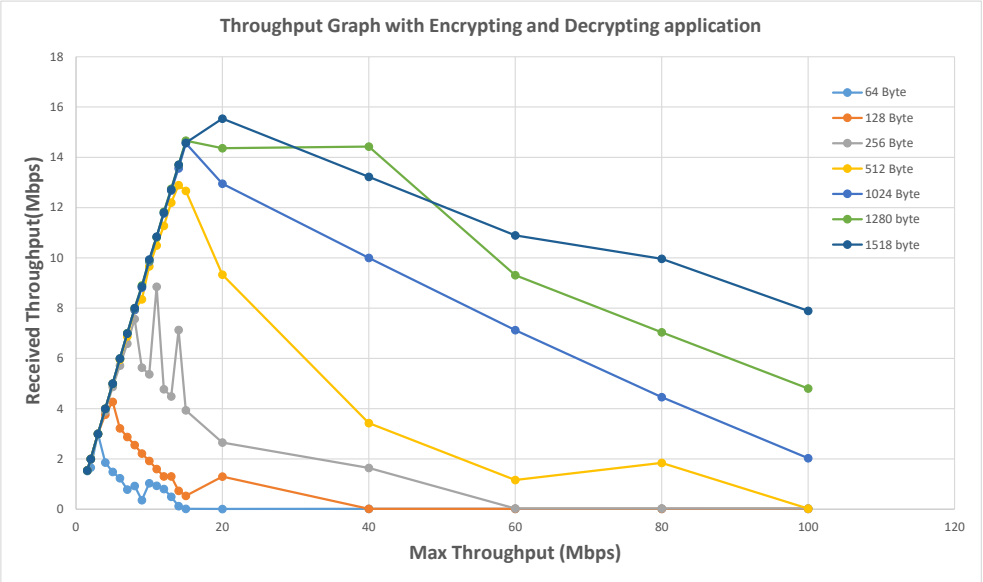

Figure 4.16: Sweep Throughput Test of ENA with Encrypting and Decrypting application

throughput test of the ENA with the IP forwarding application and with the Encrypting application is represented. Some considerations can be done considering the trends. With low throughputs, the system is able to process all informations and encrypt the data; in this case the speed for encrypting the data is not the limiter. Increasing the throughput, the number of packets that have to manage, increases, so the FPGA has to manage a lot of packets in small time, for this reason after a throughput value of 15Mbps , the machine is not able to perform the encryption process faster so, comparing the trend of the IP Forwarding application is possible to notice that after 15Mbps the FPGA computation is the "bottleneck" of the system.It is possible to see in figure 4.18 a zoom that better explains the trend. In fact the IP forwarding application reaches higher performance also above 15 Mbps. However both applications has a similar trend, in fact, after their peak, the throughput drops down. This cause is because the CPU is not able to manage so high packets and due to limits in internal buffer dimension, interrupt kernel management, and other types of process that cannot be managed at user space level.

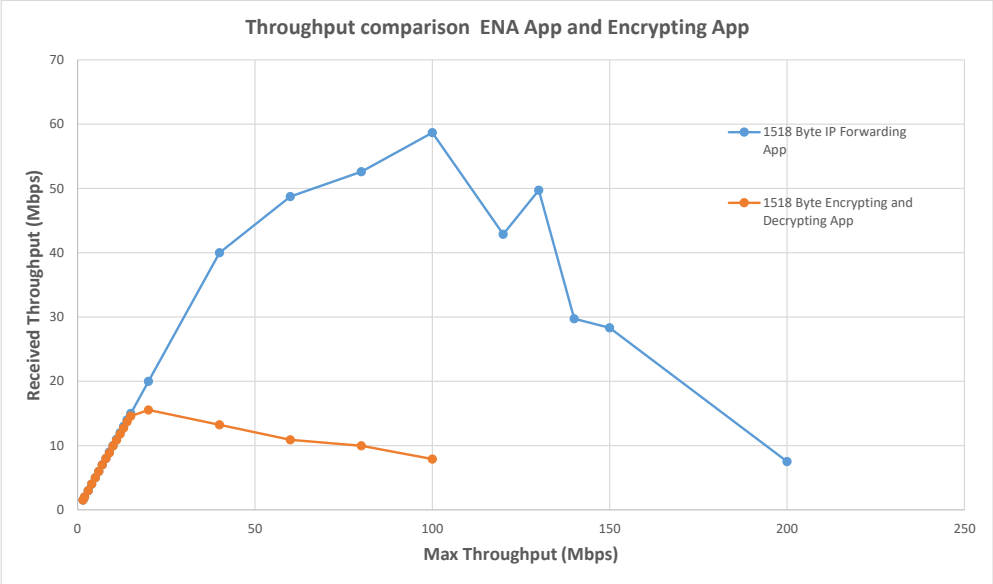

Figure 4.17: Comparison between Throughput test of IP Forwarding Application and Encrypting Application

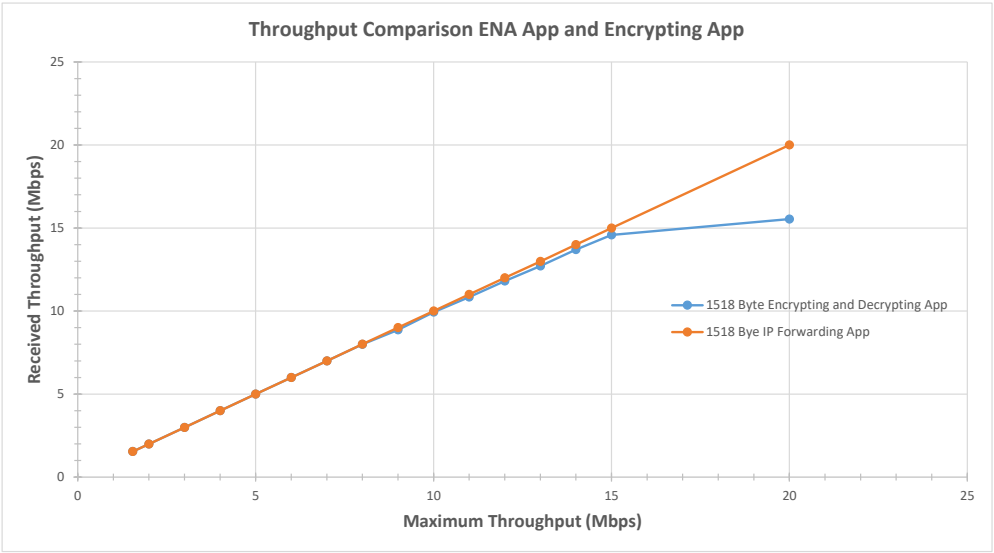

Figure 4.18: Zoom of comparison between Throughput test of IP Forwarding Application and Encrypting Application

Chapter 5

Conclusion

Thanks to the throughput tests, it is possible, in conclusion, to analyse the results and the final specification of the product, so, it is possible to highlight the limits of the microprocessor and all the restrictions. All the results are summarized and discussed in this section. The final goal of the system is reached after some intermediate steps. The first step is performed to declare and find the maximum physical throughput that the ENA can reach. So a throughput test with IP forwarding is performed and a comparison with other commercial system is done. Before this, a comparison is made between the ENA and the commercial ROJ Smarc to perform consistent tests. It is discovered that the two machines are similar and have the same hardware network interfaces but two different Linux Kernel. So, the IP forwarding is performed and throughput tests are executed.
It is discovered that a general purpose system is the best machine to perform network operations related to IP communication protocols and it is the best way to handle Ethernet frames and forward them. In fact, with a maximum throughput of 100 $Mbps$, it is able to forward all packets, without losses, and the packets are received from the Test machine at 100 $Mbps$.
After that, it is performed throughput tests on a commercial embedded system called ROJ Enuc Smarc. The performance of ROJ Enuc are lower than the general purpose system. The number of small packets(up to 256byte) at 100Mbps are higher respect bigger packet size, so the system has to handle higher number of frames, for this reason the system is not able to forward all the packets and at the receiver the number of packets are less. So the performance are very poor, with a losses higher than 96% with packet size less than 256 byte. The 100 $Mbps$ are reached with packets size equal and higher than 512byte. With packets of 512 byte all the packets are received without losses.Thanks to this results, it is possible to understand and forecast the trends of the ENA due to the similarity with the ROJ device.
The last test with Kernel IP forwarding is performed on the ENA device. The 100 $Mbps$ of maximum throughput is reached only with packets equal and higher than 1024 byte. In this condition all the packets are received and there are no losses, but a not symmetrical trend is shown. In fact, the receiver of the local machine test, receives lower packets, and consistent losses are found (24% of losses). So acceptable performance are reached only

with packets higher than 1280byte where the two receivers are symmetrical and there are no transmission losses. In conclusion, with the first throughput test step, it is possible to fix the minimum frame size that the system is able to handle. So the next tests are focused mainly on packets with size higher than 1280byte.
The second step is the introduction of the application at user space that handles the Ethernet frames. In this situation the CPU has to manage packets that arrives and from the kernel level has to send them at user space level. These operations take time and the performance consistently reduces. With 1024 byte packet size, a maximum throughput of 68 Mbps is achieved. This value fixed the maximum throughput that the ENA with IP forwarding application is able to manage. So it is expected that the maximum throughput with the Encryption and the decryption of the packets is less than this value.
The last step is the final application that allows to encrypt or decrypt the Ethernet frames that arrive. The packets are collected from the kernel, after the opening of the socket, they are saved into a buffer and send into the FPGA. Due to the 16-bit bus between the microprocessor and the FPGA, the frame have to be sent 16-bit at time; this operation requires time that lowers the maximum throughput. With small packets, despite of their size, so less number of read/write operations in the FPGA for one complete frame, the number of packets is higher, so the performance are very lower, not sufficient. The maximum throughput reachable is 7.8 $Mbps$. So this is the real received throughput that the system reaches. In this condition, however, the losses are very high, due to the fact that the packets are sent at 100 $Mbps$ and are received at 7.8 $Mbps$. So a final analysis is performed.
In this last tests, the packets are sent not at fixed frequency of 100 $Mpbs$ but at variable throughput. In this way it is possible to understand the maximum throughput where there are no losses. A sweep test is performed starting from 1 $Mbps$. With low throughput, the machine is able to encrypt and decrypt correctly the information. For instance, with 2 $Mbps$ the data are received at 2 $Mbps$ without losses. So a theory trend is a straight line. However, increasing the throughput, the number of frame increases and with small packets(so, high number of packets), the performance starts to drop. A better trend is shown with big size packets, but, at a certain throughput point, the performance dramatically drops. The maximum throughput with no losses is 15Mbps, after this value the losses start to begin consistent and the performance drops. The same tests are performed on the ENA with IP forwarding application without the encryption and it is possible to notice that the maximum peak is at about 60$Mbps$;so, the system has better performance. Thanks to this measurements, it is possible to understand that the FPGA computation is the "bottleneck" of the system. However after a peak, both trends start to drop. In theory, after the peak, the curve saturates at that value, but this no happens, because with high throughput the number of packet frames is so high that the CPU is not able to manage all these informations and drops.
So, at the end, it is possible to assert that the maximum throughput that the machine is able to manage is 15 Mbps with 1518 byte packet size. Thanks to this work, it has been possible to highlight the limits of the microprocessor and, although with the current restriction of the ENA could not be suitable for being an enterprise product, it represents a good proof of the concept of how deeply military companies have to deal with the topic of encryption of information for security applications.

Appendix A

Open System Interconnection/International Standard Organization

Open System Interconnection/International Standard Organization(OSI/ISO) is a model that standardizes the communication functions to ensure a compatible national and world-wide data communication. In this model, there are several partitions called abstraction layers. It is a seven layer architecture that defines a complete system. The different levels are:

1. Physical Layer
2. Datalink Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

- The Physical Layer is the lowest layer of the model, in fact, it defines the electrical, optical and all the physical specifications. So it includes medium(optical fiber,coaxial cable), pins, voltages, line impedence, signal timing, frequency,ecc. The transmission can be simplex, half duplex or full duplex(allows communications in both directions).
- The Datalink Layer synchronizes the information. So its main function is to make sure that there aren't errors in the physical layer. The data frames that are transmitted and received are managed by this layer. It is divided in two sublayers:

  - Logical link control(LLC) layer, that is the upper sublayer. It provides a mechanism in order that different protocols can exist together and to transport the information onto the same medium. It also provides the flow control and the request errors. The LLC is also an interface between the Media Access Control(MAC) and the Network upper layer.
  - Medium access control(MAC) layer: In this layer, the MAC provides channel access control; so it ensures that signals sent from different points across the same channel don't collide. To do this, MAC protocol is used; it assigns a unique identifier number that is assigned to all ports on a switch. The MAC address is an hexadecimal number(6 byte) that is divided into different parts in specific formats.

- The Network Layer provides the procedure to transfer different data sequences from one node to another; it acts as a network, managing the traffic. It also divides the messages into packets. It chooses the best path for each message to arrive at destination.

- Transport Layer provides services for the transport of data. It has the task to open and close the communications, to split the messages in smaller units that pas on the network layer, decides if the data transmission should be on parallel or single and handles the multiple connections.

- The Session layer controls and manages the conversation between computers. It ensures that the messages are not cut or data are not lost. So its main task is to carry out a session.

- Presentation Layer takes care that the information is sent in a "good way", so that the receiver will understand the information and is able to use it. So it is able to provide independent data compared to different representation by translating the information.

- Application Layer is the topmost layer. It interacts with the software applications that implement the communication for example an email service software.

Appendix B

IP-Internet Protocol

B.1 Overview and main characteristics

The Internet Protocol (IP) is a network protocol included into the TCP/IP protocol Suite;so it is a set of rules to exchanging messages across a series of interconnected networks. It is the main Inter-Networking Protocol of the layer 3 of the ISO/OSI model. It is a simple protocol based on connectionless packets and best-effort type. It is not reliable so it doesn't ensure any type of communication reliability in terms of error checking and flow control. The packet sent can be lost, duplicated or delayed but the IP protocol doesn't inform both receiver and transmitter. It is connectionless, this means that every packet is processed independently by other packets. So for example different packets with the same transmitter and receiver can proceed with different path and can be received or not. The first major protocol is the IPv4; there is also a new version, IPv6 that tries to solve the principal limitations of the IPv4 protocol like the number of IP address, that are almost not-sufficient but the IPv6 reaches nowadays 20% of the internet traffic, so the IPv4 remains the most used. The most important functions that the IP protocol performs are:

- It distinguishes every host that is a network card with an identifier, called IP address. A single (unicast) IP address identifies a single host, but a single host can have many unicast IP, according to the number of network cards. For example a router has many addresses because it is a sorting centre with different network cards.
- It receives the data (in form of PDU-Protocol Data Unit) from the Transport Layer (layer 4).
- It encapsulates each PDU in different packets with a maximum dimension and adds an header.
- It splits the packets during the transport to insert them into layer 2.
- It routes the packets on the network.
- It identifies the errors, but it doesn't correct them.

- It extracts the data (PDU) from the Transport Layer.
- It delivers the data to the transport layer in the order in which they arrived at destination, that can be different from the departure order.

B.2 IPv4 packet

Every IP packet is composed by a header and a data section that is the portion of data to be sent. An IP packet(IPv4) has:

- a fixed part of 20 bytes;
- a second optional part with variable length, but it must be a multiple of 4 byte.

It is structured as follows:

- Version: it is the first header field of 4 bits and it indicates the version of the protocol IP that has generated the packet. It is used by the receiver to understand the type of format of the packet.
- HLEN (4 bits): it indicates the length of the header IP in a word of 4 bytes. The header fields have a fixed length, except for the Options field that can vary.
- Type of Service (8 bit): it indicates the type of the transport that is needed to the router by decide a priority.
- Total Length (16 bits): it indicates the total length of the packet expressed in byte. Often the routers don't use this field.
- Identification (16 bits): it is used to identify the fragments of a single IP datagram. This happens when a packet is split into different packets.
- Flags (3 bits): it is used to control and identify the fragments.
- Fragment Offset (13 bits): it indicates the number of bytes of data that are present in the previous packet. If the fragment is the first the field is 0.
- Time to Live (8 bits): it is a time that indicates the living time of a packet, after that time, the packet can be rejected. The value is initialized at 255 from the sender host. Every time that the packet is processed by a router, time is decremented by 1 unit. It avoids an infinity loop inside a network.
- Protocol (8 bits): it indicates the type of the transported data, that is the type of protocol of layer 4.
- Header Checksum (16 bits): it verifies the integrity of the header when it arrives to destination.
- Source IP (32 bits): it contains the IP address of the source

- Destination IP (32 bits): it contains the IP address of the final destination of the IP datagram.
- Options: it is not often used. It is a variable length field and due to the fact that the field must have a fixed multiple length of 4 bytes, it is used to complete the packet with the padding.

Appendix C

Secure Hash Algorithm

The term SHA is the acronym of Secure Hash Algorithm and it is a family of five different cryptographic hash functions. The SHA produces a digest message that is a sort of digital print of the message, that has a fixed length even if the message is not fixed. The main feature is that it is a non reversible function. So, it is not possible to know the original message knowing only this data. It has also the feature that it is no possible to generate two different message with the same digest. The algorithms are SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512. The last four algorithms are indicated as SHA-2. The difference between the different SHA is in the number of bits of the digest. A message is processed by blocks of 512 and it requires 64 rounds to be compute. The SHA is used mainly to discover if a data was corrupted.

Appendix D

Networking

Internetworking is "the concept of interconnecting different types of networks to build a large, global network"[19], in this way any pair of host can exchange packets and can communicate. Internet is a set of networks connected by routers that are configured in such that the traffic pass among any computers that are attached to the network. Today millions of computers are connected world-wide. To create a network: a stardard scheme is needed to address the packets to any host; then a standardized protocol that defines the format and handles the transimitted packets is needed and a components that routes the packets to their destination. The smallest network is formed by two LANs of computers that are connected to each other via router. A Local area network(LAN) is a network that interconnects computer in a limited area. The internet protocol is designed to provide a not guaranteed packet service. To transfer the data, the applications must utilize an appropriate protocol such as the TCP(Transmission Control Protocol) that is a reliable protocol(so guaranteed transmission) or the use of UDP(User Datagram Protocol) that is a connection-less transport protocol, used principally for data that require real-time service like voice or video streaming. Each host and router is defined through address with fixed length, that are grouped in IP network. An IP address has a fixed length of 32 bit. They are written in sequence of four decimal number with values from 0 to 255 and separated by a dot. The address identifies the interconnection point of one host with a net. But with an address it is possible to identify not an host but one of its interface. so One host can have more than one IP address like in the designed board. The IP address is subdivided in two parts:
- Network ID : it is a prefix that identifies the network;
-Host ID : it identifies the host interface.
The Network ID occupies the left part of the address and te Host ID the right part. The Net-ID can have different dimensions and Classes are defined to differentiate the networks. Different classes of addresses are present:

- A class : The Net-ID that identifies an IP network has the first decimal number fixed. So it is possible to index numbers from 0.0.0.0 to 127.255.255.255.

- B class : Addresses from 128.0.0.0 to 191.255.255.255

- C class: From 192.0.0.0 to 223.255.255.255
- D class: From 224.0.0.0 to 239.255.255.255
- E class: From 240.0.0.0 to 255.255.255.255

It is possible to decide locally a sub-ripartition of the NET/Host ID. So it is possible to fragment the Host ID in two parts:
- The first part identifies the subnet;
- The second part identifies each interface host of the subnet;
For th subdivision it is necessary to use the Netmask; it must be configured to route the IP address correctly. For example if it is used a network of B class (ex: 192.168.0.0) it is possible to use the first byte of the Host-Id as subnet index ; in this way from the network of B class it is possible to extract 254 network of C Class dimension. To do this a netmask of 255.255.255.0 is used. The netmask is used to identify its own NET-ID from the IP address. The subnetting is identifies using a specific notation: IP Address/(Net-ID dimension + Subnet-ID). For example 192.168.5.0/24. Each packet that travels has to be route, so the routers create a structure: in this way a datagram passes from one to another router until it reaches the destination.

Bibliography

[1] ALTERA. *Cyclone IV Device Handbook, Volume 1*. Version Rev 2.0. 2016. URL: https://www.altera.com/en_US/pdfs/literature/hb/cyclone-iv/cyclone4-handbook.pdf (cit. on p. 21).

[2] William Fornaciari Carlo Brandolese. "Sistemi Embedded - Caratteristiche, tecnologie e mercato". In: *Mondo Digitale* 3 (2009), pp. 3–10. URL: https://home.deib.polimi.it/fornacia/lib/exe/fetch.php?media=papers:2009:2009_mondodigitale_embedded_p_3_10.pdf (cit. on p. 3).

[3] Micron. *Micron Serial NOR Flash Memory Reference manual*. Version Rev B. 2014. URL: https://www.micron.com/resource-details/2dd46e97-8a6c-4ed2-81c8-7d77528076c2.

[4] [Online]. URL: https://www.ic3.gov/media/annualreports.aspx (cit. on p. 1).

[5] [Online]. URL: https://www.zionmarketresearch.com/news/embedded-systems-market (cit. on p. 4).

[6] [Online]. URL: https://en.wikipedia.org/wiki/ARM_architecture.

[7] [Online]. URL: http://www.st.com/en/microcontrollers/stm32f7x6.html?querycriteria=productId=LN1902.

[8] [Online]. URL: https://developer.arm.com/products/processors/cortex-m/cortex-m7.

[9] [Online]. URL: https://en.wikipedia.org/wiki/ARM_Cortex-A.

[10] [Online]. URL: https://en.wikipedia.org/wiki/Small_form-factor_pluggable_transceiver.

[11] [Online]. URL: https://en.wikipedia.org/wiki/Ethernet#Frame_structure.

[12] [Online]. URL: https://en.wikipedia.org/wiki/OSI_model.

[13] [Online]. URL: https://en.wikipedia.org/wiki/IPv4.

[14] [Online]. URL: https://en.wikipedia.org/wiki/Microcontroller.

[15] [Online]. URL: https://www.eleccircuit.com/why-use-microcontroller/.

[16] [Online]. URL: https://www.elprocus.com/embedded-systems-real-time-applications/.

[17] [Online]. URL: https://en.wikipedia.org/wiki/Internet_Protocol.

[18] [Online]. URL: https://en.wikipedia.org/wiki/Internetworking.

[19] Bruce S. Peterson Larry L.; Davie. *Computer Networks: a systems approach.* Ed. by Anche Leggoo. Elsevier, Inc., 2012 (cit. on p. 76).

[20] NXP Semiconductors. *i.Mx 6Quad applications Processors for Industial Products.* Version Rev 5.0. 2017. URL: https://www.nxp.com/products/processors-and-microcontrollers/applications-processors/i.mx-applications-processors/i.mx-6-processors/i.mx-6quad-processors-high-performance-3d-graphics-hd-video-arm-cortex-a9-core:i.MX6Q (cit. on p. 19).

[21] STMicroelectronics. *STM32F756xxx Reference manual.* Version Rev 1. URL: https://www.st.com (cit. on pp. 12, 13, 15–17).